UNIVERSITY OF TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
DEPARTMENT OF INDUSTRIAL ENGINEERING
FONDAZIONE BRUNO KESSLER

**Doctorate Program in Industrial Innovation**

# Modelling of Functions and Malfunctions in Industrial Processes: An Applied Ontology Approach

Francesco Compagno

Advisor
Stefano Borgo
Istituto di Scienze e Tecnologie della Cognizione (ISTC) - CNR

Industrial Tutor
Paolo Galvagnini
Adige S.p.A. - BLMGroup

January 2025

# Thanks

I thank my supervisor Stefano Borgo and my industrial tutor Paolo Galvagnini for their support, attention, and patience during these years. I also thank all my colleagues at the Laboratory for Applied Ontology for the remote and live meetings they organized and animated, in particular Emilio Sanfilippo, Daniele Porello, Alessandro Mosca, Nicola Guarino, Luca Biccheri, and Claudio Masolo for the many discussions on scientific topics. For the same reason I must extend my gratitude to many researchers outside the ISTC, but in particular to Alessandro Oltramari, Sergei Chubuanov, Evgeny Kharlamov, Walter Terkaj, Sergio Benavent, and Riichiro Mizoguchi. Finally, I would like to thank those who have read this thesis, in primis Paolo Bosetti, Giancarlo Guizzardi and Melinda Hodkiewicz, but also all those who may read it in the future.

# Abstract

This thesis stems from the need to identify and evaluate issues related to information modelling and management that are typical in manufacturing companies and, more broadly, in the industry. The thesis, which was funded by the company Adige S.p.A., a laser-cutting machines manufacturer, has as main objective the solution of problems raised by the following points:

- `I1`: Information, given its variegated nature and sources, tends to be serialized, structured, and organized in a heterogeneous manner.
- `I2`: Information is generally structured in a rigid way, in monolithic, non communicating silos.
- `I3`: Information is difficult both to store into and retrieve from the underlying information systems, leading to further issues of information sharing.

The three preceding points are interrelated. For instance, it is challenging to provide a modular (non-monolithic) structure to information that is heterogeneously structured. Moreover, sharing redundant and non-modular information is difficult[1]. In a broader sense, any difficulty or inefficiency in organizing information adds to technicians' natural reluctance to allocate their energy to feeding information systems – an activity that might not be perceived as immediately and directly beneficial to their work. This reduces the utility of these systems and, consequently, their impact on the work practices of the staff.

The conceptual tools used in the thesis to face these criticalities are part of a research area known as applied ontology. Applied ontology aims to produce practical applications through the exploitation of methods and theories from logic and philosophical ontology. An introduction to applied ontology will be presented later in the thesis, for the time being we remark that the choice to work within this research area is because the highlighted issues fall into the type of problems that have already benefit by the application of ontological techniques in knowledge engineering. As reported in the literature, the use of an ontology promotes:

[1]For instance, imagine a long and complex technical document that is continuously updated, producing a series of versions. Sharing the document and keeping an alignment with the correct version is difficult. More so if the document is managed in a non-modular, monolithic way, where the change of a single line is drowned in the change of the whole document.

- terminological precision and conceptual clarity in defining concepts,
- analysis and formalization of domain knowledge, including the explicit statement and serialization of otherwise implicit assumptions in a way that facilitates certain computational tasks,
- reuse and sharing of information.

The first two points promote the serialization and structuring in a homogeneous way, helping with `I1`. The second and third points hold (also) because applied ontology can establish formal and precise mappings between different conceptual schemas, breaking down silos, a capability that addresses Issue `I2`. The ability to map between conceptual schemas facilitates modularization, which is also studied by its own accord; in addition, ontologies allow information to be stored in ways renown to be flexible (e.g., by using knowledge graphs), both these facts help with `I2`. Finally, technologies associated with applied ontology (such as (knowledge-)graph databases, automated reasoning, logic-based modelling, etc)[2] possess various techniques for storing and retrieving information that can help with `I3`. The other characteristics of ontologies are also synergistic to this end: for example, difficulties experienced by technicians in systematically entering data into an information system may be exacerbated by a complex and confusing information structure, and alleviated by a clear and simple information structure.

It must be noted that the fields of maintenance and manufacturing are enormously complex and varied, so that our analysis should have a limited scope and only focus on certain specific topics within these disciplines. The choice of this thesis is to focus on functions and malfunctions of engineering systems: in addition to having a rich history in both the engineering and applied ontology literature, they are pivotal aspects of maintenance and manufacturing and are especially relevant in Adige's use case. Therefore, we hope their analysis will particularly help with issues `I1`, `I2`, and `I3`, even though it is limited in scope with respect to the full extension of the topics that are relevant for these three issues.

It is also important to note that this thesis focuses more on the formal ontological analysis of functions and malfunctions, as this topic is sufficient to fill completely the work of the thesis. A systematic study of the connection between issues `I1`, `I2`, and `I3` and said formal ontological analysis is outside the scope of this thesis, although chapters 1 and 5 do discuss some connections, both in general and by means of examples. More specifically, the thesis focuses on answering the following research questions:

- `RQ1` What are the main fundamental issues in knowledge representation in the notions of function, failure and malfunction in engineering, and how can applied ontology contribute to their solution?

  Question `RQ1` will be addressed both for pure research interest as well as to lay down the foundational basis of the ensuing work.

[2]Note also that these technologies are currently being exploited in the context of neuro-symbolic approaches to the latest trends in artificial intelligence.

- RQ2 How can the analysis of the fundamental knowledge representation issues of RQ1 enable applied ontology and semantic technologies to represent knowledge about functions and malfunctions in a well-founded way?

  Question RQ2 is addressed because Adige's research interest in this case is mainly related to interoperability issues, and well-founded knowledge representation is known to be a good answer to such issues. While we are interested in the particular cases provided by Adige's factory and products, from a scientific perspective, we want to develop solutions applicable to a broader set of cases sharing the same domain.

  Lastly, we want to showcase, by means of examples, how these solutions can be used to address application scenarios.

- RQ3 What are some possible ways to use ontologies (and semantic technologies more in general) to drive examples of relevant enterprise applications?

  Here by 'relevant' we mean relevant to any enterprise struggling with issues I1,2,3 mentioned above.

To answer these questions, the thesis starts from foundational concepts and gradually builds up to practical applications, thereby providing an exploration of applied ontology to knowledge management in some engineering contexts of interest. In particular, the thesis is structured as follows: In the **first chapter**, the discipline of applied ontology is introduced, as well as the host of semantic technologies spawned by the Semantic Web vision, providing an overview of the techniques used in the subsequent parts and explaining why they can be used, and are being currently used, in enterprises. Then, **second chapter** chapter contains a literature review detailing the state of the art of modelling of functions and malfunctioning in engineering and ontology. Informed by this review, the **third chapter** lays the groundwork for a theoretically sound adoption of semantic technologies by an enterprise: using the applied ontology approach to modelling, functional modelling of engineering systems is analyzed. Building on this, the **fourth chapter** focuses on modelling malfunctions. The **fiveth chapter** resumes the semantic methodologies and technologies presented and developed in the first three chapters, to exemplify how to exploit them in enterprises. In particular, this chapter showcases a few practical examples demonstrating the application of the theoretical concepts discussed in the preceding chapters on real use cases, and illustrating the benefits of applied ontology. Lastly, a brief, conclusive chapter summarizes the thesis content and contributions.

It is important to note that some of the work of this thesis has been serialized into various files, which are made openly available by means of two GitHub repositories. When appropriate, the repositories are mentioned in the thesis. However, for the convenience of the reader, links to the repositories are also provided here:

- Mainly for the work in Chapter 3: `https://github.com/kataph/function-method-ontology`.

- Mainly for the work in Chapter 4: `https://github.com/kataph/MALFO`[3].

Additionally, the core work of this thesis has been published in a series of papers, although this document modifies and expands their content most extensively, and adds original material. These papers are referenced in the thesis where relevant, but are also listed here for the convenience of the reader:

- for Chapter 3: [30, 55, 54, 52, 56].
- For Chapter 4: [53].

These works do not exhaust the works published during the Ph.D., but are the most representative of this thesis.


# Keywords

*Ontology, Function, Malfunction, Failure, Knowledge Graph*


[3]Including a file with an alignment to the other repository's work.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Background: Applied Ontology and Semantic Technologies

The main theoretical framework applied by this thesis is the one of Applied Ontology. In this introductory chapter this discipline is presented, together some of the most important technologies it is connected with.

## 1.1 Applied Ontology and the Semantic Web

Applied ontology aims to solve practical applications, especially in the context of conceptual and knowledge modeling, through the use of methods and theories from philosophical ontology. We will see later some examples of these methods, while we start this section by clarifying the meaning of some (buzz)words that are commonly heard of in and around this discipline:

**Ontology:** In this context, this word does not refer to the philosophical discipline of studying what exists, as much as to either certain logical theories and(/or) their implementation as software artifacts. Various definitions of 'ontology' exist: a typical one on the more general side is that "an ontology is a formal, explicit specification of a shared conceptualization" [237]. The ontology is 'formal' to ensure it is machine-readable, this means that an ontology employs a formal language, such as the Unified Modelling Language (UML), mathematical First-Order Logic (FOL), or the Ontology Web Language (OWL)[1], etc. Note that these languages are not ontological-languages but instead ontologically-neutral in the sense that they do not take any particular ontological commitment[2]. The formal language employed can be more or less complex and more or less expressive, and there is always a tradeoff between these two characteristics. In fact, the more a language is able to describe complex ideas, facts, situations the

[1] Througout this thesis, preexisting familiarity with FOL is assumed, while OWL will be discussed later in this section. UML is another language for conceptual modelling, but we don't focus on it in this thesis.

[2] Some small commitments are taken, e.g. in OWL there is an instantiation relation between 'classes' and 'individuals'.

higher the complexity class of (relevant computational tasks[3] related to) the language is.

Additionally, in any model or specification, there are things in the cognitive domain of the model designer that are relevant to the model but left unsaid. One wants to minimize the quantity of these unsaid things and put as much as possible into words or symbols, hence the 'explicit' part. Then, the ontology is 'shared' because it encodes the understanding shared among a community (e.g. the domain experts of some domain), and this part emphasizes the goal of facilitating knowledge exchange. Finally, an ontology is a conceptualization in the sense that, roughly, it describes the target domain intentionally, by specifying characteristics and relations of the relevant entities, and not (just) by listing examples.

We do not tarry more on this kind of definition, though the interested reader can consult e.g. [95] (as well as [199] for an example of criticism). Instead, we remark that in many (but not all) communities, such as the Semantic Web Community or the Natural Language Processing (NLP) community, the term 'ontology' has usually a much narrower meaning: the specification mentioned in the previous definition is specialized to software artifact that serializes a theory written using the Ontology Web Language to model some domain. The Ontology Web Language is one of the languages most often used to develop ontologies [251] and we will expand on its characteristics in the following paragraphs.

**The Ontology Web Language (OWL), and the Semantic Web:** OWL is a logical language used for knowledge representation that constitutes a pivotal part of the Semantic Web Stack: a set of technologies developed by the (W3C) to promote data standardization on the World Wide Web by facilitating the integration of semantic content within web pages (see Figure 1.3). OWL is frequently used in the context of ontology development [251], and the reasons for this are likely to be a complex mix of scientific and historical reasons. Regardless of the precise reasons behind its use, some of its main distinctive characteristics are:

- it has good computational properties[4].
- It is a widely recognized standard.
- It has a formal semantic.

[3]For instance, a classic computational task for a logical language, such as FOL or OWL, is to determine whether a formula is derivable by a theory. For OWL there is an algorithm that can do this in finite time for every theory and formula, thus OWL is said to be 'decidable'. FOL, which is more expressive than OWL, happens to be un-decidable precisely due to its espressivity.

[4]By this we mean that certain relevant computational tasks belong to tractable (i.e. at least polinomial in time) computational classes. However, since OWL is actually a set of dialectes called 'profiles', one must take care in determining the profile and the task. This will be discussed more in depth in the following, see e.g. Figure 1.2, but consider in particular that not all OWL dialects are decidable.

- It employs the open world assumption (facts not stated are not assumed to be false by default), it rejects the unique name assumption (two entities with different identifiers may be equal), and it uses IRIs as global entity identifiers.

Before describing OWL more in-depth, we will introduce briefly the original vision that spawned OWL: the Semantic Web. Though such a topic is not strictly necessary for understanding OWL, it is nevertheless essential to understand the ratios behind OWL design choices and a part of history that is tightly connected to applied ontology.

**The Semantic Web:** In 2001, Tim Berners-Lee and colleagues kickstarted a new vision for the web, called the Semantic Web, with an article in the Scientific American [23]. In the article, they imagined a scenario where automated agents automatically carried out tasks for users, such as turning off local audio devices or scheduling medical appointments. Arguably, the main characteristics were:

- Ontologized: some words in the thought-scenario are in italics (e.g., 'location', 'time', 'volume control', ...). This is because these words are supposed to belong to ontologies, intended as "document[s] or file[s] that formally defin[e] the relations among terms. The most typical kind of ontology for the Web has a taxonomy and a set of inference rules". The goal of these ontologies was to make explicit the meaning of the content of webpages to automated agents or to answer queries. Some examples made by Berners-Lee are entity disambiguation, search result enrichment, and complex information retrieval tasks "you met [Ms. Cook] at a trade conference last year. You don't remember her first name, but you remember that she worked for one of your clients and that her son was a student at your alma mater. An intelligent search program can sift through all the pages of people whose name is 'Cook' (sidestepping all the pages relating to cooks, cooking, the Cook Islands and so forth), find the ones that mention working for a company that's on your list of clients and follow links to Web pages of their children to track down if any are in school at the right place".

- Distributed: the Semantic Web would be free from oligopolies. In 2001 Tim Berners-Lee and colleagues imagined a distribution of data up to the individual level: a web user would have their own webpage and their own web-agent exchanging data with other users' agents "[Pete's web-agent] then began trying to find a match between available appointment times (supplied by the agents of individual providers through their Web sites)".

- Crawled by trusted agents: the Semantic Web would have been traveled by automated agents whose capabilities would have exponentially grown together with the quantity of machine-readable information. The agents would have had ways of establishing trust among them and with humans.

In 2001, the ontology languages used as examples were the Simple HTML Ontology Extension (SHOE) and DAML+OIL[5], and logical proofs were suggested to be used for

[5] `https://www.w3.org/TR/daml+oil-reference/`.

establishing trust. Work was later started that produced OWL in 2004, itself strongly based on DAML+OIL. Later, in 2006, Berners-Lee introduced a new keyword: 'linked data', which indicates the data produced a set of rules summarizing the idea of the Semantic Web[6]:

- "Use URIs as names for things".
- "Use HTTP URIs so that people can look up those names".
- "When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)".
- "Include links to other URIs. so that they can discover more things".

One also speaks of 'linked open data' if, in addition, the data is released with an open license. Under the banner of linked data, various databases were created, even of enormous size, such as DBpedia (a transposition of part of Wikipedia information into a structured, ontologized form, started in 2007[7]) and Wikidata (an ontologized RDF graph that everyone can freely contribute to, started in 2012[8]). The Google Knowledge Graph was also introduced in 2012 [68], as a way to enrich Google's search results, and grew rapidly to a gargantuan size. In 2011, the owners of the biggest search engines of the time (Google, Yahoo!, and Bing) published Schema.org[9], an ontology for tagging webpages content, analogously to the use of ontologies envisioned by Berners-Lee in 2001, to make search engines better understand the webpages' semantic. The set of datasets published as linked open data is pictorially summarized by the Linked Open Data Cloud project[10], for instance, the picture corresponding to the year 2014 is in Figure 1.1.

Semantic Web ideas, however, never gained widespread application, and activities such as ontology development and linked open data publication and maintenance mostly dried up, except for a few domains, such as medicine and cultural heritage. Most of all, the Semantic Web never materialized, at least not in the precise sense described in 2001. A series of articles started to trickle down, stating the death of the Semantic Web project. We collect some of them in Table 1.1, with a concise summary of the reasons they claim caused the project failure.

Currently, efforts to produce the Semantic Web have long since evaporated: socially, the last PhD students supervised by James Hendler (co-author of the 2001 Semantic Web article) do not make a single mention of Semantic Web technologies, in place of the Semantic Web there are the API economy and the internet oligopolies; technically, Semantic Web technology adoption was always low and the rise of machine learning and neural networks (arguably) falsified the 2001-notion that a machine could not derive meaning from unstructured data.

[6] Such summary can be found here: `https://www.w3.org/DesignIssues/LinkedData.html`.
[7] `https://www.dbpedia.org/`.
[8] `https://www.wikidata.org/wiki/Wikidata:Main_Page`.
[9] `https://schema.org/`.
[10] `https://lod-cloud.net/`.

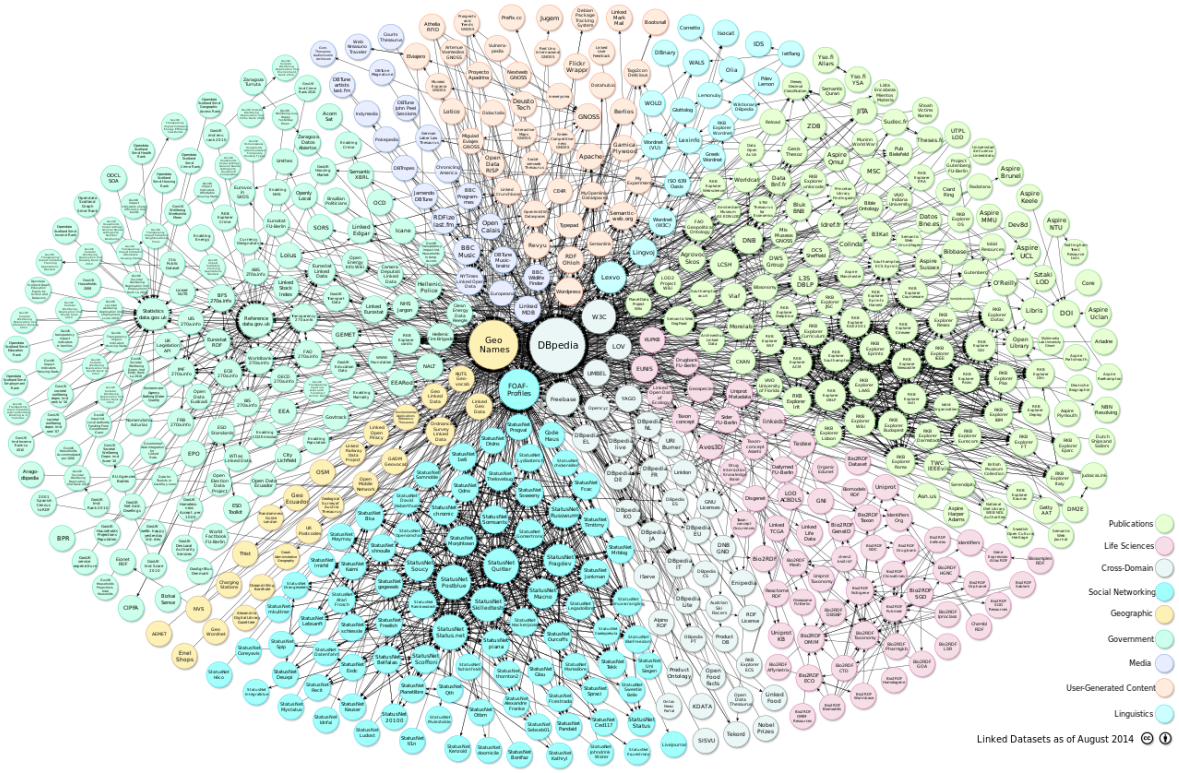



Figure 1.1: Some of the datasets published as linked open data in August 2014, and the relationships among them. Produced by the Linked Open Data Cloud project (`https://lod-cloud.net/`).

| Source | Date | Causes |
|---|---|---|
| [66] | 2001 | People would maliciously manipulate the markup. There is no unique way to classify things, and any such way carries economic interests, so organizations will not agree upon a standard. |
| [236] | 2005 | The Semantic Web would need to solve semantic heterogeneity, which has never been solved before by analogous AI efforts because it is hard. It may have limited success in producing cross-enterprise information sharing when the enterprises have a favorable business perspective, because e.g. they are especially advantaged by collaboration, or deal with semantically simple data, or if they are sufficiently big to control the market. |
| [215] | 2014 | No tool ecosystem was developed. Ontology interoperability is a hard issue. Modeling carries high effort. |
| [6] | 2014 | Putting semantic markup to a webpage carries high effort but no reward. |
| [113] | 2016 | OWL usage was never widespread, it focuses on various ways OWL could be improved. |
| [206] | 2017 | An open and decentralized location parallel to the (classical) web where to store the semantic information was not developed, and it is difficult to do so. |
| [184] | 2017 | SEO is only reason to markup (thus limited vocabulary and no link to related terms). Ontology efforts were centralized (e.g. Schema.org). Data privately held: KG not public, lack of trust. |
| [74] | 2018 | People would maliciously manipulate the markup. Metadata interoperability is a hard issue. The Semantic Web technology stack was too complex and too little expressive at the same time, and the automated inferencing was too limited. |
| [117] | 2018 | The Semantic Web technology stack was too complex. Almost everybody finds it easier to delegate its data to a centralized, privately-owned repository. |
| [182] | 2022 | The Semantic Web technology stack was too complex, and some of its underlying characteristics (non-unique name assumption, open world assumption) are unfortunate. |
| [202] | 2024 | "We can finally see the full realization of the original vision " due to successes in establishing massive knowledge graphs and the recent rise of generative AI, which should enable autonomous agents with open-ended conversational interfaces. |

Table 1.1: Summaries of the causes of the failure of the Semantic Web in the opinion of various sources, with the exception of a recent quote from Lassila that has a different tone (see why at the start of Section 1.2).

However, Bernards-Lee is still working on similar ideas with the Solid (Social Linked Data) project[11]. Moreover, it must be noted that the recent rise of generative AI, caused a renewed interest in the Semantic Web. Indeed, Lassila claims that "[w]e can finally see the full realization of the original vision" [202]

While the Semantic Web never came close to its original goal (at least for now), that is, the production of a gargantuan ontologized knowledge graph spanning the whole content of the web travelled by autonomous agents, the vision did have various impacts: linked open data are currently of interest to restricted domains, such as public administration, biology, and cultural heritage, and, more recently, manufacturing[12]; and some technologies of the Semantic Web Stack have had a considerable impact (e.g. RDF-based knowledge graphs). Whatever the reasons for the Semantic Web project's failure, they are likely less prevalent in restricted, controlled domains, as similar projects in these environments have been more successful.

**OWL, a little bit more in detail:** After the digression on Semantic Web of the previous section, we can return to OWL, and describe it somewhat more in detail.

OWL comes in many dialects ('profiles'), but all of these are fragments of mathematical first-order logic. Specifically, OWL allows for expressions containing at most two variables and limits how variables are quantified. A full treatise on OWL is beyond the scope of this introduction and can be found e.g. at `https://www.w3.org/TR/owl2-syntax/`. Here we just make some examples to clarify some key points, for instance, the following formula:

**ex1** $\forall x\ \texttt{CarPart}(x) \iff \exists y(\texttt{Car}(y) \wedge \texttt{partOf}(x, y))$

"$x$ is a car-part if and only if there is a car of which $x$ is part of."

is an OWL formula, since it makes use of two variables only and the inner existential quantification introduces a variable $y$ which is related to $x$ through a binary predicate. On the other hand,

**ex2** $\forall x, y\ (\texttt{uncleOf}(x, y) \iff \texttt{Man}(x) \wedge \exists z(\texttt{siblingOf}(x, z) \wedge \texttt{parentOf}(z, y)))$

"$x$ is the uncle of $y$ if and only $x$ is a man and is the sibling of a parent of $y$."

is a first-order logic sentence, but *not* an OWL sentence. In fact, the concept of 'being-the-uncle-of-somebody' cannot be expressed in OWL. The topic is nuanced, for instance, one can say things such as 'if $x$ is the brother of a parent of $y$, then $x$ is $y$'s uncle'. Additionally, while the following is not an OWL formula:

**ex3** $\forall x, y, z(\texttt{ancestorOf}(x, y) \wedge \texttt{ancestorOf}(y, z) \implies \texttt{ancestorOf}(x, z))$

"`ancestorOf` is a transitive property."

OWL allows nonetheless to state that a binary-predicate is transitive, using ad-hoc syntax:

**ex4** $\texttt{Transitive}(\texttt{ancestorOf})$.

[11] `https://github.com/solid/process`.

[12] See e.g. the Horizon 2020 project OntoCommons, `https://ontocommons.eu/`.

More in general, OWL (profiles) are variations of certain description logics (DLs, they are a family of fragments of FOL known for having 'good' computational properties). For instance OWL2 is based on the description logic known as $\mathcal{SROIQ}^{(\mathcal{D})}$, while OWL-DL is based on $\mathcal{SHOIN}^{(\mathcal{D})}$ [118]. The names of these description logics hints to their expressiveness:

- $\mathcal{S}$ means that appropriate restrictions of negation, intersection, and universal and existential quantification are allowed, as well as transitive binary predicates.
- $\mathcal{R}$ means that binary predicates can be said (with some restrictions) to be one the specialization of/disjoint with another (e.g. `brotherOf` is a specialisation of `siblingOf` and is disjoint with `sisterOf`), as well as reflexive or irreflexive.
- $\mathcal{O}$ means that unary predicates can be defined extensionally, by enumeration: e.g. a `Continent` is exactly one of {`Africa`, `Asia`, `Australia`, `America`, `Europe`}
- $\mathcal{I}$ means that binary predicates can be declared one the inverse of another (e.g. `hasPart` is inverse of `isPartOf`).
- $\mathcal{Q}$ stands for qualified cardinality restrictions (e.g. to say that a car has exactly 4 wheels as parts).
- $(\mathcal{D})$ stands for the use of datatype properties, data values or types (e.g. to state that the second argument of the `hasName` predicate are only string-literals).
- $\mathcal{H}$ means that binary predicates can be said (with some restrictions) to be one the specialization of another (a restriction of the expressivity entailed by $\mathcal{R}$).
- $\mathcal{N}$ stands for (unqualified) cardinality restrictions, to allow saying, e.g., that a car has at least four parts, but not to specify that 4 of these parts are wheels (a restriction of the expressivity entailed by $\mathcal{Q}$).

Additionally, the languages of first-order logic, description logics, and OWL are slightly different, so that, for example, FOL-'binary predicates' are 'roles' in DLs and 'object properties' in OWL. Similarly, FOL-'unary predicates' are called classes in both DLs and OWL. The syntax style is also different: for instance

**ex5** $\texttt{CarPart} \equiv \exists \texttt{partOf}.\texttt{Car}$

is a 'DL-style' rewriting of Sentence (ex1), while the corresponding OWL functional style syntax would be:

**ex6** $\mathit{EquivalentClasses}($ `ex:CarPart`
$\mathit{ObjectSomeValuesFrom}($`ex:partOf ex:Car`$))$

where the "`ex:`" before the classes and the object property is a prefix name: since ontology elements must identified by Internationalized Resource Identifiers (IRIs), which may be quite lengthy, a shortcut (prefix name) for an IRI (prefix IRI) is usually defined and prepended to the remaining part of the IRI (the local name, e.g. $CarPart$), forming an abbreviated IRI. The use of IRIs to identify ontology entities has the (optional)

advantage of making each entity point to a resource, in a Web-compatible way, which could e.g. consist of definitions or illustrations of the entity itself or of related entities.

As a general rule, there is a trade-off between how much a formal language can express and how difficult are reasoning tasks using that language and OWL is no exception. In addition, the optimal degree of this trade-off depends on the use case and the domain of application. This is the reason why OWL comes in dialects that vary in expressivity and complexity. For example, the consistency checking task (checking if a theory is consistent) in FOL is undecidable[13], while in OWL2-DL it is decidable[14], albeit with a considerable degree of complexity[15]. In some other less expressive profiles consistency checking is instead decidable with a lesser degree of complexity[16], see Figure 1.2 or consult Table 10 at `https://www.w3.org/TR/owl2-profiles/` for details.

Another essential characteristic of OWL is its choice of the open world assumption and refusal of the unique name assumption. The open world assumption is the assumption that facts that are not explicitly negated may be true (the opposite assumption is called 'closed world assumption'). The unique name assumption is the assumption that different identifiers (names) refer to different entities. The reason why OWL adopts the first and rejects the second is to be found in its origin in the Semantic Web project. Indeed, on the web the closed world assumption would make little sense: an ontology on the web would contain only a small part of the whole set of true facts present in the web at any given moment. Additionally, such a set of true facts would change in time. Therefore such an ontology could not assume the falsity of facts not contained in itself: to do so would contradict most facts present in other ontologies of the Semantic Web at any given time and cause problems with the addition of new facts. Similarly, since there wasn't (and isn't) an authority on the web that assigns unique names to things, the same entity will tend to have different identifiers in different ontologies, thus motivating a refusal of the unique name assumption.

**(RDF-based) Knowledge Graphs:** OWL uses IRIs: this is something it inherits from the Resource Description Framework (RDF) language, another fundamental component of the Semantic Web Stack, which OWL expands. The basic idea behind the RDF language is to represent a single statement about some resources as a subject-predicate-object triple, RDF documents themselves as then constituted by sets of triples. A triple is naturally interpretable as a labeled edge (the predicate) between two nodes (the subject and the object), so that RDF documents are naturally interpretable as graphs, and typically are called as such. The syntax of RDF is quite simple: RDF graphs correspond to labeled directed multigraphs where the labels are either (certain) IRIs, (typed) literal values, or 'anonymous' variables. With the restriction that literal nodes can only be the targets (objects) of the edge-arrows (called properties), and anonymous variables can only label nodes, which are then called blank

[13]That is, there is no algorithm that, given any FOL theory, will always determine in finite time if the theory is consistent or not.

[14]That is, the opposite of undecidable.

[15]2NExpTime complete.

[16]Polynomial or less.

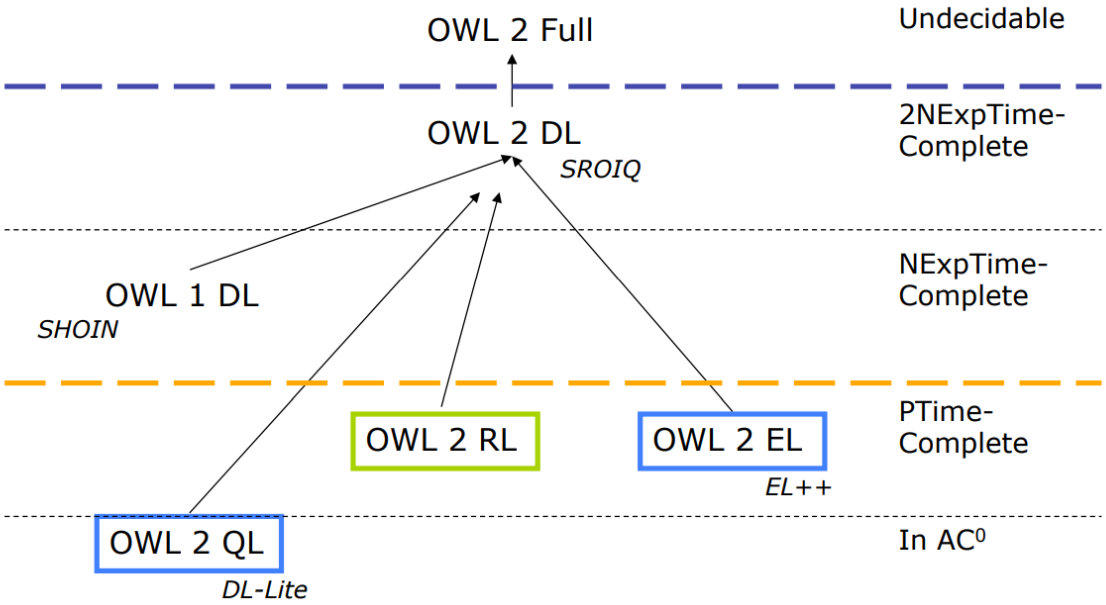



Figure 1.2: Complexity of some OWL profiles. In particular, the data complexity for conjunctive query answering or consistency checking is the same for OWL QL, RL, EL, and full is the same and is as indicated in the picture; while for OWL 1 DL and OWL 2 DL the complexity class is that of the taxonomical complexity or of the combined complexity (these two complexity classe are the same for these profiles, see Table 10 at `https://www.w3.org/TR/owl2-profiles`). The picture is taken from [2]

nodes. On the other hand, RDF puts a heavy emphasis on the 'semantic', that is, on the attribution of meaning to RDF graphs. Briefly, while the full meaning of an RDF graph may depend on many factors that cannot be understood by machines (such as human habits or common sense notions), it was decided to make use of model-theory-based formal semantics to encode a part of meaning that can be easily understood by machines. That is, the RDF standard defines entailment rules and interpretations[17]. Entailment rules show how to deduce one graph from another, while interpretations are analogous to Tarski's interpretations of FOL-theories within set-theory. So that it is formally possible to assert statements such as 'this RDF graph models this situation (as a collection of real things and their relations)' and 'if this RDF graph models true situations then this other RDF graph also models true situations'. Notice also that RDF has a vocabulary, a list of special URIs, which must be interpreted in any model of an RDF graph, together with some triples built with these URIs. The full list of these URIs is quite simple:

```
rdf:type
rdf:Property
rdf:langString
rdf:subject rdf:predicate rdf:object
rdf:first rdf:rest rdf:value rdf:nil
rdf:List rdf:_1 rdf:_2 ...
```

These URIs clearly have precise intended meaning (e.g. the last two rows are intended to be a vocabulary used for lists, while the row just above for triples-reification), but their meaning is not constrained in the RDF interpretations, if not minimally (some constraints related to datatypes, plus the request that `<ex:p>` `rdf:type` `rdf:Property` must hold for any property `<ex:p>` appearing in the graph as well as every URI of the RDF vocabulary except for `rdf:nil`, which must be interpreted as a `rdf:type` of `rdf:Property`).

RDF Schema (RDFS) is an extension of RDF, which expands the RDF vocabulary (some of the main additions are `rdfs:Class` and `rdfs:subClassOf`, `rdfs:subPropertyOf` to build taxonomies of classes, and properties, respectively; `rdfs:domain` and `rdfs:range` to state domain and ranges of properties; and the properties `rdfs:label` and `rdfs:comment` to annotate nodes with literals). The entailment regime also is extended with many rules, for instance, `ex:A rdfs:subClassOf ex:B` and `ex:a rdf:type ex:A` together entail `ex:a rdf:type ex:B`. A full list of the entailment rules can be fond at `https://www.w3.org/TR/rdf11-mt/#rdf-interpretations`.

In the previous section, we said that OWL is 'based' on the $\mathcal{SROIQ}(\mathcal{D})$ DL. This means that the entailment rules of OWL and its formal semantic are defined almost in the same way as those of $\mathcal{SROIQ}(\mathcal{D})$ (OWL extends that DL a little bit). This way of specifying the semantic of OWL is called the 'the direct semantic' of OWL[18]. However, OWL can be built also an extension of RDFS and, in that case, it is natural to describe

[17] `https://www.w3.org/TR/rdf-mt/`.

[18] `https://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/#Direct_Model-theoretic_Semantics_for_OWL_2`.

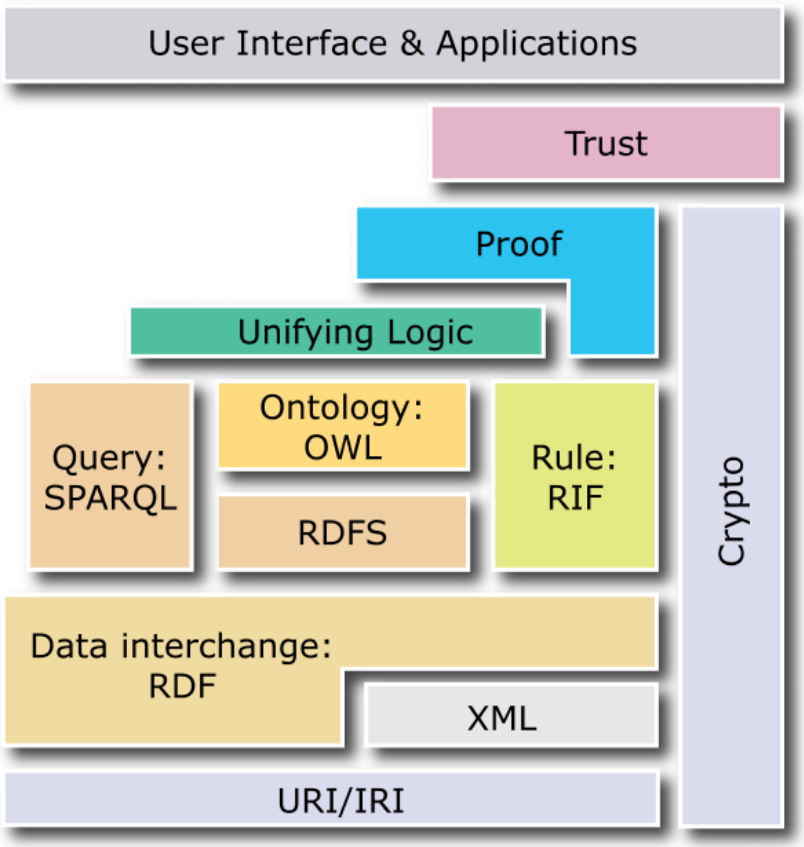



Figure 1.3: Pictorial representation of the Semantic Web Stack. The Technologies on the bottom enable the technologies on top of them, for instance, OWL is serialized on top of the RDF language, and SPARQL is a query language for RDF graphs.

its semantic as an expansion of RDFS'. This is called 'the RDF-based semantic' of OWL, and is known to be (almost) equivalent to the direct semantic[19]. For instance, the OWL sentence (ex6) can be rewritten into an RDFS graph:

```
ex:CarPart owl:equivalentClass _:AnonymousClass .
_:AnonymousClass rdf:type owl:Restriction .
_:AnonymousClass owl:onProperty ex:partOf .
_:AnonymousClass owl:someValuesFrom ex:Car .
```

Note that in this translation it was necessary to use blank nodes (the 'Anonymous Class').

This emphasis of RDF, RDFS, and OWL on semantics, together with the fact that sets of triples can be represented as graphs (see e.g. Figure 1.4 for a graph-rendition of the triples corresponding to Formula (ex6)), makes quite reasonable to call such graphs as 'semantic graphs'. However, now the alternative 'knowledge graph' sees much more use currently[20]. The term came to popularity after its use in 2012 by Google to indicate

[19] `https://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/`.

[20] As revealed by Google Trends, last checked in 2024.

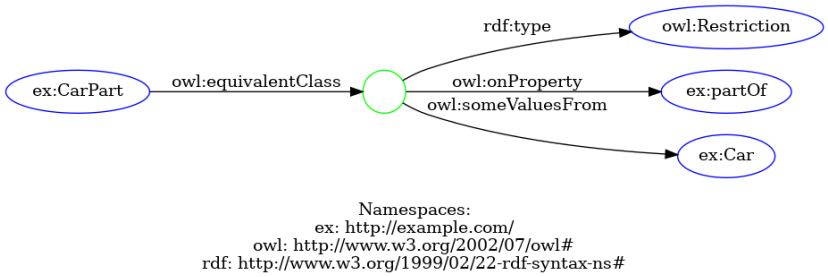



Figure 1.4: Graph that represents the RDF triples encoding the OWL Formula (ex6)

their solution to semantically enrich Google's search engine [3]. In this introduction, we are not interested into an analysis of definitions of the term knowledge graph. Instead, we simply quote Ehrlinger [68]: "the term knowledge graph does not constitute a new technology. Rather, it is a buzzword reinvented by Google and adopted by other companies and academia to describe different knowledge representation applications". Furthermore, Ehrlinger remarks that the term 'knowledge base' is often used interchangeably with 'knowledge graph', and, if a difference is present is usually that a knowledge base does not include a reasoning/entailment system, while a knowledge graph does.

Of course, the idea of representing knowledge using graphs is not new: it is, in all likelyhood, a very old idea that predates the first known appearances in the literature. Just to mention a few of these appearances, in the 1960s a graph-based formalism for knowledge representation was developed [105], which was called 'semantic nets' or 'semantic networks'. Then, in the 1980s, a joint project of the Universities of Twente and Groningen started to develop an ontology composed of a small set of relations, which was used to guide graph-based conceptual modelling efforts [14]. The Twente-Groningen team was probably the first to use the term 'knowledge graph' in systematic fashion.

**Applied Ontology:** In the previous sections We introduced Semantic Web technologies, indeed, the Semantic Web was for many years the main use case of applied ontology. However it is important to note that applied ontology is a distinct discipline that has also different use cases, interacts with many different communities, and both predates and survives the Semantic Web. One could say that, in essence, the discipline of applied ontology supplies philosophically-based methodologies to knowledge engineering. Here we exemplify this by recalling a famous example: the OntoClean methodology [98]. OntoClean offers guidelines for constructing taxonomies: it requires manual input, as each class of the taxonomy must be annotated with appropriate meta-properties, but, apart from that, it lists a series of constraints that must hold among the meta-properties belonging to different classes that can be verified automatically. If the taxonomy violates the constraints it should then be 'cleaned' up. Take, for the sake of example, Formula (ex6). It could be part of a wider ontology, including a taxonomy. A portion of such an ontology will reasonably look like the following:

...

**ex7** $\forall x\ \texttt{Car}(x) \Rightarrow \texttt{Device}(x)$

"Cars are devices."

**ex8** $\forall x\ \texttt{CarPart}(x) \Rightarrow \texttt{Device}(x)$

"Car-parts are devices."

**ex9** $\forall x\ \texttt{CarPart}(x) \iff \exists y(\texttt{Car}(y) \wedge \texttt{partOf}(x, y))$

"$x$ is a car-part if and only if there is a car of which $x$ is part of."

**ex10** $\forall x\ \texttt{Engine}(x) \Rightarrow \texttt{CarPart}(x)$

"Engines are car-parts."

**ex11** $\forall x\ \texttt{Wheel}(x) \Rightarrow \texttt{CarPart}(x)$

"Wheels are car-parts."

...

One of OntoClean meta-properties is rigidity: a class is rigid if its instances are necessarily so. Now, the modality of being-necessarily-something must be assigned by the human user. In this case, considering the domain in question, it would be reasonable to say that `Car`, `Engine`, `Wheel` are rigid, while `CarPart` is not, because something that is part of car is not necessarily so (for instance, a combustion engine may not always be installed in a car, components could be replaced form one car to another, and certain components could be part of things that are not cars – the seat of a car and the seat of a truck are not much different, say). In this case, Axiom (ex11) conflicts with OntoClean constraint that rigid classes cannot be subsumed into non-rigid classes, suggesting that the taxonomy should be modified appropriately[21].

The previous example shows how applied ontology can help with modeling, and, of course, OntoClean is far from being the only methodology/set of principles that can help to achieve high-quality schema or data models. Such sets of principles may space from pure heuristics (e.g., compare [249]) to heavily-philosophically-founded (e.g. [263]). One classical principle is the use of foundational ontologies, which we will touch briefly in the following.

**Sharing meaning with ontologies, inter-ontology interoperability, and top-level ontologies:** In disciplines such as knowledge engineering and data governance, ontologies and semantic graphs are not the only way to represent knowledge, and organize and store data. They are (or should be) used when the goal is to *share meaning*, which is the main point of ontologies. In practice, this means that ontologies will most often appear when interoperability and data integration are most sought after.

Whatever the reason for needing an ontology, if a suitable ontology already exists, that one should be reused instead of developing a new one; in the same way that, if a library of a programming language that carries out efficiently a task exists already, one

[21] Indeed, classes similar to `CarPart` are often ill-advised.

should not spend the effort to develop a new one. However, due to a variety of reasons, not least the fact that an ontology may have been not shared publicly or it may not precisely answer each and every need of the ontology developer, ontologies for the same domain or application may be re-developed multiple times.

In addition, since application domains often overlap (consider e.g. the couples mechatronics and electric engineering, manufacturing and resource planning, maintenance and failure analysis), corresponding ontologies will likely overlap. This was especially true for the Semantic Web project, and its successive incarnations, due to the envisioned size of the project (consider, for instance, this quote about Linked Open Data "[w]e work towards a future in which people share the power to collect and organize the data that shapes humanity's understanding of the world"[22], which appears to envision a a patchwork of ontologies covering the whole of human knowledge).

Since domains also have different levels of generality (e.g. manufacturing versus addictive manufacturing versus defects in addictive manufacturing), ontologies can even be completely 'embedded' one into the other. It is natural, therefore, to consider the issue of how to make the ontologies themselves interoperate, both 'horizontally' (i.e. between ontologies of comparable levels of generality) and 'vertically' (i.e. between ontologies of different levels of generality).

Therefore one must add links between the ontologies, explaining how they are related. On a practical, operative level, this typically translates to enumerating a series of inclusion or equivalence axioms, to say which class or property in one ontology is more specific or equivalent to which class or property of the other ontology.

Using an ontology of a more general scope that is already linked to both ontologies would solve or at least greatly simplify the problem, since the desired links can then be deduced completely or partially. Doing so, however, would seem more complicated, since now one must relate the more specific ontologies to the more general one, thus needing two alignments in place of a single one. Nevertheless, there are two advantages:

- first, using higher generality ontology is a natural way to start and guide the alignment process, simplifying it.
- If there are not just two ontologies, but, say, $n$ ($n >> 2$) ontologies that have to be aligned, then aligning each ontology to all others requires $O(n^2)$ alignments. Instead, repeatedly using higher-generality ontologies to bridge couples of ontologies naturally produces a tree structure among ontologies, which needs just $O(n)$ alignments (the remaining alignments should be then either automatically deducible, or at least easy to derive).

It is natural, then, to suggest high-generality ontologies, usually called upper-level ontologies, as guides for the ontology development process. In fact, their use for that activity has the additional advantage that basic patterns or even complex axiomatics encoding precise meaning can be reused without having to redevelop these.

Of course, there is no unique way to model any given situation, and this is also true for high-generality entities and relations, so that multiple upper-level ontologies were

[22] `https://meta.wikimedia.org/wiki/LinkedOpenData/Strategy2021/Joint_Vision`.

developed over time, such as BFO [16], GFO [115], YAMATO [195], UFO [101], and DOLCE [176]. In this thesis, we focus mostly on DOLCE, so that we will summarily introduce it in the next section. However, before doing so, let us take a step back to OntoClean and to the formal languages used for modelling, and see an example of how the use of an upper ontology can fit in and enhance these. OntoUML [102] is an extension (a 'profile') of UML that introduces terminology and constraints from the upper ontology UFO. UFO itself uses and extends[23] the meta-properties of OntoClean, and speaks of rigid properties (properties that are necessary for their instances, as discussed briefly in the paragraph about OntoClean above), sortals (properties that provide a criterion, called identity criterion, for determining if two instances of the properties are the same. E.g. 'person' would be a sortal class because there is a way, in principle, to say that two people are the same or different; while 'being red' would not be a sortal, because it does not have an identity criterion), dependent or independent entities (e.g. 'size' is a dependent class because it is always the size *of* something), etc. UFO uses these meta-properties to describe meta-classes: for instance, a 'kind' is an (existentially) independent rigid sortal [24] and similarly other meta-classes are defined, such as category, role, phase, mixin, .... UFO's axiomatization then restricts the meaning of such meta-classes: similarly to OntoClean, roles (which are antirigd) cannot subsume kinds (which are rigid), and so on. UFO's meta-classes and axioms are then translated into OntoUML as 'stereotypes' and constraints. Stereotypes are used to annotate the classes of a UML class diagram, the constraints then ensure that the UFO's axioms are respected, forcing the user to comply with UFO's ontological commitments and thereby improving the quality of models [250]. This showcases how one can reuse complex semantic constraints in modelling without having to redevelop them.

**DOLCE:** The Descriptive Ontology for Linguistic and Cognitive Engineering was first fully formalized in 2001 [177]. As reflected by its name, has a descriptionist approach, in the sense that it tries to capture the ontological categories used in the human commonsense and language: DOLCE categories are to be intended as cognitive artifacts depending on human perception and conventions. As a consequence, DOLCE does not make use of four-dimensional entities (a DOLCE-object is therefore wholly present at each instant it is present).

DOLCE is an ontology about particulars (entities that do not have instances) and not about universals (entities that do have instances). That is, it uses universals to categorize and talk about particulars, but doesn't talk about universals themselves (e.g. one can say 'John's car is an (instance of) object', but cannot say 'the object-universal is a category'). Furthermore, it adopts a multiplicative approach, since it cares for expressivity without trying to reduce the number of primitives, and accepts the existence of co-located entities (e.g. the clay a statue is made from and the statue are different entities).

[23]Note that UFO has a much broader terminological scope than OntoClean.

[24]The definition of kind is more specific than 'an independent rigid sortal', since that applies also to 'quantity' and 'collective', but these are the key characteristics and we simplify for the sake of brevity.

DOLCE taxonomy can be seen in Figure 1.5. It has four top categories: endurants, perdurants, qualities, and abstracts. The distinction between endurants and perdurants is at the core of the ontology: endurants (e.g., a car, a tree, a crack in a wall, a liter of water, etc.) are wholly present at any time they are present. On the other hand, Perdurants (e.g., a marathon, an action, a storm, the fall of a leaf), on the other hand, are made by the sum of parts that exist at different times: when they are present, they are only partially present.

Endurants can be physical or not, depending on if they have a spatial location. Additionally, if they are physical, they can be: (i) features ($\texttt{F}(\cdot)$), if they depend on some other entity for their existence (e.g. a crack in a wall depends on the wall existence); (ii) physical objects ($\texttt{POB}(\cdot)$), if there is a relation that holds among their parts, which can be temporary, and make them a coherent whole (e.g. a tree); (iii) otherwise they are amounts of matter ($\texttt{M}(\cdot)$), e.g. a given set of sand grains. Physical objects can be agentive ($\texttt{APO}(\cdot)$) or non-agentive ($\texttt{NAPO}(\cdot)$). Non-physical endurants that depend on a community of agents are called social objects. A subclass of social objects we are especially interested in are roles. Roles were introduced as an extension to DOLCE [179] and have now become integral to the expanded taxonomy [31]. Roles are characterized as antirigid and dependent (also known as founded) classes [98, 97, 179]. An antirigid class implies that its instances are not inherently so, while a dependent class stipulates that all its instances existentially rely on an external entity, often referred to as the context. For instance, a particular individual may assume the role of a student, but no individual is inherently a student. In fact, it is expected that an individual ceases to be a student after a certain period. In contrast, every individual inherently belongs to the class of persons, irrespective of external entities. An ontological class is considered dependent or founded on another if the presence of an instance of the first class necessitates the presence of a corresponding instance of the second. For example, the citizenship of an individual depends on the country, as every citizen is associated with a specific country. Such a dependence is usually identified with some flavour of DOLCE's specific dependence relation ($\texttt{SD}(\cdot,\cdot)$):

**df1** $\texttt{SD}(x,y) \iff (\exists t(\texttt{PRE}(x,t)) \land \forall t(\texttt{PRE}(x,t) \implies \texttt{PRE}(y,t)))$ "$x$ is specifically dependent on $y$ if and only if every time $x$ is present so is $y$ (and $x$ is present at least at some time to exclude an empty quantification)."

While this chapter will delve into different forms of dependence and founding, the basic concept is encapsulated in the citizen-country example.

Furthermore, some authors categorize roles based on the nature of their context. For instance, [168] distinguishes relational, processual, and social roles, classifying them depending on whether the context is a relation, a process, or a social object. In DOLCE, roles ($\texttt{RL}(\cdot)$) are reified and treated as concepts ($\texttt{CN}(\cdot)$), which, in turn, fall under the class of non-agentive social objects ($\texttt{NASO}(\cdot)$), as it can be seen in Figure 1.5.

To differentiate perdurants ($\texttt{PD}(\cdot)$), DOLCE employs three mereological properties: cumulativity, homeomericity, and atomicity. Cumulativity holds for a perdurant-type when the sum of two instances of a type yields the same type, indicating closure of that type under mereological sum. Consider the example of 'walking': if we contemplate

two instances of walking activities, the composite activity remains of the 'walking' type. Perdurants of a cumulative type are termed *stative*, whereas those that lack cumulativity are labeled *eventive*. Homeomericity applies to a perdurant type if any parts of its instances are instances themselves. For example, in the case of 'sitting', portions of a sitting action remain classified as sitting actions. Homeomeric stative perdurants are termed *states* ($\mathtt{S}(\cdot)$), while those that are stative but have parts of different types are designated as *processes* ($\mathtt{PRO}(\cdot)$). An illustration is walking, where completing a specific leg movement is a prerequisite for it to be considered a walking movement. Another instance is the buzzing of a clapper (or buzzer): the clapper alternates between two states during clapping (opened/closed circuit), and neither state is inherently of the buzzing type. DOLCE refers to the temporal relationship between a perdurant and the object involved in it as *participation*, denoted as $\mathtt{PC}(\cdot,\cdot,\cdot)$: in the aforementioned example, the clapper is a participant in the buzzing.

DOLCE defines various relations holding between perdurants, that we will also use in the oncoming chapters. They are parthood ($\mathtt{P}(\cdot,\cdot)$), specialized in either temporal parthood ($\mathtt{P}(\cdot,\cdot)_T$) or spatial parthood ($\mathtt{P}(\cdot,\cdot)_S$); temporal inclusion ($\subseteq_T$); and spatial inclusion ($\subseteq_S$). Reciprocal temporal or spatial inclusion is called coincidence ($\approx_T$, $\approx_S$). Except for parthood, these are all defined relations. Definition of temporal and spatial part and temporal inclusion is straightforward, while the spatial inclusion makes use of the spatial location of perdurants, which is stipulated to be the same spatial location as that of their maximal total participant. Introduction of the maximal total participant is slightly effortful, and we don't report it here, in any case the spatial location of perdurants is inherited by the location of their participants. We report the remaining definitions here for sake of clarity (they are copied from [177, p.29-30]):

**df2** $x \subseteq_T y \iff \exists t, t'(\mathtt{ql}(t,x)_T \wedge \mathtt{ql}(t',y)_T \wedge \mathtt{P}(t,t'))$

"$x$ is temporally included in $y$ if the time extension of $x$ (indicated by the special quale relation $\mathtt{ql}(\cdot,\cdot)_T$) is part of that of $y$."

**df3** $x \approx_T y \iff x \subseteq_T y \wedge y \subseteq_T x$

**df4** $x \subseteq_S < y,t > \iff \exists s, s'(\mathtt{ql}(s,x,t)_S \wedge \mathtt{ql}(s',y,t)_S \wedge \mathtt{P}(s,s'))$

"$x$ is spatially included in $y$ at time $t$ if the time spatial extensions of $x$ at time $t$ (indicated by the special quale relation $\mathtt{qt}(\cdot,\cdot)[\cdot]_S$) is part of that of $y$ at the same time."

**df5** $x \subseteq_{ST} y \iff \exists t \mathtt{PRE}(x,t) \wedge \forall t(\mathtt{PRE}(x,t) \implies x \subseteq_S < y,t >)$

"$x$ is spatio-temporally included in $y$ if at all times $x$ is present it is spatially included in $y$ at that time (and there is at least one such time)."

**df6** $x \approx_{ST} y \iff\iff x \subseteq_{ST} y \wedge y \subseteq_{ST} x$

**df7** $\mathtt{P}_S(x,y) \iff \mathtt{PD}(x) \wedge \mathtt{P}(x,y) \wedge x \approx_T y$

"Perdurant $x$ is a spatial part of $y$ if it is a part of $y$ and it temporally coincides with $y$."

**df8** $\mathtt{P}_T(x,y) \iff \mathtt{PD}(x) \wedge \mathtt{P}(x,y) \wedge \forall z((\mathtt{P}(z,y) \wedge z \subseteq_T x) \implies \mathtt{P}(z,x))$

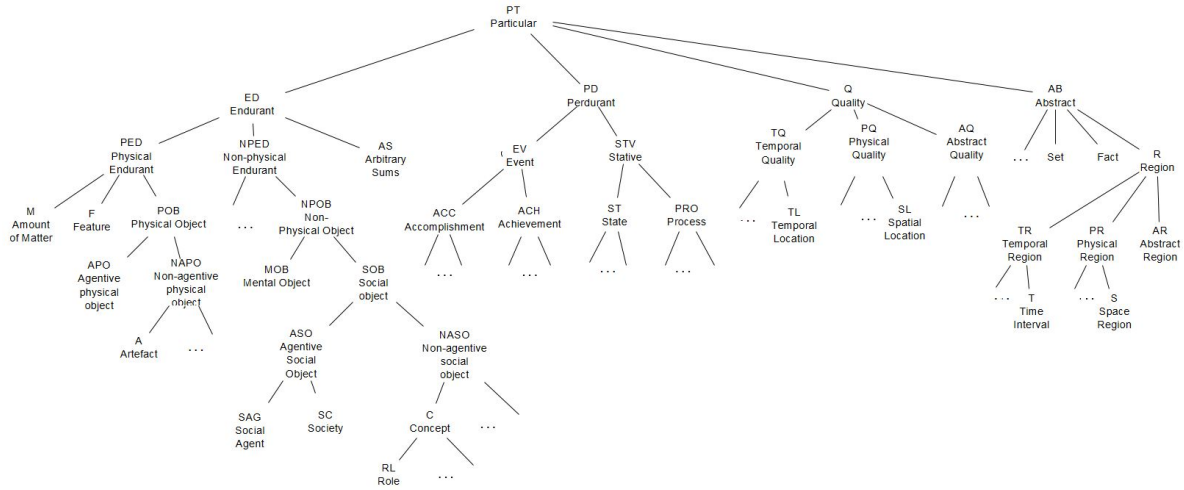



Figure 1.5: DOLCE taxonomy, taken from [31].

> "Perdurant $x$ is a temporal part of $y$, which is maximal with respect to all other parts of $y$ that are temporally included in $x$."

DOLCE-qualities were influenced by trope theory [44] but exhibit some differences from those. DOLCE-qualities, such as the weight of a car or the color of a flower, represent specific attributes inherent to individual objects. Consequently, the weight property of a given car is unique to that particular car, which is referred to as the 'bearer' of the quality. Each quality functions as an individual entity and is affiliated with its bearer. For instance, if we consider a group of five cars, the colors of these cars manifest as five distinct color qualities, each associated with a specific car. This occurs despite the potential visual indistinguishability of the five cars regarding their color. In contrast to tropes, qualities are linked to values that can undergo changes over time. For instance, the color quality of a flower may transition from red in summer to brown in fall. The values of qualities, termed 'quales', are separate entities from the qualities themselves. This enables a flexible assignment of values to qualities, following the schema of object-quality-quale (or bearer-individual quality-value). Quales are always part of an appropriate space, which in DOLCE is an instance of the region category, which is the most important sub-category of abstracts. Spaces of quales are called quality spaces and are based on the theory of conceptual spaces [77]; also each quality type has a corresponding quality space [177] and individual qualities are unique given a bearer and a quality type. In [177] qualities have to have the same time extension of their bearer, however, in a more recent axiomatization ([175]) this has been relaxed and qualities may have shorter lives than their bearer, so it is possible that a bearer gains or loses a quality. It is also possible that a bearer gains or loses the same quality multiple times, as there is no commitment on the convexity of the time region constituting the life of the quality. Since this is desiderable property for the framwork of the thesis we adopt the axiomatization of [175] in this regard. In DOLCE, qualities constitute the class $\mathtt{Q}$ (with the predicate $\mathtt{Q}(\cdot)$), and the (direct) inherence relation is denoted as 'direct-

| Class | Name |
|---|---|
| ED | Endurant |
| PED | Physical Endurant |
| NPD | Non-physical Endurant |
| AoM | Amount of Matter |
| POB | Physical Object |
| APO | Agentive Physical Object |
| NAPO | Non-Agentive Physical Object |
| NASO | Non-Agentive Social Object |
| CN | Concept |
| RL | Role |
| A | Artefact |
| TQ | Temporal Quality |
| PQ | Physical Quality |
| PD | Perdurant |
| E | Event |
| S | State |
| ACH | Achievement |
| ACC | Accomplishment |
| SV | Description |
| PRO | Process |
| Q | Quality |
| R | Region |
| T | Time interval |

Table 1.2: DOLCE's classes that are most relevant for this thesis.

quality-of' (written as $\mathtt{dqt}(\cdot,\cdot)$). There is also an indirect quality-of relation ($\mathtt{qt}(\cdot,\cdot)$), the difference is that the indirect version is transitive, while the direct version does not present intermediate dqt relations whenever it holds; moreover the indirect version is used to represent quality dimensions: for example, the color of a rose directly inheres in the rose and the brightness of that color directly inheres in the rose's color itself, while indirectly inhering in the rose. A temporal quale relation associates a quality with a specific value, which can vary over time and is expressed as $\mathtt{ql}(\cdot,\cdot,\cdot)$.

Since we are going to use DOLCE as an upper ontology to guide the ontology development throughout this thesis, in Table 1.2 and Table 1.3 the main classes and relations of DOLCE that will appear in logical formulas are listed.

**On the choice of foundational ontology and implementation language** As it is implicitly suggested by the subjects of the above sections, this thesis will proceed on using the language of first order logic, and using the ontology DOLCE as a guide on foundational ontological issues. The ensuing ontologies will also be translated into OWL, which is (more) implementation-oriented than FOL.

One could object to this choice: why not YAMATO, or perhaps GFO, or BFO,

| Relation | Name | Holds between |
|---|---|---|
| `qt` | Indirect Quality | A quality, endurant, or perdurant, and any of its qualities |
| `dqt` | Direct Quality | A quality, endurant, or perdurant, and any of its qualities |
| `ql` | Quale | A quality and the corresponding value (at a given time) |
| `PRE` | Presence | An entity and a time it exists at |
| `P` | (Temporary) Parthood | An endurant, perdurant, or abstract entity, and another endurant, perdurant, or abstract entity that contains the former (at a given time, in the case of endurants) |
| `CP` | Constant parthood | Same as temporary parthood |
| $\texttt{P}_T$ | Temporal parthood | Two perdurants |
| $\texttt{P}_S$ | Spatial parthood | Two perdurants |
| `K` | Constitution | An endurant or perdurant, and another endurant or perdurant that is constituted by the former (at a given time) |
| `CK` | Constant constitution | Same as constitution; defined in the same way as for constant parthood |
| `PC` | Participation | An endurant participating in a perdurant (at a given time) |
| $\subseteq_S$ | Spatial inclusion (at a time) | Two physical endurants, two perdurants, or two physical qualities |
| $\subseteq_T$ | Temporal inclusion | Two endurants, two perdurants, or two qualities |
| $\subseteq_{ST}$ | Spatio-temporal inclusion | Two endurants, two perdurants, or two qualities |

Table 1.3: DOLCE's predicates that are most relevant for this thesis.

or UFO as a choice of foundational ontology. And why OWL and not some language from the logic programming community (some recent version of Datalog, say) implemented in some deductive database, or perhaps something more near to object-oriented programming with UML or SysUML, or something else (possibly also another language still from the Semantic Web community, such as SHACL)? On an even more applicative level, why focus on RDF and SPARQL and not, say, property graphs and Chypher[25].

Indeed, it is important to note that DOLCE is far from the only upper-level ontology, and users may prefer another ontology of choice and argue that DOLCE has alleged limitations. For instance, users of BFO or UFO will likely highlight that DOLCE does not contain a category for dispositions, and users of UFO may also point to the fact that the theory of relational qualities is not much developed, and so on. DOLCE users may in turn point to other alleged limitations of the other ontologies and so on. This could start a debate, that we do not address (except for Jansen and Röhl's argument [218, 133] against internally-grounded functions, which is a classical in the literature): we simply use DOLCE as a guide in foundational matters because it is some good characteristic to do so. In particular:

- DOLCE is a prominent upper ontology, which underwent extensive testing across various applications and is an ISO standard (part of the ISO 21838).

- DOLCE demonstrated remarkable stability over two decades, underscoring its reliability, in contrast to the frequent reworkings observed in other upper ontologies, which were often extensively reworked along the lines of DOLCE.

- We follow Jansen and Röhl's opinion [218, 133] on the relation of functions with malfunctions, in particular, we agree that (some) functions are externally grounded. Thus, our reluctance to adopt ontologies like BFO, which espouse opposite ontological views.

- The last, and most important point is that DOLCE "aims at capturing the ontological categories underlying natural language and human commonsense" [177], in this sense it is a descriptive ontology. In addition to making theories formulated using DOLCE more accessible and comprehensible, this is critically important because of our choice of a descriptive approach towards engineering functions.

Of course, the choice of DOLCE is still (up to a point) arbitrary, from a scientific point of view, and putative users of this thesis' framework may want to make use of other foundational ontologies. In order to minimize potential issues derived from this, this thesis follows the broad design principle of taking minimal ontological commitment on foundational ontological issues: this ensures that the thesis framework is as lean as possible and that mapping it to other ontologies is as easiest as possible. Consider e.g. the case of the ontology MALFO developed in Chapter 4. The intended mapping with DOLCE is just five relatively simple axioms: (mp3), (mp4), (mp5), (mp6), (df50). It is

[25]The most known query language for property graphs, see `https://neo4j.com/docs/getting-started/cypher/`.

easy then for a user to swap these mappings in favour of mappings with another ontology. This design choice also facilitates modularization (as exemplified by the MALFO ontology-module), which helps with issue `I2` from the abstract.

On the other hand, this means that many times we will not commit to some point of view and instead we will just say that multiple points of view are possible, without supplying further indications. This could be considered a downside of the underlying design choice.

Coming to a more applicative level, the choice of OWL is similarly arbitrary (up to a point): such a language does have some good properties, listed in the sections above, but, as with all other modeling languages, it is not perfect. Manifest, well-known alleged issues, which users of other languages may point to, are the open world assumption and the non-unique name assumption. These fundamental choices entail that reasoning in OWL is infamous to be counter-intuitive and too 'weak' to constrain information. In fact, especially in the database community and enterprise environments, the opposite choices are often made: entities have unique identifiers, and information not stated is assumed to be false. More in general, it is critical to remark that Semantic Web technologies are not the only way to do conceptual modeling: OWL is not the only language that can be used to formalize ontologies (some other languages we have mentioned above); nor RDF, a language for graph-based knowledge representation that will be described in the following, is the only way to serialize knowledge in the shape of a graph, for instance property graphs, also called attributed graphs[26] [5] constitute one competing serialization solution. This could be the start of another debate, which is also outside the scope of the thesis. Again, we proceed on with OWL even though we recognize that it has critical aspects, but we try to do so in a 'neutral' way: the use of OWL should also be understood as an exemplification of an application-oriented language, whose discussion introduces various characteristics that are critical to every implementative language used for knowledge representation. We also try to go (admittedly slightly) beyond the use of OWL: for instance, the ontology MALFO is also serialized in (a certain version) of Datalog, and in a FOL-based format used by tools from the automatic theorem provers community. The use of FOL as a basic modeling language also helps with trying to be neutral w.r.to more implementative languages: FOL has the virtue of being free from implementative details, and is considerably expressive and flexible; using FOL-theories as references, putative users can then translate them into more applicative languages.

## 1.2 The Use of Semantic Technologies in Enterprises: Why and How

This section aims to continue the historical narration of the previous section and to provide a reason for the adoption of semantic technologies in enterprise environments.

[26] They are called so because they store information, attributes, in the graph edges. Semantic Web style graph serializations do not.

Note, in particular, that this section is not a motivation behind the specific research activities carried out in this thesis, but it provides a context to undestand why and how semantic technology (including the artefacts developed in the thesis) can be fruitfully used.

In Section 1.1 we have touched upon characteristics and genesis of semantic technologies[27], mentioning that one of the main visions that pushed for semantic technologies, the Semantic Web, did not materialize. However, this should not lead the reader to think that semantic technologies have no use or application. In fact, the adoption of these technologies has never been bigger. For instance, (some of) the open knowledge graphs spawned by the Semantic Web efforts[28] have reached gargantuan dimensions and their use for, say, natural processing language or machine learning tasks is routine. Also, the adoption by companies of semantic technologies has recently been trending and growing [227]: For example, Ora Lassila recently declared the 'victory' of the Semantic Web project in the sense that enterprise-level of huge and efficient knowledge graph has already came into being, and exemplifying this with some projects realized by Amazon Neptune [202], and observed that the shifting of focus from the Web to the enterprise environment has been a good thing due to the reduction in scope. Additionaly he (among many others) expresses optimism for the future of the Semantic Web project due to the recent rise of generative AI: open-ended conversational interfaces for automatic agents are now possible.

In this section, we will touch on (some of the main reasons of) how and why semantic technologies currently find their space in enterprises.

**The problems of integrating and sharing the meaning of data** As mentioned in Section 1.1, semantic technologies are mostly employed when there is a need for data integration or sharing the meaning of data. Let us see how this need could arise. Speaking in very general terms, data and knowledge[29] have always been important for enterprises, and their importance has continuosly increased: data was called "the new oil" in 2006 [8], and, since then, its importance has only grown [264].

In the context of an enterprise, we can distinguish the roles of data sources or data producers, which generate the data, and the roles of data consumers, which process the data to pursue some business goal. An example of a data source could be a plane technician on an aircraft carrier who is required to fill out forms about his interventions (an example from [201]) or, say, a component in an assembly line owning a rich array of sensors (a typical example in I4.0 contexts). An example of a data consumer could be a data analyst, trying to find meaningful knowledge from the data, or a manager, trying to ground a business decision on knowledge based on data.

The process of using data for some goals leads, in practical settings, to some typical

[27]Semantic technologies is an old buzzword, still in use, which indicates a broad set of technologies and methodologies, including those described in the previous section.

[28]Such as, e.g., YAGO `https://yago-knowledge.org/`, KBpedia `https://www.kbpedia.org/`, DBpedia `https://dbpedia.org`, or Wikidata `https://www.wikidata.org/`.

[29]Data and knowledge are sometimes used interchangeably. Here, by knowledge we mean the information gained by giving meaning to the data, that can be used e.g. to make business decisions.

issues, especially in the interface between producers and consumers: since data producers and data consumers are generally different entities[30] they may not share the same interpretation of the data. For instance, a data analyst may receive some data from a relevant data engineer, but could realize of not being able to exploit the data, because he/she does not understand their meaning. This will likely prompt the analyst to ask the engineer for a thorough description of the data, causing significant time loss. Worse, the data analyst may misunderstand the data without realizing the error, and use it erroneously, potentially causing some failures in downstream work. Similar issues, which are issues in sharing data meaning, see are multiplied when many, separated, sources and/or consumers are involved in the process. This latter case is typical, and occurs, for instance, because every branch/team/technician stores data on different, non-aligned databases, which at some point must be all used coherently. And, of course, these issues grow in severity as the underlying enterprise(s) grow bigger and more complex and are notorious for their potential costs [87, 227].

**Issues preventing data interoperability** Since they are so important, let us discuss a little bit more in depth and categorize data interoperability issues. Interoperability is often divided into syntactical and semantic interoperability [238], so we shall divide interoperability issues into two corresponding categories: syntactic (semantic) interoperability issues are those interfering with syntactic (semantic) interoperability. By syntactic interoperability we mean, in the context of this thesis, the interoperability caused by the harmonization of data/information format and structure (e.g. data exchange between two databases is easier if there are similarly named tables, with similar columns containing similarly formatted fields). We define semantic interoperability as the ability to communicate data and information ensuring a mutual, clear understanding of their meaning. Issues related to semantic interoperability are usually considered as instances of 'semantic heterogeneity' [109], which could be defined as the opposite of semantic interoperability and synonymous with 'semantic interoperability issues'. Notice that data quality is known to be a 'make it or break it' factor [140], for this reason, data quality issues will be included in the enumeration of data interoperability issues. Additionally, though when talking about interoperability one imagines the case that there are two systems between which communication is being attempted, many of the issues preventing interoperability may manifest in a single system without the need to reference a second system. Therefore in the following list we do not always reference two systems when speaking of an issue. For example, synonymy is a possible issue hindering interoperability, it happens when there are two different terms sharing (approximately) the same meaning. The terms may belong to the two different systems that need to communicate, or they can belong to a single system. Synonymy could be an issue in both cases, because in both cases both systems need to know the synonym relationships to share meaning.

Note that the following is no pretense of systematical classification of interoperability issues, just one of the possible ones, hopefully useful, where we stay on a very general

[30] We say entity because they could be people but also, say, software programs.

level and do not commit to a precise context. More complex classification efforts can be found in e.g. [200, 21, 22].

Syntactic interoperability issues:

- Missing items: for instance, a relational database table could be missing a record pertaining to some real entity, or, within a given record, some field value may be missing.
- Wrong or inconsistent format: when a symbol should have a shape following some predetermined requirements but those are violated, or when heterogeneous syntaxes are used to represent similar things. For instance, dates in a database could be represented with inconsistent format (e.g. DD/MM/YYYY and MM/DD/YYYY, etc.). Similarly, in the context of natural, written language, a word could be misspelled, or various different abbreviations and acronyms could be used to write the same word. In the context of software systems, this issues also occurs when dealing with, say, different file extensions
- Incorrect values: values can be either false and plausible (e.g. a wrong but plausible date of birth in an HR database) or false and meaningless (e.g. a gibberish string in a name field). Sometimes it is possible to determine what data is wrong, sometimes one can only determine that some data is wrong, without being able to pinpoint the exact items that are wrong. The latter is the case that an inconsistency in the data is discovered. The inconsistency here could mean different things: for instance that there is data that violates a business rule (e.g. date of hiring precedes the date of the enterprise foundation) or that violates a constraint in a database (e.g. the zip code and country fields are '85577' and in 'Germany', then the city must be 'Neubiberg' [75, p.30]); or that, if one makes use of logic-based databases (such as OWL ontologies), there is a formal, logical inconsistency.

Semantic heterogeneity issues:

- Different symbols sharing the same meaning: in the written, natural language, this reduces to the presence of synonymous words (e.g. 'tank' could mean either a water container for fishes or a military vehicle [24], or 'couch' and 'trainer' could be both acceptable values for an occupation field [75, p.38], or 'latex hand protection' for 'rubber glove' [236]). Another well-known example regards primary keys in relational databases, or, more generally identifiers in information systems: a given entity should have a unique identifier within a given system. Already this could go wrong if there is an error in a given system[31], and, more frequently, different systems may attribute different identifiers to the same entity.
- Same symbol having different meanings: this is the case of, e.g., homonymity and polysemy. For instance, 'salary' meaning either "salary after taxes in French Francs including a lunch allowance" or "salary before taxes in US dollar" [236].

[31] Of course, relational databases actually prevent insertion of data that would violate a uniqueness constraint.

- Adoption of a symbol having a different but related meaning in place of the original symbol. In linguistics, this is the case of metonymy and synecdoche. There are some noticeable subcases: replacement of an instance for a type; replacement of a hyponym with a hypernym or vice versa, which happens when the two meanings are concepts related by the subsumption relation (e.g., the 'command [device]' in place of 'valve'); replacement of a whole with a part or vice versa, when the relation is meronymy (e.g., 'cable' for the numerous copper wires contained therein); the constituent for the object, e.g. 'iron' in place of a tool made of iron. In manufacturing, a common case is also the replacement of an object with the object's supplier (e.g. a 'Pentax' for 'camera').

  This is also the case of trying to communicate between different natural languages, or between two ontologies describing similar entities, or between two relational database management systems having different schemas despite containing similar information.

- Different sets of symbols are employed, with different meanings: in this case the issue is the difficulty in relating the symbols between the two separate set and their meanings, to produce a coherent whole. The addition of said relations enriches the information attached to the symbols and can be understood as a process of discovery (e.g. there is a list of drugs and a list of illnesses, and the effectiveness of a drug w.r.to an illness must be determined), but also as wasted time if the relations were known in advance and were never communicated (e.g. somebody had already carried out some data analysis to link the drugs to the illnesses, but this was never shared). The latter case is an issue stemming from the absence of a unique, explicit, and universally (at intra- or inter-enterprise level) shared master information source.

Data integration and data interoperability both aim to lessen the previous issues. These two terms are sometimes used interchangeably. However, we define data integration as the activity of merging data from different sources into a semantically consistent whole, while for us data interoperability is ensuring beforehand that the generated data is easily integrated (by using e.g. standard-based and vendor-agnostic technologies).

**The main case for using semantic technologies in enterprises.** Not having integrated data has detrimental effects while having integrated data may lead to advantages, and there are many success stories[32]. In addition, the need for data integration and interoperability is reinforced by the tendency of most enterprises of creating data silos, in a number that grows with the enterprise size [248].

On the other hand, pursuing a data integration project is costly. For instance, in [227] data warehousing[33] projects are estimated to take at least 6 months and cost 1 million USD.

[32]E.g. data warehousing within Amazon, in addition to data warehousing as a business-to-business service from Amazon.

[33]Data integration can be achieved in at least three ways: federation, mediation, and materialisation (data warehouse) [42]. In a federated database system each component controls its interaction with

Therefore, it is critical to lower the difficulty of data integration, and attack the syntactic and semantic interoperability issues discussed above. Semantic technologies are a solution of choice for this goal, for various reasons. In fact, while currently in enterprises one will find most typically relational databases, and data integration approaches are also usually implemented with additional relational databases, knowledge graphs have various advantages and this is a critical reason for their adoption [227]. Their growing adoption may become evident by consulting e.g. appropriate reviews of 'data federation' systems: for instance, in [91] 24 of the 48 systems reviewed support SPARQL as a language for querying the system.

One advantage is that the emphasis of ontologies on terminology makes it possible for the data consumers to access data using their own terminology, reducing semantic interoperability issues and tremendously improving the easiness of gathering high-quality data to answer business questions. This doesn't strictly need to happen only with ontologies, and relational databases could spend the same attention to terminological matters. However, de facto, for both historical and technical[34] reasons, actual relational databases tend to sacrifice the integrity of their conceptual schemas (which is the same as an ontology) in favor of the efficacy of their logical and physical schemas in the context of the application to be developed [227]. This often results in schemas that cannot be understood by data consumers, even though data consumers know the underlying domain the data that follow the schema is about. This advantage mitigates the issue of sharing data meaning.

Moreover, knowledge graphs are very flexible: their schema can be easily modified even when they are currently storing data, while in relational databases this may be very complicated. In fact, using the RDF language, there is not even a strict separation between the schema and the data. This advantage mitigates the costs that a global schema carries, since it makes schema changes (e.g. adaptations of current or future local schemas to the global schema, or modifications of the global schema itself) cheaper. Additionally, this flexibility in schema modification makes it so that the integration of knowledge graphs can be carried out in a native, obvious, way: it is sufficient to add appropriate concepts and relations to the knowledge graphs to be merged.

SPARQL has also a native way to carry out explicit (in the sense that the user specifies which part of a query is addressed to which source) data federation, by using the `SERVICE` keyword[35].

For these reasons, and for the fact that relational databases are the standard legacy database technology, current knowledge-graph-based federation systems focus on the

the other components. Mediated database systems, instead, contain a mediator component, which consists of a global virtual database that allows integrated access to the components. The mediator is virtual in the sense that the data it provides access to is not stored locally, but must be retrieved from the original source when requested. The data warehouse architecture is similar to a mediated system, only the mediator is now a materialized centralized database (in the sense that the data is copied, duplicating it from the sources).

[34] For instance, using OWL one can formulate concept definitions, this cannot be done in relational databases.

[35] `https://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/`.

case that the federated sources are relational, while the offered global virtual schema[36] is a graph.

On top of this, semantic technologies can use reasoning to infer new knowledge from the data. This is not used often, but the more advanced systems (such as ontology-based data integration, mentioned below) make effective use of the reasoning.

All the previous points are related to data integration *per se*, but semantic technologies are naturally useful also to ensure that the data is interoperable:

- they are *standard-based*, due to the rich galaxy of well-specified standards that the W3C has produced throughout the years.

- they are *vendor-neutral*, mainly as a consequence of them being consistently standard-based: a knowledge graph managed by the system of a given vendor, may be put as-is into the system of a different vendor. Another thing that is difficult to do with relational databases [248, p.249]

Notice that semantic technologies can be used in both materialization-based[37] (by copying the data into a knowledge graph) and virtualization-based approaches to data integration (by supplying access to the data through a virtual knowledge graph). The most known case is the so-called ontology-based data integration (OBDI), often reduced, for the sake of simplicity when studying academic topics, to ontology-based data access (OBDA)[43], that is, to the case that there is only one source database[38] (in this case there is no data integration between distributed databases, but there still is the issue of sharing data meaning to be solved). OBDI is a type of mediated data federation architecture, in which the mediator is an ontology. This has the effect that (i) the global schema is actually a semantically rich description of domain concepts and their relations, and (ii) ontology reasoning is actually pivotal for the tasks that the mediator has to carry out, especially query answering.

For instance, in the flagship implementation of OBDI, Ontop[39], query answering is carried out through *query rewriting*. In particular, the SPARQL query received by the mediating ontology is first expanded by rewriting it using the knowledge contained in the OWL ontology axioms, then it is expanded again into a SQL query, this time by using the knowledge of the mapping, which can be expressed either in a native format or in the R2RML[40] mapping language[41]. In this way, the ontology axioms allow queries to return also implicit information, and act, effectively, as a way to compress

[36] This terminology refers to the mediated data integration architecture, see Note 33.

[37] Again, refer to Note 33.

[38] Actually, since the authors of [43] apparently consider the disjoint union of source schemas as a valid case of a federated database, there is little difference between OBDA and OBDI, since, at that point, a virtual ontology layer on top a trivially federated union of schemas would have to do the whole data integration by itself.

[39] `https://ontop-vkg.org/guide/`.

[40] `https://www.w3.org/TR/r2rml/`.

[41] The alternative to query rewriting would be *materialization*, where the query is first rewritten into a SQL query using the mapping, the query is then executed on the relational database, and finally the ensuing results are expanded by recursively applying the ontology axioms.

the queries, by delegating some logic to the axioms. This is very important, at least because a good deal of business knowledge is usually included in business rules which are encoded into programs and queries located in and around database management systems. Classically, this results in interoperability issues, which can be dispelled by putting the business logic in the ontology. Of course, there are also various limitations. In particular, the expressivity of the language allowed for the axioms must be rather limited: for example Ontop allows at most OWL 2 QL[42].

**Other uses of semantic technologies in enterprises.** Above, we focused on data interoperability and integration, which are important aspects of data management, but not the only ones. Data quality, data governance, metadata management, and data security are other essential aspects.

One discipline with well-developed semantic-technologies tools (in addition to data integration) is data quality assurance, and, in particular, data validation. Indeed, data validation is the goal of the Semantic Web standards SHACL and ShEx [163], which are well-known in the industry (at least in the context of semantic technologies). Unfortunately, valid data, i.e. data that respects a series of predetermined rules, is not the only ingredient that makes data accurate: data is accurate when it contains true values that are correctly represented [201]. For example, if a database contains the information that 'John Smith' has the address '10339 W Grand Ave' and that 'Mary Sue' has the address '1a39 W.! Grand', say these are two rows in a table. Then, the second address is invalid (it does not have the form a U.S. address should have), while the first is valid (it looks like a correct address) but is inaccurate because the address does not exist. Accuracy is not the only relevant aspect of data quality: timeliness (the data is not stale), completeness (no data is missing), intelligibility (the meaning of the data is understood), and trust are others; however, accuracy is the most important [201].

In any case, data validity is still an important part of data quality, and SHACL and ShEx can be used to check that, both on the level of a single data item and on the level of relations between data items. Violations of the latter case can also be found by reasoning engines deliberating that a set of statements is inconsistent.

This concludes our brief, introductory excursus in the history of applied ontology and in the uses of semantic technologies in industry.

[42] See e.g. Figure 1.2.

# Chapter 2

# State of Art Review: Functions and Malfunctioning in Manufacturing

In this chapter we review the state of the art of the literature about functionality and malfunctioning in engineering and ontology.

## 2.1 State of the Art of Function Modelling

Functional reasoning, that is, thinking of entities in terms of their purpose, is an essential part of science[1] and technology[2], not least because it's natural for scientists and technicians alike to think in teleological[3] (functional) terms when analyzing a system, be it a biological system, or an engineered product[4].

We are especially interested in the last case, this means that we focus on the activities of an engineered product and not, say of a natural or social system. Additionally, this also means that many properties of an engineering product are outside the scope of our work, such as its form, cost, history, aesthetic design, among others, which are usually called 'non-functional' properties.

Consider, for the sake of discussion, that motors are often described as having the function of obtaining mechanical energy by converting some other type of energy (say, electrical energy in electrical motors or chemical energy in combustion motors). Surely producing electrical energy is something that motors do, but, as motors do also other things such as making noise or vibrating, functionality cannot be reduced to the set of

[1]Consider e.g. Ayala that states "teleological[i.e. functional] explanations in biology are not only acceptable but indeed indispensable"[11].

[2]Consider e.g. "[t]he concept of a function is of great importance in [engineering] design" from [123], and the considerable attention dedicated to functions in priminent engineering methodologies [205, 49].

[3]The adjective 'teleological' here indicates explanation by reference to purpose, goal, or, indeed, function. For instance, to say that a sheeps exists to provide wool is a teleological explanation the existance of sheeps, while to say that they exist because a certain evolutionary process took place is not.

[4]Indeed, consider as examples of this Dimanzo that describes functional reasoning as apparently ubiquitous in human behavior [65], and Love that states "Darwin's primary reasoning style was functional" [169].

actions that a device (or an organ, or a person) carries out. Indeed, what functions are, exactly, has been the object of lasting discussion. In engineering, this discussion is motivated by strong practical interests as, for example, engineers commonly use functions in order to design [204] and diagnose devices [164], while philosophers and ontologists focus is towards understanding what functions are and what is the meaning of expressions such as "the function of X is to Y-ing", where X and Y usually are a noun and a verb, respectively.

The review in this section is divided in three parts: first the content of the main engineering methodologies developed to deal with functions is discussed, then the same is done with the main ontological treatment of functions, lastly a final section summarizes some selected main points.

### 2.1.1 Early history

Whatever functions are, they have been studied as full-fledged research subjects in engineering, since at least the 70s. For example, in [79][5], they are heavily involved in the design process, and are conceptualized as 'black-box transformations' that acts on inputs to produce outputs (both input and outputs are called *flows*). As engineers make strong use of graphical languages, the idea of functions as transformations has graphical counterparts. The names used to refer to these graphical languages are many, for example, an early instance from 1962 can be found in [110] (but their use by engineers is surely older) and is called 'functional block diagrams' (see e.g. Figure 2.1). In this language, functions are represented as blocks, possibly labeled with appropriate names (e.g. 'discriminator') or drawn with conventional shapes; while the flows they act upon are represented as arrows, possibly labeled with names (e.g. 'video-in') and properties (0.2-2.5V).

### 2.1.2 The German school of funktionenstrukture

The idea of functional block diagrams grew and differentiated into various subsequent methodologies. We focus in particular on engineering of manufactured artefacts, but, of course, many other engineering disciplines employ this kind of modelling: for instance, the Structured Analysis and Design Technique methodology does so and was developed in the late 1960s [219] in the context of software engineering.

Coming back to functions of engineered artefacts, the contributions to their study by some German researchers working on engineering design since the late sixties are especially important in this context. In particular, Rodenacker [216] proposed a methodology for design consisting in the decomposition of a system function into sub-functions, which should, in turn, be decomposed into further sub-functions and so on, up to the point that the final sub-functions can be realized by mechanical element chosen from a pre-existing catalog. Roth authored one of the most known catalogs [220], which consists in tables of mechanisms or engineering systems organized depending on their function and other relevant properties (such as symmetry or shape). Within the framework of

[5]This is the original German version, which was afterward translated in English [204].

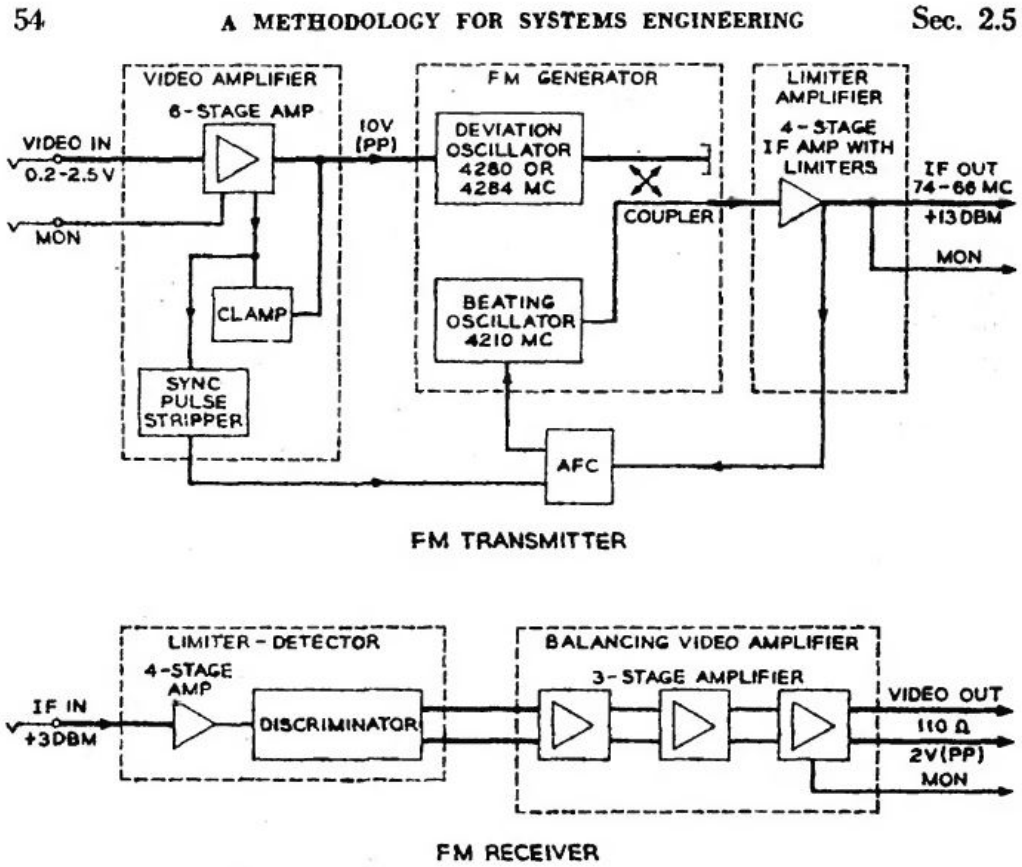



Figure 2.1: Functional block diagram of some FM radio terminal, taken from [110].

these design catalogs, functions and the mechanisms that execute them are aligned on a common abstract-concrete spectrum, varying based on their 'degree of embodiment'. In this approach to design, which we shall refer to as the *German school of funktionen-strukture*, functions play an essential role, because the functional decomposition of a product largely determines its structure.

In particular, in the works of Pahl and Beitz in the 70s, collected in their classical book about engineering design [204] (original German edition [203]), functions are conceptualized as "relationships between the inputs and outputs of a plant, machine or assembly", where the inputs and outputs are divided into three types: energy, material, and signal.

**Flows** The three types of energy, material, and signal are usually exemplified by enumerating some of their sub-types, for instance:

- energy can be thermal energy, mechanical energy, electrical energy, and so on.
- A material could be categorized by its aggregation state or as a component, a workpiece, etc.
- Signals can carry information about the status of a system or can be commands sent to a system.

Thus, a function is a (or a series of) *transformations* (*umsatzes*) of these three entities. For instance, an electrical motor converts electrical energy into mechanical energy and, as a waste product, thermal energy; while materials can be transformed by e.g. being reshaped, coated, mixed, transported, and so on; and signals can be received, elaborated, transmitted, and so on. It is argued that such transformations can be conveniently considered as *flows* (*flusses*) of the operands. Using this metaphor, the operands flow in the functions, are there transformed, and finally flow out. Since the operands are consistently divided into the three categories of material, energy, and information, there are thusly three kinds of flows: *material*-flow (*stoff-fluss*), *energy*-flow (*energie-fluss*), and *signal*-flow (*signal-fluss*). Naturally, this means that are three corresponding types of transformations (*stoff-umsatz*, etc.), and corresponding functions.

This tri-partition is most probably taken from the work of Weizsäcker [262], and is expanded, in particular, by Roth [220], see Figure 2.2. However, this tri-partition was probably already used before the German school and is still used in contemporary engineering. Further, it is assumed that one can identify a *main-flow* (*haupt-fluss*) of an engineering system, which is the operand of the most important transformation carried out by the system. For instance, the main-flow-kind of a television is signal-flow, while the main-flow-kind of a motor is energy-flow.

**Function vocabularies** The term 'transformation', by itself, can be used with a vast quantity of meanings. Therefore, many controlled vocabularies were proposed in the literature to capture such meanings. Pahl and Beitz [204] themselves use one introduced by Krumhauer [160], which divides functions into types depending on the property

| Eigen-schaft | Ruhemasse besitzend | Vermögen Masse zu beschleunigen | Unbekanntes vermindernd |
|---|---|---|---|
| Nr. | 2 | 3 | 4 |
| 1 | **St** Stoff (Masse) | **E** Energie | **I** Information |
| 2 | **Su** Substanz | | |
| 3 | | **Si** Signal | |
| 4 | **Ge** Geformter | ____ | Stoff |
| 5 | **M** Materie | | |

(a)

| Feature | Have rest mass | Able to accellerate mass | Lessen the unknown |
|---|---|---|---|
| Nr. | 2 | 3 | 4 |
| 1 | Material (Mass) | Energy | Information |
| 2 | Substance | | |
| 3 | | Signal | |
| 4 | Formed | __ | Material |
| 5 | Matter | | |

(b)

Figure 2.2: Original and translated table showing different categorizations of flows, taken from [220]. "Rest mass" is interpreted in the relativistic sense. The union, in the relativistic sense, of material and energy is called "substance", the union of information and energy is called "signal", and the union of information and material is called "formed material". Finally, the union of information, energy, and matter is called (self-organized) matter, the latter concept is taken from [267, 58].

(time, number, magnitude, place, or type) of the operand that the function transforms (notice that this division is perpendicular to the division of functions depending on their main-flow-type), see Figure 2.3. Such function-types are called *generally valid functions* to highlight their independence from any possible engineered system that can execute them. Therefore generally valid functions, which are the most abstract functions in this approach, can be used to classify engineered systems in an implementation-independent way.

Generally valid functions are divided into five types depending on the type of flow property that is changed by the function: type, magnitude, number, place, and time (see Figure 2.3).

## 2.1.3 Other engineering approaches to functions

### 2.1.3.1 Functional Basis (FB)

Based on the German School various other vocabularies for engineering functions were produced, one that had considerable diffusion is the Functional Basis (FB) [235, 116]. This vocabulary is also used, by its authors, to enable a methodology for functional modeling, so that, in this thesis, by FB we refer to both the vocabulary and the methodology. The FB vocabulary contains two categories of entities: functions and flows.

Flows are essentially the things engineering devices operate on when carrying out their functions. Consistently with the German School, flows are divided into three main types: energy (e.g., mechanical energy, electrical energy, and such), material (the fuel used in a combustion motor, the workpieces a machine operates on, and so on), signal (e.g., the control signal sent by a yoke to an airplane elevator, the acoustic signal emitted by a car alarm when beeping, ...). Signal flows are recognized as peculiar in

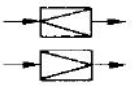
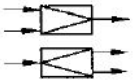
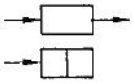
| Characteristic Input (I)/Output (O) | Generally valid functions | Symbols | Explanations |
|---|---|---|---|
| Type | Change |  | Type and outward form of I and O differ |
| Magnitude | Vary |  | I < O<br>I > O |
| Number | Connect |  | Number of I > 0<br>Number of I < 0 |
| Place | Channel |  | Place of I ≠ 0<br>Place of I = 0 |
| Time | Store |  | Time of I ≠ 0 |

Figure 2.3: The generally valid functions of Pahl and Beitz [205].

the sense that they are energy (e.g. low voltage electrical signals) or material flows (e.g. ink on paper, a punched card) that have the role of carrying information. For instance, a radio signal is an electromagnetic wave, which by itself could be considered an energy flow, but, since it is used to transfer information, say a news bulletin, it is classified as a signal.

The second category of entities is functions, which are conceptualized as (black-box descriptions of) operations carried on the flows. The function terms are always verbs (convert, supply, store, branch, ...), while the terms for flows are nouns. This is because, since functions operate on flows, one can rend this linguistically by a verb-object (function-flow) format, for example 'the motor *converts electrical energy* into mechanical energy'. Functional terms (as well as flow terms) are organized into a three-level taxonomy in order of specificity. The full taxonomy is reported in Figure 2.4. Functional models developed using FB then take the shape of graphs whose nodes are boxes labeled with function terms, and whose edges are arrows labeled with flow terms.

#### 2.1.3.2 Functional Representation (FR)

Another relevant methodology is Functional Representation (FR), proposed by Sembugamoorthy and Chandrasekaran [226] with the goal to build "deep" device models enabling reasoning about such devices, especially diagnostic reasoning.

They use the term "deep" because they claim their models *generate* novel diagnostic rules based on underlying functional knowledge. Instead, shallow models would be those that make diagnoses based on explicit rules linking symptoms to malfunction hypotheses. They call shallow models "compiled", because the symptom-malfunction

| Class (Primary) | Secondary | Tertiary | Correspondents |
|---|---|---|---|
| Branch | Separate | | Isolate, sever, disjoin |
| | | Divide | Detach, *isolate*, release, sort, split, disconnect, subtract |
| | | Extract | Refine, filter, purify, percolate, strain, *clear* |
| | | Remove | Cut, drill, lathe, polish, sand |
| | Distribute | | Diffuse, dispel, disperse, dissipate, diverge, scatter |
| Channel | Import | | Form entrance, *allow*, input, *capture* |
| | Export | | Dispose, eject, *emit*, empty, *remove*, destroy, eliminate |
| | Transfer | | Carry, deliver |
| | | Transport | Advance, lift, move |
| | | Transmit | Conduct, convey |
| | Guide | | Direct, shift, steer, straighten, switch |
| | | Translate | Move, relocate |
| | | Rotate | Spin, turn |
| | | Allow DOF | *Constrain*, unfasten, unlock |
| Connect | Couple | | Associate, connect |
| | | Join | Assemble, fasten |
| | | Link | Attach |
| | Mix | | Add, blend, coalesce, combine, pack |
| Control | Actuate | | Enable, initiate, start, turn-on |
| | Regulate | | Control, equalize, limit, maintain |
| | | Increase | *Allow*, open |
| | | Decrease | Close, delay, interrupt |
| Magnitude | Change | | Adjust, modulate, *clear*, demodulate, invert, normalize, rectify, reset, scale, vary, modify |
| | | Increment | Amplify, enhance, magnify, multiply |
| | | Decrement | Attenuate, dampen, reduce |
| | | Shape | Compact, compress, crush, pierce, deform, form |
| | | Condition | Prepare, adapt, treat |
| | Stop | | End, halt, pause, interrupt, restrain |
| | | Prevent | Disable, turn-off |
| | | Inhibit | Shield, insulate, protect, resist |
| Convert | Convert | | Condense, create, decode, differentiate, digitize, encode, evaporate, generate, integrate, liquefy, *process*, solidify, transform |
| Provision | Store | | Accumulate |
| | | Contain | *Capture*, enclose |
| | | Collect | Absorb, consume, fill, reserve |
| | Supply | | Provide, replenish, retrieve |
| Signal | Sense | | Feel, determine |
| | | Detect | Discern, perceive, recognize |
| | | Measure | Identify, *locate* |
| | Indicate | | Announce, show, denote, record, register |
| | | Track | Mark, time |
| | | Display | *Emit*, expose, select |
| | Process | | Compare, calculate, check |
| Support | Stabilize | | Steady |
| | Secure | | *Constrain*, hold, place, fix |
| | Position | | Align, *locate*, orient |

Figure 2.4: Full taxonomy of functional terms, reported from [116]

rules are obtained by compiling (synthesizing) the deep knowledge of domain experts into an actional form (see [50] for a more detailed discussion of deep vs. compiled diagnostic systems).

Coherently to its original purpose of achieving deep diagnostic reasoning, and in opposition to design-centric methods, FR requires a model of the physical structure of the already-existing device. In particular, the device is modeled as a set of components, each one with relevant parameters and input-output ports, as well as relations between the devices (examples of relations are electrical connection or the containment of a component into another). Then, different device states are described by the use of state variables, and the device behavior is characterized by a network of transitions among these states. Such networks are called "causal process description" (CPD), since

they are directed graphs whose edges are interpreted as the causation relation among neighboring states. Each transition in a CPD is justified with an explanation, of which there are three types:

- CPDs: another causal process description can be invoked in the explanation of an initial CPD (e.g. the process of 'cooking pasta' may use the process of 'boiling' in its explanation).

- Component functions: the function of another device, say one of the components of the original device, may be invoked in the explanation of a causal link.

- Domain laws: known physical principles may be invoked during explanation, for instance Ohm's law in electronics.

In this context, typically, functions are themselves state transitions from a precondition to a function goal, and they are themselves explained with a CPD. Both the precondition and the function goal are states, and the interpretation of the functional goal depends on the function type, while the precondition is always the initial state of the CPD. In fact, functions are divided into the four types introduced by Keuneke [138]: "toMake", "toMaintain", "toControl", "toPrevent". In functions of "toMake" type the functional goal is (or holds in) the final state of the CPD explaining the function, and must come into being due to the causal action of the device operations. Analogously, in functions of "toPrevent" type the functional goal must not hold for any state of the CPD, while in the "toMaintain" case it must be maintained. Finally, in the "toControl" case, a predetermined functional relation must hold between some independent controlling variables and some dependent controlled variable, whose value is causally determined by the controlling variables (a full explanation of the four function types, as well as a formal discussion of CPDs is present in [130]).

In addition to its original purpose of diagnosis of engineering systems, FR has been proposed as the basis for design environments allowing designers to ensure that the designed device is going to satisfy all required functions, by carrying out a simulation of the device behavior in a way strictly linked to its functional description [259, 131]. Moreover, FR has been argued useful to designers in order to capture the causal aspect of the rationale behind various design activities [47]

In later works, Chandrasekaran expanded on the concept of function (the previous works can be regarded as focusing on the descriptions of functions more than defining functions): first functions were defined as "behavior constraints" [48]:

> Let F be a set of constraints defined in some W[environment]. Let $\Theta$ be a new object introduced into W, in a relationship R($\Theta$,W) and let $W_\Theta$ be the extension of W when $\Theta$ is included. If $\Theta$ causes F to be satisfied in W, we say that $\Theta$ plays, performs or has the role F in $W_\Theta$. If F is intended or desired by some agent A, then we say that $\Theta$ has or performs, the function F in W for A. Often A is understood to refer to a designer or a user, and is dropped.

Here, a behavioral constraint is defined as logical expressions about devices properties, e.g. "If diameter of pipe 1 is greater than the length of pipe 1, then the diameter of pipe 2 should be greater than the length of pipe 3" [48], additionally, the relation R(Θ,W) between the device and the environment is called "mode of deployment". Then, in [49], the previous definition is called "environment-centred" definition of functions, which are now heuristically described as "roles+intentions". Moreover, a different kind of definition of functions ("device-centred") is introduced:

> Let F be a set of behavioural constraints defined on, and satisfied by, an object D. If F is intended or desired by an agent A, then D has function F for A.

Essentially, this new definition is foreign the interaction of the device with the environment.

Some acknowledged limits of FR are:

- the difficulty in representing "passive" functions [46, 47], such as that of a chair that supports the weight of a person, or the one of a pipe that allows the movement of fluid just by virtue of its physical shape. In fact, functions such as these are not naturally explained by a succession of states and thus are not naturally explained by CPDs: a more natural explanation consists of describing the function as a single state made possible by the virtue of the physical properties of the underlying devices. In this sense, the more natural explanation of such functions is a 'static' one. Still, engineers do sometimes make use of 'dynamic' language even in these cases: for instance, one can talk about how a beam *transmits* the load, or how the load is *split* among the chair legs. In such a way, one can recover the use CPDs to explain also "passive" functions. To our knowledge, modeling of "passive" functions with FR was only carried out in Korda's thesis [159].

- In addition to "passive" functions, there are other classes of phenomena for which an explanation as state sequences may not work well, the only one of such classes explicitly mentioned by Chandrasekaran are "behavior that depends on shapes of objects" [47].

- Some occurrences that one would like to make part of CPD are not states. For example, a sawtooth signal is not a state by itself, but it would be more likely represented by the continuous cycling among a charged and a discharged state [47].

#### 2.1.3.3 Function and Behaviour Representation Language (FBRL)

This framework [150, 223]commences with a structural model of the system, akin to the structural models in FR, obtained through a distinct module known as 'device ontology'. In FBRL, a function is a role played by the behavior of a device within the context of a system, when contributing to the attainment of predetermined goals. Behaviors and functions are separated and this separation clarifies how a device can possess different

functions while its behavior remains constant (for instance, a heat exchanger can either cool or warm a liquid based on the system configuration: the behavior is the same, while the function changes depending on the system configuration). Another notable point is that the vocabulary of functional terms (such as 'to join') is distinguished from the vocabulary of 'way-of-achievements' (such as 'welding', a possible way to realize the 'to join' function): in this framework, functions are considered domain-independent and can be stored in a unique, shared ontology, while way-of-achievements are domain-dependent and can be stored in domain-specific databases. Both the functions and the way-of-achievements are organized in rich taxonomies. To provide a teleological explanation of a system, a set of 'meta-functions' is introduced, which consists of purely teleological relations between functions (e.g., the 'give-heat' function of the boiler drives the 'rotate-shaft' function of the turbine). Note that the introduction of these meta-functions allows the engineer to draw functional diagrams on a purely teleological level, while in other methodologies (e.g. FB) the functions in the diagrams are linked by arrows (the flows) which represent concrete entities (e.g. the 'wood' and 'electricity' flows link the 'cutting' function to other functions of an electric bench saw, and these flow refer to concrete entities: pieces of wood, and quantities of electric energy; on the other hand, the 'driving' meta-function that links the give-heat function to the rotate-shaft function in the previous example, does not refer to any concrete entity).

#### 2.1.3.4 Other functional methodologies

In addition to those mentioned above, numerous other modeling frameworks have been developed to address the functional and behavioral representation of engineering systems, such as Umeda et al.'s FBS (Function Behaviour Structure) [246], Goel et al.'s SBF (Structure Function Behaviour)[85], and Qian and Gero's FBS (Function Behavior Structure)[209]. Briefly explaining the characteristics and differences among all these frameworks would be challenging. However, simplifying, commonalities can be isolated: these frameworks generally view the structure of a device as a set of parts (components) along with their attributes and the relations between them. Typically, they describe behavior as a sequence of states of the components, and the definition of function primarily recognizes a dependence of functions on behaviors and of behaviors on the structure.

Despite these commonalities, there are key differences. For instance, in Chandrasekaran's FR, (functional) behavior refers to a causal process, while in Gero's FBS representation, behavior is linked to the properties of structural components. Another distinction lies in the fact that Gero, Chandrasekaran, and Goel explicitly allow for the decomposition of functions into behaviors: for instance, in Goel's approach, the functional decomposition of a system can mix different kinds of entities, since a function is a state transition, which could be explained by either functions or behaviors, alternatively, at different levels of granularity. On the other hand, in Sasajima's approach, functions only decompose into other functions.

### 2.1.4 De Kleer's principles

It is necessary to mention another set of foundational works, (partially) contemporary to the development of the German School: those of De Kleer [61, 60, 153]. We are interested in these works due to two foundational principles they outline[6]: the principle of the locality of functions and the 'no function in structure' (NOFIS) principle.

The principle of locality "demands that the laws for a part cannot specifically refer to any other part"[153]. More precisely, a device should have ports[7] that allow for connection to other devices, and the functional description of the device may at most reference the value of some quantity of interest at these ports, so that, say, 'the voltage drop across the resistor is half of the voltage supplied by the battery' is not a local description of the resistor function because a battery is mentioned.

The NOFIS principle was originally stated as [61, p.287]:

> the rules for specifying the behavior of any constituent part of the overall system can in no way refer, even implicitly, to how the overall system functions.

Respecting this principle is tricky, since, for example, describing a switch by "if the switch is off, no current flows; and if the switch is on, current flows"[153] violates the principle. Indeed, so tricky that De Kleer considers its complete achievement impossible [153, p.16], and proposes a weakened version, in which one can make assumptions about the context of the device, if these assumptions are *class-wide* (e.g. ignoring compressibility of fluid for most hydraulic devices). The goal of this principle is to ensure the production of *robust* models simulation or explanation of for engineering system functions. De Kleer explains later that a model is "robust with respect to its device structure, if the questions that can be asked about the device structure can be answered correctly" [61, p.289]. However, by accurately reading his work, it appears that his idea of robustness is more similar to the idea that a model can answer correctly when the default, standard assumptions regarding a system characteristics and context are perturbed: e.g., if a component malfunctions or is wired to the circuit with reversed polarity. Clearly, then, the more context-free functional[8] descriptions of devices are, the more robust will be the models explaining how systems produced by the combination of these parts are.

The NOFIS principle is central among the principles predicated by De Kleer and the one that was mostly discussed in the literature. Indeed De Kleer himself says that the locality principle helps with the satisfaction of the NOFIS principle. In particular, Keuneke and colleagues [137] discuss the principle in depth. In summary, their conclusion is that there is a tradeoff between how much functional descriptions of devices are context-free and how complex must be the background theory of physics used for simulation or description of the system workings. E.g., if one assumes the device is a component in a linear electrical circuit, then a few simple equations suffice; if one would

[6]There are other principles, but they are less relevant for this discussion.

[7]The term port is not actually used by De Kleer.

[8]Note that in [61] there are no stated difference between a system function and a system behavior.

like to give a context-free description of the behavior of a piece of wood that can explain the behavior of the wood both as a wave breaker (groin), as the keel of a boat, and as foundation for a house in Venice, then one needs very complex fluidodynamics and chemical theories. For this reason is better to violate the NOFIS principle, but, similar to De Kleer's class-wide assumptions, limit this violation to assuming knowledge about "classes of devices about which some knowledge is available, and which are meaningful to domain experts".

From our point of view, these principles are useful because they naturally classify functional descriptions depending on the level of their violation: for instance, the meta-function-based description "the 'give-heat' function of the boiler drives the 'rotate-shaft' function of the turbine", as a description of the boiler function strongly violates the NOFIS principle, since it refers to other devices belonging to a given system, while a "the boiler function is to-give-heat" does not (at least not at the same level). We will refer back to this point in Section 5.1.

### 2.1.5 Functions in philosophy and applied ontology

Of the sizeable literature related to functions in ontology, We focus only on a few things that are especially relevant for this thesis: the main philosophical approaches to functions, some classification of functions and especially some classical dualities related to functions, the treatment of functions in upper ontologies, and classical list of desiderata about theories of functions.

**Dualities about and classification of functions.** The two dualities that are probably most considered by philosophers are:

- Intrinsic vs extrinsic: there exist a debate on whether functions are entities intrinsic to their bearer or something else. For example, Kitamura and Mizoguchi [147] speak of 'capacity functions', which are properties that belong to the artifact, live 'inside' it, and depend on it; differently from 'actual functions', which are instead related to the device performance and are external to it and less dependent on it. They also state that capacity functions are the capacity to perform an actual function. Rhöl and Jansen make a similar distinction [218]: they say that functions are externally grounded, that is 'optional given the physical structure', while the entities that are internally grounded are dispositions.

- Essential (or proper) vs accidental: intuitively, there is a difference between the screw-screws function of a screwdriver, and the cut-paper-envelope function that a user may sometimes use the screwdriver for. For example, Rhöl and Jansen say [218] say that functions are essential properties to their bearers, while it is roles (which are not functions) that are accidental, so that they would probably say that only the function the screwdriver was designed for is a proper function (let us call it the 'design function'), while the function devised by the user (let us call it the 'use function') and not by the designer is actually a role. Kitamura

and Mizoguchi [147] instead consider both such things as functions, and just call essential the former and accidental the latter.

In addition to these, for the sake of coherence, we append two other dualities, which are however more specific to engineering functions, and are not discussed as much in other domains.

- Function vs behavior: this is a classical distinction present in many engineering methodologies. We just mention that is frequent in engineering to define functions as intentionally selected device behaviors, and redirect the reader to the previous sections of this chapter, as well as Section 2.1.8.1, for more information.

- Device (or component) function vs Environment (or effect or external) functions: this duality is founded on the separation between a system and the environment it operates in. Roughly, the idea is that functions can be seen as the effect a device/system can have on its external environment (e.g. 'the function of the light switch is to light the room'), or they can be seen as their contribution to the functioning of a system their bearer is embedded in (e.g. 'the function of the light switch is to allow current to pass through'). Chandrasekaran tackles this duality with his device-centred and environment-centred functions (recall Section 2.1.3.2), while Kitamura, Takafuji, and Mizoguchi [147, 152] talk about component or device functions (which 'represents changes of entities within the system boundary' [152, p.3] and depends on a functional decomposition of a device, which supplies a so-called 'system function context') and external functions (which depends on a user supplying an 'external function context').

All these dualities naturally lead to rich classifications/taxonomies of functions, since one could speak of accidental functions, essential functions, etcetera. In addition, richer classifications that can not be derived just from the previous dualities have also been considered. We mention only three examples, the reader can find others in the references of [152].

Burek [39, Section 7.4] divides functions also depending on the time length of their goal (e.g. 'to maintain a high temperature in the room' would be a continuous function, 'to move a car in the truck' would be an instantaneous function), depending on the relations they have with other functions (so that there are 'enabler' functions, and so on, similar to the relata of SOFAST' meta-functions), depending on the agent that intentionally ascribes the function (design function, user functions, ...), depending on whether they are active (e.g., moving something) or passive (e.g., allowing something), and so on. Kitamura, Takafuji, and Mizoguchi [152] produced a complex taxonomy of functions (see Figure 2.5) that classifies functions depending on the value of their descriptors, which are attributes, participants, or parts of a more general 'goal-oriented context'. For example, flowing-object functions are functions in which an agent acts on operands that are 'flowing objects' (generalized input-outputs entities); while in inter-device functions both agents and operands are devices. Finally, Rausand and Øien [211] consider also some classes of functions that are somewhat different from those considered by the previous classification systems: protective functions, which protect people, the

environment, and the equipment; information functions, whose output operands are information; and superfluous functions, which may be present in a device if it operates in an operation environment different from its specified one.

**General philosophical accounts of functions.** Let us now take a more holistic approach and consider theories about functions on a more general level. First, intentionality is a pivotal aspect of engineering functions, in particular, one can divide functions depending on the agent who project an intention on the function object, for example, the users, or the designers and manufacturers.

Second, approaches to functions can be divided into (at least) two main branches: causal and etiological [33]. Causal accounts of functions (e.g. [59]) identify the function of a trait or object based on the role it plays in bringing about (causing) the achievement of certain goals within a system, specifically, on being disposed to have that causal role. Etiological accounts (e.g. [187]) identify the function of a trait or object in terms of its history and how it has been selected and maintained through evolutionary processes. For the sake of simplicity, we also call etiological those accounts of functions based on intentional attribution by some agent (designer, manufacturer, user, etc.) at some point of the life of some entity.

Both approaches have known limitations [33]: causal accounts do not have a way to distinguish accidental from essential functions: there is no clear systemic criteria that could differentiate between, say, the contribution of the nose to breath from the contribution of the nose to read by supporting glasses. Also, they may have difficulties in supplying a way to separate the cases of not having a function at all and not having a function due to accidental malfunctioning. Etiological accounts do not take into account the actual behavior of an individual entity, only its history or even the history of its kind. Therefore, it appears difficult that these accounts can explain all phenomena related to function, e.g. the aforementioned possibility of malfunctioning.

Now, note that these philosophical approaches include the analysis of engineering functions but also embrace other domains. For example, etiological accounts are typical in (philosophy of) biology. However, both approaches mentioned above may and their discussion may be applied to the engineering domain. The same can be said for (some of) the approaches described in the following section, which focuses on applied ontology.

### 2.1.6 Functions in Applied Ontology.

Despite the potential benefits of ontological analysis of functions in the engineering domain, the existing attempts have been (relatively) limited. Some research endeavors have formalized aspects of the Functional Representation (FR) system [27], initiated the formalization of the Functional Basis vocabulary and its underlying conceptualization using the upper ontology DOLCE [26, 28], compared the Functional Basis with either the FR system [76] or both FR and FBRL [151]. The function decomposition relation has also been scrutinized using ontological techniques [254, 256].

Furthermore, applied ontology has left its mark on engineering literature, as evidenced by the depiction of functions as roles played by behaviors [193, 141, 49], or by

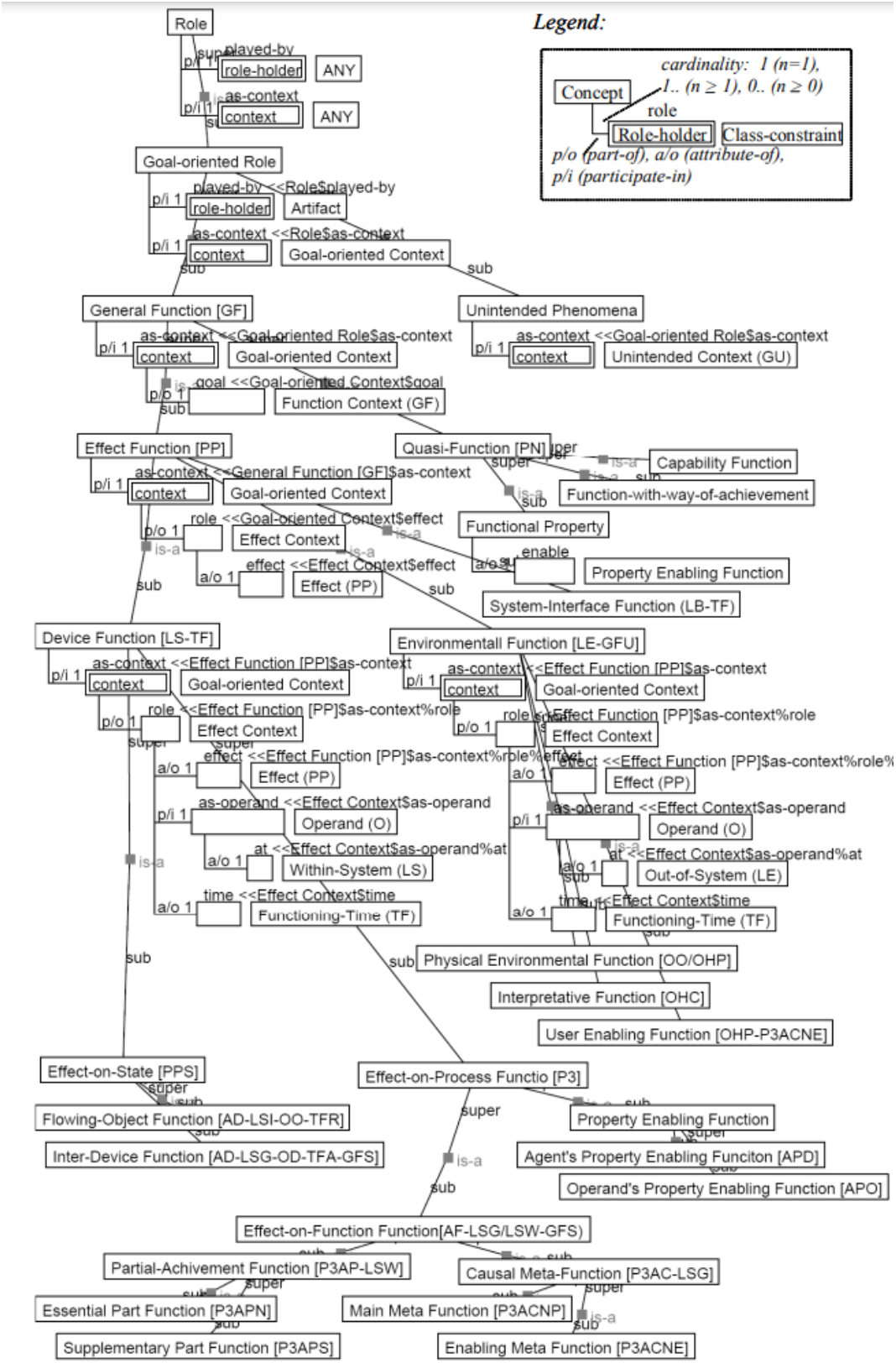



Figure 2.5: A reference taxonomy of functions, reported from [152]

the categorization of behaviors as occurrents [141].

Nevertheless, a universally accepted reference ontology for functions is still absent, and top-level ontologies generally do not delve into theories of functions as understood in engineering. For instance, DOLCE doesn't incorporate the concept of function within its specifications [177]. While in YAMATO, functions are categorized as roles, and behaviors are classified as processes[9]. Specifically as processes that are actions carried out by non-intentional actors ('doers'). On the other hand, BFO defines functions as dispositions inherent in their bearer's physical makeup, such that the physical makeup came into existence through (at least in the case of engineering) intentional design. Similarly, GFO acknowledges that functions can be realized by other entities but does not mention dispositions. Instead, it terms functions as 'intentional entities' [115]. Another important upper level ontology, born explicitly as an industry standard, is ISO 15926. Such an ontology went through many changes throughout the years, but in both the second-last iteration (ISO 15926-14 [154]) and the latest one (The Industrial Data Ontology – IDO, which is ISO CD 23726-3 [128, 37]) adopt BFO's perspective, viewing functions as dispositions realized in specific processes. Finally, "[t]here is no explicit definition of function in UFO" [101], however in UFO functions are exemplified as dispositions "The category of modes include [sic.] dispositions (e.g., functions, capabilities, capacities, vulnerabilities, etc.)"

In truth, despite the limited attention given to functions in these upper-level ontologies, their authors have more extensively discussed functionality in various research papers. For instance, the creators of DOLCE propose viewing functions as specific types of events, while describing behaviors as 'qualifications of the participation relation', elucidating the specific manner in which a device participates in a particular event [34, 28, 76]. The authors of YAMATO delve deeper into their functional ontology concept in additional papers, including [141, 148, 194], where they reaffirm and elaborate on their assertion that functions are roles played by behaviors, with behaviors themselves being a specific class of processes (it is noteworthy that both YAMATO and FBRL were developed by the same research group).

The perspective of GFO's authors shares some similarities with that of FBRL. Although functions are not roles themselves, they are connected to 'functional items', which are roles played by entities realizing the function. Functions are complex entities, comprising a functional item, an identifying label, and descriptions of the initial and final states corresponding to the function execution, referred to as 'requirements' and 'goals', respectively [38, 40]. A typical example is the function 'to transport oxygen', linked to the functional item 'oxygen transporter', which can be performed by a red blood cell. A requirement for the realization of the function is the presence of oxygen in the red blood cell environment, and the goal is the presence of oxygen in the body cells requiring it for cellular respiration. Another resemblance to FBRL is the establishment of teleological relations between functions, akin to meta-functions. For instance, [38] mentions relations like 'support', 'enable', and 'prevent', depending on

[9]Note that processes exist as a category both in DOLCE and YAMATO, but they are different things: in YAMATO, processes exist wholly at each point in time, while in DOLCE this is not true since DOLCE-processes are a specialized form of perdurants.

whether the goal of the first function fulfills partially, completely, or is incompatible with the requirement of the second function (cf. with [142], which also employs terms like 'enable' and 'prevent', among others).

Coming to BFO, the notion of functions as specialized dispositions has been elaborated and defended by its creators [7, 232], while some philosophers, such as Jansen and Röhl [218, 133], have criticized this perspective. A comprehensive discussion of this debate is beyond the scope of this thesis, but it's worth noting that while BFO maintains functions as a subclass of dispositions, Jansen and Röhl argue that functions and dispositions are distinct but related categories.

Specifically, BFO defines functions as follows [232]:

> "f is a disposition & f exists in virtue of its bearer's physical make-up & this physical make-up is something that this bearer possesses because it came into being, either through evolution (in the case of natural biological entities) or through intentional design (in the case of artifacts), in order to realize processes of a certain sort".

This implies that when a function ceases to exist the physical make-up of its bearer changes as well (since this holds true for all dispositions in BFO). On the other hand, Jansen and Röhl argue that functions are "externally grounded". Another significant point of contention involves malfunctions, with Jansen asserting that malfunctioning entities possess a function but lack the means to realize it, for example, due to a loss of required dispositions. In contrast, Smith and colleagues assert that when an entity undergoes malfunction, it forfeits its function, either partially or entirely. The entity is then recognized as a member of its kind solely based on its history, if at all. For instance, a cancerous lung is not classified as a lung in this perspective. It's important to note that in both Smith's and Jensen's perspectives, functions and roles are assumed to be separate subclasses of realizable entities, because roles are considered accidental for their players, while functions are deemed essential for their players.

THe approach of the Industrial Data Ontology (IDO) is similar to BFO's. Such an ontology gives a very similar definition of functions "A 'capability' [useful disposition] which is the reason for existence of the bearer. The bearer's physical design (or evolved form, in the case of natural biological entities) is there in order to enable participation in activities of the specified kind, in a specified way" [37, github repository]. This is almost the same definition of BFO (in BFO functions are capabilities, which are dispositions, and IDO adopts this), however there is a key difference: while the end of a BFO-dispositions entails a change in the physical structure of the bearer, this does not hold for IDO-dipositions. Of course, this entails a significant difference of IDO-dispositions w.r.to BFO-dispositions that may make them escape Jensen's argument, but further discussion on this topic is beyond the scope of this thesis[10].

Neither YAMATO nor BFO commit to an axiomatization of functions (except for stating the place they occupy in the respective taxonomies). From this point of view,

[10]see the discussion on externally grounded dispositions in [243] for an example of work further analyzing this topic.

GFO is an exception, since its OWL version of GFO[11] explicitly relates functions to goals, requirements, and functional items. Additionally, in GFO functions can be realized not only by process-like[12] things (e.g. the act of transporting some goods) but also by entities existing fully at some time instant (the given example is the situation of a moth camouflaging on a tree)[13] [114].

**Classical desiderata for theories of function.** Finally, let us come to what philosophers considered the good properties that a theory of functions should have. We rephrase[14] a list by Rhöl and Jansen [218], which they re-elaborateed from Artiga [9], Houkes and Vermaas [122], and Burek [39]:

- Teleology[15]: the existence of the function bearer should be explained by the function itself.
- Restriction: the theory explains the duality between essential and accidental functions.
- Normativity[16]: functions can be performed well or badly according to some norm that is present in the function ascription.
- Malfunction: A thing can have a function even though it fails to perform according to the function. In particular, malfunctions and non-functionings should be accounted for.
- No Epiphenomenalism[17]: the fact that a thing has a function or not is explained (also) by the performance of the thing and not only by 'causally inert' facts

[11] `https://github.com/Onto-Med/GFO`, last accessed in January 2024.

[12] GFO divides entities not between endurants/continuants/objects and perdurants/occurrences/happenings like DOLCE, UFO, and BFO. Instead it divides entities into presentials and (GFO)-processes [114]. Presentials are things that are can be "entirely present at a timepoint" [114] and do not have a time extension, while GFO-processes do have a time extension.

[13] The entities existing fully at some time instant would be the *presentials* of GFO, which are, for GFO, different from continuants/endurants. Presentials also include the situation of a moth camouflaging.

[14] In particular, note that the couples of items Normativity and Malfunctions, and No Epiphenomenalism and Support are located into the same item in the original list, due to their tight connection.

[15] The word 'teleology' here stands for a nominalization of the adjective 'teleological' used before in the text. In this context, it indicates that the explanation of some entity should reference its purpose, goal, or, indeed, function. For instance, to say that a sheeps exists to provide wool is a teleological explanation the existance of sheeps, while to say that they exist because a certain evolutionary process took place is not.

[16] Normativity is the social phenomenon of designing certain things as good and others as bad. E.g. 'it is good for children to eat vegetables' is a normative statement.

[17] Epiphenomenalism is a term that suggests that there is some process that accidentally produces something else (the epiphenomenon). In this context, the epiphenomenon is the function and the process is some kind of selection process (natural selection being the prime example, but generalizations to, e.g., social processes are included). Epiphenomenalism is then the defect of etiological theories (i.e. theories pertaining to causal history, they were introduced in Section 2.1.5) of functions, for which functions do not need to be explained by the current performance of their bearer, but only by their past history.

such as the thing's history or decisions made by the thing's observers, users, manufacturers, or designers.

- Support: The structure of the thing should be involved in explaining the fact that the thing has a given function.
- Innovation: the theory makes possible to ascribe new functions to innovative artifacts or to newly evolved organisms.
- Continuants[18]: functions and their realizations should be differentiated.
- Domain bridging: the theory should encompass functions of different domains, e.g. both engineering and biological functions.
- Processes: the theory should allow for also processes to be bearers of functions.
- Decomposition: functional decomposition should be accounted for.

We will come back to this list in the oncoming chapters of this thesis, after having laid down our approach to functions (cf. Section 3.0.11).

### 2.1.7 Summary of approaches to functions

In conclusion, 'function' is a term with multiple meanings, and multiple approaches exist to deal with these. We do not carry out a schematic (e.g. tabular) listing of various meaning and approaches, since reviews doing that exist already. For example, the throughout work of Erden [70] sums up 18 proposed functional frameworks, and we have already mentioned some classification schemas in Section 2.1.5. More interestingly, it is natural to wonder how to handle these different meanings of function. Carrara and colleagues [45] suggest that there are essentially three possible ways: the revisionary strategy, which aims to identify one single concept of function and a corresponding definition; the overarching strategy, which aims to identify multiple concepts of function and recursively capturing their similarities until one single overarching concept of function is found that subsumes all others; and the descriptive strategy, which tries to pinpoint the main concepts that engineers use when speaking of functions and formalize them in a single framework.

Since it appears that the multiplicity of meanings is useful for engineers [45, 253], we will not attempt a revisionist strategy, but a more descriptive one, defining various 'sub-concepts' of function within a unifying framework. In particular, nor the term 'function' not 'engineering function' will be given a definition. This will be done in Chapter 3.

[18]Here the term 'continuant' is philosophical jargon that can be taken to be synonymous with DOLCE's endurants and antonym with occurrents/perdurants: a house is a continuant, the event of that house being on fire is a perdurant. This term is used to constrast functions with their realizations: the latter should be perdurants, the former continuants.

### 2.1.8 Some related concepts

A review of functionality in engineering cannot be thoroughly complete without mentioning some important related concepts. We briefly review some of these, whose ontological status is quite complex: behaviors, capabilities, capacities, and affordances.

#### 2.1.8.1 Behavior

As highlighted by the previous literature review, there is a widespread acknowledgment that functionality is contingent upon the agent's intention, exemplified by the designer's intent [145], also referred to as "design rationale" [47]. Consequently, functions are not entirely objective. It raises the question of what aspects of a function are genuinely objective. Typically, the response lies in behavior, defined as the actions of a device determined by physical laws.

Of course, the term "behavior" is also employed with ambiguity, lacking a universally shared definition. This ambiguity has prompted some authors to categorize the various uses of behavior in the engineering literature. For instance, Kitamura et al. [145] identify four types of device behavior[19]. The classification is based on whether the device alters the state of another device, its own state, or the state of one of its operands. Within the latter behavior type, two additional cases are distinguished: the operand can either remain in the same position or 'move' from the input to the output ports. The latter case is termed *B1 behavior*. For example, "a motor converting electrical energy to torque" and "the signal intensity increased by the amplifier" are instances of B1 behavior, while "the temperature in the room increased by the oven" is not B1 behavior, nor is "the turbine is rotating". Chandrasekaran and Josephson categorize six types of behaviors, primarily depending on their duration (instant, interval, or unspecified) and the values of the state variables during that time.

The relation between behavior and function is a reflection of the one between objectivity and intentionality, indeed a typical distinction is that behavior is what a device does, while function is what a device is for [153]. Others describe behavior as a sequence of state changes and functions as abstractions of behaviors with a goal in mind [246]. Another classical distinction is saying that functions and behaviors answer the questions "why?" and "how?", respectively. For example, Goel [86, 85] maintains that behaviors are generic transitions between system states that express how a purpose is achieved, while functions express that purpose itself. Goel's definition, though, in a typical case of conceptual opacity, does not completely clarify what the difference is, especially since they also say that functions are merely a subset of system output behaviors. Additionally, functions are, de facto, used as annotations of state transitions, together with physical laws, behaviors, and other explanations, and no clarification is given about why functions differ from other annotation types.

Kitamura's team, instead, defines a function as "a teleological interpretation of a B1 behavior under a goal". Then, they explain that 'interpretation' is to be understood in terms of the ontological concept of role. Therefore, the difference between functions and behaviors is the same as between roles and their players.

[19]Note that static behaviors such as supporting are explicitly excluded by Kitamura et al.

In the work by Sasajima et al. [223, 224], it is stated that function is composed of behavior plus additional teleological information called "functional topping". The functional topping allows distinguishing different functions that a device can execute. For instance, a heat exchanger could be devised either for heating or for cooling some environment or apparatus, and the functional topping in these two cases is different, though the device's behavior is the same. More specifically, given a component, the functional topping of that component's behavior is constituted by a few different aspects: the goal of the given component, its function type, a focus on a certain subset of input and/or output flows, and the (non-)attribution of necessity to the presence of a focused input or output flow. These aspects are somewhat complex to explain in detail and the interested reader can consult [223] for a fuller explanation. Here we just offer a brief exemplification: consider the example of the heat exchanger, assuming that it has two input and two output flows corresponding to the two different fluids that are being thermally coupled, considered as input flows before the thermal coupling and as output flows after. The heat exchanger goal would be made of state descriptions such as 'output coolant does not exceed 500°C'; its function type would be a choice of one of Keuneke's function types, e.g. 'toMaintain' [138]; the focused flows would be the two corresponding to the lower temperature flow if the function is 'to heat fluid', or to the two corresponding to the higher temperature flow if the function is 'to cool fluid'; finally if the function of the heat exchanger is also to being used to heat some further thing, then the thermal energy of the heated fluid at the output is labelled as necessary, while if the heat exchanger was being used to cool something, then the thermal energy of the cooled fluid at the input is labelled as not necessary.

In another perspective presented in [49], functions are considered as the sum of intentions and sets of behavioral constraints, where the latter are defined as causal relations between device states.

#### 2.1.8.2 Capabilities and capacities

'Capability' is another term that has a fast link with the concept of function. It is also frequently accompanied by the term 'capacity', especially in the context of manufacturing resource modeling [134, 222, 34]. These two terms are often employed with varying, and at times opposing, interpretations.

In the standard MANDATE [126], capability is defined as "The quality of being able to perform a given activity. [...] The capability is defined by a group of characteristics that describes functional aspects of manufacturing resources or system[s]". Furthermore, the standard says that, while capacity is a quantitative term indicating the "capability of a system, subsystem or resource to perform its expected function from a quantitative point of view", and capability is fundamentally a "functional and qualitative concept". For instance, a machine tool's capacity could be 'the quantity of workpieces the machine can process in an hour', whereas one of its capabilities could be 'the ability to drill a counterbore hole'. The same standard warns against reducing capacities as characteristics of capabilities and forbids reducing capabilities as characteristics of capacities. This is quite different from, say, Solano [231], for whom a

"Capacity is a Capability expressed in terms of amount of production".

In the literature, there is also a terminological confusion between capabilities and functions, as well as between capabilities and (manufacturing) processes. For example, the ReCaM project deliverables [212] state "a capability is a *functionality* of a resource, such as "milling", "drilling", "screwing" and so on" (emphasis is ours). While the MaRCO ontology[20] uses a "process-oriented definition of capabilities" to allow for "the matching of a product's processing requirements with the resource capabilities" [134]. For example, 'hammering' is a class representing a capability in the context of a capability taxonomy. That class is also a subclass of a different but homonymous class, this time representing a process and belonging to a taxonomy of manufacturing processes. Additionally, some names, like 'storing', align with traditional function names, while others are explicitly labeled as functions, such as 'RivetingTool*Function*' (emphasis is ours). An analogous ambiguity arises when Köcher et al. [157, 156] adopt the standard VDI 2860 [252] to construct a capability taxonomy. That standard contains a vocabulary of actions that machines carry out when manipulating products, such as 'guide', 'rotate', etc. However, the standard specifically refers to these actions as "Teilfunktionen des Handhabens" [partial functions for handling]. Consequently, Köcher uses the term "capabilities" for what the VDI standard identifies as "functions" and is not clear what is the difference, if any.

Various ontological analyses tried to resolve the lack of consensus about the meaning of capabilities and capacities. For instance, the aforementioned MaRCO ontology [134] (which is not aligned with a top-level ontology) defines capability as "the inherent ability of a resource or system to perform change in the properties of materials, parts or assemblies, or to perform activities which may be required to change the properties of material, parts or assemblies, in order to produce tangible products". In MaRCO, capabilities are either simple or combined: combined capabilities are made of simple capabilities. For instance, 'joining parts by blind riveting' is a combined capability that requires simple capabilities of basic capabilities such as 'moving' and 'pulling'. Furthermore, capabilities are characterized by specific parameters, such as velocity and acceleration, which describe the 'moving' capability. These parameters are used to differentiate the different degrees that various resources possess the same capability.

A different research line expanded the top-level ontology BFO, adding capabilities as a subclass of dispositions [222, 155, 183]. Sarkar and Šormaz [222] belong to this research line, and they define capabilities as dispositions constraining the functionality of resources by describing "the extent by which some function $f$, inhering in artifact $a$, is realized when the $a$ participates in some process $p$'. Merrel et al. [183] instead do not mention functions in the definition of capabilities: for them, capabilities are those dispositions "in whose realization someone (or some organism, or some group of organisms) has or had an interest". Given this definition, they claim that functions are a subclass of capabilities: functions are those capabilities that belong to an entity either by design (if the entity is an engineered artifact) or by natural selection (if the entity is an organism). Similarly, works from UFO's authors take a similar stance: they "understand that capabilities should be interpreted as types of dispositions (disposition

[20] Available at `https://resourcedescription.rd.tuni.fi/ontology/resourceModel.owl`.

universals in UFO)" [12].

A third approach is instead based on the top-level ontology DOLCE [34, 239, 54]. Here, capabilities are (DOLCE-)qualities, this entails that they always inhere in a bearer on whom they depend. Precisely, capabilities are (DOLCE-)qualities inhering in resources and describing their ability to participate in the execution of a function, by contributing to achieve the final state brought about by the function execution [34]. Capacities are also qualities of resources, but they describe the extent the resource can execute a function. Take, for the sake of example, a container: one of its capabilities is 'storing', which is characterized by its volume, which is a capacity. This approach is similar to the one of Kitamura and colleagues [149]: they talk about 'capacity functions' that are properties ascribed to devices that indicate the capacity of the device to perform a function. Kitamura's capacity functions and the capabilities of the DOLCE-based approach are analogous because they are both classes of entities inhering resources: "a capacity function is a thing (property) that a device has (or is ascribed to a device), exists inside the device, and is dependent on the device" [149]; because they both represent the "possibility to perform an actual function" [149]; and because they are both based on "a set of appropriate properties (e.g., physical attributes, structure, geometry and material)" [149]. Interestingly, Kitamura and colleagues [152] distinguish between capacity functions and capability functions, the difference being that capability functions are properties of entities signifying their ability to perform an activity that has no effect on others (e.g. a person has the capability function of walking); moreover they introduce 'functional properties', which are qualities inhering in entities that (partly) explain the entities' capacity functions. Therefore, Kitamura-functional properties are more similar to what other authors call capacities, while Kitamura-capacity functions and Kitamura-capabilities are different disjointed subsets of 'typical' capabilities.

As mentioned above, capabilities and capacities are frequently employed in resource modeling [134, 222, 126, 231, 221]. This is because the main goal of resource modeling is, at least in the context of the manufacturing system design [240], to make it possible to match the various requirements to manufacture products with corresponding resources. Capabilities and capacities enter the fray here, since the ability of a resource to implement a requirement is based on its capabilities/capacities. For this reason, it has been argued that it is crucial to model capabilities and capacities in conjunction with process and production plans [239], because process plans provide specific instructions on the operations necessary to transform a workpiece into its intended form, while production plans allocate appropriate resources for executing the required operations.

This is the approach of Sanfilippo and colleagues [221], which compare three different conceptualizations of manufacturing resources using first-order logic. These conceptualizations highlight the connection between resources, and their capabilities and capacities; and activity occurrences, activity requirements, and manufacturing goals. In addition, a fourth approach that unifies these conceptualizations is suggested, and its application in a use case is compared with a preexisting model [247] based on the Industry Foundation Classes standard [167]. The main characteristics of this approach are:

- its alignment with DOLCE.

- the conceptualization of manufacturing activities, along with specifications of technical artifacts, capabilities, and states, are conceptualized as DOLCE-descriptions. This allows experts to explicitly represent the process plans or design models that activities and artifacts must adhere to.

- The reuse of the Process Specification Language (PSL, [89]) ontology. In particular, activities and activity occurrences are distinguished: the former represents the types of activities and the latter represents the processes occurring in time.

- The conceptualization of manufacturing resources are objects that play specific roles. Precisely they are the objects that meet the requirements for the execution of a manufacturing activity. For example, a machine tool becomes a manufacturing resource when it complies with the specifications of a specific activity, such as being able to drill a hole with a given depth, diameter, and precision in a given material.

#### 2.1.8.3 Affordances

The last engineering concept tightly linked to functions that we analyze in this thesis is the notion of affordance. Gibson, a psychologist, first introduced affordances in 1979 [82], and they were later popularized in the context of engineering design by Maier and Fadel in a series of works ([172, 171], among others). Those works described affordances as a shift in paradigm from a function-based design to interaction-based design, with a focus on all potential user interactions with a product, including those unintended by the designer and not strictly related to the product's function. Gibson originally defined affordances, using an animal interacting in its environment as an example, as "what it [the environment] offers the animal, what it provides or furnishes, either for good or ill" [82]. In other words, they represent potential behaviors that the environment, or a part of it, allows an agent to perform. Gibson's original example was a flat surface that affords footing to an animal, while other examples are a pool full of water, which affords a person the ability to swim, or a briefcase, which affords a person the opportunity to be grabbed. Thus, one could say that 'swimmability' and 'grabbability' are affordances of the pool and the briefcase, respectively. In an engineering context, Gibson's definition would shift to "the set of all the behaviors that an artifact allows a user to perform" [36], however, this definition is not universally accepted. The concept is not surprisingly difficult to pinpoint, to the extent that some authors have called for their abandonment [197]. Currently, there is an ongoing debate in disciplines that utilize the concept of affordances regarding their meaning [36].

**Conclusion** In the previous sections, we analyzed the literature about engineering functions, various related concepts, and engineering systems dealing with those. These various topics, including the connection between functions and malfunctions, the distinction between functions and behaviors, and the methodology for decomposing a function, have been approached differently by these systems. Overall, there doesn't appear

to be a consistent and shared perspective emerging, indeed the discussion regarding the use of these systems and functional vocabularies is not settled yet, with various research paths currently being investigated. This encompasses topics such as simulating system behavior derived from a functional model [162] or formally and automatically creating the functional model itself [84, 161].

Additionally, frameworks utilized for modeling functions in engineering often lack grounding in, or alignment with, ontological theories. We argue that connecting these frameworks and ontological theories would be beneficial, given the prevalence of such engineering systems and the importance of clarifying underlying terminological and conceptual issues. Moreover, engineers frequently eschew the use of formal languages like first-order logic or OWL[21]. Instead, they prefer natural language or pseudocode to produce informal descriptions[22].

Given this, in this thesis, we set about to outline and formalize a unifying conceptual framework encompassing engineering functionalities and related concepts. To do this, in Chapter 3, we shall use DOLCE as a guide, leveraging, and extensively modifying and extending previous works in this direction [55, 54, 239, 221].

## 2.2 State of the Art of Malfunction Modelling

### 2.2.1 Fault analysis from an ontological point of view

The are many terms employed by engineers to indicate that within an engineered system something is not as it should be. In this chapter, the most common among these are reviewed along with their use and associated methodologies. Maintaining the ontological viewpoint of this thesis, we will sometimes intermix the engineering literature with the philosophical one, especially for some terms (e.g. 'malfunction') that are more typical in the latter.

**Failure** The main term used to indicate that something in an engineered system is not as it should be is probably 'failure'. This term has a wide range of meanings: it may stand (at least) for the "condition in which an engineering member exhibits plastic deformation" [180, p.16] or for the "loss of ability to perform as required "[124, 192-03-01]. Many other definitions of failure can be found in the literature, and many of those cannot be reduced to one another. Conveniently, Del Frate carried out a detailed analysis, collecting more than 50 definitions from the literature [63] and clustered them into three general, independent types [62]:

- **Function-based failure**: the termination of the ability of an item to perform a required function [*due to an internal state*]. (this is the same definition as in the

[21]It's worth noting that OWL was first published in 2004, while many functional modeling works predate this.

[22]Exceptions are, for example, Kitamura et al. [141] and Yang et al. [266], which present an implementation of an industrially deployed version of FBRL called SOFAST, and an OWL and SWRL formalization of the Functional Basis, respectively.

IEC vocabulary [124], we added the part in square brackets as it is an important part of the intended meaning).

- **Specification-based failure**: the termination of the ability of an item to perform as specified provided it has been operated under the stated operational environment for which it is designed.

- **Material-based failure**: any permanent change in the values of geometrical or physicochemical properties of the materials of an item which (i) renders the item unable to perform as specified or (ii) increases substantially the likelihood that the item will become unable to perform as specified.

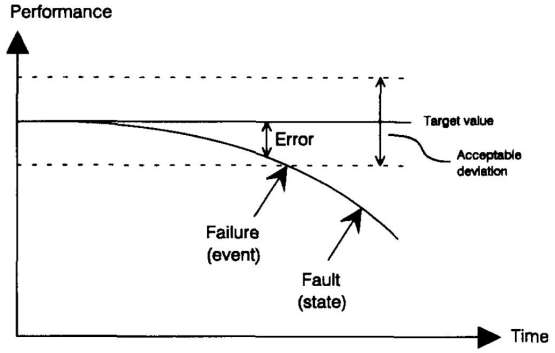
Del Frate then presents a definition, which he argues possesses favorable properties, particularly for interdisciplinary application: "Failure is the inability of an engineering process, product, service or system to meet the design team's goals for which it has been developed" [63]. Notice that the distinction between the first and the second definitions is twofold: the specification-based definition imposes more restrictions compared to the function-based one. Specifically, failures caused by operators' improper use of equipment are not considered specification-based failures. Furthermore, an item may still perform its function despite not adhering to design specifications. For instance, a pipe could still transfer fluid even if its diameter is less than the specified value due to corrosion.

**Fault** Usually, with some exceptions, a failure is typically referred to as an event, whereas a fault is described as a state that arises from a failure[23]. This is done by, e.g., Del Frate, which individuates a corresponding fault concept for each of his three concepts of failure [63], by the IEC vocabulary [124], and by Rausand and Øien [211] (see also Figure 2.6a).

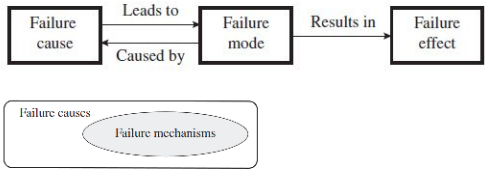
Other authors propose the opposite interpretation: for Avizienis the results of faults are failures, by means of an error, which is the deviation between the expected and actual states of a system [10]. This interpretation and the previous ones, though they appear as opposites, may not need strictly irreconcilable (cf. Section 4.1.1), however, it is inevitable that their coexistence will lead to confusion. Vesely attributes yet another meaning to the terms [260]: in this case, failure has a similar meaning to function-based failure, while the term fault is a strictly general case of failure (a hypernym) that also includes the scenario in which the incapacity to execute is not due to an internal state and is instead the result of a malfunction in a distinct component. The distinction between failure events and fault states is crucial when conducting failure analysis since it "describes the borderline between what is a failure and what is not" [211].

The term 'down state' (sometimes also 'failure state' [265]) [1] is another term closely associated to faults and failures. In the EN 13306 standard, a down state is the "state

[23] In Section 1.1, we provided further explanation on the distinction between events and states. Here, we emphasize that a failure event can occur suddenly (e.g., the instance when a shaft ruptures), while a fault state can persist for an undetermined duration (e.g., a fractured shaft). Also note that the DOLCE's interpretation of states and events is not the unique one in applied ontology.




(a) A diagram taken from Rausand work [211], which shows the difference between failures and faults. 'Error' is also present, with the same meaning as in IEC vocabulary: "discrepancy between a computed, observed or measured value or condition, and the true, specified or theoretically correct value or condition"




(b) One possible understanding of the causal relations among failure mechanisms, modes, and effects. Taken from [210]. Precisely, the top half of the figure, constituted by three boxes and arrows is Figure 3.5 at page 62 of [210], while the bottom half is Figure 3.12 at page 71 of the same reference. The caption of the latter recites '[f]ailure causes and mechanisms. A failure mechanism is a specific type of failure cause".

Figure 2.6

of an item being unable to perform a required function due to preventive maintenance or a fault". Therefore, down states appear to be strictly more general than faults: down states include the case that an item is unable to perform due to a scheduled preventive maintenance shutdown. EN 13306 also introduces the terms 'disabled state' (or 'outage') and 'external disabled state'. The former is defined as "state of an item characterised by its inability to perform a required function, for any reason", while the latter is the "subset of the disabled state when the item is in an up state [i.e. is able to perform the required functions under the correct conditions], but lacks required external resources or is disabled due to planned actions other than maintenance". Since external disabled states are a subset of disabled states and their complement is the set of down states, down states are also called 'internal disabled states' in EN 13306. A summary of these terms is shown in Figure 2.7.

Failure states are again mentioned in [67], however this time they appear to be similar to Del Frate's faults, because they are 'brought about' by a failure occurrence (in particular, this makes them incompatible with the failure states of [265]). For the rest the work in [67] is somewhat similar to Avizienis', with the noticeable exception that errors in the former ('Erroneous User Action') do not correspond to Avizienis' errors but instead to Avizienis' 'Human Fault'.

**Malfunction** The term 'malfunction' is frequently used as a synonym for failure or fault in engineering: in the IEC vocabulary it is a "situation for which the electrical equipment does not perform the intended function due to a variety of reasons [...]".

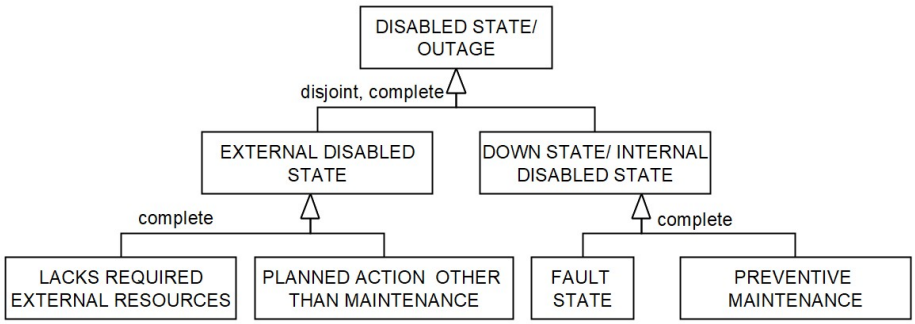



Figure 2.7: Taxonomy of some of the states described in the sandard EN 13306, with the top one being 'disabled state'. All child sets are complete and the top one is also disjoint.

Applying IEC's definitions, malfunction then means 'being at fault'. In engineering, malfunction is perhaps a less used variation of these two terms, after all there exists an engineering journal titled 'Engineering Failure Analysis', while none other of the 414 engineering journals listed on Scimago[24] contains the word malfunction. Instead, in philosophy and applied ontology, malfunctions, referred to as such, have been the focus of various analyses. For instance, Baker [13] provides a definition of malfunction that bears resemblance to the concept of function-based failure by Del Frate, and posits that malfunctions, although defined by the negation of functional performance, hold the same ontological significance as artifacts and functions: "There are no technical artefacts without functions; there are no functions without the possibility of malfunction".

Jansen [133] draws attention to the distinction between malfunction and malfunctioning, which is comparable to the distinction between function (a property) and functioning (an occurrence). According to Jansen, "a material object x has a *malfunction* at t with respect to the function to F-in-situation-S if and only if x has the function to F-in-situation-S but would not F in S because, at t, it lacks the disposition to F-in-situation-S". On the other hand, "a material object x is *malfunctioning* at t with respect to the function to F-in-situation-S if and only if, at t, x is in situation S but does not F". Based on this definition, Jansen argues that functions are not dependent on or inherent in dispositions. This conclusion is supported by the existence of malfunctions, which involve the presence of a function without "the matching disposition". A similar perspective can be found in Mizoguchi et al. [192], where it is suggested that malfunction can only be explained if functions, which are "roles played by behaviors in a function context", do not inhere in objects. Vieu and Borgo also have a similar view [261], where malfunctioning occurs if an artifact does not have all the capacities (abilities) required from it by somebody. Spear instead opposes such an approach [232], using the BFO upper-ontology to state that (i) functions inhere into objects and (ii) an object's life ends if it meets a (complete) malfunction.

[24] `https://www.scimagojr.com/journalrank.php?category=2201`, last accessed in November 2022.

**Failure mode, failure mechanism, and failure effect** Other common terms in this context, which are linked to 'failure', are failure mode, failure effect, and failure mechanism. These terms are most often encountered when dealing with FMEA (Failure Mode and Effect Analysis) since that methodology explicitly gives their definition and prescribes to collect instances of such terms during the analysis [186]. In FMEA, failure mode is defined as "the manner by which a failure is observed. Generally describes the way the failure occurs and its impact on equipment operation." [186]. This is quite different from the laconic definition present in the IEC vocabulary: "manner in which failure occurs", and from the definition given, e.g., in [69]: "The consequence of the mechanism through which the failure occurs, i.e., short, open, fracture, excessive wear".

Irrespective of their definition, reliability engineers compile tables filled with comprehensive examples of failure modes, such as [213, 69, 71, 225]. Examples of failure modes found in these tables are, for an actuator, 'leaking', 'seized', 'worn', 'contaminated'; for a circuit breaker 'opens without stimuli', 'does not open' [213]; for electric generators 'abnormal instrument reading', 'breakdown', 'external leakage', 'fail to start on demand', 'vibration', ... [225]; for a pinion 'spalled' (i.e., a portion of the gear surface flaked away), for a shaft 'snap ring not installed properly causes sun gear to interfere with plate' [71]. Examining these examples, we see that in practice things quite different from each other can be failure modes, such as failures to execute the required function ('fail to open'), or conditions that may cause failures in the future ('worn', 'vibration').

There are, at least, two common ways in which failure modes are defined: they may be specific parts of a causal chain, together with 'failure effects' and 'failure mechanisms' (cf. [69], in which failure modes are 'consequences', or Figure 2.6b). In these causal chains, failure mechanisms are processes that lead to failure [124, 210], which itself has the failure mode as a consequence, which, in turn, causes a 'failure effect'. Therefore, in this view, failure modes are intermediate elements of causal chains started by failure mechanisms and producing failure effects.

The alternative perspective characterizes failure modes as the immediately perceivable symptoms of a malfunction (e.g., "a failure mode is a manifestation of the failure as seen from the outside" [211], "The effect by which a failure is observed on the failed unit" [225, p. 41]). According to this viewpoint, the crucial aspect of failure modes is their ease of observation. Consequently, a failure mode of a valve can be "internal leakage", whereas "wear of the valve seal" is not, since the latter can only be detected by disassembling the valve (see [211]).

Failure modes are sometimes associated with some attributes [186, 25], [211][25], we collect some of these in Table 2.1. These attributes are also employed for categorizing failure (modes) into taxonomies based on the attribute values they possess [211]. Perhaps bafflingly, these attributes, which could be reasonably called modes-of-failure, are often not used identify to failure modes.

Perhaps surprisingly, these attributes, which could reasonably be called modes-of-failure, are often not used to identify failure modes. For example, a failure mode identified by the attributes of Table 2.1 could be 'extended-partial-gradual failure' or 'sudden,

[25]The latter mentioned above associates the attributes directly with failures instead of failure modes.

high-criticality leak'. Instead, the terms typically associated with failure modes, such as 'abnormal instrument reading', 'breakdown', 'external leakage', and others, are not always directly related to the attributes of Table 2.1, at least not terminologically.

| ATTRIBUTE NAME | DESCRIPTION | EXAMPLE VALUES |
|---|---|---|
| Persistence | The time extension during the item is at fault. Intermittent means that the fault will cease spontaneously (and possibly return). Extended means that to interrupt the fault a repair is necessary | Intermittent, Extended |
| Severity (or Degree) | The degree to which the affected function is altered | Complete, Partial |
| Suddenness | The speed with the failure mechanism produces the failure. Gradual (sudden) means that there is (not) enough time to forecast the failure. | Gradual, Sudden |
| Criticality | The extension of the negative effects caused by the failure, taking into account their probability of occurrence | High, Medium, Low, Negligible |
| Phase of occurrence | The life-cycle phase during which the failure happens | Design, Assembly, Transportation, Installation, Operation |

Table 2.1: Attributes most frequently associated with failure modes, present in the engineering literature at least since a work of Blache and Shrivastava [25]. The terms in the table are not standard in the literature, except for 'criticality', which has seen widespread use since at least from 1949 [186]. Note that complete and sudden failures are sometimes referred to as 'catastrophic'.

**Cause, root cause, immediate cause, and other causes** Causality is critical in the analysis of failures: one must determine the cause of a failure to understand

how to prevent a similar failure from reoccurring and/or to assign blame for the failure. Therefore, a rich terminology for causes and effects was developed in the related literature.

The most common expression is simply 'failure cause'. For the IEC vocabulary it is the "set of circumstances that leads to failure" [124], but many other definitions exist. Moreover, specialized terms (hyponyms) are used to individuate more precise types of causes, such as 'direct' and 'root' causes. 'Direct' or 'proximate' [210] or 'immediate' [260] causes are those causes that are considered as adjacent to the failure in both temporal and spatial terms, and are obvious to determine. 'Basic' [260] or 'root' [210] causes, on the other hand, may have occurred a considerable time before the examined failure and are considered to provide a more comprehensive explanation for the failure analysis[26]. Note that, for the view for which failure modes are intermediate elements of causal chains started by failure mechanisms and producing failure effects (cf. the previous section), failure mechanisms must also be failure causes, in some sense.

Root causes are often the most analyzed type of causes, since it is argued that their individuation may prevent similar failures from reoccurring. Del Frate [64] surveyed definitions of the root cause found in the literature, dividing them into two main types: those that espouse a forward-looking view, which highlights the effects that correcting the root cause will have on the system, and those that have a backward-looking view, which emphasizes the role of the root cause in leading to the failure. Important difficulties that these approaches must solve are separating the search for a correction from a search for blame assignment (which is an undesirable side-effect, in some cases), and identifying the root cause in place of keep looking for further causes, respectively.

Ultimately, Del Frate tries to reconcile these two approaches by proposing a reconciled definition of root causes: "that element of the factors and causes that is, if corrected, the most likely to prevent failure phenomena similar to the one under investigation from happening again".

Another point of discussion is the interplay between causal and mereological relations: whenever a system is divided into hierarchical layers, one can consider how causation propagates between the layers. One typical case is the subdivision of a system into different functional levels, such as 'system', 'subsystem', 'equipment', and 'part', taken from ISO 14224 [127]. In that case, a fault from, say, a piece of equipment may cause a failure to a subsystem, and so on. Following the causal propagation of events among hierarchical layers, Rausand and Øien [211] suggested that the triple made from failure cause, mode, and effect can be instantiated by different triples of events depending on the layer that one is concerned with. That is, they suggest that what was the failure mode on, say, the equipment level is the failure cause on the subsystem level, and so on. See Figure 2.8 for a clarification.

Koji et al. [158] also investigate the propagation of a malfunction in a low-level function within a functional decomposition of a system to higher-level functions. For instance, the in left side of Figure 2.9 a causation tree explains why a wire, used to

[26]This is not a rule, as, for instance, Matthews [180] employs the adjective 'proximate' (meaning proximate *in efficiency* concerning other probable causes) in a manner akin to 'root ' cause and makes reference to the use of such an adjective from English law.

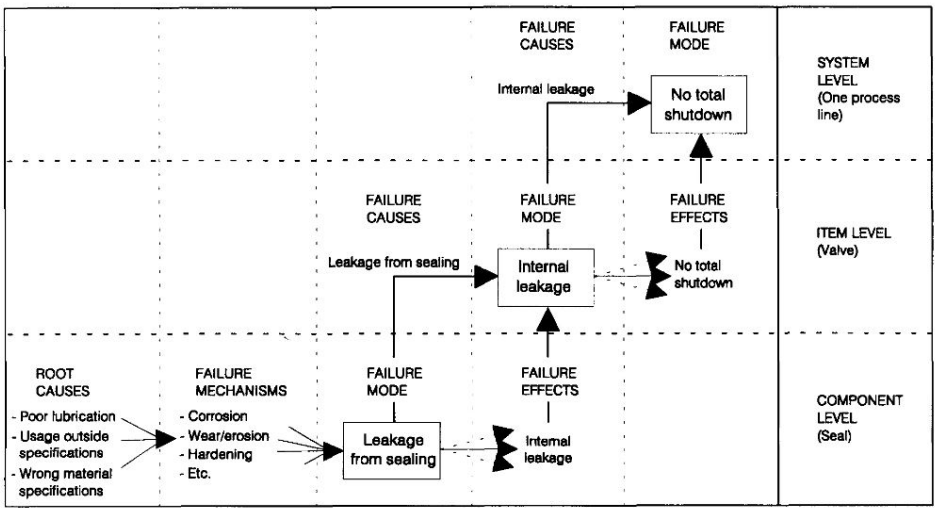



Figure 2.8: The causal propagation of a failure throughout the different hierarchical levels of a system. Taken from [211]

split silicon ingots, broke: the heat caused by friction weakened the wire. Moving to the right side, the execution of a low-level function ('application of frictional force to the ingots') is prevented by the wire rupture and this causes cascading malfunctionings of upper-level functions, up to the interruption of the main function of the wire saw itself: splitting ingots.

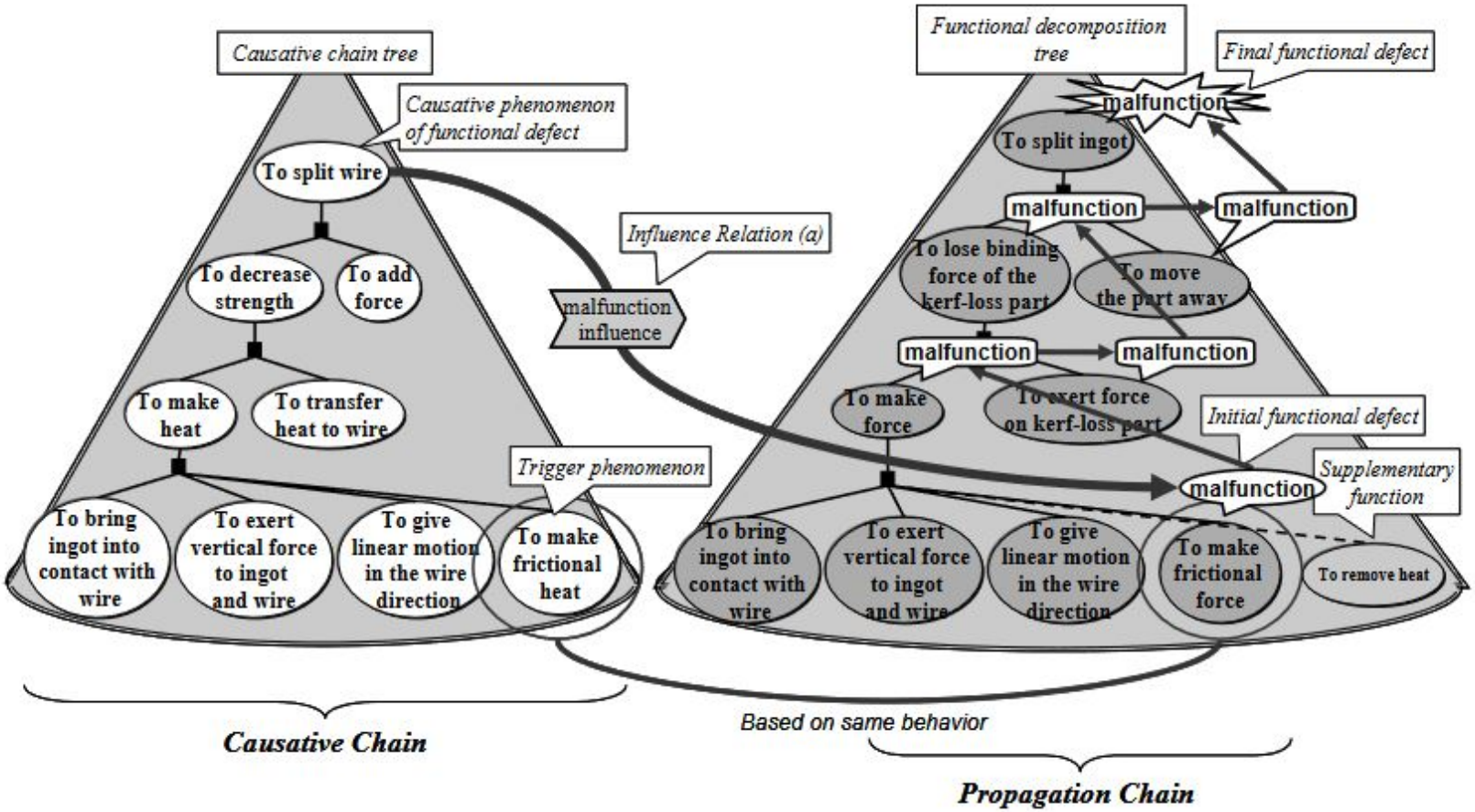



Figure 2.9: Example of the propagation of a malfunction from low-level to upper-level functions in a functional decomposition of an engineered system. It should be understood as follows: the picture contains two trees. The one on the right is a functional decomposition of a wire-saw machine for silicon ingots. The one on the left is a decomposition of a malfunction occurrence, which is done in the same way as the functional decomposition on the right: the whole malfunction is taken as a goal to be achieved, and then recursively decomposed into sub-events/goals. The malfunction (the wire splits) is caused by the excessive heat generated by the normal operation of the machine. When the wire splits, a causal chain starts that goes up the functional decomposition on the right, affecting various functions, until the top-function 'to split ingot' is debilitated.

### 2.2.2 Engineering methodologies for diagnosis, recording, or analysis of failure and related occurrences

In this section, we list some of the most common engineering methodologies that are used to analyze failures, before comparing them and highlighting some of their criticalities.

**Failure Modes and Effects Analysis (FMEA)** The FMEA methodology aims to identify potential failures within a system, prioritize them depending on their effects, and determine corrective or preventive actions [186, 69]. It was originally developed by the U.S. Department of Defense. Typical steps are:

- Divide the system under analysis into its distinct components and/or functions, for instance utilizing a functional block diagram. Note that instead of the components of a material system one could consider a set of relevant processes, in that case one talks of process FMEA.
- For every component and function, identify the specific ways in which they fail their purpose (the 'Failure Modes' in FMEA), as well as the resulting impacts on the overall system (the failure 'Effects'). Then, determine appropriate corrective actions to address these failures. Note that sometimes one links modes to only components, sometimes to only functions, sometimes to couples <component,function>.
- If the FMEA activity incorporates criticality analysis[27], then, one calculates the failure rate, relative frequency of the failure mode, and the probability of system failure for each failure mode of a particular item. These three values are then multiplied to obtain the criticality index of the failure mode for the specific item.
- After having collected the previous information, risk mitigation strategies should be carried out. One typical example is modifying the system design to use more reliable components or introduce redundancies.

FMEA is usually applied during design to inform the design process itself, but it may also be applied to an already-developed product, to, for example, get a better understanding of the product criticalities and inform maintenance strategies.

**Failure Reporting, Analysis, and Corrective Action System (FRACAS)** FRACAS, similarly to FMEA, aims to record, analyze, and correct failures; and it was also developed by the U.S. Department of Defense [185]. Differently from FMEA, the FRACAS methodology analyzes actually-occurred failures during the use of a system, focusing on managing and learning from them. FRACAS basic steps are:

- Record instances of failures using standard forms.

[27] So that FMEA expands to Failure Modes, Effects, and Criticality Analysis (FMECA).

- Investigate the failures to identify their cause(s). This step may require more or less detailed failure analysis methodologies, such as any methodology for root cause analysis mentioned in an oncoming section.

- Identify, document, and implement corrective measures to prevent or reduce the chances of recurrence of the failures.

These steps should be carried out continuously, in a closed loop. For this reason this methodology is also called closed-loop analysis and correction action process [111]. This methodology can also work in tandem with FMEA[28]: an occurred failure mode can either have been detected during FMEA already or not. If it has been, then possible corrective actions should already have been identified; if not, the FMEA should be updated with the new possible failure mode and the relative corrective actions.

**Fault Tree Analysis (FTA)** The FTA methodology represents the sequences of events culminating in a system failure [260]. It represents potential failure causes using a tree structure that shows cause-and-effect relationships. These relationships can occur in parallel (when multiple events contribute to an effect simultaneously) or in series (when events happen sequentially, forming a chain of causes and effects). Parallel contributions are represented as logical gates (e.g.m two events merging into an 'OR' gate will cause the event at the output of the gate if both or only one among them happens). The events in the tree structure are categorized into different types: basic events (simple events that don't require further explanation and may refer to a single component), undeveloped events (events that could be explained further but are not convenient to do so), and intermediate events (non-leaf nodes of the FTA tree). The analysis continues until the events reach the desired level of detail, such as when only undeveloped or basic events remain. Each FTA focuses on a specific system failure and should be repeated for every relevant failure.

For any given FTA tree minimal cut sets can be identified. These sets consist of leaf-level events that have the potential to cause the top failure if they occur simultaneously. By conducting a qualitative analysis of the failure causes, corrective action can be taken to prevent them during the design. If the probabilities/distributions of the leaf-level events are estimated, along with knowledge about possible dependencies between events, a quantitative probabilistic/statistical analysis can be performed. This analysis can, for example, rank the minimal cut sets based on their probability of occurrence or determine the rate at which the system failure occurs.

**Human Factors Analysis and Classification System (HFACS)** The HFACS is based on four taxonomies of human errors [230]: One for 'unsafe acts of operators' (such as decision errors, perception errors, routine violations, ...), another for 'preconditions for unsafe acts' (fatigue, ...), another one for 'unsafe supervision' (e.g. bad training), and a last one for 'organisational influences' (say, no appropriate funding for training).

[28]Of course, all the methodologies described in this section can be carried out in synergetic groups. The coupling of FMEA and FRACAS just happens to be one of the most common.

The basic idea is that any occurrence of human error can be attributed to a failure within a broader organizational context (e.g., the operator made an incorrect decision due to inadequate documentation). Specifically, any human error should be in, or have origins in, each of the four organizational levels mentioned earlier.

This framework can systematically identify (latent) failures within an organization that (may cause) have caused an accident, making it possible to amend these by appropriate interventions.

**Root Cause Analysis (RCA)** RCA analyzes past occurrences of failures to prevent or mitigate reoccurrences. In particular, the main goal is the identification of the 'root' cause of such occurrences. RCA is a single coherent methodology, but rather a collection of various guidelines and techniques [17, 214], [207, p. 1] are used to make RCA into a structured activity.

One of these methodologies is the '5 whys' [228]. Such a methodology recursively identifies the cause of a failure at least five times. By continuously digging deeper into the causes, the aim is to avoid getting stuck with superficial explanations of the initial failure.

Another methodology is the '6M': this time investigators are required to divide the cause(s) of an event into 6 distinct categories: 'Man', 'Machine', 'Method', 'Measure', 'Mother-Nature' e 'Material'[29]. For each type, one or more causes should be listed, and each cause can be accompanied by a dedicated further analysis. For instance, a workplace accident could have been concomitantly caused by the distraction of the personnel ('Man'), by a defect in a tool supplied by a third party ('Material'), and by a lack of implementation of workplace safety norms ('Method'). A fishbone-, or Ishikawa-, diagram [125] may be used as a graphical tool to assist this activity.

Overall, The various methodologies mentioned above can be divided into inductive (which proceed from particular causes to system-level effects, such as FMEA, FRACAS, HFACS) versus abductive[30] (which, oppositely, go from the effects to the causes, such as FTA, RCA); a-priori (which can start before the failure takes place, such as FMEA, FTA) versus a-posteriori (which focus on a failure already occurred, such as FRACAS, RCA, HFACS); they can also direct their attention towards human errors (HFACS) or not. Additionally, certain methodologies may have initially been used in specific engineering domains (e.g. HFACS in aviation and FRACAS in the military) and later extended to other disciplines.

Whatever the category they belong to, these methodologies are known to be problematic, for various reasons. For one, causality is a challenging subject, and these methodologies do little to alleviate the inevitable terminological and conceptual issues that arise [53]. Consequently, terms such as 'Failure Mode,' 'Failure Cause,' 'Failure Effect,' 'Fault,' etc., which are the fundamental concepts of these methodologies, are used without providing clear definitions, resulting in inconsistent usage.

[29]Sometimes the number of 'M's varies. Same for the names of the categories.

[30]Note that we are following Vesely with this division, who actually he speaks of inductive and deductive metodologies [260, I-8]. However, the term abductive seems more fitting.

These approaches also cannot be used to consistently and systematically investigate causation: regarding the 5 whys, for example, the number 5 is arbitrary and may not be suitable for complex problems, the questions ("why did this happen?") may not provide helpful insights, and the answers may lack consistency [228]. Moreover, a-priori methods enable systematic enumeration of failures but do not address causation, since the relationships between a failure mode, its cause, and its effect are left with a considerable degree of arbitrariness. While inductive methods are limited to addressing single point failures (with the exception of very small systems, as seen in the Double Failure Matrix analysis [260, p. II-5]), as the number of possible failure combinations would become unmanageable. Deductive methods, instead, may allow for the analysis of multiple causes (e.g., FTA, 6M, HFACS) or not (5 whys).

Storing the knowledge generated by these methodologies in databases that are easily accessible, preferably in an interdisciplinary manner, also poses challenges, particularly for a-posteriori methods (FMEA and FTA results have a convenient-to-store structure: tables and trees, respectively). After all, many RCA techniques, such as 5 whys or 6M, aim to facilitate brainstorming, not effective organization of data in databases or their use in knowledge-retrieval systems.

Finally, significant sociological issues exist. Among these issues are the 'political' implications associated with assigning blame, as blame is closely tied to any causal analysis of pertinent failures. Such implications may impede the analysis process. Another issue is the challenge of taking effective action following a root cause analysis, often due to organizational inertia [207].

The discussion carried out in this section will inform Chapter 4, where we will develop a formal ontology of malfunctions. In the next chapter we will finally address the core of our proposal, by studying engineering functions, through the lenses of applied ontology.

# Chapter 3

# An Ontology of Functions in Engineering Systems

Here we propose a formal, overarching DOLCE-guided conceptualisation of functions, informed by the review in Chapter 2, with the goal of offering a a unifying framework for (engineering) function modelling. Note that the content of this chapter is based on previous works in this direction ([30, 55, 54, 52, 56]), but most extensively modifies and extends them with novel elements and results.

### 3.0.1 Motivation

As observed in Section 2.1.6, the term 'function' carries ambiguity even for ontologists. Furthermore, it is generally not regarded as a top-level concept, receiving limited attention in top-level ontologies. To the best of our knowledge, the ontology community has yet to produce a shared middle or domain-level ontology specifically addressing functions[1].

Considering the current state of affairs, an ontological analysis elucidating the realm of functionality would be highly beneficial. This thesis endeavors to contribute towards a comprehensive ontological framework for functionality, at least in the engineering domain. While it may be acknowledged that the ambiguity in function terminology is considered essential and rational by engineers [253, 45], we contend that formalization is imperative for ensuring interoperability and highlighting distinctions among potential approaches. Furthermore, such formalization proves valuable, if not essential, for the development of applications that rely on functional reasoning. In any case, we consider seriously the presence of this 'strategic' ambiguity, and, for this reason, we opt for a descriptive approach in the sense of Carrara and colleagues [45], that is, we will isolate and describe various sub-concepts of function in a unique framework.

We are especially interested in answering the following research questions (RQ):

**RQ1** How can the wide range of 'function' concepts be ontologically differentiated and explained?

[1] A possible taxonomy of such an ontology was proposed by Borgo et al. in [34] and [29].

**RQ2** How can the functional decomposition of a system be explained and formalized?

**RQ3** What are capabilities and capacities, what is the difference between them (if any), how do they relate to functionality?

### 3.0.2 Behaviors, systemic functions, and ontological functions in DOLCE

We will use DOLCE as upper-level ontology as a guide in foundational ontological issues. The reasons for this choice are discussed at the end of Section 1.1.

We are especially interested in roles, since, we anticipate, it will be argued that a certain meaning of the word function corresponds to a subcategory of roles. Therefore we start the analysis with them. As mentioned in the introduction, roles are founded, meaning that they depend in some sense on some external entity. This is particularly important in this thesis, thus, we will proceed on with a formalization. In the following formulas, $\mathtt{CL}(\cdot,\cdot,\cdot)$, 'classified-by', is a relation holding between concept and one individual, at a certain time, if the individual is recognized as an instance of that concept (cfr [179]); while $\mathtt{K}(\cdot,\cdot,\cdot)$ ('constitutes') is DOLCE's relation linking substrata to their object; and $\mathtt{O}(\cdot,\cdot,\cdot)$ is the temporal overlap relation that is obtained by parthood in the usual way[2]; and $\mathtt{PRE}(\cdot,\cdot)$ stands for being present at a time (cfr Table 1.3).

**df9** $\mathtt{externalTo}(x,y) \iff \neg(\mathtt{qt}(x,y) \vee \mathtt{qt}(y,x) \vee \exists t(\mathtt{K}(x,y,t) \vee \mathtt{K}(y,x,t) \vee \mathtt{O}(x,y,t)))$

"$x$ is external to $y$ if and only if $x$ is neither a quality of $y$, nor $y$ of $x$, nor $x(y)$ is one of $y(x)$'s constituents (at any time), nor $x$ and $y$ have parts in common (at any time)[3] (at any time)."

**df10** $\mathtt{SD}(x,y) \iff (\exists t(\mathtt{PRE}(x,t)) \wedge \forall t(\mathtt{PRE}(x,t) \implies \mathtt{PRE}(y,t)))$

"$x$ specifically (existentially) depends on $y$ if and only if $x$ exists at some time and at any time when $x$ exists so does $y$."[4] Note that this is the same as Definition (df1): we duplicated the definition for sake of readability.

We also specialize the founding relationship to concepts:

**df11** $\mathtt{instantiationFoundedOn}(x,y) \iff (\mathtt{CN}(x) \wedge \mathtt{CN}(y) \wedge \exists z,t(\mathtt{CL}(z,x,t)) \wedge \forall z,t(\mathtt{CL}(z,x,t) \implies \exists w(\mathtt{externalTo}(z,w) \wedge \mathtt{CL}(w,y,t))))$

"the concept $x$ is instantiation-founded on the concept $y$ if and only if, whenever $z$ is a given entity that plays as $x$, there is also an external entity $w$ that is an instance of $y$."

[2]Two entities overlap if and only if they have a common part.

[3]Substrata, parts, and qualities may not exhaust all possibilities if a different top-level ontology were used.

[4]Without the existential quantifier something that is never present would depend on everything. Due to this logical technicality, analogous existential quantifiers will be required by axioms having a similar structure.

For us the case that the classified entity is a perdurant (e.g. an engineering behavior) is especially important. In Masolo's original work on roles, concept-instances are limited to DOLCE-endurants. Therefore, we explicitly extend the domain of the classification relation to perdurants. Additionally, we consider a non-temporalized version of CL holding for perdurants.

**ax1** $\texttt{CL}(x, y, t) \implies (\texttt{ED}(x) \vee \texttt{PD}(x))$

"if $x$ is classified by a concept at some time, then $x$ is either a perdurant or an endurant."

**ax2** $\texttt{CL}(x, y) \implies \texttt{PD}(x) \wedge \forall t(\texttt{PRE}(x, t) \implies \texttt{CL}(x, y, t))$

"$x$ is (constantly) classified by $y$ if it is a perdurant and is classified by $y$ at all times it is present."[5]

We also introduce another type of founding relation, which we term definition-founding ($\texttt{defFoundedOn}(\cdot, \cdot)$). We refrain from formalizing this relation in detail, as it entails discussing the formal definition of roles, a topic beyond the scope of this thesis. Instead, we provide an informal interpretation:

**df12** "$x$ is definition-founded on some entity $y$ if and only if $y$ is used to define $x$."

For instance, if the role of a doctor is defined as someone who treats sick individuals, then the doctor-role is definition-founded on the sick-person concept. A slightly more extensive comparison with Masolo's work on social roles is also present in Section 3.0.9. It's worth noting that instantiation-founding and definition-founding may not always coincide. For instance, an individual may be a doctor without actively treating any sick individuals at a given time. In such cases, the doctor-role is not instantiation-founded on someone being sick.

Given all this, we can explicitly say that DOLCE's roles are founded:

**ax3** $\texttt{RL}(x) \implies \exists y\ \texttt{defFoundedOn}(x, y)$

"if $x$ is a role, then there is a $y$ on which it is definition-founded."[6]

[5]Note that this formula is actually highly non-trivial, ontologically speaking: the temporal argument of the ternary CL relation can account for antirigidity of role-concepts: (anti-)rigidity requires a modality to be expressed, and time can be such a modality (I am a student today, day, but not tomorrow). If the CL relation loses the temporal argument, how can the antirigidity of role-concepts be accounted for? It is still possible: in our case, the modality that makes a perdurant (say, a device behavior) into a role (say, a function), is the subjective evaluation of an agent in the context of a causal analysis of an engineered system, under some goal, cf. (df16). In particular, the same perdurant may be a function in one analysis but not in different one.

[6]Note that in Masolo's [179] a concept ($x$) is said to be founded if there is another concept ($y$) such that, in our terminology, $x$ is instantiaton-founded (df11) and definition-founded (df12) on $y$. However, we consider only this weaker axiom and exclude the instance-founding because it may exclude the example of doctors and patients. Note that one could still ensure instance-founding of the doctor-role, e.g. if one instance-founds it on individual medical licences. However, doing so would require the addition of such social objects in the conceptualization and a modeller may or may not prefer to do so. In any case, Axiom (ax3) is compatible with both approaches since it is compatible with Masolo's axiom.

For instance, in the context of the definition provided in (df11), the role of 'husband' ($x$) is instantiation-founded on the concept of 'marriage' ($y$). This is because whenever a person ($z$) is a husband, there exists an individual marriage ($w$) involving that person and another individual, which is external to the husband.

Furthermore, considering DOLCE's view of roles and concepts as reified classes, there must exist a binary relation corresponding to the subsumption relation between classes. Such relation is called 'sub-concept-of' ($\texttt{subConceptOf}(\cdot,\cdot)$).

**df13** $\texttt{subConceptOf}(x,y) \iff (\texttt{CN}(x) \wedge \texttt{CN}(y) \wedge \exists z,t(\texttt{CL}(z,x,t)) \wedge \forall z,t(\texttt{CL}(z,x,t) \implies \texttt{CL}(z,y,t)))$

> "A concept $x$ is a sub-concept of a concept $y$ if and only if all instances of $x$ are also instances of $y$."

As an illustration, consider the scenario where the role of the Italian Prime Minister is a sub-concept of the role of Prime Minister. It's important to note that this notion of sub-concept is ontologically weak. Ideally, one would want to introduce a modal characterization, indicating that Definition (df13) holds necessarily across all possible worlds. However, delving into these logical technicalities is not particularly pertinent to this thesis. Therefore, in this thesis, we will utilize this formula and similar ones, under the assumption that in appropriate languages, such as first-order modal logic, a suitable formula would be substituted. Additionally, note that we refrained from calling this relation 'specialization' because sub-concept-of and specialization are sometimes distinguished (e.g. Masolo does it [179]), however we will sometimes use specialization as a synonym for sub-concept-of, to improve readability[7].

We also need to introduce a specialization of DOLCE's qualities, dividing them into 'intrinsic' and 'relational' (or extrinsic) [174]. Mass, length, and shape, are standard examples of intrinsic qualities. On the other hand, weight (which varies depending on the gravitational field)[8], personal marathon records (which are tied to the societal notion of marathons), and distances between objects (which rely on the presence of multiple objects), are examples of relational qualities. Relational qualities are inherently tied to external factors beyond the object itself. For instance, the distance of the Moon from the Earth, seen as a property of the Moon, cannot be thought of without considering the Earth, indicating that it is a relational quality of the Moon (likewise for the distance of the Earth from the Moon). Conversely, if a certain brick possesses a given tensile strength, this does not depend on other entities, rendering tensile strength an intrinsic quality[9]. Similarly, other mechanical or chemical qualities, such as ductility or the structure of atomic lattices in crystals, can be deemed intrinsic. Another example is the voltage at a point of an electric circuit, which necessarily requires a second point used as reference (the ground of the electrical circuit). Moreover, intrinsic and

[7] Also, in some oncoming formulas containing the sub-concept-of relation, such a relation could be replaced with the stronger specialization relation.

[8] This one is very debatable. With less controversy, we can say that, if mass and weight must be said one intrinsic and the other extrinsic, then mass is the intrinsic one and weight is the extrinsic one.

[9] One needs other entities to measure the tensile strength value, but note that this dependence is relative to establishing the value only, and is not related to the tensile strength's existence.

relational qualities are, arguably, present in the engineering literature. For example, in [209], Qian and Gero discussed various types of 'behavioral variables': structural, such as the dimensions of a room or the size of a water tap, and exogenous, such as water flow through a tap. In the latter case, water flow is deemed an exogenous variable because "water is not part of the water tap design, it is only related to the design". It is natural to see a correspondence between this structural-exogenous duality and the intrinsic-relational duality.

Delineating relational qualities is a nuanced task that extends beyond the scope of this thesis. Interested readers may explore a proposed framework on relational and intrinsic qualities in [73]. Hence, we just introduce intrinsic and relational qualities as primitive subcategories of DOLCE qualities, which will be used in the following sections:

**ax4** $\texttt{RelationalQuality}(x) \implies \texttt{Q}(x)$

"Relational qualities are qualities."

**df14** $\texttt{IntrinsicQuality}(x) \iff (\texttt{Q}(x) \wedge \neg\texttt{RelationalQuality}(x))$

"Intrinsic qualities are those qualities that are not relational."

The last brick we need before coming to functions and behaviors more properly, is a concept for engineering systems and their components. Various terms are utilized to denote these, such as part, component, tool, machine, technical artifact, functional object, etc. We adopt the term 'technical artifact'[10] to signify any physical object[11] (ax5) originating from an intentional technical process. Examples include cars, planes, tooling machines, and, broadly, devices designed for specific tasks. We'll also treat the terms 'device' and 'technical artifact' interchangeably. Moreover, we employ the term 'system' to emphasize the mereological structure of a complex artifact. While the concept of a system is intricate and cannot be fully encapsulated by that of a technical artifact, its precise delineation remains an ongoing challenge (as discussed in, e.g., [191]). Here, we establish the notion of a system as fundamental, introducing a primitive class, $\texttt{System}(\cdot)$, wherein technical artifacts are encompassed. The mentioned mereology is constructed conventionally from the temporalized parthood relation of DOLCE, denoted as $\texttt{P}(\cdot,\cdot,\cdot)$. Additionally, DOLCE provides a non-temporalized parthood relation, $\texttt{CP}(\cdot,\cdot)$, defined in (df15). We also incorporate DOLCE Axiom (ax6) (recall that $\texttt{POB}(\cdot)$ represents the category of physical objects, see 1.2).

**ax5** $\texttt{TechArtifact}(x) \implies \texttt{POB}(x)$

"a technical artifact is a physical object."

**df15** $\texttt{CP}(x,y) \iff \exists t(\texttt{PRE}(y,t)) \wedge \forall t(\texttt{PRE}(y,t) \implies \texttt{P}(x,y,t))$

"$x$ is constantly part of $y$ if and only if whenever $x$ exists, it is part of $y$."

**ax6** $\texttt{P}(x,y,t) \implies (\texttt{PRE}(x,t) \wedge \texttt{PRE}(y,t))$

[10] For a deeper exploration of technical artifacts, refer to [32].

[11] Of course, one could also consider non-physical objects such as software artefacts or methodologies and call also these 'technical artefacts'. However, since the focus of this thesis is on physical technical artefacts, we follow [32] and consider ohysical objects only.

"if $x$ is part of $y$ at time $t$, then $x$ and $y$ are both present at time $t$."

After these DOLCE-related preliminaries, we want to start building a framework to characterize engineering artifacts behaviors and functions, in such a way to explain the engineers' speech in different areas, including engineering design, manufacturing, maintenance, process and product planning, etc.

**Engineering behaviors.** The behavior of an artifact refers to how it realizes specific interactions between itself and elements of the environment, as exemplified by statements like "[the car] rattled when it hit the curve" [49]. While DOLCE addresses the occurrence of interactions as perdurants, ontologically understanding behavior is more complex. In the literature, the term 'behavior' is employed with diverse connotations. It may denote simplified parts or descriptions of processes, as indicated by excerpts such as "The causal rules that describe the values of the variables under various conditions" [49], or "the behavior represents objective conceptualization of its input-output relation as a black-box" [141], or "situation-independent conceptualization of the change between input and output of the device" [194]. In the YAMATO framework [196], behaviors are modeled as processes involving an agent or agent-like object as a doer. On the other hand, Borgo et al. [27] characterize behaviors as relational qualities that describe the specific manner in which an object participates in individual events.

In order to make a conscious decision on how to assimilate the concept of behavior in this thesis, we will discuss several key ontological properties that help distinguish among various definitions of behavior. There are at least four axes along which different interpretations of behavior can vary in the literature.

First, there's the occurrence-property dichotomy: behaviors can be viewed as something that unfolds in time and in which the entity in question actively participates. In this perspective, behavior is often described as a transition between states or simply as staying in a state. Examples of this view can be found in works by Goel [86], Chandrasekaran [49], Umeda [246], and the authors of YAMATO [194]. Alternatively, a behavior can be conceptualized as a quality inherent to the behaving entity itself. Examples of this perspective are provided by Borgo et al. [27], and similar notions are discussed by Vermaas and Gero and colleagues, who refer to attributes or dispositions [257, 80].

Next, there's the token-type distinction: behaviors can be considered as a class or concept, as argued by Mizoguchi [194], alternatively, a behavior can be seen as an instance or entity relative to a specific event, as proposed by Chandrasekaran and Josephson [49], and by Borgo et al. [27], respectively.

Moreover, there's the external-internal axis: a behavioral description of an artifact may solely refer to characteristics intrinsic to the artifact itself, or they may need to reference external entities. For instance, "the electric switch can alternate between open and closed states" is an internal behavioral description, while "the current passing through an open switch is zero, if the applied voltage stays within operating conditions" is an external description. Different authors may use behavior with either an internal or external meaning, as exemplified by works such as those by Zhao et al. [270] and Kitamura et al. [146].

Lastly, there's the modal axis: behaviors can be perceived as either expected (as envisioned by engineers) or actual (what actually occurs). This duality is implied by Gero [81], particularly in the context of design activity, but it can also be applied to discussions of malfunctioning. This axis also encompasses discussions of causal laws or relations, as they could be conceptualized as changes of a system under certain modalities, i.e., changes that necessarily occur when specific conditions are met. While not advocating for this conceptualization of causal laws, it's important to acknowledge this perspective, as it's encountered frequently in literature.

In the context of this thesis, we chose to lean towards the occurrence, type, and external directions. The main reason is that this is the most natural fit to our oncoming characterization of (certain) functions as roles. In particular, we introduce 'engineering behavior' as a DOLCE perdurants subcategory, whose instances are related to external entities.

Engineering behaviors, referred to simply as behaviors hereafter, encompass perdurants in which two processual[12] roles can be identified: an active role termed 'doer' (the subject(s) of the behavior) and a passive role referred to as 'operand' or 'flow' [205] (the object(s) in the behavior). This role-based conception of behavior is motivated by and limited to the domain of functional modeling, where it underscores a prevalent engineering perspective. It complements the broader ontological viewpoint of behavior as a relational quality between an entity and its environment, as elucidated in [27]. While the role-based approach portrays behavior as a perdurant and aligns naturally with the ontological view of function that will be explored later, the ontological understanding of behavior can be refined to differentiate between the behavior of the doer and that of the operand, as necessary. For example, consider a cutting action, which involves a subject (such as a saw or the system comprising a person or machine using the saw) and an object (such as a beam of wood). Simultaneously, the saw exhibits its own distinct manner of interacting with its environment (to which the wood belongs), i.e., its unique ontological behavior, and similarly, the wood demonstrates its behavior concerning its environment (to which the saw belongs). This distinction applies to other behaviors described by transitive verbs, such as pumping, joining, and so forth.

The behaviors described above, which we denote by the predicate $\texttt{Behavior}(\cdot)$, are external behaviors, given that the flow and the behaving artifact (the doer) are external to each other. We model the two processual roles using two specializations of DOLCE's participation relation: $\texttt{participateAsDoer}(\cdot,\cdot,\cdot)$, indicating participation in a process of an entity with the role of an agent or agent-like doer, and $\texttt{participateAsFlow}(\cdot,\cdot,\cdot)$, representing the role of an operand:

**ax7** $\texttt{participateAsDoer}(x,y,t) \implies (\texttt{TechArtifact}(x) \vee \texttt{Agent}(x)) \wedge \texttt{Behavior}(y) \wedge \texttt{PC}(x,y,t)$

"if $x$ is a doer in $y$ at time $t$ then $x$ is a technical artifact or an agent, $y$ is a behavior, and $x$ participates in $y$ during that time."

**ax8** $\texttt{participateAsFlow}(x,y,t) \implies \texttt{Behavior}(y) \wedge \texttt{PC}(x,y,t)$

[12] following the terminology introduced by Loebe [168], which refers to general perdurants, not to the category of DOLCE's processes specifically.

"if $x$ is a flow in $y$ at time $t$ then $y$ is a behavior, and $x$ participates in $y$ during that time."

**ax9** $\texttt{Behavior}(x) \implies \exists y, z, t(\texttt{participateAsDoer}(y, x, t) \wedge \texttt{participateAsFlow}(z, x, t) \wedge y \neq z)$

"$x$ is a behavior only if there are at least a doer and a flow that participate in $x$."

From Axiom (ax9), it follows that behaviors are perdurants. Additionally, we assume that behaviors can be combined to provide causal explanations of complex perdurants, such as the 'beam was cut, therefore it fell to the floor'. This is formalized using a causal relation between perdurants: `posCausalCon`. This relation will actually be defined and discussed in the next chapter (df44), but we anticipate its use here.

The roles of doer and flow are also not expanded upon, and left as primitive notions. For a deeper discussion of these notions, refer to [168]. The combination of the role-based approach with the causal contribution relation allows us to model behavior from the perspective of the participating device, as outlined in Axiom (ax9). However, it also introduces certain limitations. For example, consider the case of an artificial agent altering its own settings, which represents a function where the doer and flow coincide. Such a case is ruled out by Axiom (ax9), but, for example, is discussed and called 'change over' in [29]. A comprehensive analysis of the relationships between the two approaches to behavior (relational quality vs. perdurants with roles) has not yet been conducted in this thesis, but it is worth noting again that these approaches are mutually compatible within DOLCE. Henceforth, when referring to behavior, we will generally mean engineering behavior unless otherwise specified.

Having touched on behaviors as role-based views of perdurants, we come now the states of engineering systems. In DOLCE, states are cumulative and homeomeric perdurants, as discussed in Section 1.1. These properties should also apply to states of engineering systems. If we understand states, as many engineers do, as conditions determined by constraints over state variables, then such conditions are homeomeric (if a device satisfies a constraint over a time period, then it also satisfies it during a fragment of that period) and cumulative (if a device satisfies a constraint over some time periods, then it satisfies it during the union of those periods). However, the engineers' use of the term is often more liberal than DOLCE's concept of states. For example, engineers use the "state" to describe oscillating phenomena and similar occurrences (e.g., the buzzing action of a clapper, which alternates between two different DOLCE-states while buzzing, namely open-circuit and closed-circuit). While one could consider an oscillating phenomenon as characterized by a variable constrained to a constant value (say, the frequency equal to some value), ontologically, it is intrinsic to, say, the buzzing motion to be composed by more than one subphase (e.g. contact or non-contact between the clapper's components). Consequently, the appropriate DOLCE category to conceptualize system states is that of stative perdurants. To avoid terminological confusion, we will use the term 'state' following engineering terminology, while ontologically, it is understood as a 'stative condition' in DOLCE.

Given that engineers typically understand how to define the desired state for a system, we assume that specific 'types' of states are selected as goals for a given technical

artifact. These goals typically involve selecting the conditions that a state must satisfy, which refers to a concept rather than a specific instance of a state. Consequently, we can state:

**ax10** $\texttt{Goal}(x) \implies \texttt{CN}(x) \wedge \forall y, t(\texttt{CL}(y, x, t) \rightarrow \texttt{STV}(y))$

"a goal is a concept that classifies stative perdurants only."

Hence, expressions such as 'the temperature at the port B of the heat exchanger is between 80 and 110 Celsius degrees' and 'the buzzer is clapping with a frequency of at least 10kHz', which describe states, may correspond to goals.

**Systemic functions.** we come now to propose a preliminary formalisation of ontological functions, based on the concepts introduced in the previous paragraphs. We start with defining 'systemic functions', primarily inspired by Mizoguchi's definition of such very term in [193] (cfr. also Cummings' definition in [59]): a function is seen as the contribution of an individual component's activity to the activity of the system as a whole, and teleological aspects are introduced through goals imposed on the system. This definition is based on the so-called 'systemic view' of devices, which posits that devices are complex aggregates of components, whose interaction with the rest of the system contributes to generating the activity of the whole system. Note that our 'behavior' is different from Mizoguchi's [193], in particular, our approach facilitates decomposition (and recomposition) of a system's behavior into the behaviors of its components: the latter are parts of the first.

**df16** $\texttt{systemicFunctionOf}(x, y) \iff (\texttt{RL}(x) \wedge \exists z, g, b, t(\texttt{System}(z) \wedge \texttt{goalOf}(g, z) \wedge \texttt{CL}(b, x, t) \wedge \texttt{P}(y, z, t) \wedge \forall b', t'(\texttt{CL}(b', x, t') \implies (\texttt{Behavior}(b') \wedge \texttt{participateAsDoer}(y, b', t') \wedge \texttt{P}(y, z, t') \wedge \texttt{posCausalCon}(b', g)))))$

"$x$ is a systemic function of $y$ if and only if $x$ is a role and there exists a system $z$ and a goal $g$ for $z$ such that $x$ classifies only behaviors which have $y$ as doer and part of $z$, and that causally contribute to achieve $g$."[13][14]

**df17** $\texttt{SystemicFunction}(x) \iff \exists y\ \texttt{systemicFunctionOf}(x, y)$

"$x$ is a systemic function if and only if it is the systemic function of some object $y$."

There are numerous function conceptualizations in the literature, each pertinent from a distinct engineering perspective. We commence with Definition (df16) because,

[13]This implies that the term 'systemic' in 'systemic function' denotes the reliance of such a role-concept on a system, without suggesting that the player artifact itself is a system. For instance, in the case of a wood table held together by screws, each screw serves a systemic function within the table-system: it connects one of the table-legs to (a side of) the table-top, despite none of the screws being a system.

[14]This definition entails that systemic functions always classify some behavior at some time. But what of a system that is never operated or a system-as-designed? If one wishes to consider these cases they must accept (only-)possible perdurants as entities in the formula. This topic is however ontologically complex, cf. the discussion of non-actual events in the next chapter.

unlike others, it provides ontological clarity and aids in elucidating the assumptions underlying alternative meanings.

Upon observing Definitions (df16) and (df17), one realizes that systemic functions are specifically-dependent on systems. Moreover, it also appears that systemic functions are definition-founded on systems. However, this cannot be proven since we gave no definition of definition-founding, and must be state explicitly:

**ax11** $\texttt{SystemicFunction}(x) \implies \exists y(\texttt{System}(y) \wedge \texttt{defFoundedOn}(x, y))$

"each systemic function is definition-founded on some system."

**Ontological functions.** Engineers discuss functions even independently of any specific system, especially in early system design phases. For example, when addressing the task of identifying devices capable of realizing a required transformation, these devices cannot be inherently associated with a particular system at the outset, as there is no system defined yet. The system remains a variable input in the problem-solving process. Therefore, to address such scenarios, a more general concept of function is required, as discussed in [29]. In this thesis, these functions are referred to as 'ontological functions', as they are relative solely to the upper ontology being employed. The idea behind ontological functions is to delineate very general transformations without specifying how such transformations occur or which entities they apply to. Informally, ontological functions are categorized primarily based on the ontological distinctions they impose between input and output states, independent of systemic functions.

For instance, consider modeling the activity of walking as the realization of a moving function. While walking may seem like a simple action, it is actually a highly coordinated, continuous, and dynamic process. To simplify its modeling, one might choose to 'constrain' the walking model to certain aspects, such as the variation in distance of the feet from the floor, the trajectory of the body's center of mass, the changing contact forces between the feet and the floor, and the forces developing within the leg joints, among others (cf. [93]). There are numerous aspects to focus on, leading to the question of which aspects should be utilized.

Assuming a fixed upper ontology, one examines the state transitions made available by the ontology that are relevant to the transition of interest. In the case of walking, the pertinent feature(s) characterizing walking at the upper level of DOLCE are the changes in the entity's location. At this high level, the ontology provides only space and location qualities among the entities relevant to walking, without characterizing or committing to classes such as foot or center of mass, or qualities like speed and force. Consequently, the perdurant cannot be classified as walking since DOLCE lacks a walking category or the requisite ontological commitments to introduce one. If walking is characterized only by the change in location quality at this level, it cannot be distinguished from swimming or flying (depending on the richness of the upper-level ontology). However, even at this level of characterization, substantial differences can be made; for instance, cutting can be distinguished from walking. Cutting is characterized by the presence of one entity at the initial state and two or more entities at the final state, which walking lacks. (In DOLCE, entities can be distinguished by number, thus cutting

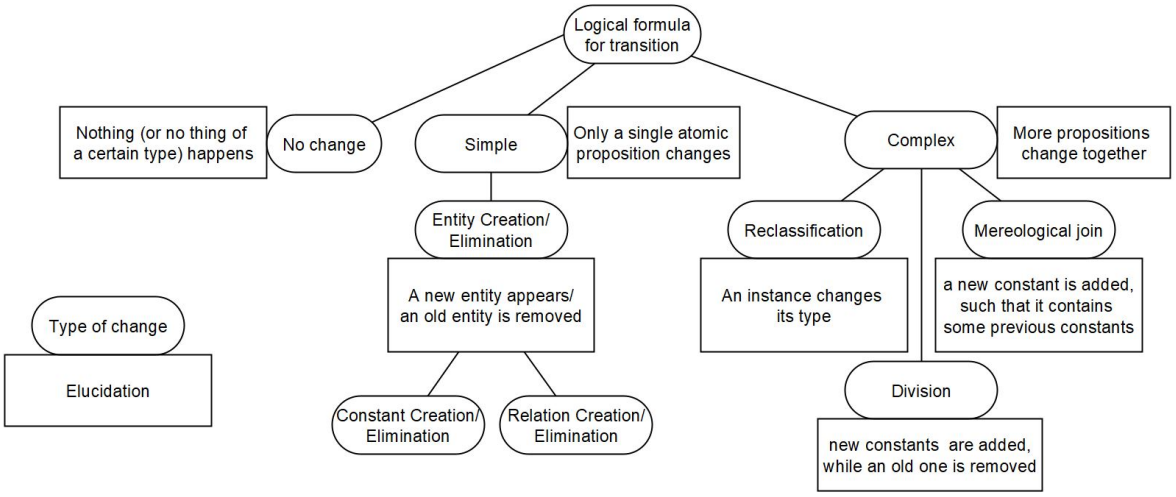



Figure 3.1: A non-exhaustive taxonomy of the logical formulas that describe differences between states, based on what predicates differ between initial and final states.

and walking are different types of perdurants even at this level.) Since functions are realized by perdurants, we can utilize this ontology-driven categorization of perdurants to introduce a categorization of functions.

We refer to these functions as ontological functions, which are carried out by transformations characterized by the ontological commitments present in the upper ontology. This provides only a high-level (coarse) characterization of functions. Since the resulting taxonomy of functions strictly depends on the adopted ontology, we use the qualifier 'ontological' and refer to them as ontological functions (when more than one upper-level ontology is used, one should be more precise by indexing the ontological functions by the ontology they depend upon, e.g., 'DOLCE-ontological functions'). Continuing the discussion with DOLCE as the upper ontology, the ontology offers categories such as region, quality, physical object, amount of matter, etc. (see Figure 1.5), which give rise to several ontological function-types based on change in quality-value (e.g., 'the controller increased the internal temperature of the oven'), change of physical object (e.g., 'the employee assembled four legs and a table-top to form a table'), change of amount of matter (e.g., 'the water was vaporized by the boiler'), and so on, as shown in Figure 3.2.

Note that an ontological category can be implicated in a transition in various ways: an instance of that category could cease to exist, if the instance was present in the state before the transition but not after. Conversely, an instance could be created, if it was not present before the transition but only afterward. Another possibility is a change in a relation of the ontology; for example, in the initial state, there may be two individuals, one part of the other, say, while in the final state they are not. Additionally, in most cases, the same event may be classified by several ontological functions. For instance, a cutting-by-knife event is also a squeezing event (i.e., a compression, a change of the shape quality of the initial object) and a moving event (a change of the

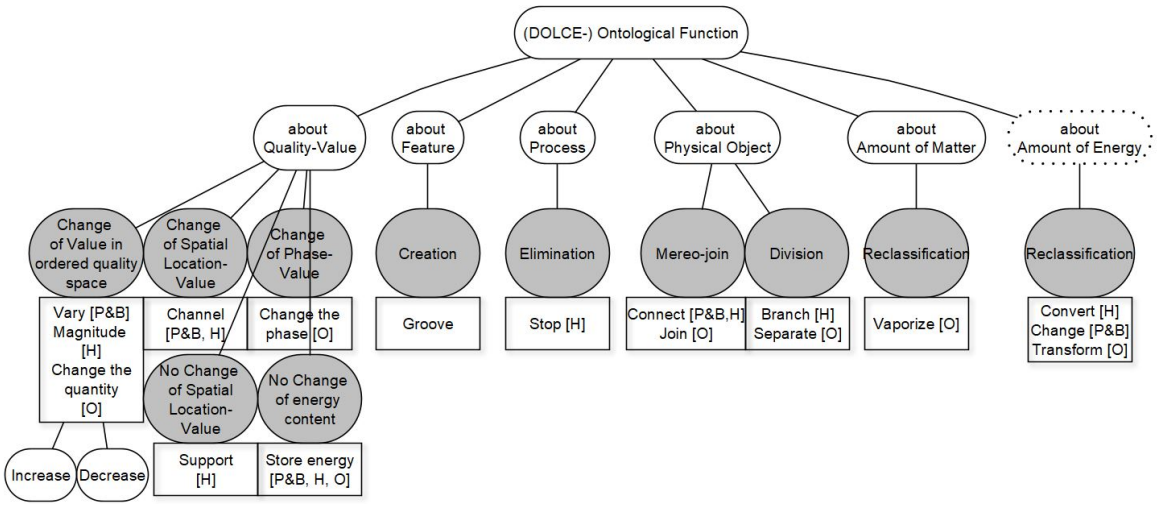



Figure 3.2: An illustration of how a taxonomy of ontological functions might be structured. The rounded white rectangles encompass the categories of ontological functions (classes of perdurants with identified doer and flow roles). The gray circles denote functional concepts utilized to specialize ontological functions as defined in (df22), referencing Figure 3.1. The rectangles with sharp corners enumerate alternative terms used in the literature to denote that very functional concept: 'H', 'P&B', and 'O' correspond to [116], [205], and [223], respectively. (It's worth noting that terms may have different meanings in the literature than in this figure, and that the same occurrence could be categorized by multiple functional concepts). The inclusion of 'Amount of Energy' is purely for illustrative purposes; it is not a part of DOLCE.

device's location relative to the object's location). In such a way, more complex changes could be described by composing simple examples such as these. Some of these cases are illustrated in Figure 3.1, which is not exhaustive. These distinctions are naturally similar to those obtained via the ontological analysis of events by Guarino et al. [93], however, while Guarino and colleagues focus on what they call 'qualitative events' (essentially our changes in quality value), we must encompass a wider variety of cases, e.g. creation/destruction events. Another similarity is with UFO-B, in whose axiomatization [20, Section 3.9] creation and destruction of objects is explicitely considered, and in which would be therefore easier to define creation and destruction events. However, in such a work UFO-predicates (mainly situations and causal relations between situations and events[15]) are used to formulate the definition in such a way that a linear comparison with our oncoming approach is difficult.

The function names used in this paragraph are inspired by the literature, which is not homogeneous. For instance, 'reclassification' (change where an instance changes type) of energy is sometimes called 'convert' [116, 223] and some other times 'change' [205] or 'transform' [146].

Functional terms associated with domain-level concepts (e.g., 'vaporize', 'moisten', 'heat') are not encompassed at the level of ontological functions and necessitate a domain or middle-level ontology containing relevant concepts (e.g., vapor, humidity, temperature) to extend the framework described earlier to that level of ontological distinctions. The ontological status of certain concepts, such as 'energy', remains unclear [181], despite being fundamental in engineering and physics. However, with an ontology containing a conceptualization of energy, it becomes feasible to introduce a corresponding ontological function, which may differ in another ontology with a distinct conceptualization of energy.

For example, one may conceptualize energy as a quality, as demonstrated in [28]. This is applicable when referring to the energy 'of something', such as the energy of a battery. In this scenario, our ontology would feature an 'energy content' quality type, giving rise to a corresponding 'increase energy', or 'store energy', etc. ontological functions. Alternatively, energy can be conceptualized as an enduring entity that can inhabit entities, as demonstrated in [223, 194] where energy is considered an "operand" which inhabits a "medium", and has the possibility of changing its location and its type (e.g., 'before the impact the energy was in that body, after the impact part of the energy was transferred to that surface with its type junged from kinetic energy to heat energy', etc.). In this case, the 'convert energy' ontological function would represent a reclassification-type transformation.

An ontological function essentially serves as an event classifier, examining the alterations (or continuities) occurring within the event. This approach implies that specific ontological functions may be defined by comparing initial and final states. For instance, consider introducing a 'connect' function in the following manner, where $\texttt{sState}(y, s_i, t_i)$ ($\texttt{eState}(\cdot, \cdot, \cdot)$) means that $s_i$ is a state with time extension $t_i$, and is part of $y$ such that no parts of $y$'s time extension precedes (follows) $t_i$:

[15]These primitives can also express our initial and final states introduced in the following.

**df18** $\texttt{Connect}(x) \iff (\texttt{CL}(y,x) \iff$
$\texttt{E}(y) \wedge \exists a,b,c,s_i,s_f,t_i,t_f\ (\texttt{sState}(y,s_i,t_i) \wedge \texttt{eState}(y,s_f,t_f) \wedge \texttt{Goal}(s_f) \wedge \texttt{POB}(a) \wedge$
$\texttt{POB}(b) \wedge \texttt{POB}(c) \wedge \texttt{PC}(a,s_i,t_i) \wedge \texttt{PC}(a,s_f,t_f) \wedge \texttt{PC}(b,s_i,t_i) \wedge \texttt{PC}(b,s_f,t_f) \wedge \texttt{PC}(c,s_f,t_f) \wedge$
$\texttt{P}(a,c,t_f) \wedge \texttt{P}(b,c,t_f)) \wedge \neg\exists c'(\texttt{POB}(c') \wedge \texttt{PC}(c',s_i,t_i) \wedge \texttt{P}(a,c',t_i) \wedge \texttt{P}(b,c',t_i)))$

"A 'connect' function is a classifier of events that are transitions between a starting state ($s_i$, with time extension $t_i$) and a final state ($s_f$, with extension $t_f$) and two entities ($a$ and $b$) participate in such states such that in the final state, but not in the starting one, they form a single object ($c$)."

Notice that `Connect` represents the act of forming a single object out of two. To talk about, say, an 'assembly' function, the final object should also be required to be a single, self-connected, piece. Additionally, reversing the initial and final states would produce a definition for the 'divide' function. Note also that in (df18) `CL` appears as a binary predicate, while, when it was originally introduced in [179], it was a ternary predicate holding between an endurant and a concept at a given time. We extend the original predicate to the case of a concept classifying a perdurant. In this latter case, we suppress the temporal argument and we mean that the classification is w.r.to the whole perdurant and not just a temporal slice of it.

Then, to define 'convert' one could write (where $\Phi$ and $\Psi$ are types belonging to the set $\mathcal{NR}$ of all non-rigid types of the underlying ontology):

**df19** $\texttt{Convert}(x) \iff (\texttt{CL}(y,x) \iff \texttt{E}(y) \wedge \exists s_i,s_f,t_i,t_f,a,\Phi,\Psi\ (\texttt{sState}(y,s_i,t_i)$
$\wedge\ \texttt{eState}(y,s_f,t_f) \wedge \texttt{Goal}(s_f) \wedge \Phi \in \mathcal{NR} \wedge \Psi \in \mathcal{NR} \wedge \Psi \cap \Phi = \emptyset$
$\wedge\ \texttt{PC}(a,s_i,t_i) \wedge \texttt{PC}(a,s_f,t_f) \wedge \Phi(a,t_i) \wedge \Psi(a,t_f)))$

"A 'convert' function is a classifier of events that are transitions between a starting state ($s_i$, with time extension $t_i$) and a desired final state ($s_f$, with extension $t_f$) and, moreover, there is an entity ($a$) that at time $t_i$ is instance of a type ($\Phi$), while at time $t_f$ is instance of a different, disjoint, type ($\Psi$)."

In the previous definition, we quantified over the set of all non-rigid classes of an ontology ($\mathcal{NR}$). When such a set is finite, as in the case of DOLCE, this quantification corresponds to a finite disjunction of predicates. Therefore, Formula (df19) becomes a first-order formula. Additionally, upper-level ontologies typically consist only of rigid categories, meaning that an individual categorized as a physical object will always remain a physical object. Hence, the convert function becomes particularly useful when modeling changes in entities that undergo phases [98], or when modeling entities that change their processual role, as described earlier, or in combination with non-rigid categories provided by middle or domain-level ontologies.

Ontological functions involving changes in quality values (cfr. the qualitative events of [93]) could be described as:

**df20** $\texttt{ChangeQualityValue}(x) \iff (\texttt{CL}(y,x) \iff \texttt{E}(y) \wedge \exists s_i,s_f,t_i,t_f,a,q,v_i,v_f$
$(\texttt{sState}(y,s_i,t_i) \wedge \texttt{Goal}(s_f) \wedge \texttt{eState}(y,s_f,t_f) \wedge \texttt{qt}(q,a) \wedge \texttt{PC}(a,s_i,t_i) \wedge \texttt{PC}(a,s_f,t_f) \wedge$
$\texttt{ql}(q,v_i,t_i) \wedge \texttt{ql}(q,v_f,t_f) \wedge v_i \neq v_f))$

"A 'change of quality-value' function is a classifier of events that are transitions between a starting state ($s_i$, with time extension $t_i$) and a desired final state ($s_f$,

with extension $t_f$) with a participating entity ($a$) which has a quality ($q$) with different values, $v_i$ and $v_f$, at times $t_i$ and $t_f$, respectively."

These changes in quality values can be further specialized by specializing the quality types in this formula. For instance, if the quality is a spatial location quality, the ontological function will become a 'channel'. If the quality takes numerical values (e.g., 'temperature', 'humidity', etc.), then the ontological function becomes 'vary', possibly an 'increase' or 'decrease', as discussed in [116, 205]. Functions like 'store' and 'support' imply the absence of change. For example, a battery effectively stores energy if it maintains its charge over time, and a load-bearing wall supports the roof only if the roof remains in place. In these cases Formula (df20) should be adjusted to ensure that the initial and final values of the quality are identical, possibly within some tolerance.

So far we have given examples of possible definitions of the ontological functions present in most of the branches of Figure 3.2: if we assume that reclassification-type ontological functions about amounts of matters are defined analogously to reclassification-type ontological functions about amounts of energy, we miss only the 'creation' and 'elimination' branches. We give just one example of an ontological function specializing 'creation' and 'about feature', definitions for elimination-type ontological functions or creation-type ontological functions about different ontological categories should be analogous. To do this, suppose that we have a domain ontology containing the class '`Groove`' as a specialization of DOLCE's category of features. Then we can define:

**df21** $\texttt{ToGroove}(x) \Longleftrightarrow (\texttt{CL}(y, x) \Longleftrightarrow \texttt{E}(y) \wedge \exists s_i, s_f, t_i, t_f, a, f$
$(\texttt{sState}(y, s_i, t_i) \wedge \texttt{Goal}(s_f) \wedge \texttt{eState}(y, s_f, t_f) \wedge \texttt{Groove}(f) \wedge \texttt{SD}(f, a) \wedge \texttt{PC}(a, s_i, t_i) \wedge$
$\texttt{PC}(a, s_f, t_f) \wedge \neg\texttt{PRE}(f, t_i) \wedge \texttt{PRE}(f, t_f)))$

"A 'change to groove' function is a classifier of events that are transitions between a starting state ($s_i$, with time extension $t_i$) and a desired final state ($s_f$, with extension $t_f$) with a participating entity ($a$) that has[16] a feature ($f$) of '`Groove`' type at time $t_f$ but not at time $t_i$."

Note that we do not give a list of definitions that exhausts all the nodes of the tree in Figure 3.2: they would be too many and would distract from our main point that is to show how such definitions can be constructed in general. In addition, note that Figure 3.2 itself is not exhaustive and certain nodes (e.g. vaporize) have been added only to exemplifies selected instances from the literatue.

At this point, it's important to make three observations:

- Ontological functions, by their nature, lack fine-grained detail. They are constrained by the language of the upper ontology and do not account for the dynamics between initial and final states. For instance, consider a temperature-controlled oven where the controller adjusts the heat output to maintain the temperature within a specific range. As a result, the temperature fluctuates around

[16]Note that, in DOLCE, the dependence between features and non-agentive physical objects is, in general, constant generic dependence [177, p.23], which is weaker than specific dependence. However, in this definition, 'having a feature' stands for the stronger specific dependence. This is because, in this particular case, specific dependence seems more appropriate.

the target value, a process that cannot be inferred solely from observing initial and final states. Therefore, distinguishing events that maintain a stable temperature from those with oscillations requires additional constraints on the temperature values during the event (e.g., via variation patterns, see [93]). This aligns with our understanding of ontological functions, emphasizing the need to consider only conditions expressible within the ontology's language.

- Events are often intricate, involving numerous entities, as seen in activities like walking from a biomechanical perspective. This complexity, combined with the acknowledgment that ontological functions can also describe the absence of certain changes (e.g., 'support'), means that many events may be categorized by a combination of ontological functions. This aligns with the inherent flexibility of teleological thinking. Consider a load-bearing wall as an example. Its function could be described as 'supporting' the weight of the roof, but it may also 'transmit' the load to the floor, 'prevent' wind from entering the house, or 'partition' the space into smaller rooms, among other functions. Determining the most relevant aspect in a given context is a task for engineers or technicians, who use a contextual viewpoint to identify the key aspects to focus on and define the target function. This underscores the rationale for using the term 'function' in conjunction with 'ontological' to characterize these classifiers. Given the inherent openness to interpretation of events, multiple perspectives may arise.

- Finally, ontological functions are quite flexible. Throughout the preceding discussions, functional terms were predominately sourced from the Functional Basis [116] or the influential perspective presented by [205]. While these vocabularies are proficient in expressing a considerable quantity, if not the entirety, of functions employed within engineering domains, they may not always do so in the most intuitive manner. For instance, imagine working with tooling machines that create holes, slots, grooves, and so forth in workpieces. Relying solely on the Functional Basis confines one to employ the term 'remove [a specific material]' to describe the functions responsible for producing such features in the workpiece. Conversely, employing a domain ontology containing concepts like hole, slot, groove, etc., enables the development of corresponding functional terms such as 'make hole', 'make slot', 'to groove', and so on, defined analogously to the previously discussed formulas. These terms may offer a more intuitive representation than simply using 'remove' and may even diverge significantly from it depending on their precise formalization. Consider another example: within the Functional Basis vocabulary, the term 'stop' denotes 'to cease the transfer of a flow', as illustrated by the phrase "A reflective coating on a window stops the transmission of UV radiation through a window" [116]. However, what if perdurants play a significant role in our domain, and we wish to express that something 'stops a process' (e.g., "The addition of a respiratory inhibitor stops the absorption of amino acids")? Here, we're not merely describing the cessation of the flow of amino acids and materials, but rather the halting of the absorption process itself, where absorption is a crucial element of the domain. In such cases, deriving functional terms from an

appropriate domain ontology that manages concepts related to domain processes could yield valuable insights.

In the previous paragraphs, We have defined examples of types of ontological functions, but We did not define the concept of ontological function. A possibility is to define ontological functions by comprehension of a list of specific types of ontological functions, provided that such a list is actually exhaustive. For example:

**df22** $\texttt{OntologicalFunction}(x) \iff (\texttt{Connect}(x) \vee \texttt{Divide}(x) \vee \texttt{Convert}(x) \vee \dots)$

On the other hand, one could supply an intrinsic definition, along the lines of, say:

**df23** "A concept is an ontological function if and only if it classifies exactly those events consisting in a transition such that the changes (or the absence thereof) between the initial and final states is characterized in terms of a formula expressed in the language of an upper ontology."

However, a formalization of this latter definition would require higher-order logic. Since we don't want to wander that far with the theory, we refrain from giving an intrinsic definition of ontological functions.

**Link between systemic functions and ontological functions.** Observe that systemic functions are system-dependent functional descriptions, while ontological functions are system-independent functional descriptions and, therefore, more general:

**ax12** $\texttt{OntologicalFunction}(x) \implies (\exists y(\texttt{subConceptOf}(y, x) \wedge \texttt{SystemicFunction}(y)) \wedge \forall y(\texttt{subConceptOf}(y, x) \wedge \neg\texttt{OntologicalFunction}(y) \implies \texttt{SystemicFunction}(y)))$

"any ontological function $x$ has only systemic functions or ontological functions as sub-concepts, additionally, any ontological function has at least one systemic function as sub-concept."

For example, if the system is a tooling machine such as a CNC lathe using two electrical motors, one for rotating the spindle and the other for moving the cutting tool horizontally, then both motors execute the same 'convert' electrical energy into mechanical energy (an ontological function). They also perform each an additional systemic function specializing the ontological function in the context of the lathe-system: the first motor 'converts (electrical energy into mechanical energy) to rotate the spindle', and the second motor 'converts (electrical energy into mechanical energy) to translate the spindle'. The underlying idea is that systemic functions can be grouped based on shared characteristics, as demonstrated by definitions (df18) to (df20), abstracting away from specific systems.

Additionally, note that the two categories of functions introduced in this paragraph reflect at least two distinct notions of functions employed by engineers: there are 'general' functions utilized by engineers when describing or gathering information from diverse systems. For instance, in his analysis of failure experience data [51], Collins discusses 'elemental mechanical functions', which are universal characterizations of fundamental functions. Similarly, Pahl et Beitz [205] mention 'generally valid functions' as

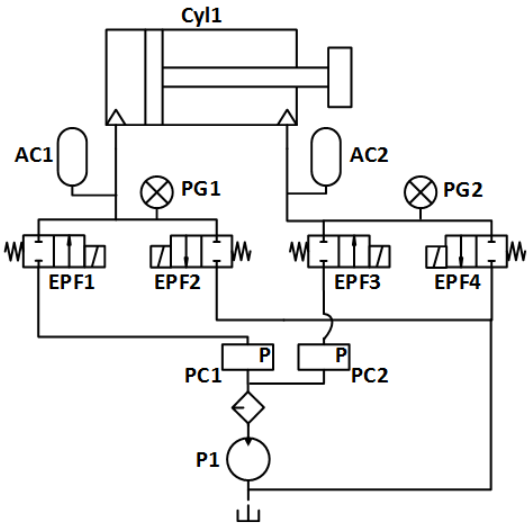



Figure 3.3: Scheme of a hydraulic system, from [170].

references for organizing design knowledge concerning function implementation. These types of functions can be represented through ontological functions. Engineers use also system-specific functions: when engineers focus on a particular system, they employ different terminology to refer to its components. Though the terminology varies, common terms include 'serial number', which identifies a specific component instance, 'component code', denoting the type of component or assembly, and 'functional location' or 'tag', which considers the component's position within a system. For instance, Figure 3.3 illustrates a hydraulic system with four solenoid valves, labeled EPF1 to EPF4. These labels cannot solely indicate the valve type, as all four valves might share the same type, nor can they solely denote valve instances, as schematics typically represent multiple, different, system instances. In our conceptualization, it is natural to identify tags with functional roles played by system components: since the functions of the components contribute to the system function, tags can be formalized as systemic functions.

### 3.0.3 Functional decomposition

In engineering, a critical feature of functions is their potential for decomposition into sub-functions, this enables a finer level of detail in system descriptions.

First, observe that such decomposition cannot be reduced to a partial order relation between functions. That is, if we represent the functional decomposition of a function, say $\mathtt{f}$, into sub-functions, say $\mathtt{f}_1, \mathtt{f}_2, \ldots, \mathtt{f}_n$, as $\mathtt{decom}(\mathtt{f}; \mathtt{f}_1, \mathtt{f}_2, \ldots, \mathtt{f}_n)$, then it does not seem possible to find a parthood relation such that $\mathtt{decom}$ reduces the mereological sum. This is caused by, at least, the following reasons:

First, it's important to note that such decomposition cannot be reduced to a partial order relation among functions. In other words, if we represent the functional decomposition of a function $\mathtt{f}$ into sub-functions $\mathtt{f}_1, \mathtt{f}_2, \ldots, \mathtt{f}_n$, as $\mathtt{decom}(\mathtt{f}; \mathtt{f}_1, \mathtt{f}_2, \ldots, \mathtt{f}_n)$, then

it is difficult to establish a mereological relation such that $\mathtt{f} = \mathtt{f}_1 + \mathtt{f}_2 + \cdots + \mathtt{f}_n$. This is due to at least the following reasons:

- Functions are teleological entities, which need corresponding objective substrata to be executed, encompassing both behaviors and objects. Then, any decomposition of functions must consider the decomposition of the underlying objective substrata.
- The sub-functions of a function must 'organize' (compare [254]) to realize the decomposed function. Specifically, the composition of a mere set of functions is not unique because sub-functions can be organized in different ways.
- A single function can be decomposed in multiple ways, therefore the relationship of (possible) decompositions within functions is necessarily many-to-many. However, not all combinations of sub-functions are feasible due to physical and technical constraints. Moreover, among all possible combinations, engineers typically recognize certain patterns and employ them systematically.

Additionally, Vermaas has demonstrated that modelling functional decomposition as some parthood relation, using the Functional Basis approach – where functional models are represented graphically as directed graphs with flows as edges and functions as nodes, leads to contradictions [256]. These contradictions arise assuming that if a function-token is part of another function-token, the same holds for their corresponding types, and that flow loops are feasible. Consequently, there are formal barriers hindering the application of classical mereology to functional decompositions. For now, we refrain from proposing a solution to this challenge of harmonizing mereology and functional decomposition; instead, we assume the existence of the `decom` relation and concentrate on its description, while a more in-depth discussion will occur in Section 3.0.8.3.

Taking inspiration from the work of Kitamura et al. on the 'ways of functional achievement' [141, 144], we start with introducing engineering methods, conceptualized as individual DOLCE's non-agentive social objects. Engineering methods represent the shared engineering knowledge about ways of implementing functions through functional decomposition:

**ax13** $\mathtt{EngMethod}(x) \implies \mathtt{NASO}(x)$

"Engineering methods are non-agentive social objects."

Formally, such individual non-agentive social objects can be understood as reifications of functional decomposition relation instance[17]. More precisely, we introduce relational[18] roles $\mathtt{MainFunction}(\cdot)$ and $\mathtt{SubFunction}(\cdot)$ such that:

**ax14** $(\mathtt{MainFunction}(x) \vee \mathtt{SubFunction}(x)) \implies$
$(\mathtt{RL}(x) \wedge \exists \mathtt{m}\ (\mathtt{EngMethod}(\mathtt{m}) \wedge \mathtt{instantiationFoundedOn}(x, \mathtt{m})))$

[17] Instantiation of a relation in the sense analogous to instantiation of a class: if, say, 'being-part-of' is a binary relation, then the fact that a crtain handle is part of a certain door is an instance of that relation.

[18] Relational as in the sense of Loebe in [168] or Masolo [178].

"main-functions and sub-functions are roles instantiation-founded on some engineering method."

**ax15** $(\texttt{MainFunction}(x) \vee \texttt{SubFunction}(x)) \implies$
$\exists y(\texttt{OntologicalFunction}(y) \wedge \texttt{subConceptOf}(x, y))$

"Each main-function or sub-function role is a sub-concept-of an ontological function concept." That is, main- and sub-functions are always functions of some ontological-function-type.

Additionally, we assume that methods are linked to a main-function and a certain number of sub-functions (axiom $(\text{ax16})_{meta}$ is actually an axiom-schema, for this reason we mark it with a $_{meta}$-subscript. Given a finite set of methods, such an axiom can be substituted with a finite set of axioms in FOL).

**ax16**$_{schema}$ $\texttt{EngMethod}(\texttt{m}) \implies \exists! n\ \texttt{engMethodNum}(\texttt{m}, n)$, and $n$ is integer

"Each method, say $\texttt{m}$, has a (unique) number, say $n$, of sub-functions that are contextualized by the method."

**ax17**$_{schema}$ $\texttt{engMethodNum}(\texttt{m}, n) \iff \exists! \texttt{main}, \texttt{sub}_1, \ldots, \texttt{sub}_n(\texttt{MainFunction}(\texttt{main})$
$\wedge\ \texttt{SubFunction}(\texttt{sub}_1) \wedge \ldots \wedge \texttt{SubFunction}(\texttt{sub}_n) \wedge$
$\texttt{instantiationFoundedOn}(\texttt{main}, \texttt{m}) \wedge \texttt{instantiationFoundedOn}(\texttt{sub}_1, \texttt{m}) \wedge \ldots \wedge$
$\texttt{instantiationFoundedOn}(\texttt{sub}_n, \texttt{m}))$

"For any engineering method of functional decomposition, say $\texttt{m}$, having a given number of sub-functions, say $n$, there exist a main-function role and $n$ sub-function roles, which are instantiation-founded on $\texttt{m}$ and are uniquely determined."

Given that the roles of the main-function and the $n$ sub-functions of a method are uniquely determined, they can be denoted them using functional symbols. Specifically, we write $\texttt{main}^m$ for the main function and $\texttt{sub}_1^m$, $\texttt{sub}_2^m$, ..., for the sub-functions corresponding to the method $\texttt{m}$, in accordance with Axiom (ax17) (for simplicity, we omit explicitly denoting the functional dependence of $n$ on $\texttt{m}$). The connection between methods and decompositions is outlined in the subsequent definition schema:

**df24**$_{schema}$ $\texttt{decom}(\texttt{f}; \texttt{f}_1, \texttt{f}_2, \ldots, \texttt{f}_n) \iff (\texttt{SystemicFunction}(\texttt{f}) \wedge \texttt{SystemicFunction}(\texttt{f}_1)$
$\wedge \ldots \wedge \texttt{SystemicFunction}(\texttt{f}_n) \wedge \exists \texttt{m}(\texttt{EngMethod}(\texttt{m}) \wedge \texttt{subConceptOf}(\texttt{f}, \texttt{main}^{\texttt{m}}) \wedge$
$\texttt{subConceptOf}(\texttt{f}_1, \texttt{sub}_1^{\texttt{m}}) \wedge \ldots \wedge \texttt{subConceptOf}(\texttt{f}_n, \texttt{sub}_n^{\texttt{m}})))$

"$\texttt{f}$ is decomposed in $\texttt{f}_1, \ldots, \texttt{f}_n$ if and only if they are all systemic functions and there is a method with corresponding main-function $\texttt{main}^{\texttt{m}}$ and sub-functions $\texttt{sub}_1^{\texttt{m}}$, ..., $\texttt{sub}_n^{\texttt{m}}$, which are specialized by $\texttt{f}$ and $\texttt{f}_1, \ldots, \texttt{f}_n$, respectively."

This approach offers numerous advantages. Firstly, it facilitates the organization of method types within hierarchical taxonomies, allowing for nuanced categorization. For instance, 'spot welding' can be classified as a specialization of the broader method 'welding', which, in turn, serves as an implementation method for the function 'to join'. Secondly, it enables the description of method properties, such as their underlying working principles. For example, one can specify Kirchhoff's law as the working principle for a method like 'voltage divider'. Furthermore, this approach enables the introduction

of properties related to functional decomposition. For instance, if one desires to assert that all functions are decomposable, they can state:

**ex12** $\texttt{SystemicFunction}(x) \implies \exists y\ \texttt{MainFunction}(y) \wedge \texttt{subConceptOf}(x, y) \wedge x \neq y$

"For each systemic function ($x$) there is a main function ($y$) such that $x$ is a proper subconcept of $y$."

Instead, if one wishes to state that a function, say `f`, is not decomposable:

**ex13** $\neg\exists y\ (\texttt{MainFunction}(y) \wedge \texttt{subConceptOf}(\texttt{f}, y) \wedge \texttt{f} \neq y)$

"There is no main function such that `f` is a proper subconcept of $y$."

Finally, this approach can be used to formally define a new function concept: *solved functions*, which we define to be functions that are main-functions of some method.

**df25** $\texttt{SolvedFunction}(x) \iff \texttt{MainFunction}(x)$

"Solved functions are exactly main-functions."

Hence, solved functions represent the roles that systemic functions can fulfill in the context of a functional decomposition. For example, in the lathe scenario mentioned earlier, it's likely that the systemic functions of the two motors are both realized through the method of, let's say, 'three-phase electric motor'. Consequently, they assume the role of solved functions. In such a case, the functional decomposition implied by the method encompasses sub-function roles such as 'supply electrical energy' (one for each phase), 'drive the motor', and 'output mechanical energy'.

### 3.0.4 Capabilities and capacities

In this section, we try to analyze capabilities and capacities exploiting our previous analysis of functions. Critically, we will base this distinction on an ontological argument that goes beyond the perspective of ISO 15531-31[126], which proposes associating capacities with a quantitative viewpoint and capabilities with a qualitative viewpoint solely on practical grounds. Recall that the characteristics of a technical artifact constitute its physical makeup and determine its interactions with the environment. For instance, an individual pump is constructed in a manner that enables it to pump water at a certain flow rate. Both the physical attributes of the technical artifact and its 'ability' to perform a function can be conceptualized as DOLCE individual qualities, since they are, as all DOLCE qualities are, specifically dependent on their bearers, and also unique to their bearer (once a given capability/capacity-type is fixed), and this will be our choice.

It is natural to state, in light of our theory, that an entity possesses a capability if it can participate as a doer in an ontological function: an entity has a capability of a certain type (e.g., the capability of dividing) if and only if it can be the doer of an ontological function of corresponding type (e.g., it can be the divider in the 'divide' function). Hence, we posit that each ontological function-type defines a corresponding type of capabilities, and an entity has a capability only if it can perform (participate as a doer) the corresponding ontological function. These very abstract capabilities, which

could be termed 'ontological capabilities', can then be specialized, again reflecting what happens with functions, so that functions like cutting, loading, etc. can characterize cutting capability, loading capability, and so forth.

Another critical characteristic of capabilities and capacities, we argue, is their relational nature. For instance, one cannot discuss a device's capacity, like a machine's ability to process a certain number of items in a given time, without necessarily referencing (at least) another entity, for instance the (standard) item being processed. Relational qualities rely on an entity external to their possessor. However, the precise nature of this dependency varies across the examples that were made in Section 1.1. Specifically, in the case of capacities, the dependency is 'potential', as a device may possess the capacity to process a product even in the absence of the actual product. Conversely, regarding, say, relative distance, the dependency is 'actual', as a physical object is consistently positioned at a certain distance from another. This is a fine ontological point that we shall not delve into. Regardless, we assume that capabilities and capacities are relational qualities.

**Capabilities vs. capacities** Each realization of a capability depends on various external entities, in particular, on various qualities of the underlying artifact, which make possible the realization of the capability. Using our terminology it is (at the very least) specifically dependent on those qualities. For instance, the capability of a pump to exert pressure is dependent on having a flow rate capacity for some type of material. Returning to Gero and Qian's example of a water tap mentioned in Section 1.1, which bears similarity to that of the pump, we might state that the faucet has the capability of delivering water when requested, which is dependent on its flow rate capacity, which in turn is dependent on its diameter quality (Gero and Qian suggest that the flow rate is 'controlled' by the diameter [209]). Now, in this example, diameter is an intrinsic quality, whereas the flow rate is a relational quality. This suggests that 'capacities' are those relational qualities upon which a capability is dependent on, analogously to the approach in [34]; while capacities themselves are dependent on intrinsic qualities of the bearing object. The key idea is that capacities are relational qualities that offer insights into how the capability that is dependent on them can be realized in practice, and this is the key difference between capabilities and capacities.

Consider electronic components like resistors, transistors, and others, which are typically accompanied by datasheets detailing their technical specifications. These specifications often include properties such as failure rate, maximum temperature, current, or voltage that the component can reliably handle under standard conditions. Given the above discussion, all these properties are considered capacities: they can manifest only when the component is integrated into a functioning electrical circuit. As mentioned earlier, a prime example is voltage, which requires a fixed reference point for measurement due to its potential nature. These properties serve to parameterize certain capabilities of the component, such as the ability of a resistor to induce a voltage drop. For instance, the maximum operating current depends on the material properties of the conductor metal, such as its diameter and its resistivity.

Given the previous discussion, we characterize capacities as follows:

**ax18** $\texttt{Capacity}(x) \implies \texttt{RelationalQuality}(x) \wedge$
$\exists y(\texttt{SD}(x,y) \wedge \texttt{IntrinsicQuality}(y) \wedge$
$(\texttt{CP}(\texttt{topBearerOf}(y), \texttt{topBearerOf}(x)) \vee \texttt{CK}(\texttt{topBearerOf}(y), \texttt{topBearerOf}(x))))$

"A capacity ($x$) is a relational quality and is specifically dependent on some intrinsic quality, which is carried by the a bearer that is a constant part of (or a constant constituent of) $x$'s bearer." Note that, since CP is not antireflexive, this formula allows for the case that the bearer of the capacity and the bearer of the intrinsic quality are the same; it also allows for the case that the intrinsic quality is beared by some sub-part of the capacity's bearer (a sub-system of a functional complex, say).

**ax19** $\texttt{Capacity}(x) \implies \exists y(\texttt{Capability}(y) \wedge \texttt{SD}(y,x))$

"For each capacity there is a capability that is specifically dependent on that capacity."

In Formula (ax18), the $\texttt{topBearerOf}(\cdot)$ is the (top)[19] bearer of a quality, which is unique. We speak of 'top' bearer because in DOLCE contains both a direct quality relation (dqt), and its transitive, indirect version (qt). This allows for, and can only happen when, qualities inhere in other qualities. Thusly, a quality can indirectly inhere in more than one bearer (while the direct beare is unique, since the entity-quality relation in DOLCE is reverse-functional). Moreover, a chain of indirect inherences must always end with a unique bearer that is either a perdurant or an endurant (see [177, axioms Ad46, Ad47, Ad48]), we call said bearer as 'top bearer':

**df26** $\texttt{topBearerOf}(x) = \imath\, y(\texttt{qt}(x,y) \wedge (\texttt{PD}(y) \vee \texttt{ED}(y)))$[20]

"the (top) bearer of a quality is the perdurant or endurant that has the quality (the only one the quality inheres in)." For example, if $r$ is a rose, $c$ is the rose's color, and $b$ is the brightness of the color, then $b$ and $c$ directly inhere in $c$ and $r$ respectively, and the top bearer of $b$ is $r$. Instead, the direct bearer of $c$ is $r$, which is also the top bearer of $c$.

Axioms (ax18) and (ax19) characterize capacities only partially. A more throughout modelling would require further investigations. A possible such a way defines a capacity as a 'parameter' of the capabilities that depend on it, a statement that could be made precise as follows:

**ex14** $\texttt{Capacity}(x) \iff \texttt{RelationalQuality}(x) \wedge$
$\exists y, z(\texttt{SD}(x,y) \wedge \texttt{IntrinsicQuality}(y) \wedge \texttt{SD}(z,x) \wedge \texttt{Capability}(z)$
$\wedge\, \texttt{P}(\texttt{topBearerOf}(y), \texttt{topBearerOf}(x)) \wedge \texttt{P}(\texttt{topBearerOf}(x), \texttt{topBearerOf}(z)) \wedge$
$\forall u, t(\texttt{ql}(u,z,t) \implies \exists v(\texttt{ql}(v,y,t) \wedge \texttt{P}(v,u))))$

"Capacities are precisely those relational qualities that are specifically dependent on some intrinsic quality (whose top bearer is part of the capacity's top bearer), and

[19] We sometimes drop the 'top' part for simplicity.

[20] Note the analogy between topBearerOf and UFO's ultimateBearerOf [101, d3], indeed using the same definition structure for topBearerOf as for UFO-ultimateBearerOf (i.e. $\texttt{topBearerOf}(x) = \imath\, y(\texttt{qt}(x,y) \wedge (\neg\texttt{Q}(y)))$ ) would not change the predicate due to DOLCE's axiomatization.

on which some capability (whose top bearer contains the capacity's top bearer) depends, such that every time that the capability takes a range-value ($u$), the capacity takes a value ($v$) which is part of $u$." Note that, as in Formula (ax18), the case that all three top bearers are the same is allowed.

The formula says that capacities are the parameters of capabilities and gives an ontological foundation to practical approaches like that of ISO 15531-31[126]. However, this is only a possible interpretation that we do not fully commit to (also, further discussion on (ex14) will happen in Section 3.0.6).

### 3.0.5 Capabilities vs. functions

The ability to perform a function is a modal concept, entailing discussion of possible events. This is, after all, the reason for the various dispositional accounts of capabilities. Therefore, the relationship between capabilities and functions-occurrences (behaviors) of specific types could be expressed as follows (where $e$ represents a possible event, which may not necessarily be actual):

**ex15** $\exists c(\texttt{PumpingCapability}(c) \wedge \texttt{qt}(c, x)) \iff$
$\exists e, f, t(\texttt{participateAsDoer}(x, e, t) \wedge \texttt{PumpingProcess}(e) \wedge$
$\texttt{PossiblePerdurant}(e) \wedge \texttt{CL}(e, f) \wedge \texttt{ConvertEnergyIntoFluidMotion}(f))$[21]

"An object $x$ carries an individual capability of pumping if and only if it executes some possible perdurant of type 'pumping process' executing a function of type 'convert energy into fluid motion' ($f$) at some time."

Example (ex15) seems to suggest that capabilities depend on types of functions (not just processes or behaviors, for any behavior such as that of (ex15) will always execute a function, at least when speaking of capabilities of engineering interest), which are used in their definition. Such dependence can be captured by the definition-founding relation introduced in Section 3.0.2. Also, as said before, we treat capabilities as individual qualities[22]:

**ax20** $\texttt{Capability}(x) \implies \texttt{RelationalQuality}(x) \wedge$
$\exists y(\texttt{OntologicalFunction}(y) \wedge \texttt{defFoundedOn}(x, y))$

"$x$ is a capability only if it is a relational quality definition-founded on some ontological function."

Additionally, we suppose that each capability is characterized by at least one capacity (cf. (ax19)):

**ax21** $\texttt{Capability}(x) \implies \exists y(\texttt{Capacity}(y) \wedge \texttt{SD}(x, y))$

"A capability is specifically dependent on some capacity."

[21] Reformulations of this formula using modal logic are natural, however we do not indulge in them for the sake of simplicity.

[22] An alternative is to consider them dispositions, see [222], and [173] for this modelling path and a broader discussion on dispositions.

As mentioned earlier in this section, the underlying idea is that capabilities are relational qualities that link their bearers to ontological functions, necessitating a reference to those functions which is in our framework the definition-founded relation. It's worth noting that they are not instantiation-founded, as defined in (df11), since an artifact could possess the capability to function without actually doing so. Additionally, the function must be ontological rather than systemic because, otherwise, the bearing object would be *a priori* associated with a predetermined system, which is something that is not necessary when talking of capabilities.

Our characterization of capacities and capabilities ((ax18), (ax21), (df20)) explains:

- the strong connection between functions and capabilities (capabilities are definition-founded on ontological functions);
- the difference between capacities and capabilities (capabilities are specifically dependent on capacities, but not vice-versa).

The potential aspect remains unaccounted for, explicitly. Therefore, only a partial characterization of capabilities is supplied (and it is possible that additional aspects are still unaccounted for). One could think of extending DOLCE with a primitive subcategory for dispositions (or realizable qualities, or something similar) collecting exactly those qualities with a potential character. However, in that case, if an interesting ontological analysis is to be made of such a category, genuine issues would arise. For example, Guarino [92] argues that most qualities can actually be considered to have a dispositional character: for example, a color quality could manifest/realize when reflecting light, and even mass could be defined as the disposition to maintain a body's velocity. Guarino, instead, states that qualities can be more (e.g., fragility) or less (e.g., length) complex, and both simple and complex qualities could be seen as dispositional or not. Guarino's argument is (one) main reason that we do not introduce a disposition category. Instead, our approaches tries to order qualities by complexity using specific dependence, which is the relation mentioned by Guarino.

Coming to the distinction between capacities and capabilities on a quantitative vs qualitative basis, this is only marginally elucidated by our approach, which, instead, opts to differentiate capacities and capabilities on a hierarchical basis (using the order-like relations of dependence and parthood).

### 3.0.6 The issue of capacity and capability quality spaces

Having conceptualized capacities and capabilities as DOLCE-qualities, the issue of their corresponding conceptual spaces arises (in DOLCE each quality type is associated with a conceptual space called quality space). Up to our knowledge, this issue was never tackled before. Here we propose a possible, interesting approach coherent with the rest of our theory, however do note that the content of this section is marked for future work and needs a more detailed analysis. To go on, we have to give some details on the theory of conceptual spaces.

Gärdenfors' theory of conceptual spaces [77], on which DOLCE qualities and quality spaces are based, states that there is a correspondence between basic adjectives (e.g,

red, heavy, long) and regions in abstract, mathematical structures called conceptual spaces. For example, the biconical hue-saturation-intensity model for colors is a conceptual space and color is a region (subset, part) of the space. Gärdenfors also treats different linguistic elements in the same way, so that also verbs (e.g. 'walking') and nouns (e.g., 'apple') correspond to a region in some conceptual space. Each simple adjective, which Gärdenfors call property, correspond to a region within a conceptual space composed of *integral* dimensions, known as a domain. Complex nouns or verbs, referred to as concepts, correspond to regions in the product space of multiple domains. More specifically, Gärdenfors says that two dimensions are integral if "one cannot assign an object a value on one dimension without giving it a value on the other" [77][23], then a domain is a maximal set of integral dimensions and two dimensions that are not integral are separable. Gärdenfors emphasizes that regions corresponding to concepts are convex and the convexity is used to establish a link with prototype theory. Finally, Gärdenfors notes that abstract terms, such as 'democracy', may be difficult or impossible to analyze from this point of view.

Now, since we want to extend the conceptual space machinery to treat capabilities, it must be highlighted that Gärdenfors, and cognitive scientists more in general, aims to find structures that are characteristics of the human mind, and not necessarily to find structures that are ontologically-sound and/or useful for some task. Instead, we aim to model some aspects of the engineering domain and are more interested in the ontological soundness and usefulness of putative structures required to model capabilities. Additionally, Gärdenfors has already tried to model with conceptual spaces what he calls 'functional properties'[24], which he exemplifies with artifact functions and affordances (see Sections 3.0.6.1 and 2.1.8). In fact, after some initial doubts about the feasibility of this[25], he suggested to reduce functional properties to the actions that an object affords, which, in turn, are to be represented through an appropriate conceptual space for actions, itself based on dynamic force patterns [106]. Other authors follow a similar approach, for instance, Janowic and Raubal compare the moveability of desks with that of books based on spaces constructed using appropriate numerical descriptors of these affordances, then compare different entity types taking an average of the similarities between affordances characterizing the types [132].

While these approaches are interesting, they focuse on the actions of the human body (e.g., pushing, walking, running, throwing, etc.) and on affordances. Instead we focus on technical functions of designed artefacts and their capacities and capabilities, therefore we will not follow them verbatim. Instead, we start our analysis by noting that cognitive scientists, and Gärdenfors in particular, sometimes hypothesize that a

[23]Gärdenfors intends this not in the sense a dimension cannot exist without the other, but more in the sense that one dimension cannot be perceived, or has no cognitive existence, without the other. For example pitch and loudness are integral dimensions, though there is no logical/physical restriction on their coupling. However, the distinction is nuanced [78].

[24]We will adopt this term in the first part of this section, until we will give more precise terminological definitions.

[25]In [77] he warns that "it is clear that the theory of properties and concepts based on conceptual spaces, as developed in this book, applies primarily to natural kinds, and it may only be indirectly useful for an analysis of the other types [including functional properties]".

certain set of meaningful domains really is the one employed by the human mind to describe a (class of related) concept(s) (see e.g. [19, 77, 106]). To corroborate such a hypothesis, there is (at least) a fundamental argument: one can obtain similarity measures among individual examples of different concepts, e.g. by questionnaires, then employ multidimensional scaling[26] to determine a mapping between the examples and points in an $N$-dimensional vector space, and finally check if there is a correspondence between the vector space dimensions and the domain (see e.g. [19]). One can see, then, that the possibility of comparing entities belonging to a concept or property is a necessary and fundamental aspect of the conceptual space theory. Comparability between product types and, in particular, their capacities/capabilities is also an essential aspect of engineering practice. Therefore we wonder: can capabilities (and capacities) be compared among themselves, and if so how?

First, let us consider a counterexample: Borgo and Vieu [35] speak of *the* capacity of an object as an individual quality representing the sum of *all* the abilities of the object to behave in some way. For example a pen capacity includes the ability to fit in one's palm, write fluidly on paper, make a certain noise when crushed by a rock, etcetera. It seems very difficult to compare this BV-capacity[27] of an object to the one of another, due to the very abstract nature of such things, that apparently are an infinite sum of other entities, also complex and abstract. Even restricting to the case of comparing the BV-capacity of the same object at different times seems undoable: take for instance a round pen that has a clip at $t$, then, at $t' > t$ the clip is broken. It would seem that the pen's BV-capacity at $t'$ is smaller than at $t$, since the sum now lacks the addendum of being able to be clipped to something; however, it also has gained the ability to roll on an inclined plane since it is now the clip would not get in the way. Since one is not limited to useful behaviors of the pen, one could likely enumerate a host of useless, specific abilities gained by the clip-less pen, possibly even more than the useful behaviors. Therefore, a slight change in the object properties could result in a big, nonmonotonic change in the respective BV-capacities, rendering the feasibility of a meaningful comparison doubtful. Therefore, we argue, there cannot be any meaningful quality space associated with such BV-capacities and thus they are not DOLCE-qualities, at least not in the sense of [177].

Constraining the set of abilities to be somehow useful[28] may be necessary. Indeed, engineers can and do compare different systems, products, or services and the capabilities of these are a main point of comparison. In this context, the comparison is usually called benchmarking. Take, for sake of example, Quality Function Deployment (QFD) [268]: it is a methodology that aims to ensure the highest product quality as a result of design processes and can be used to (among other things) technical benchmarking different products, say to choose the better design option or to understand

[26] A set of algorithms that, given a predetermined number of dimensions ($N$, say), a vector norm, and a matrix of numbers representing similarity (or dissimilarity) between a set of (say, $M$) entities, assign to each of the $M$ entities an $N$-dimensional vector such that the dissimilarity between a couple of entities is very similar to their distance in the norm.

[27] Since Borgo and Vieu capacities are different from the capacities of this thesis, they are prefixed with 'BV-' in this section, for the sake of clarity.

[28] Borgo and Vieu speak of attributed capacities, see Section 3.0.9.

competitors advantages and weaknesses. One key tool of QFD is the House of Quality: a matrix that correlates the (qualitative) customer requirements with the (quantitative and measurable) product technical characteristics [112]. Each technical characteristic is weighted depending on how well it correlates to important customer requirements and its value in a given set of competitor products is assessed. This way, an existing product or a product-to-be can be compared against existing products (or other designs) and thoughtful decisions can be taken on how to increase its quality. Additionally, note that typically it happens that some technical characteristics correlate positively with some requirement, but negatively with some other requirement, entailing the presence of a trade-off between the maximization of the first requirement with the second. This is to be expected, since trade-offs are a natural, inescapable feature of engineering design. Now, some customer requirements do not consist of abilities[29] (e.g., 'being cheap') but those that are functional requirements do (e.g., for a vacuum cleaner robot, 'can move silently'), and we note a suggestive resemblance of these with capacities/capabilities. Moreover, we note an analogy with the method of Janowicz and Raubal of comparing entity types by using affordances and numerical affordance-descriptors [132] (the analogy would link product types with general entity types, functional requirements with affordances, and technical characteristics with affordance-descriptors).

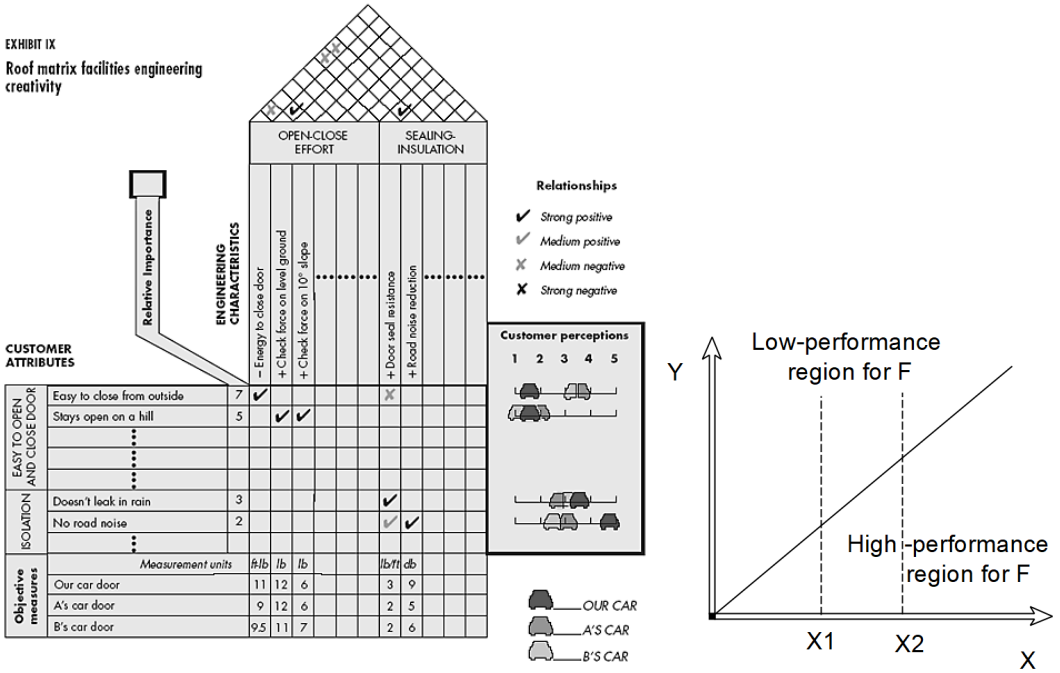
Given all this, it seems natural to try to compare capacities and capabilities by proxy of other, simpler qualities. However, we must be careful in the details of how this is done. In particular, we must ensure the ontological soundness and the compatibility with DOLCE of our approach, as well as its ability to inform essential engineering tasks such as trade-off evaluation and product comparison.

Now, in Axioms (ax19) and (ax21) we used specific dependence as the relation holding between capabilities and capacities, and capacities and intrinsic qualities. Therefore, before pointing to the exact nature of capacities and capabilities, we should wonder if the correlation relation is compatible with specific dependence, assuming that the items related by a House of Quality correlation matrix are indeed qualities.

Consider the example of a car with an 'easy to open and close door' requirement that is positively correlated with the '[low] energy to close the door' and negatively correlated with the 'door seal resistance', taken from Figure 3.4a. The correlation relation is $n$ to $m$ in general, in fact 'door seal resistance' is also positively correlated with the requirement 'isolation'. The fact that the relation is $n$ to $m$ suggests to exclude that it is interpretable with DOLCE's inherence relation, which is functional. This also excludes that the matrix relates qualities with the dimension-qualities describing the underlying quality space. Instead, it suggests that there is a relation of causal nature[30],

[29]Indeed, in typical use, the elements linked by the House of Quality seems to escape any systematic rule regarding their ontological categorization, since requirements can be described by adjectives, adjective-qualified nouns or processes, nouns of technical artifacts, etc., while technical characteristics can be physical qualities, subjective evaluations, number of technical artifacts, etc. In any case, note that we do not want to ontologize the QFD process, only take inspiration from it to produce an ontological modelling that takes into account some essential phenomena present in QFD.

[30]Note that the correlation relation in the matrix does not always appear to be a genuine causal relation: e.g. 'is light'/'is cheap' correlated with 'made of aluminum', 'second hand component' do not appear to be genuine causal relations. However, correlations regarding functional requirements do




(a) Example of House of Quality correlation matrix, taken from [112]. The columns correspond to technical characteristics, the rows to product requirements. Checks (crosses) in the cells indicate the presence of a positive (negative) correlation. Cells in the square grid relate requirements to characteristics, cells in the triangular grid relate characteristics among themselves. On the bottom there are evaluations of the characteristics of a few compared products.

(b) Example of the quality space of a functional property built upon the spaces of its correlated characteristics. $F$ is some functional property, whose performance is determined by the formula $X-Y$. The region $X-Y > 0$ determines high-performance, the region $X-Y < 0$ determines-low performance.

Figure 3.4

in the sense that when the seal resistance is high, this causes a high isolation value, while obstacling the action of opening or closing the door. More precisely, it appears that positive/negative correlation between two qualities is to be interpreted as a positive/negative[31] causal statement between instances of types of occurrences constituted by the change or stasis of the corresponding properties. Since these occurrences, when they are stasis, consist in states described by the fact that a certain quality has a certain value, we could reasonably call them 'simple states', modifying the terminology from Guarino and colleagues[32] [93]. Given the causal relation between stases, then,

appear to be of causal nature, we are interested in those. Consider also Note 29.

[31]These adjectives are intended in the sense of Toyoshima's causal ontology, which will be explained in Section 4.0.1.1. Positive would be 'normal' causation, where the cause facilitates the effect, negative causation is the case of causes that completely or partially hinder something from happening.

[32]In [93] simple events are either direct or indirect. Direct simple events are "the occurrence of a change (or a stasis) in an object with respect to one of its qualities", while in the indirect case the

necessarily, changes in a technical characteristic also cause changes in the correlated qualities and both changes appear to be examples of Guarino's simple events [93]. If the requirement-quality is a functional property (as in the car example), it should be interpreted as the property of being able to participate in a certain way to a certain type of process along the lines of (ex15). Then one can say, using our terminology, that the correlated qualities, in base of their value, contribute to causally determine the behavior of their bearer[33]. However, since in this section we are interested in qualities and since behavior-based explanations bring into question entities of potential/possible nature while quality-based explanations are actual, we proceed on by considering the case of causal links between qualities.

It appears that these causal relations characterize the corresponding product types, in the sense that, for each individual car of the analyzed type, this car has the qualities listed in the correlation matrix, and every time one of the technical-characteristic-qualities has a certain value (e.g. the door seal strength is high) this influences the values of the correlated qualities (e.g. the car isolation is high and the ability to open/close the door with ease is low). Therefore, we claim that the correlation relation is compatible with specific dependence, because if a requirement-quality is correlated with a technical characteristic, then every time the former quality is present the latter is also present. Specific dependency in the opposite direction may not hold, if one considers e.g. the case that a product completely loses a requirement-quality correlated to multiple technical-characteristic-qualities due to the loss of one of the latter (e.g. one cannot open the car door anymore due to the lock mechanism being broken, while the car seal has nominal strength). Moreover, the correlation relation appears to be strictly stronger than specific dependence due to its causal nature.

Let us come now to discuss the structure of the qualities spaces of these qualities, taking into account the presence of the causal relations discussed above. Technical characteristics are to be easy to model with simple spaces, like e.g. a single positive half-line with an adequate unity of measure for the seal strength, since they are prescribed to be (easily) quantifiable by the QFD methodology. Under certain cases, functional requirements may also be likewise easy to model (consider e.g. the case of 'stays open on a hill' from Figure 3.4a, which could be modeled simply with the graph of the function that links a given inclination level with the corresponding strength required to open the door). However, more typically, it appears a complex endeavor: in the cases that the functional property is associated with a human action or cognitively similar occurrences, one could follow the path of Gärdenfors and build a multidimensional conceptual space grounded on dynamic force patterns (see e.g. [83, 107] and Figure 3.5). However, engineers often discuss functional properties not directly reducible to human actions, for which this approach looks difficult (what would be the action space corresponding to, say, 'high-efficiency laser cutting'?). From the previous discussion, in particular from the efforts of Gärdenfors, and Janowicz and Raubal [77, 106, 132], and the example of QFD, it seem natural to build the quality spaces of functional properties from the spaces

changing quality inheres in a proper part of the object.

[33] Again, this does not hold for non-functional properties, for whom the correlation relation may also not be interpretable through genuine causation (cfr Note 30).

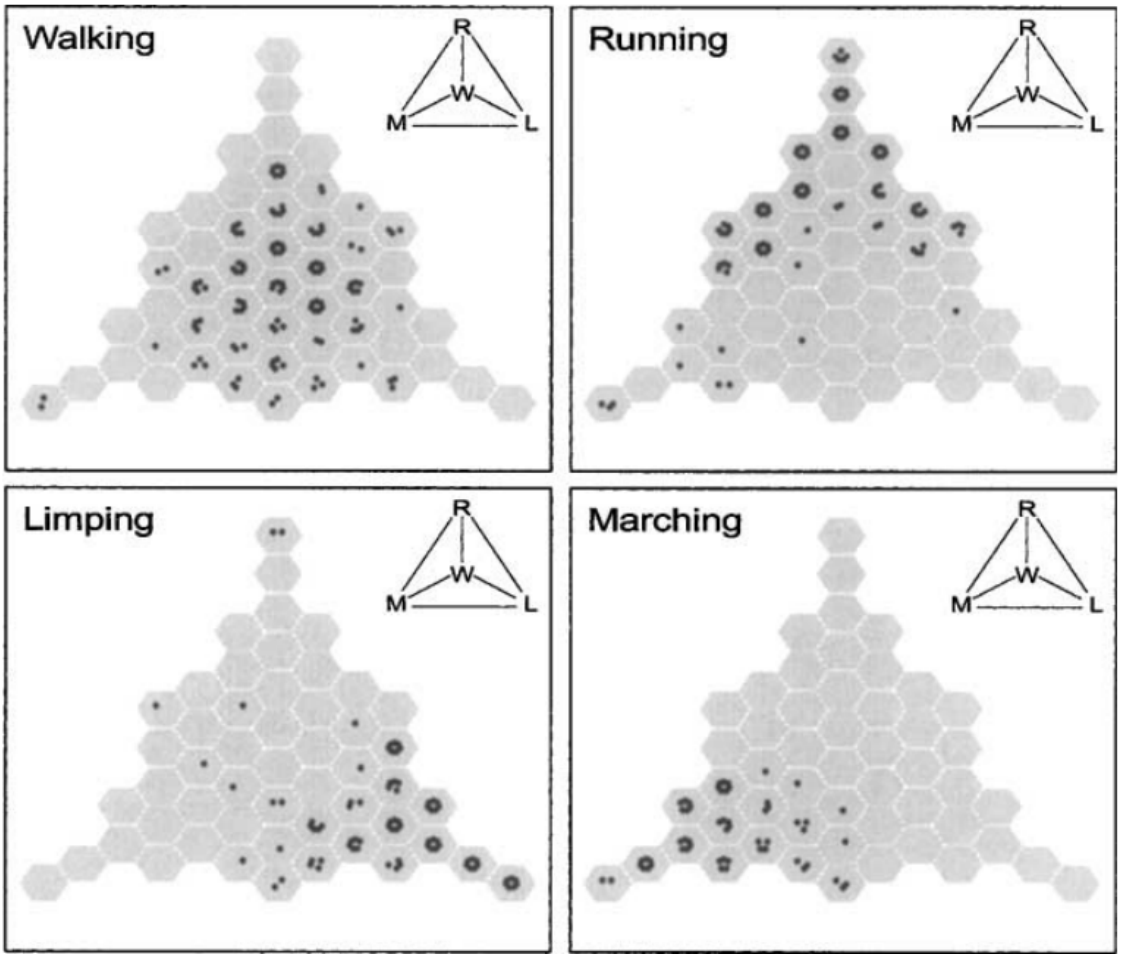



Figure 3.5: Action space for walking, limping, marching, and running. In the triangle there are four poles, one for a prototype of each action (walking in the center, the other three in the triangle vertices). The prototype are constituted by recordings of the actions performed by a subject, on which points corresponding to certain anatomical references were marked. Each of the other hexagonal cells also indicate a video showing a dynamical pattern of moving dots, this time obtained by weighted interpolation of the four prototypical patterns (the nearer to one of the poles, the higher the corresponding weight). The dots in the hexagon correspond to the number of test subjects that categorized the corresponding video as the type written in the upper-left corner of the small squares. The picture is taken from [83].

of the qualities they causally depend on. Let us see how. Take, for sake of example, a functional property $F$ of some product $P_1$, which is positively correlated with a technical characteristic $X$ and negatively correlated with the characteristic $Y$. Assume for simplicity that these are the only relevant correlated characteristics for $F$, that they take values on the positive real half-line, and the trade-off between the two is such that there is a high-performance region for $F$ when $X > Y$, and a low-performance region when $X < Y$. Then one would naturally consider as quality space for $F$ the cartesian product $\Pi \coloneqq [0, +\infty) \times [0, +\infty)$ endowed with an additional structure consisting in a (almost-)partition $\mathcal{P} \coloneqq \{\{(x, y) \in \Pi \mid x > y\}, \{(x, y) \in \Pi \mid x < y\}\}$ (see Figure 3.4b), which could be generalized to finer partitions if one desires a finer level of granularity for the performance regions. The ensuing space, call it $S_F$, also takes into account that there is (at least) an ordering between the partition elements, because we know that performance-wise it is better if $X > Y$ and worse if $X < Y$[34]. If there was another functional property, say $F'$, correlated to positively to $Y$ and negatively to $Y$ its quality space would be still built upon the product $\Pi$ and partition $\mathcal{P}$, but the partition member would be ordered in the opposite way w.r.to $S_F$, so that, in particular $S_F \neq S_{F'}$. This procedure could be generalized to more complex cases, it is sufficient to have some information about the how the performance of the functional property causally relates to the underlying qualities. Now, in the very simple example of $S_F$, one could also argue that $S_F$ is not made by $\Pi$ and $\mathcal{P}$, but, since $F$ is basically such that $F = X - Y$, one could say that the quality space of $F$ is simply the line of real numbers ($\mathbb{R}$), and the dependency $F = X - Y$ is something that holds in addition and independently from that. However, in more complex cases (think e.g. $F$ as a vector function of several components having additional independent variables, $F = (f(X, Y), f_2(X, Y, Z, \dots), f_3(X, Y, Z \dots), \dots)$, such that the range of $F$ is not a simple subset of $F$'s codomain[35]) it is probably more effective modeling-wise to focus on the qualities the functional properties causally depend on, and we are considering exactly these complex cases. Note that this and the previous considerations are analogus to similar considerations of Guizzardi and colleagues in [103]: they also privilege the explicit modeling of more basic properties (in approriate cases). Moreover, a strong analogy is that they also propose to do so especially in the case that there are causal relations between such properties (which would mainly be modes, in their language) and in that case causal properties should also be represented explicitely.

In any case, the resulting quality space structure may not be cognitively-founded in the same way that classical conceptual spaces are, but it encodes information about the quality correlations, including synergies and trade-offs. Moreover, it is founded on engineering domain knowledge, thus one could point to, say, appropriate engineering facts and methodologies to ground the quality space, in place of the cognitive features of the human mind.

[34]Note that, in the simple example of Figure 3.4b, the relation from $X, Y$ to $F$ is functional, however this does not need to be true in general. Regardless, when it is true one could also chose to identify $S_F$ with the graph of the functional relation plus some criteria to order the function values.

[35]Also note that a faithful mathematical model of the matrix in Figure 3.4a would resemble more this latter case than the simple $F = X - Y$, due to the high number of items.

However, comparability between so-obtained quality spaces raises some issues: suppose e.g. that there is second product type $P_2$, which has a functional property $F_2$, apparently of the same type as $F$ (e.g. $P_1$ is the car from above and $F$ is 'easy to open and close door', $P_2$ is an armored door that has as $F_2$ its own individual property of 'easy to open and close door'), then if we construct the quality space of $F_2$, it will be founded on a different set of technical characteristics (there is no seal strength, say). Since in DOLCE quality types correspond to quality spaces, we are forced to conclude that $F_2$ and $F$ actually belong to different types (they may still belong to the same category on a higher level of abstraction than the one of *types*, for instance they could belong to the same *category*[36] of qualities based on the fact that they are associated to the same *ontological function*). In particular, we observe that the comparability may be fully absent, partial, or complete depending on the quantity and importance, and relations holding w.r.to the shared characteristics. Therefore, if we accept that quality spaces of functional properties can be constructed in this way, we are forced to accept that their types are associated to the precise way the functional properties is achieved, suggesting a link with the way-of-achievements. In fact, we could strengthen Axiom (ax20) with the addition of:

**ax22** $\texttt{Capability}(x) \implies \exists y(\texttt{EngMethod}(y) \wedge \texttt{defFoundedOn}(x, y))$[37]

"capabilities are definition-founded on engineering methods."

We now need just one last ingredient to speak about capacities and capabilities: we need to interpret their quantitative vs qualitative duality. To do this, we go back to the difference between Gärdenfords-concepts and Gärdenfords-properties, itself based on the the fact that the conceptual spaces of the former are made of multiple, separable domains, the latter correspond to single domains (recall that domains are sets of integral dimensions), and we suggest the following:

**ex16** [38] "Capabilities are those physical qualities that (i) are relational qualities, (ii) signify the ability of their bearer to participate in a certain way in processes of a certain type, which specializes some ontological-function type, (iii) their corresponding quality space is built (in the way described above) upon the product of multiple, separable domains." In particular, they points (i) and (ii) entails the satisfaction of Axiom (ax20).

**ex17** "Capacities are those physical qualities that: (i) are relational qualities, (ii) characterize some aspect(s) of how their bearer is able to participate in a certain way in a process of a certain type, which specializes some ontological-function type,

[36] Consult e.g. [99] or [97] for a discussion on the differences between such meta-level terms as 'category' and 'type'.

[37] In passing, if capabilities are definition-founded on engineering methods and each engineering method is associated with an ontological function through its main-function role, it is natural to assume this entails that capabilities are definition-founded on ontological functions, so that this axiom entails the last conjunction of Axiom (ax20). However, since we supplied no axiomatization of `defFoundedOn`, this cannot be proven.

[38] Note that this definition and the ones below are given for sake of example, and we postpone our full commitment to them after some future work will be carried out to analyze them more holistically.

(iii) their corresponding quality space is built from a single domain."

Adding to this the postulates that:

**ex18** "Every capacity is causally dependent on some intrinsic quality."

**ex19** "Every capability is causally dependent on some capacity."

We deduce Axiom (ax21) (because causal-dependence would be a specialization of specific dependence) and we are one step away from Definition (ex14). In particular, the last conjunction of (ex14) is to be interpreted in the sense that every region in a capacity quality space is isomorphic to a region of the quality space of the corresponding capability[39], in the sense that the region $X_1 < X < X_2$ (call it $v'$) of the quality space of $X$ in the example of Figure 3.4b injects in the rectangle between the dotted lines (that is $\{(x, y) \in \Pi \mid X_1 < x < X_2\}$, call it $u$) since it is isomorphic to $\{(x, 0) \mid X_1 < x < X_2\}$ (call it $v$). Then the statement $\mathtt{P}(v, u)$ in (ex14) is interpreted with $v \subset u$. Note that our interpretation of quantitative and qualitative entails that capabilities are more complex than capacities[40] in the cognition-based sense of Gärdenfords. Also, note that in the case of capacities, one could chose to represent the quality space separately from the causal relation with simpler properties (e.g. representing $S_F$ above with $\mathbb{R}$ or representing the quality space of 'stays closed on a hill' with just a strength-required-to-open-space, separated from its dependency on the inclination level); putative guidelines for the choice of a modeling path over the other would require additional study and analysis.

Let us discuss some examples to showcase the ideas behind (ex16) and (ex17). In the example of the car door, the door seal strength is a capacity because (i) it is relational[41] in the sense that it exists to seal *something* by preventing the penetration of *something else* in that something (it is, in some way, existentially depended on some external entity), (ii) it characterizes one aspect of how its bearer[42] can perform the function of sealing, (iii) its quality space can be modeled with a single positive half-line with some strength unit. Additionally, it also satisfies the postulate (ex19), because there are some intrinsic physicochemical properties that causally determine the strength value.

Similarly, the 'stays closed on a hill' property from Figure 3.4a is a capacity because, briefly, (i) it makes sense only in the context of an environment, (ii) it characterizes how its bearer can participate in the 'easy to open and close' process type, (iii) its quality space can be modeled with an integral domain (e.g. the graph of the function linking the inclination level to the strength required to open the door, as mentioned above). The 'easy to open and close' property looks more similar to a capability, instead, because its quality space appears to be non-reducible to a single domain: in fact, in Figure 3.4a it is 'decomposed' into various sub-properties, (at least some of) which seems to be

[39]The existence of that capability is postulated in (ex14) itself.

[40]This does not necessarily mean that a capacity is easy to represent, since a domain can be, in principle, a complex, highly-dimensional space.

[41]We recognize that the relational/intrinsic duality is sometimes fuzzy, and likely depends on the precise way an agent conceptualize the underlying quality and its relation with the environment. However, said duality is a classic presence in philosophy, and a foundational analysis of how (and if) to un-fuzzy it is beyond the scope of our analysis.

[42]Either the seal, the door, or the car depending on the conceptualization details.

capacities. The quality space of 'easy to open and close' can then be built from the product spaces of the 'decomposing' properties endowed with a structure taking into account the causal relations between these properties themselves and between these properties and the performance of the opening/closing process.

As a final example, consider the property of color: for Gärdenfords it is linked to an integral domain, the biconical hue-brightness-intensity color space. Coherently, color is the archetypical example of DOLCE-quality type. Prima facie, color looks like an intrinsic quality, inhering in its bearer irrespective of any external entity. However, consider e.g. the case of commercial carbon nanotube-based black coatings [166]. The objects coated with these products are extremely black, by virtue of the coating structure, and this blackness is the main feature of the products. To the physicist engineering this feature, the coating color appears to be the result of a complex phenomena involving the interaction of light waves and the material structure. Is this color the same quality as that whose quale is perceived by a casual observer? The issue is complex, but it appears that the previous discussion suggests a negative answer: to the casual observer the black object has a color which is a DOLCE-quality taking values in regions of the biconical color space very near to the black pole; instead, the physicist conceptualizes a color quality that is capacity, which should probably be better called 'reflectance', that takes values in the region of "uniformly low reflectance over a broad range of wavelengths from the visible to far infrared" [166], and is causally determined by "the intrinsic properties of graphitic material as well as the morphology (density, thickness, disorder, and tube size)" [166]. The quality space of the latter should then be different from the biconical color space, since it should also take into account the causal relations among reflectivity and the intrinsic properties.

We postpone further analysis on the quality spaces and causal relations between capacities and capabilities to further work, and start looking to the topic of malfunctions.

**Performing badly versus not being able to: losing, and gaining capabilities** Since we are also interested in malfunctions (see Chapter 4), we want to address bad-performing and broken objects as well. In our framework, it is natural to identify the performance of an object (w.r.to a function) as the value taken by its corresponding capability in its quality space (which is also determined by the values of certain, appropriate capacities as postulated in (ex19)).

Given this identification, poor performance can be explained in two qualitatively different ways: in the first way, the quality space of the capability can be divided into different subregions, corresponding to various degrees of performance, so that if a performance-value falls in (is part of a) a low-performance region, we can say that the object is performing bad. For example, a system of coupled gears has the capability of transmitting and/or amplifying or reducing torque, and is therefore able to participate (in a certain way) to appropriate torque-transmitting processes. We assume that the capability quality space has multiple domains, coherently with (ex16), and we focus, for the sake of simplicity, on a single domain corresponding to the gears backlash[43].

[43]The gap between the teeth of coupled gears that cause them to move independently of each other

The latter could also be identified as the quality space of a certain capacity (e.g., 'the capacity to backlash', as discussed above) on which the capability is dependent on, since it can be modeled with a single integral domain: e.g. some arc of circumference marked with, say, sexagesimal degrees as a unit of measure[44]. Depending on the specific application[45], a subregion of the space may be specified as the region of correct performance (say, the interval $0.14^\circ \pm 0.1^\circ$) and another as the region of degraded performance (say, the interval $[0.15^\circ, 0.18^\circ]$), yet another as the region of unacceptable performance (say, $[0.18^\circ, \infty)$. Coupled gears with a backlash value of $0.15^\circ$at time $t_0$ will be said to have an acceptable performance, if at time $t_1 > t_0$ the backlash has increased to $0.17^\circ$due to wear, the gear system will be said to have a degraded performance[46].

In the second way, an object can completely lose its ability to perform a function. This happens when the capability of the object is lost. That is, the object at that time does not have the capability-quality. Continuing the example above, if a gear system breaks, say the axis of one toothed wheel fractured and the wheel fell off, the gear system does not have the capability of transmitting torque anymore. Indeed, in that case it would be meaningless to speak of the backlash of the missing wheel w.r.to other wheels, so quality space values could not be assigned, and also the capability could not be present, since is specifically-dependent on the backlash-quality. An object that has lost its capability to perform a desired function will typically be disposed of or repaired. In the case it is repaired, it recovers the capability quality. As remarked in the introduction (Section 1.1), in this case the time extension of the capability life is non-convex.

Let us formalize the gear example to clarify: assume that there is a simple gear system[47], $(s)$, a gear train made by four tooted-wheel gears $(g_1, g_2, g_3, g_4)$ coupled by meshing teeth ($g_1$ with $g_2$ and $g_3$ with $g_4$) or co-axiality ($g_2$ with $g_3$). The only properties of the system considered are the backlash between the gears: $b_{12}, b_{34}$. The capability of the system of transmitting and amplifying torque is called $c$. We assume that at $t_0$ the value of $b_{12}$ and $b_{34}$ is $0.12^\circ$, then the value of $b_{12}$ decreases and at time $t_1 > t_0$ is $0.14^\circ$, while $b_{34}$ value stays the same. Then, at time $t_2 > t_1$ the axis of $g_1$ breaks and that wheel falls off; later, at $t_3 > t_2$ the wheel has been repaired and the backlash has been regulated back to $0.12^\circ$. Then the following holds at every time:

**ex20** $\mathtt{Q}(b_{12}) \wedge \mathtt{Q}(b_{34}) \wedge \mathtt{Q}(c) \wedge_{i=1,2,3,4} \mathtt{POB}(g_i) \wedge \mathtt{POB}(s) \wedge \mathtt{PR}(0.12^\circ) \wedge \mathtt{PR}(0.14^\circ)$

> "The backlashes and the capability are qualities, the gears and the gear system are physical objects, the backlash values are physical regions."

for a small distance when starting or inverting motion.

[44] In this case, we are separating the backlash-capacity space from its dependencies (e.g., the teeth shape, the distance between the gears, etc.) and leaving the latter implicit, again for the sake of simplicity.

[45] The ideal backlash level is application dependent.

[46] In passing, note that even an apparently simple quality as backlash could be argued to be a disposition, since it is usually relevant only during start and inversion of rotating motion. As argued before, this ambiguity of many qualities if they are or not potential is one reason we do not introduce any category of realizable qualities and instead focus on the characterization of capabilities and capacities.

[47] This example is roughly inspired by `https://khkgears.net/new/gear_knowledge/gear_technical_reference/gear_backlash.html`, Section 6.4, last accessed in November 2024.

**ex21** $\mathtt{qt}(b_{12}, g_1 + g_2) \wedge \mathtt{qt}(b_{34}, g_3 + g_4) \wedge \mathtt{qt}(c, s)$

"The backlash $b_{12}$ ($b_{34}$) inheres in the mereological sum of the first (last) two gears, the capability $c$ inheres in the gear system."

Instead, the relations that hold at some time are:

**ex22** $\wedge_{i=2,3,4}\mathtt{P}(g_i, s, t_0 + t_1 + t_2 + t_3) \wedge \mathtt{P}(g_1, s, t_0 + t_1 + t_3) \wedge \neg\mathtt{P}(g_1, s, t_2)$

"The last three gears ($g_2, g_3, g_4$) are part of the system at all times, while the first gear ($g_1$) is part of the system at all times except for $t_2$."

**ex23** $\mathtt{ql}(0.12^\circ, b_{12}, t_0+t_3) \wedge \mathtt{ql}(0.14^\circ, b_{12}, t_1) \wedge \mathtt{ql}(0.12^\circ, b_{34}, t_0+t_1+t_2+t_3) \wedge \neg\mathtt{PRE}(b_{12}, t_3)$

"The quale of $b_{34}$ at all times is $0.12^\circ$; while $b_{12}$ has this value at $t_0$ and $t_3$, that changes to $0.14^\circ$at $t_1$, while $b_{12}$ itself is not present at $t_2$ (therefore, it also does not have a value)." Regarding the values of $c$, we assume they are completely determined by the gear backlash. This is, of course, a simplification for the sake of this example as gear-system performance in general depends on more factors (also if there were no other factors we would be forced to declare $c$ a capacity and not a capability by account of our own definitions (ex16) and (ex17)). Therefore, the value of $c$ is a function of the value of the backlashes at those times. Precisely, assuming that the ratio between the diameters of the second and first, and fourth and third gears is 0.5 in both cases, the formula is $c = b_{12} + 0.25 * b_{34}$, which is the value of the 'composite' backlash affecting the system. Therefore at times $t_0$ and $t_3$ the value of $c$ is $0.15^\circ$, increasing to $0.17^\circ$ at time $t_1$ (if more factors were to be considered, these two values would be only part of the quale of $c$ at the corresponding times). At time $t_2$ the capability is not present, therefore it has no value:

**ex24** $\mathtt{ql}(0.15^\circ, c, t_0 + t_3) \wedge \mathtt{ql}(0.17^\circ, c, t_1) \wedge \neg\mathtt{PRE}(c, t_2)$

"At times $t_0$ and $t_3$ the value of $c$ is $0.15^\circ$, at time $t_1$ the value is $0.17^\circ$ and at time $t_2$ $c$ is not present."

Adopting the choice of performance regions of the previous example, we can say that the gear train has nominal performance at $t_0$ and $t_3$, poor performance at $t_1$, and is broken at $t_2$.

**Subsumption between capability classes** Since in this section we discuss capabilities in relation to their conceptual spaces, it is natural to wonder about how conceptual spaces influence the subsumption of capability classes. For example, it is natural to say that the capability of 'being able of moving' is a generalization of 'being able to move silently', which in turn is specialized by 'being able to move silently and smoothly from A to B in less than 10 seconds', etcetera. Can these generalizations be explained in terms of quality spaces associated to quality spaces? To answer we have to consider first how are the conceptual spaces associated to such quality classes (if they exist at all). Such putative conceptual spaces cannot be built using qualities inhering in a capability bearer, because the construction we proposed in the previous sections depends on the precise way in which is the moving is realized (see e.g. Axiom (ax22)): these conceptual

spaces do not have the necessary generality to be associated to generic classes such as 'being able to move silently'. One could still think that, since each moving occurrence has temporal qualities (such as the time extension of the occurrence, if the moving action was smooth or clunky, if it was silent or noisy, any attribute that can be attributed to the occurrence itself), such temporal qualities could be used as dimensions of the conceptual space. Then subsumption hierarchies of capability classes could be explained by restricting the allowed quality values. However, adding temporal qualities dimensions to the capability quality spaces (which are otherwise physical) would make the regions of the ensuing space heterogeneous: some of its parts would be temporal regions, some other parts would be physical regions. But temporal regions are closed w.r.to parthood in DOLCE, and the same holds for physical regions. Morevoer, in DOLCE objects cannot have qualities that take values in temporal regions, only in physical regions. Therefore, the possibility of heterogeneous quality space should be rejected.Since we don't see other ways to associate a conceptual space to such capability classes, we are forced to state that these capability classes are not associated to conceptual spaces, and, therefore, they are not qualities types. They must be a more general, weaker, sort of quality-classes. Since these classes appear to be rigid, we opt to consider them categories [97]. So that, though one can group capability-types depending on the type of function they execute into categories, and these categories can be specialized into other categories by specializing the properties of the function execution, these classes are not associated with quality spaces. As a corollary, there cannot be individual qualities belonging 'just' to a category such 'being able to move', but they must always belong to a type since individual qualities do have a quality space, for instance 'being able to move as a vacuum cleaner robot' is such a type and its conceptual space is built using some qualities corresponding to vacuum cleaner robots. In passing, note that Guarino and colleagues in [93] make a similar argument, although in their terminology they call 'kind' what we say type: transposing their argument here, they would likely say that 'being able to move as a vacuum cleaner robot' is an event-kind(type) because it supplies 'individuation criteria', while this does not hold for 'being able to move'.

consider the stabbing of Caesar by Brutus and the ensuing death of Caesar and state that 'killing' and 'stabbing' individuate the same event because "from the fact that a killing event occurred we get no information on the nature of such event". The anlogous to stabbing would be 'to move water using gear pumps' and 'to move water' would be analogous to killing. Then, if one espouses [93] and takes $z$ to be the instance of 'to move water' spation-temporally coincident with $x$ and $y$, it must be that $z = x$.

**Summary.** To summarize:

- Functional properties are difficult to model with the tool of conceptual spaces. We suggest that, at least in certain cases, it is appropriate to build their conceptual space from the spaces of the properties they causally depend on, along the lines of $S_F$ from Figure 3.4b and the other examples discussed.

- In particular, capabilities and capacities quality spaces can be also built in this way, and the difference is that capability quality spaces are more complex (since

they are composed by multiple domains), while capacity quality spaces are simpler (a single domain). In this sense capabilities are more similar to Gärdenfords concepts than to Gärdenfords properties. For the rest both are relational qualities and both (albeit in different ways) predicate about how their bearer participates in certain processes, and capabilities causally depend on capacities (see (ex17) to (ex19))

- In this framework, poor performance phenomena can be modeled by either locating a capability quale in a designed subregion of the quality space, or by eliminating the capability itself (see the backlash example above).

- Additionally, a capability type can only be the capability of executing a function *in a certain way* and individual capabilities must always instantiate a capability type. Capabilities of 'just' executing a certain function are subcategories of the category of capabilities and are not associated with quality spaces. Such categories can be into subclasses by specifying qualities of the function execution, but these subclasses remain categories, not types.

#### 3.0.6.1 Affordances

Affordances, which are described (in engineering) as "interaction[s] between artifact and user in which properties of the artifact offer a potential use to the user" [171] (see also the end of Section 2.1.8), form another important class of functional properties. They are, however, less interesting for this thesis, which focuses more on functional decomposition and less so on, say, human-machine interaction. In any case, we suggest a possible way to relate them to capabilities:

**ex25** "If an engineering artifact affords a user (or another artifact) to do something then the user (or the other artifact) has the capability to that something and the engineering artifact's affordance is its relational quality corresponding to the user's (or another artifact's) capability. Corresponding in the sense that, if the Earth's distance to the Moon is a relational quality of the Earth, the *corresponding* relational quality is the Moon's distance from the Earth."

From this perspective, our analysis of capabilities may provide some insight into affordances.

Of course, it must be noted that our brief discussion of affordances is just a passing mention and is marginal in the context of this thesis. The interested reader may benefit of more serious ontological analyses, such as that of Toyoshimas and colleagues [241].

### 3.0.7 Other relevant meanings of functions: essential and accidental functions

In the previous sections we discussed some possible meanings of 'function', that we termed systemic functions, solved functions, and ontological functions. As mentioned

in Section 2.1.7, we detailed these three meanings to describe some of the concepts that engineers associate to functions, not to claim that these three concepts are the correct ones or even just the only ones. Indeed, there are many others relevant concepts of functions that are significantly different from these three. For example, let us consider the properties of intentionality (is there an agent that attributes the function to something?) and essentiality (is the function essential for some entity?). Ontological functions are types of events (for what concerns us in this thesis, types that are relevant for engineers), therefore, intentionality is only slightly relevant for them. Additionally, while being classified as a given ontological function type is an essential property of an event (e.g., a 'divide' event is necessarily so), the ontological function type is not by itself a property ascribed to some other entity. Systemic functions do relate to other entities (the system component $y$ in (df16)), however, they are only roles of the component behavior, so their attribution to the component or to its behavior is contextual, not essential. In addition, in their definition, intentionality has a background character, being hidden in the goal $g$. Lastly, what holds for systemic functions holds also for solved functions since we defined the latter to be systemic-functions as-implemented by an engineering method. Since essentiality and intentionality are clearly important in the topic of functionality (consider e.g. that for both Jansen and Smith functions are essential properties of their bearers, and are not roles for these reasons [133]), it appears that the triplet of concepts defined up to here are severely lacking from the point of view of the range of meaning that they cover.

To ameliorate this issue, we want to introduce another concept of function characterized by these two properties. From Section 2.1.5, essential functions are typically explained using an etiological/intentional approach, that is, in the engineering domain context, they are essential because they were ascribed to an entity as essential properties by an agent, usually the designer. However, etiological/intentional accounts are known to lack the desiderata 'No-epiphenomenalism'/'Structure' [218, 258], because they usually do not explain the strong link between the structure of the function bearer and the function ascription to such a bearer. However, one can mix causal and intentional accounts to solve this issue, as done e.g. by Houkes and Vermaas with their ICE theory [121]. Taking inspiration from this, and also from previous DOLCE-based works such as [35], we introduce the following definition (where `intentionalSel` is a predicate describing an event in which an agent intentionally selects a certain quality of some entity):

**df27** $\texttt{hasFunction}(x, f) \iff \exists a, q \texttt{intentionalSel}(a, e, x, q) \wedge \texttt{defFoundedOn}(q, f) \wedge \texttt{Capability}(q) \wedge \texttt{qt}(q, x)$

> "An entity has/bears/carries/has ascribed a function iff there is some agent ($a$) that intentionally selects (though event $e$) a capability ($q$) of that entity that makes the entity able of executing the function." Here we express the link between the capability and the function with the definition-founding relation, as was done before. Notice that $f$ may be an ontological function, but also some more specific class, such as a systemic function.

We call any such function a 'selected function':

**df28** $\texttt{SelectedFunction}(f) \iff \exists x(\texttt{hasFunction}(x, f))$

> "Selected functions are, by definition, exactly those attributed to an entity by an agent."

By specialization of the agent type in the definition, selected function can be specialized. For instance, we can talk of 'design function', or 'user function', depending on the role of the agent (designer or user). One usually gives special importance to the design function and may call it *the* proper function (or essential function) of an object, while a user function is called accidental. However, Definition (df28) does not change when changing the role of the agent doing the selection, so that it is difficult to believe that in one case the selected function is essential, while in the other is accidental. The usual argument applied in these cases is to strengthen the definition of selected function by saying that a selected function is an essential function when its bearer[48] came to be precisely with the goal of executing the essential function (for instance, since a product is manufactured exactly for the goal of executing the function ascribed to it by its designer, the product's design function is essential; on the other hand, if a user makes use of the product to execute a different function –using a hammer as a door stop, say– this latter user function is accidental, because the product was not manufactured for that). However, this argument is not without weaknesses: imagine, say, the case that a drug is designed to cure a certain illness and it is therefore manufactured and sold to that scope. Later, the same non-redesigned drug is discovered to have the capability of curing another, different condition, and is also manufactured and commercialized to (possibly only) that new goal[49]. Note that this phenomenon, known as drug repositioning, is not discountable since it was estimated that in 2014 repositioned drugs accounted for a quarter of the revenue of the whole pharmaceutic industry [135]. Which is then the function to execute whom the drug came to be, the one it was designed for or the one it was manufactured and sold for? Borgo and Vieu would likely claim that there are three relevant types of entities: two different drug-artefacts, the first born from the selection of the capability to cure the first illness, and the second from the selection of the capability to cure the other illness, both constituted by the physical-drug-object or substance the drug is made (which would be the $y$ in Definition (df37)). Then, these three entities may be instantiated by co-located individuals: the constituent is always co-located with the constituted, and even two different individual drug-artefacts could be co-located, if different agents select the same underlying object or substance. Finally, each drug-artefact bears its selected function as an essential property (quality), while the constituent object or substance does not bear a selected function as a quality

This approach explains which selected function is essential to which artefact, but (i) it requires a multiplicativist point of view, which may or may not be desirable and (ii) it does not account for accidental functions, since it is symmetric w.r.to user and design function. Point (i) depends on the ontological commitments one wishes to make and its discussion is out of the scope of this section, while point (ii) could be hand-

[48] Of course, in our framework, the function is not a quality and does not have a bearer. The quality is the selected capability, and in this sentence we use a metonymy for the sake of simplicity.

[49] Aspirin is the typical example of this phenomenon.

waved by saying that there are different types of agents and of selection events, and some are more relevant than others (e.g., designers, manufacturers, and the design and manufacturing processes are more important/essential than any capability-selection by a user), ignoring the cases when there is an ambiguity or conflict in the important types (as in the drug-related example above), and say that, informally:

**df29** a selected function ($f'$) is an accidental function of the artefact $a$ if and only if it is a user function (more generally, a selected function of a less important agent-type) borne by a different artefact ($a'$) co-located with the first artefact ($a$) that bears a design function ($f$, more generally, a selected function of a more important agent-type)[50].

After all, etiological/intentional accounts are considered to deal easily with the distinction between essential and accidental functions [218], so it seems excessive worrying about such issue in the context of an intentional account. Moreover, the reason to consider accidental functions in the engineering domain is mainly to discuss unintended uses of a product by the users: depending on the context, the designer may need to discourage, or prevent or reduce possible side-effects of the unintended uses.

In passing, note also that systemic functions look similar to selected functions, since both are attributed to objects, either by the `hasFunction` or by the `systemicFunctionOf` relation. Additionally, systemic functions are also linked implicitly to an agent (the one supplying the goal $g$ in Definition (df16)). However, they are different concept classes. The biggest differences are that systemic functions depend on a system and selected functions do not; and selected functions depend on a selection event, while systemic functions are contextualized by a causal analysis of a system behavior. For example, consider the hydraulic system in Figure 3.3: 'convert mechanical energy to fluid pressure for Cyl1' is a systemic function of pump P1, while the function selected by, say, the pump's manufacturer is 'to convert mechanical energy to fluid pressure'. In this example, all realizations of the systemic function are realizations of the selected functions. One could then extrapolate

**ex26** $\texttt{systemicFunctionOf}(\texttt{f}, x) \wedge \texttt{hasFunction}(x, \texttt{f'}) \implies \texttt{subConceptOf}(\texttt{f}, \texttt{f'})$

"If a systemic function and a selected function are attributed to the same entity, the systemic functions is a sub-concept of the selected function."

However, while this may be the typical case, it entails some kind of concordance between the goal for the system and the intention of the agent selecting the desired properties of the object. This needs not to be always the case, therefore we do not commit to this formula in our system.

Note that the function bearer (the $x$ in Definition (df27)) is usually intended to be an object, like an engine or a pump, say. However, some authors question that this is a necessity. For instance, Burek [39] argues that any kind of entity can bear a function: a flight process may have the function of transporting goods, the color quality value of a moth has the function of camouflaging. Our framework may be extended to functions

[50]Note that in this definition we assume that $f$ and $f'$ specialize different ontological-function-types, otherwise $a$ and $a'$ would have been the same artefact.

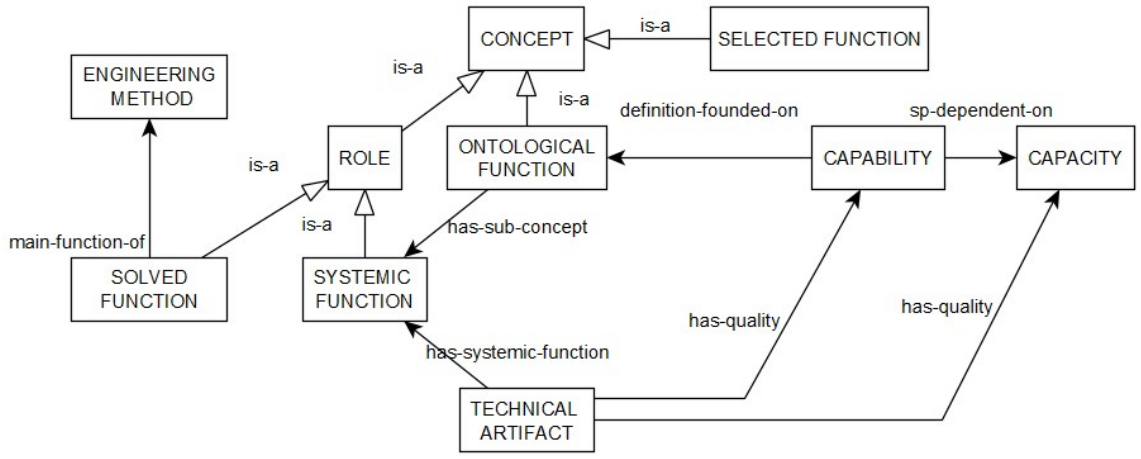



Figure 3.6: The primary concepts and relationships explained in this section at (class level only). For simplicity the diagram is based on the corresponding OWL model. Note that a filled black arrow between between classes indicates that the object property corresponding to the edge's label enjoys an existential restriction axiom between the classes (e.g. any solved function is main-function-of some engineering method).

of these other categories may very well be possible, but it would require further work. For instance, one would have to study the attribution of capabilities to such categories as quality values and processes. Depending on the case this may sound acceptable ('flying has the capability of transporting goods') or dubious ('this gray-brown color has the capability of camouflaging'). In DOLCE, it would also entail that capabilities may be physical, temporal, or abstract qualities depending on their bearer. Since these are strong ontological commitments, which are also not necessary to model the cases we focus on, we do not explore such possibilities.

As a final note, we add a picture that shows a schema indicating the main relations and concepts used in our theory of engineering functions in Figure 3.9.

### 3.0.8 Relations holding between functions

Having discussed in the previous sections the function concepts considered by this thesis, in this section we move to the relations holding between these concepts. The main references for this topic we consider here are Burek's Ph.D. thesis [39], which proposes an overall theory of such relations; Vermaas and Garbacz discussion about functional decomposition [255]; and FBRL-related literature about metafunctions [142].

The most common semantic relations in modeling are instantiation (Socrates is a man), specialization (or subsumption, e.g. man are mortals), aggregation (parthood, constitution, composition), and attribution (associating a property to an object). These relations typically hold between concepts named with nouns. Indeed, in linguistic, subsumption corresponds to hypernymy/hyponymy and parthood to holonymy/meronymy. However, when discussing functions, which are typically named with verbs, using these

relations may not be as appropriate as in the case of noun-concepts. In fact, while WordNet also uses meronymy and hyponymy as semantic relations between nouns, WordNet uses entailment, troponymy, causation, and presupposition to relate verbs [72]. A verb 'to $X$' entails a verb 'to $Y$' if for each individual $X$ing there is a corresponding $Y$ing. Depending on the relation holding between the time extension of the verb-executions, different kinds of entailment are defined: if the temporal extensions are the same one talks of troponymy, which replaces hyponymy for verbs[51] (to $X$ is a troponym of to $Y$ if to $X$ is to $Y$ in some manner, e.g. 'to lisp' is a troponym of 'to talk'). If one temporal extension precedes the other the relation is either backward presupposition (e.g., to forget presupposes to know) or causation (to feed causes to eat) If the temporal extension of $X$ing properly includes or is included into that of the $Y$ing there is no special name and one just talks of entailment. For example to snore entails to sleep and is contained in the time extension of sleeping, while to buy entails to pay, but contains the time extension of paying.

Since verbs naturally describe types of perdurants, including perdurants that are behaviors, and moreover we used verbs to denote the various function concepts we introduced, it is natural to expect that WordNet semantic relations holding between English verbs correspond to the relations between functions. In particular, since in our framework functions classify behaviors perdurants, any relation holding between behaviors can be lifted to a relation between functions by an appropriate composition with the classification relation.

DOLCE defines various such relations holding between perdurants, some of which were described already in Section 1.1. The main ones are parthood (a 'John traveled by car from Rome to Paris' – call it $X$ – has as part 'John crossed the Alps'), temporal inclusion ('Mary traveled from Rome to Turin immediately after John's departure' is temporally included in $X$), and spatial inclusion ('the engine of John's car had low oil level the whole time' is spatially included in $X$). These are all specializations of specific dependence. Similarly, by introducing ordering relations between time intervals (Allen's algebra, say), DOLCE could define temporal precedence and subsequence. In this way, we argue, all WordNet semantic verb relations would be covered by DOLCE: entailment is the specific dependence between perdurants, while WordNet-causation and precondition between perdurants could be defined as follows, (assuming a time-ordering relation is present and written as $\leq$ – it could e.g. be the union of Allen's precedes and meet relations):

**df30** $\texttt{precedes}(x, y) \Longleftrightarrow \exists t, t'(\texttt{ql}(t, x)_T \wedge \texttt{ql}(t', y)_T \wedge t \leq t')$

"$x$ precedes $y$ if the time extension of $x$ precedes that of $y$."

**df31** $\texttt{follows}(x, y) \Longleftrightarrow \texttt{precedes}(y, x)$

"$x$ follows $y$ if and only if $y$ precedes $x$."

Finally, troponymy can be arguably interpreted as either subsumption between perdurant-types or as spatio-temporal coincidence: in the first case any lisping is also a talking,

[51] Perhaps confusingly, hypernymy is maintained, meaning that the relation hypernym/hyponym for nouns is called hypernym/troponym for verbs.

in the second case a lisping can be co-located with a talking and these two actions can be different individual entities.

Having noted this, how are these relations translated to functions? To answer, we group the relations in taxonomical, dependence-like, parthood, and causal. We start the discussion with taxonomical relations. Dependence and parthood will then follow. However, notice that the discussion of causal relation will be postponed in the next chapter (in Section 4.1.4) because it needs Toyoshima's causal ontology therein discussed.

#### 3.0.8.1 Taxonomical relations between functions

Specialization of functions can happen in various ways, focusing on various different semantic elements. For instance, a 'to move' ontological function could be specialized by type of operand ('to move a fluid', 'to move water'), by goal ('to move water in the tank'), by precondition ('to move water from the lake to the tank'), by temporal properties of the underlying endurant ('to move water from the lake to the tank in less than 1 hour'). Burek introduces different kinds of specialization relations for all these cases [39], we don't, as that level of granularity does not directly benefit our framework. However, we do make two critical distinctions (that Burek does not highlight): in case of the semantic element that specializes the event-type is the engineering method (or means, or way) of execution, i.e. the way-of-achievement in SOFAST-speech (e.g., 'to move water using gear pumps'); and in case the semantic element is (part of) a system (as in the examples above where a specific lake or tank is mentioned). In the latter case we speak of systemic functions, which specialize more general function-categories. In the former case, there is an explicit relation between the function, which is termed `SolvedFunction`, and the engineering method: the solved function is the `mainFunctionOf` the method. Then, the solved-function-type is subsumed by a more general function-type (in particular, by an ontological function: each 'to move water using gear pumps' is a 'to move water'), but is not subsumed by the method. Instead, if we assume that engineering methods can have instances, then any instantiation of a 'to move water using gear pumps'-function is spatio-temporally coincident with an instance of 'using gear pumps'. Since DOLCE does not force spatio-temporally coincident perdurants to be the same, one has two choices: either the two descriptions correspond to different perdurants or they indicate the same perdurant. Both are compatible with our framework, but we don't make an assumption either way: we only assume that solved functions specialize ontological functions and are related by spatio-temporal coincidence with engineering methods (cf. Axiom (ax17)):

**ax23** $\texttt{CL}(x, \texttt{main}^{\texttt{m}}) \implies \exists y(\texttt{CL}(y, \texttt{m}) \wedge x \approx_{ST} y)$

> "The main function of an engineering method is such that any of its realizations is spatio-temporally coincident with an occurrence of the engineering method." Again, this requires assuming that methods can be instantiated. For instance, if methods are conceptualized similarly as plans, this means to assume that they can

be executed in a similar way as plans are (cfr. DOLCE-ultralite[52]).

In passing, the work in [93] is very much relevant here: let us take a certain instance of 'to move water using gear pumps' and call it $x$. Let us also call $y$ the corresponding instance of 'using gear pumps' that is coincident with $x$. Then the issue in the paragraph above is to decide if $x = y$ holds or not. In the framework of [93], $x$ and $y$ are both 'complex events', i.e. they are made from sets of simpler 'simple events', and to decide if they are the same or not one should check if they are made by the same set of simple events. Simple events are changes in a single quality of a certain entity, so that we should focus on what are the qualities that change in $x$ and $y$: if the same qualities change in both, then they are made from the same set of simple events and $x = y$; if some quality-change differentiates them, then they are different. Now, there is a prominent participant that could be most relevant in this matter: the water. That is, the water position surely is relevant in $x$, this is less sure for $y$. In particular, one could see an analogy with an example from [93], where that work says that in a gesticualting event the heart of the gesticulator does not participate in the event: "the heart of the gesticualating person is not a participant in the gesticulating event, despite the fact that a proper heart beating necessarily must occur when such an event occurs. This is because the heart beat is not cognitively relevant for describing how the gesticulation event occurred". Accepting that this is true, the question now becomes 'is water movement cognitively relevant for describing the using-gear-pumps event'. Formulated this way, and committing to the assumption that the gear pumps in questions are not strictly water-only pumps, it appears that the answer is 'no', and, therefore, what explained above entails that $x! = y$. However, the above line of reasoning needs various requirements to work, first of all the commitment to the work in [93]. This is compatible with our framework, but we shy away from adopting such a work due to our design choice of minimal commitment. As a further digression, we observe there is another example in [93] that has an obvious analogue in this section: Guarino and colleagues consider the stabbing of Caesar by Brutus and the ensuing death of Caesar and state that 'killing' and 'stabbing' individuate the same event because "from the fact that a killing event occurred we get no information on the nature of such event". The anlogous to stabbing would be 'to move water using gear pumps' and 'to move water' would be analogous to killing. Then, if one espouses [93] and takes $z$ to be the instance of 'to move water' spation-temporally coincident with $x$ and $y$, it must be that $z = x$.

Coming back to the topic of taxonomical relations among functions, in summary, the only taxonomical relation that we consider between functions is sub-concept-of (as in Axiom (ax12)), though other more specific relations could be considered. In addition, we consider a spatio-temporal coincidence between solved functions and corresponding engineering methods (ax23)

#### 3.0.8.2 Dependence-like relations between functions

We come now to dependence-like relations. By dependence-like, we mean a relation that is defined similarly to specific existential dependence. Specific existential depen-

[52] `http://www.loa.istc.cnr.it/index.php/dolce/`.

dence (between two individuals) would be Definition (df1), while specific existential dependence holds between two classes (say $\Phi$ and $\Psi$) iff for all instances of the first class there is an instance of the second class on which the instance of the first class depends:

**ex27** $\mathtt{GD}(\Phi, \Psi) \Longleftrightarrow \forall x \Phi(x) \implies (\exists y \Psi(y) \wedge \mathtt{SD}(x, y))$

By 'similar' we mean that we consider both syntactical variations and specializations of formulas (df1) and (ex27) to be dependence-like relations.

Using this definition, spatio-temporal coincidence is one example of dependence-like relation (because it specializes specific dependence), another example is the relation between a method and its subfunctions. `precedes` and `follows` are also important examples of dependence-like relations. The oncoming definitions (df32) to (df35) are also examples of dependence-like relations since they are syntactical variations of (ex27). Definitions (df31) and (df30) hold among perdurants, but can be lifted to types of perdurants and reified concepts in the obvious way (in the following $\Phi$ and $\Psi$ are perdurant-types and `c` and `d` are DOLCE concepts):

**df32** $\mathtt{precedes}(\Phi, \Psi) \Longleftrightarrow \forall x (\Psi(x) \implies \exists y (\Phi(y) \wedge \mathtt{precedes}(y, x)))$

"$\Phi$ precedes $\Psi$ if for every instance of $\Psi$ there is an instance of $\Phi$ that precedes it."

**df33** $\mathtt{follows}(\Phi, \Psi) \Longleftrightarrow \forall x (\Psi(x) \implies \exists y (\Phi(y) \wedge \mathtt{follows}(y, x)))$

"$\Phi$ follows $\Psi$ if for every instance of $\Psi$ there is an instance of $\Phi$ that follows it."

Mutata mutatis, the previous definitions hold for reified concepts:

**df34** $\mathtt{precedes}(\mathtt{c}, \mathtt{d}) \Longleftrightarrow \forall x (\mathtt{CL}(x, \mathtt{d}) \implies \exists y (\mathtt{CL}(y, \mathtt{c}) \wedge \mathtt{precedes}(y, x)))$

"`c` precedes `d` if for every $x$ classified by `d` there is some $y$ classified by `c` that precedes $x$."

**df35** $\mathtt{follows}(\mathtt{c}, \mathtt{d}) \Longleftrightarrow \forall x (\mathtt{CL}(x, \mathtt{d}) \implies \exists y (\mathtt{CL}(y, \mathtt{c}) \wedge \mathtt{follows}(y, x)))$

"`c` follows `d` if for every $x$ classified by `d` there is some $y$ classified by `c` that follows $x$."

Notice that $\mathtt{precedes}(\Phi, \Psi)$ does not entail $\mathtt{follows}(\Psi, \Phi)$, while this is true on an instance level.

We used 'follows' and 'precedes' as terms to remark the difference between other causal relations that were introduced and will be introduced in the thesis. In fact, at a type-level, the first argument of WordNet-presuppose can be interpreted as a necessary precondition for the second, while the second argument of WordNet-causes can be interpreted either as physically/logically-necessary consequence (e.g., to die follows to kill) or as a probabilistic consequence (e.g., to feed an animal, in a broad sense, does not necessarily entail that the fed animal eats: it could refuse to eat) of the first argument. Our `follows` and `precedes` can reproduce these meanings, in any case we will touch upon causal relations more in depth later, after Section 2.2.1, so we postpone the discussion.

`follows` and `precedes` relations between functions are common, at least between systemic functions: for instance, the presence of a flow-arrow between two functions

in a FB model, entails that any realization of the target systemic function follows a realization of the source systemic function. Similarly, in the context of a functional decomposition, if the subfunctions are arranged in a sequence, that sequence is ordered by precedes/follows relations.

The same relations are less common among functions at the type level (ontological functions level): they are present if there is a couple of ontological function types such that every time a systemic function specializing the first type is realized by a behavior-perdurant, such behavior is preceded or followed by another behavior instantiating a function of the second type. However, it is difficult for this to happen, because given a system in which (ontological) functions are realized in a certain temporal order, it is easy to find a different system in which the arrangement between (ontological) functions is different. For instance, consider the function 'transport water from an A to a B': it could be preceded by an instance of a 'transport water from a C to an A' but this is not a necessity, since the water could have been there all along (location A is a lake, say). Similarly, a 'transport water from an A to a B' may be followed by a 'rotate turbine' function (e.g., if the water is in a hydroelectric plant), but also by an 'irrigate fields' function (e.g., if the water is part of an irrigation system of a farm). Indeed, it appears that a `follows` or `precedes` relation holding between ontological-function-types independently of a given, implemented system would constrain the design space of any product, since only certain combinations of function would be possible, and the same holds for spatial and temporal inclusion and coincidence.

Since such constraints could help with the design process, it would be unsurprising to find bodies of work dealing with such a topic. In fact, we know of at least one such body of work: Function grammar rules. Consider for example [233, 234], those works aim to automatize design leveraging the Functional Basis methodology, in particular they aim to automatize the development of Functional Basis graphs. They do so by establishing constraints on functions and flows so that the design space size is reduced. These rules are distillate from domain knowledge and reverse engineering of various products. One such rule is depicted in Figure 3.7 and states that every time an electric energy flow is imported into a system, it is later transmitted and actuated. In our speech we can say that this rule entails that the type 'import electric energy' is followed by the type 'transmit electric energy'.

#### 3.0.8.3 Parthood relations among functions

Decomposition of a main-function into subfunctions is usually considered akin to a parthood relation. We already discussed the relations between a main-function and a method, and between a method and a subfunction. In our framework these relations are not parthood-like, but (only) dependence-like. However, this does not say much about the nature of the relation holding between a main-function and any of its sub-functions: for one, dependence does not exclude nor entail parthood, additionally, the composition of the main-function-to-method and method-to-sub-function relations may have different properties than its two constituents. We mention this because researchers have argued the impossibility of viewing such relation as a parthood relation [256]. In

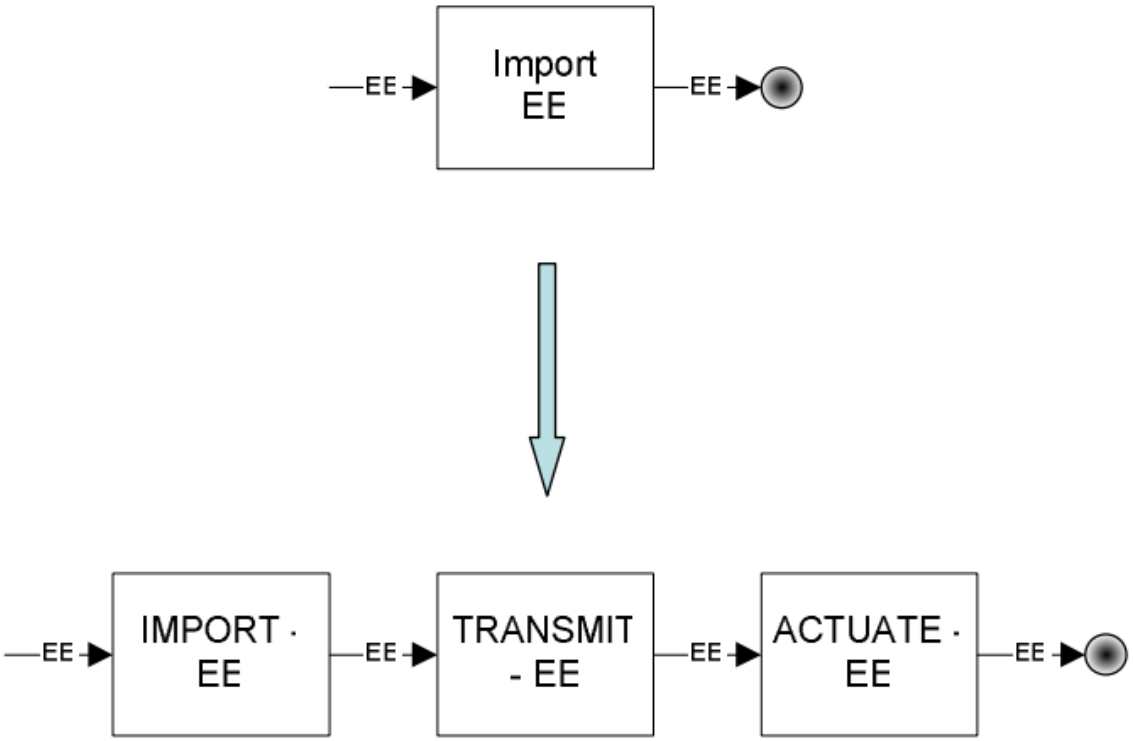



Figure 3.7: A rule for constraining the design space when using the Functional Basis methodology, taken from [233]

this section we will therefore analyze such discussion in relation to our framework and explain a few possible ways to introduce parthood relations among functions.

First of all, note that Vermaas distinguishes two types of hypothetical (non-temporalized) parthood relations: one between "function-tokens", and one between "function-types". Let us call them $*P_{in}(\cdot,\cdot)$ and $*P_{ty}(\cdot,\cdot)$ (starred predicates and axioms belong to other frameworks, or that in any case are used just for sake of discussion, exemplification, or mapping). The type level relation is lifted by the instance level by mere possibility:

**ax24*** $\forall \mathtt{f}, \mathtt{f}'(*P_{in}(\mathtt{f},\mathtt{f}') \wedge \mathtt{F}(\mathtt{f}) \wedge \mathtt{F}'(\mathtt{f}') \implies *P_{ty}(\mathtt{F},\mathtt{F}'))$ where $\mathtt{f},\mathtt{f}'$ are function-token, $\mathtt{F},\mathtt{F}'$ are corresponding function-types

> "If a function token of some function type is (a token-level) part of some function token of some second type, then the first function type is (a type-level) part of the second function type."

Moreover, Vermaas follows closely the Systemic Design and Functional Basis literature and identifies function-token with couples of finite sets of flow-tokens. For instance, if $a, b, c, d$ are flow-tokens a function is $< \{a,b\},\{c,d\} >$, and another is $< c,b >$ (removing singleton brackets for brevity). Flows have types, and a function-type is determined by the types of its input and output flows. With the convention that flow-tokens identified by the same letters (e.g., $c$ and $c'$) belong to the same type, an example of two functions of the same type is $< c,b >$ and $< c',b' >$. We also convene that different letters entail different flow-tokens. Finally, Vermaas defines the composition ($*Comp(\cdot,\cdot)$, a binary operation) of two functions as the function whose inputs are the union of the inputs of the composed functions, minus the inputs of one function that are also outputs of the other; and whose outputs are the union of the outputs of the composed functions, minus the outputs one function that are inputs of the other. Therefore, for instance, $< a,b >= *Comp(< a,c >,< c,b >)$, and $< \{a,b,a',b',e',f'\},\{d,d'\} >= *Comp(< \{a,b\},\{c,d\} >,< \{c,a',b',e',f'\},\{d'\} >)$, $< c,b >= *Comp(< a,b >,< c,a >)$, etcetera.

Given this formalization, Vermaas supplies various counter-arguments to the thesis that $*P_{in}$ and $*P_{ty}$ are parthood relations linked to the $*Comp$ relation. The main one is as follows:

- assume, by absurd, that the composition relation is linked with the instance-level parthood relation in the natural way:

**ax25*** $\mathtt{f} = *Comp(\mathtt{f}_1,\mathtt{f}_2) \implies *P_{in}(\mathtt{f}_1,\mathtt{f}) \wedge *P_{in}(\mathtt{f}_2,\mathtt{f})$.

- Consider then the functions $\mathtt{f}_1 =< a,b >$, $\mathtt{f}_2 =< c,d >$, $\mathtt{f}_3 =< d,c >$.
- Define $\mathtt{f}_4 \Longleftrightarrow< \{a,c\},\{b,d\} >$. Then, $\mathtt{f}_4 = *Comp(\mathtt{f}_1,\mathtt{f}_2)$, therefore $*P_{in}(\mathtt{f}_1,\mathtt{f}_4)$.
- Note that $*Comp(\mathtt{f}_4,\mathtt{f}_3) = \mathtt{f}_1$. Therefore, $*P_{in}(\mathtt{f}_4,\mathtt{f}_1)$.
- By antisymmetry of parthood, $\mathtt{f}_1 = \mathtt{f}_4$. A contradiction, since we assumed that different letters correspond to different tokens.

This counterexample can be read as a functional model of a chemical plant in which a liquid ($a$) is transformed to another liquid ($b$) through a reaction taking place in a tank ($\mathtt{f}_1$), if such reaction also needs a stream of air entering the tank ($c$) and leaving the tank, polluted by some gases ($d$), before being depurated and sent back to the tank ($\mathtt{f}_3$)

Now, regardless of specific examples, Vermaas composition relation manifestly has properties that make it unsuited for functional decomposition: for one it is commutative: $*Comp(\mathtt{f}, \mathtt{f}') = *Comp(\mathtt{f}', \mathtt{f})$, while, in general, the order or function realization matters. It is also degenerate, in the sense that a null-function is admitted, which can be decomposed into non-null functions: $*Comp(< c, d >, < d, c >) =< \emptyset, \emptyset >$, that is, the result of doing various things may be doing nothing. Therefore, our basic argument is that Vermaas formalizes an algebra of boxes with input/output arrows, but does not formalize a composition relation between functions. Note that this argument is independent of stating that the composition of solved functions is linked to a parthood relation or not.

Let us see how our framework would deal with Vermaas' chemical plant example. First, let us refer to the type of liquid $a$ as $A$, the type of liquid $b$ as $B$, the type of the purified air as $C$ and of the polluted air as $D$, and the type of chemical reaction happening in the tank as $CH$. For Vermaas, token functions are made up from token flows, and token flows are "specific flow[s] that [occur] only once. A token flow of electrical energy $a$, for instance, is a flow of energy that exists within one specific period in time and at one specific place in space" [256]. Therefore, in our framework, the token functions $\mathtt{f}_i (i = 1, 2, 3, 4)$ must be interpreted as particular behaviors, and the token flows as particular participants to those behaviors. Some of these behaviors are also classified by corresponding individual functions. Those individual functions can belong to the categories we introduced, such as `SolvedFunctions`, `OntologicalFunctions`, or `SystemicFunctions` (multiple individual functions of different categories can classify the same behavior). Intuitively, $\mathtt{f}_2$ should not be considered a function, since it is an unwanted side effect. However, we can still introduce a perdurant-type to classify it: 'to pollute air using reaction $CH$' ($\mathbf{poll_{air,CH}}$, say). Additionally, we introduce another perdurant, $\mathtt{f}'_2$, part of $\mathtt{f}_2$, of type 'to inject air in the liquid to facilitate reaction $CH$' ($\mathbf{inj_{air,CH}}$), which is a function. We also introduce three ontological functions: 'to transform' ($\mathbf{tr}$), 'to transform liquid $A$ to liquid $B$' ($\mathbf{tr_{AB}}$), 'to depurate air' ($\mathbf{dep_{air}}$); one engineering method: 'use chemical reaction $CH$ facilitated by injecting back the depurated gases from $CH$ in the solution' ($\mathbf{m_{CH}}$), with corresponding main function 'to transform liquid $A$ to liquid $B$ using chemical reaction $CH$' ($\mathbf{tr_{AB,CH}}$), and subfunctions 'to depurate air polluted by reaction $CH$' ($\mathbf{dep_{air,CH}}$). Then, the following relations hold:

**ex28** $\mathtt{CL}(\mathtt{f}_1, \mathbf{tr_{AB,CH}}) \wedge \mathtt{subConceptOf}(\mathbf{tr_{AB,CH}}, \mathbf{tr_{AB}}) \wedge \mathtt{subConceptOf}(\mathbf{tr_{AB}}, \mathbf{tr}) \wedge \mathtt{CL}(\mathtt{f_3}, \mathbf{dep_{air,CH}}) \wedge \mathtt{subConceptOf}(\mathbf{dep_{air,CH}}, \mathbf{dep_{air}}) \wedge \mathtt{CL}(\mathtt{f'_2}, \mathbf{inj_{air,CH}})$

"$\mathtt{f}_1$ is classified by $\mathbf{tr_{AB,CH}}$, which is a subconcept of $\mathbf{tr_{AB}}$, itself a subconcept of $\mathbf{tr}$. Additionally, $\mathtt{f}_3$ is classified by $\mathbf{dep_{air,CH}}$ which is a subconcept of $\mathbf{dep_{air}}$, and $\mathtt{f}'_2$ id classified by $\mathbf{inj_{air,CH}}$."

**ex29** $\mathtt{P}(\mathtt{f}_2, \mathtt{f}'_2)$

"$\mathtt{f}_2$ is part of $\mathtt{f}'_2$."

**ex30** $\mathtt{mainFunctionOf}(\mathtt{f}_1, \mathbf{m_{CH}}) \wedge \mathtt{subFunctionOf}(\mathbf{f_2}, \mathbf{m_{CH}}) \wedge \mathtt{subFunctionOf}(\mathbf{f_3}, \mathbf{m_{CH}})$

"$\mathtt{f}_1$ is the main function of the method $bfm_{CH}$, which also has $\mathbf{f_2}$ and $\mathbf{f_3}$ as sub-functions."

Regarding $\mathtt{f}_4$, it could be described as an occurrence of the class 'to transform liquid $A$ to liquid $B$ using chemical reaction $CH$ *and* to pollute air using reaction $CH$'. Such a class is, however, not a natural perdurant type (nor a function, since polluting is not a linked to a goal for some agent in this context), and $\mathtt{f}_4$ is more naturally described as the sum of $\mathtt{f}_1$ and $\mathtt{f}_2$ using DOLCE-perdurant mereology: $\mathtt{f}_4 = \mathtt{f}_1 + \mathtt{f}_2$.

Indeed, we never introduced a composition relation among functions in our framework, only a *decomposition* relation. Still, we go on replacing Vermaas' composition with perdurant sum: for one it is natural to do so and the sum is also symmetric, moreover Axiom (ax25)* becomes a classical property of the mereological sum[53].

Now, assume that behavior $\mathtt{f}_1$ is a DOLCE-event resulted by the culmination of a process that kept going continuously during a certain time interval $t$, delimited, say, by two time periods of inactivity due to scheduled maintenance. Then $\mathtt{f}_2$, $\mathtt{f}_3$, and $\mathtt{f}_4$ are also events ongoing that took place during $t$. Therefore, all these four perdurants are temporally coincident. Do some parthood relations hold among the behaviors? For sure, by definition of mereological sum, $\mathtt{P}(\mathtt{f}_2, \mathtt{f}_4)$ and $\mathtt{P}(\mathtt{f}_1, \mathtt{f}_4)$; and we also stipulated that $\mathtt{P}(\mathtt{f}'_2, \mathtt{f}_2)$. For the rest, DOLCE does not commit strongly to how parthood between perdurants should be interpreted, and leaves open the possibility that events may be spatial, temporally, or spatiotemporally coincident and still be different. Neither is specified any correspondence between perdurant parthood and spatial, temporal, or spatiotemporal inclusion (except that the time extension of the parts is included in the time extension of the corresponding wholes). Therefore, we have some freedom in modelling. We chose to state that the air pollution and depuration are part of the liquid transformation: $\mathtt{P}(\mathtt{f}_2, \mathtt{f}_1)$, $\mathtt{P}(\mathtt{f}_3, \mathtt{f}_1)$. In particular, they are spatial parts since they are temporally coincident with their whole. However, $\mathtt{P}(\mathtt{f}_2, \mathtt{f}_1)$ plus the definition of $\mathtt{f}_4$ as the sum of $\mathtt{f}_2$ and $\mathtt{f}_1$ entail that $\mathtt{f}_1 = \mathtt{f}_4$, like in Vermaas' proof.

Is this a contradiction? It is not. The key point is that in Vermaas' framework the identity criteria for tokens is equality of the sets of their input and output flows, that was the reason $\mathtt{f}_1$ had to be different from $\mathtt{f}_4$. On the other hand, DOLCE's perdurants do not necessarily coincide if they have the same participants. Indeed, both $cstf_4$ and $cstf_1$ are participated by $a, b, c$ and $d$, let us see why: $cstf'_2$ reasonably is participated by the purified air ($c$) only (at any given instant $t'$ in the interval $t$, $c$ is made from the amount of purified air present at $t$, note that this amount varies with $t$, so $c$ is more correctly not a constant but a function of time). $cstf_2$ is participated by the purified air ($c$) and by the polluted air ($d$, note that also $d$ is a function of time), $cstf_3$ is again participated by $c$ and $d$. Finally, both $cstf_4$ and $cstf_1$ are participated by $a, b, c$ and

[53] I.e., that the mereological sum of some entities has those entities as parts.

$d$ because participants of parts are also participants of the whole (Axioms $A_d58$ from [175]), moreover, since the air and the polluted gases are integral parts of the conversion process, it is only natural that they are both participants in $\mathtt{f}_1$.

Moreover, in our framework, the $\mathtt{f}_1$ is decomposed into $\mathtt{f}'_2$ and $\mathtt{f}_3$: $\mathtt{decom}(\mathtt{f}_1; \mathtt{f}'_2, \mathtt{f}_3)$. As discussed above, Axiom (ax25)* (with $*P_{in}$ replaced with DOLCE's $\mathtt{P}$) can be accepted without contradiction. Note that such an axiom may be accepted but we do not commit to accept it in this framework. Additionally, the other two criticalities of Vermaas' composition relation, symmetry and degeneracy, are also bypassed in our framework: the `decom` relation, as it has been defined, is not symmetric (it appears symmetric in the example above, but it is not so in general). Regarding degeneracy, we must first establish what a null event is in DOLCE: if null events are events with no participants, then these do not exist in DOLCE, since in DOLCE every event has some participant at some time, thus the `decom` relation cannot be degenerate. If null events have participants, then they should be interpreted as stasis, i.e. non-changes (e.g. 'nothing happened to my garden yesterday'), then these null events should be properly interpreted as DOLCE-states. Due to the homeomericity of the category of DOLCE-states, any part[54] of the stasis should also be a null event, therefore a null event cannot be decomposed into non-null events, provided that we adopt (ax25).

Regarding the relation of parthood between function types, we come instead to the same conclusion of Vermaas: if it is defined by Axiom (ax24)* (mutata mutatis to adhere with our notation), then it cannot be a parthood relation. The reason is simple: the above example and (ax24)* entail, in particular, that $*P_{ty}(\mathbf{dep_{air}}, \mathbf{tr})$. It is then sufficient that a 'to depurate air' function is decomposed into a series of functions containing at least one function of type 'to convert' (and the existence of such decomposition is certainly possible if not certain given the variety of devices and functional decompositions in any given engineering discipline) to have that $*P_{ty}(\mathbf{tr}, \mathbf{dep_{air}})$ and thus $\mathbf{dep_{air}} = \mathbf{tr}$, which we reject as absurd since it is an equality between two different concepts.

**Summary.** We claim Vermaas' proof for the impossibility of token-level function parthood does not apply in our framework. The key point is that the identity criteria of Vermaas' function-token are different from the identity criteria of the behaviors of our framework, which are DOLCE-perdurants. Additionally, while Vermaas composition relation is symmetric ($*Comp(\mathtt{f}, \mathtt{f}') = *Comp(\mathtt{f}', \mathtt{f})$) and degenerate (there are non-null functions that compose to a null function), our decomposition relation is neither. Therefore, (ax25)* may be accepted without contradiction, though we do not commit to it.

On the other hand, in our system as in that of Vermaas, type-level parthood between functions defined from token-level parthood by mere possibility is not acceptable.

[54] Actually, any *temporal* part of the stasis. However, this is a technical subtlety and given that, in DOLCE, for any part of a perdurant there is a corresponding temporal part (consider e.g. the fusion of all the parts temporally included in the starting part: said fusion is the smallest temporal part containing the starting part), we can assume that the parts mentioned in our argument are temporal parts, and our application of homeomericity is valid.

In addition, note that the issue of functional parts of organisms and artefacts is an important topic, which is different from but connected with the topic of parthood between functions addressed in this section. It is also a topic outside this analysis, however, for the sake of completeness, we refer the interested reader to the works [100, 242].

### 3.0.9 Comparison with some previous formalizations

Starred predicates are taken from other works.

**Comparison with Masolo's Social Roles** The attentive reader could have noticed that while we base our approach to roles on (parts) of the work of Masolo et al. on social roles [179], we consciously maintain some differences. Masolo defines roles as antirigid and founded concepts. Regarding rigidity, we already mentioned at the start of Section 3.0.2 that, while for Masolo roles classify endurants only, we allow roles to classifies perdurants, and that in that case the antirigidity of roles can be obtained by considering a modality other than time. Regarding roles being founded, for Masolo this means that roles are (i) generically existentially dependent on some other, external, concept and (ii) definitionally-dependent on the same concept from (i). Generic existential dependence on an external concept would be the same as our `instantiationFoundedOn`, modulo some minor aesthetic differences in the formalization. Instead, definitional dependence is analogous to our definition-founded relation. A first difference is that we consider (i) and (ii) separately, to be more flexible: depending on the precise conceptualization, a role-player could sometimes be present while no instance of the concept on which the role definitionally-depends is present (as in the example of the doctor and patient roles).

Another difference is that we introduce definition-founding only informally, while Masolo defines definitional-dependence by means of other, primitive predicates:

**ex31**$^{*55}$ $^{*}definitionallyDependentOn(x, y) \iff \exists d\, ^{*}definedBy(x, d) \wedge {}^{*}usedBy(y, d)$

> "The concept $x$ is definitionally dependent on the concept $y$ if and only if there is some description ($d$) that defines $x$ and uses $y$."

Where defined-by is a relation linking a concept to the one description that defines it, which always exists and is unique in Masolo's role theory, and where used-by links a concept to a description that contains the concept in some form (e.g. the Italian constitution, if conceptualized as a description, uses various concepts, including that of 'republic'). We did not adopt such a formal machinery, because it would bring into play various topics and issues that are tangential to this thesis. However, we argue that the theory developed in this thesis could be extended along the lines of (ex31), by identifying definition-founding with definitional dependence without any issue.

Actually, some issues do seem to appear with the ontological categories of the relata: for Masolo definitional dependence must relate concepts only, while for us definition-founding seems to sometimes relate concepts (as in the doctor and patient example) and some other times individuals and concepts (as in axioms (ax20) and (ax22)). However,

this is not an insurmountable obstacle: one can, for instance, consider concepts classifying unique individuals (i.e. definite determiners) and their definitions. Then, by an innocuous abuse of notation consisting of identifying the definite determiner with the underlying individual, definition-founding can be reduced to definitional dependence and still relate individuals and concepts.

**Comparison with Vermaas' formalization of parthood between functions** An example was already discussed in depth in Section 3.0.8.3. Here we just summarize that Vermaas argument about the impossibility of a parthood relation applies to our framework at the type-level only, while at token-level it is possible to link a parthood relation to function decomposition.

**Comparison with IOF Core ontology** The Industrial Ontology Foundry[56] Core ontology[57] adopts BFO's concept of function, which, as discussed at the start of Section 3.0.2, espouses dissimilar ontological commitments than ours. Regardless, IOF Core contains a definition of 'design functions' (also called 'artifact functions'):

**df36*** $^*ArtifactFunction(x) \iff {}^*Function(x) \wedge \exists d({}^*DesignSpecification(m) \wedge {}^*prescribedBy(x, d))$

> "Artifact functions are exactly those functions prescribed by some design specification."

And it is clear that this definition is strongly similar to our definition of design function discussed in Section 3.0.7, provided that one interprets the prescription of design specifications under the light of an intentional selection event.

**Comparison with Burek's ontology of functions relations** Burek's work is quite detailed, and here we can only summarize and paraphrase it briefly. First, note that his concept of functions is the same as GFO's (discussed in Section 2.1.6), so we don't repeat it here. Instead, we are especially interested in the part of his theory that describes relations holding between functions. Burek analyzes many of these, in particular, instantiation, subsumption, parthood, and causal relations. Essentially, his strategy regarding subsumption (which he distinguishes into subsumption proper, specialization, and individualization) and instantiation is as follows: in his framework functions are complex statements including a goal (the desired output of the function), a functional item (the actor executing the function), a requirement (the state preceding the function execution). A function is thus determined by a triple $(Requirement, Item, Goal)$. Each of the three elements of the triple is called function determinant and may be a class or a single individual. A function is an individual if all its determinants are individual, otherwise is a class. Then, a function-class is subsumed by another if their determinants are classes and each class of the triple of the first is subsumed by the corresponding class

[56] `https://oagi.org/pages/industrial-ontologies`.

[57] We refer specifically to the commit `https://github.com/iofoundry/ontology/commit/7903ea00e44edca1da56bae827cb78e25a8a8cdf`.

of the triple of the second; a function-class specializes another if it is subsumed by it and one of the three element subsumptions is strict; a function-class individualizes another if some determinants of the first are instances of the corresponding determinants of the second; an individual function instantiates a function-class if all the determinants of the first function instantiate the corresponding determinants of the second. For example, 'today the White House was renovated' instatiates 'to renovate the White House', which individualizes 'to renovate a house', which specializes 'to renovate a building'.

Burek employs the same strategy of lifting determinant-level relations for parthood: a function is a function-part of another if the requirement and the goal of the first are part of the corresponding determinants of the second function. A function can also be made up from a sequence of sub-functions (where the goal of one element of the sequence is part of the requirement of the next element), in that case each sequence element is a sequence-part of the whole function.

Similarly, if the goal of the first function-class ($f_1$) is part of the second class ($f_2$), then $f_1$ supports $f_2$. $f_1$ enables $f_2$ if, instead, the goal of $f_1$ is part of $f_2$'s requirement. If the goal of $f_1$ is incompatible with the requirements of the second function, then $f_1$ prevents $f_2$.

The previous three relations hold between function-classes, and have a causal nature since they are interpreted as either leading to or removing necessary preconditions. Similar relations hold between function instances[58]: an instance triggers another if it directly causes it, while an instance improves another if it neutralizes its side-effects.

A summary mapping of his relations with those of our framework can be established as follows: Burek's hierarchical relations (instantiation, subsumption, specialization, and individualization) are not directly subcases of ours (`CL`, `subConceptOf`[59]), since Burek analyzes cases that are more specific than what we are interested in considering, due to his analysis of function determinants. Neither are our hierarchical relations direct subcases of Burek's, mainly due to our emphasis on differentiating functions and way-of-achievements. The approach to parthood is very different: Burek lifts the parthood relations between function determinants, while we adopt DOLCE perdurant mereology at token-level and reject the possibility of a parthood relation between functions at type-level. Burek's causal relation between function universals should be interpreted, in our framework, with the `precedes`, `follows`, and similar dependence-like relations holding at type-level. Finally, the causal relations at token-level should be interpreted with relations that will be introduced in the oncoming chapter (Section 4.1.4), after the introduction of Toyoshima's causal ontology, so we don't discuss them here.

Burek discusses also two other important classes of relations, this time holding between functions and non-functions: ascription and realization. As anticipated in Note 58, the discussion about realization is too long for the size of this section, moreover its

[58]Burek actually talks about function realizations, and for Burek function the instantiation relation is a strict subcase of the realization relation. However, since the discussion about realizations is complex, and its essence is preserved by the instantiation relation, we consider only function instantiations.

[59]These two are relations holding among tokens and reified concepts, if one whishes to make away with reification and uses only non-reified classes, the analogue of `CL` is instantiation and the analogue of `subConceptOf` is subsumption.

essential core is preserved by the instantiation relation, thus we have considered only the latter. Instead, regarding the ascription relation, it corresponds arguably to our `hasFunction` relation defined in (df27).

**Comparison with Borgo and Vieu artefacts** From [35] we took mainly the intentionally-selects predicate that was originally (Formula A2 at p. 297) defined as:

**df37*** $^*PhysArt[x] \iff \exists e,p,y,q\ ^*intentionalSel(e,p,x,y,q)$

"$x$ is a physical artefact if and only if there is some event $e$ where the agent $p$ obtains the artefact $x$ by intentionally selecting the material entity $y$ and attributing to it capacity $q$."

**ax26*** $^*intentionalSel(e,p,x,y,q) \implies \mathtt{E}(e) \wedge (\mathtt{APO}(p) \vee \mathtt{ASO}(p)) \wedge {}^*PhysArt(x) \wedge (\mathtt{M}(y) \vee \mathtt{NAPO}(y)) \wedge {}^*AttributedCap(q) \wedge \mathtt{qt}(q,x) \wedge \exists t(\mathtt{ql}_T(t,e) \wedge \mathtt{PC}(y,e,t) \wedge \mathtt{PC}(x,e,t) \wedge \mathtt{PC}(p,e,t) \wedge \mathtt{K}(y,x,t))$

"In the intentional selection predicate, $e$ is an event, $p$ an agentive social or physical object, $x$ is a physical artefact, $y$ is an amount of matter or non-agentive physical object, and $q$ is an attributed capacity. Additionally, $q$ is a quality of $x$ and during the time extension of $e$, $y$, $x$,and $p$ are participants in such an event, and $y$ constitutes $x$."

First, note that what they call 'capacity' in this thesis is referred to a 'capability', while the capacities of this thesis go in the direction of grounding the complexity of capabilities as individual qualities, which is a putative direction of research considered in [35]. Second, the relations between the modified and original intentional selection predicate arguably are:

**mp1** $^*intentionalSel(e,a,x,y,q) \implies \mathtt{intentionalSel}(e,a,x,q)$

"By ignoring the underlying amount of matter or object in intentional selection predicate from [35] we obtain the analogous predicate from Definition (df27)."

**mp2** $\mathtt{intentionalSel}(e,a,x,q) \implies \exists y\ ^*intentionalSel(e,a,x,y,q)$

"Every time an agent ($a$) intentionally selects (though event $e$) a capability ($q$) of some artefact $x$, there is some amount of matter or non-agentive physical object $y$ such that the intentional selection predicate from [35] holds between $a, e, q, x$, and $y$." Here we express the link between the capability and the function with the definition-founding relation, as was done before. Notice that $f$ may be an ontological function, but also some more specific intentional predicate from Definition (df27)

That is, we adopt the same intentional-selection account, only ignoring, for sake of simplicity, the possibility that the artifact, after the selection event, becomes a different object but colocated with the object it was before. We still admit that possibility, only do not mention it explicitly. Additionally, in [35] the authors speak of *the* capacity of an object, as, essentially, the sum of all the abilities of the object to behave in some way: "the quale corresponding to the capacity of an entity at a given time collects all the various dispositions or behaviors the entity is able to express at that time".

However, they recognize the possibility of needing to associate multiple, different types of capacities of smaller scope to an entity, which is the direction we pursued. And, in any case, the attributed capacity values are typically smaller than the whole capacity.

**Comparison with UFO** A full comparison with UFO, which would inevitably beckon a full comparison between DOLCE and UFO, is far beyond the scope of this thesis. However, a brief, limited comparison with UFO's analogues of DOLCE-qualities is interesting, especially in light of our proposal about capacities and capabilities in Section 3.0.6 (especially the latter, since UFO literature focuses on those). The main reference that we use for this section is Guizzardi's thesis [99].

Very briefly, UFO introduces an inherence relation, similar to DOLCE's, and a class 'moments' made exactly from those (UFO-)endurants that inhere in some other UFO-endurant[60]. Additionally, UFO divides moments into intrinsic and relational. Put this way, UFO's moments would appear to be the same as DOLCE's qualities, with intrinsic (relational) moments aligning to the intrinsic (relational) qualities introduced in this thesis. However, there are some critical differences: for one, Gärdendfords-inspired qualities are also called qualities in UFO and make up a class strictly smaller than intrinsic moments: they are defined as those intrinsic moments that are associated to a 'quality structure' [99, Axioms (24, 25)], roughly a conceptual space made from an integral domain. Second, UFO-qualities are all *intrinsic* moments, in contrast with our liberal use of *relational* DOLCE-qualities. Moreover, the complement of UFO-qualities within intrinsic moments is the class of 'modes' [99, Axiom (41)], and modes are described as "conceptualized in terms of multiple separable quality dimensions" and exemplified by (also) "skills", suggesting a strong similarity with our treatment of capabilities. A further suggestion in this direction is the introduction in UFO of externally dependent modes, which (despite being intrinsic) are existentially dependent on some entity independent[61] from the mode's bearer [99, Axiom (42)]: the replacement of our term 'relational' with the locution 'dependent on some entity independent from their bearer [or just external]' would take care of the mismatch caused by the presence of relational qualities. However, externally dependent modes are also founded by some event [101, Axiom 71], in the sense that, e.g., the conjugal commitment of the husband to the wife is founded by the event of their marriage, and we did not consider analogues of founding events for capabilities.

Additionally, UFO divides qualities into simple and complex: simple qualities are those that do not bear other qualities, complex qualities are those that do [99, (28),(29)], and complex quality can only bear simple quality. Due to this, there cannot be long chains of qualities inhering in other qualities; modes instead can bear their own moments, either modes or qualities [101]. We did not define classes for simple and complex qualities[62], because we eventually used the term simple (complex) to refer to spaces

[60] Already we can see a mismatch with DOLCE, since in DOLCE endurants do not inhere in other entities. However, this point is very high-level and thus outside the scope of this section.

[61] Independence in UFO is defined as reciprocal non-dependence, which is different from our definition of external-to (df9), but has the same goal.

[62] In any case it is possible to copy UFO's definition in DOLCE without any issue.

made from a single (multiple separable) integral domain(s). This is a minor mismatch that could also be amended with an appropriate change in terminology[63].

Another important category in UFO is that of relators: relators are (non-intrinsic) moments that mediate between the relata of a material[64] relation. The mediation relation holds between relators and endurents that are relata of a material relation. In order for such relata to be mediated by the relator, they must bear another type of moments: qua individuals. Qua individuals are UFO-externally-independent-modes that inhere in the relata of material relations, are part of the corresponding relator (one relator-instance corresponds to one material-relation instance), and exemplify the set of properties that the relatum they inhere in has in the scope of the relation. For example, the relation relating together the buyer, the acquired good, and the shop of an individual purchase is a material relation. Therefore, for any such individual purchase there is a relator individual that mediates the relevant relata. For instance, if John buys a copy of a book on Amazon, then there is a relator (say, $r$) that mediates with John, the book, and Amazon; there are also three qua individuals, which are all part of the relator: the qua individual corresponding to John represents all of John's properties relevant in the context of the purchase (e.g. John's willingness to buy the book). UFO-relators are especially important in the context of this thesis because they could be used to, e.g., conceptualize entities that appear to be tightly linked to material relations, such as affordances and relational qualities. In particular, in [94] relational qualities are mentioned in this context and in [96] is said that "relational qualities typically come in bundles called relators". However, we do not examine the issue of UFO-relata further for the sake of brevity.

More importantly, capabilities are considered as dispositions [12, 188], which is a modeling path we have consciously decided to avoid, although we do not reject necessarily, preferring to highlight the dependence relations between capabilities and other qualities instead. Moreover, there are differences in the attribution of values to qualities: UFO uses a time-slice formalization, in the sense that its predicates do not have time as an argument and instead the modal formalism of possible worlds is used to allow for reference to different situations, including to the same entities at different times. Even discounting this difference as one related to the logical formalism, quality spaces for modes remain (to the best of our knowledge) not developed in UFO, in fact in [101] it is stated that modes cannot be projected into a value space (although they can bear qualities themselves that can be used to qualify them). Additionally, (what looks like UFO's version of) capabilities performance (by which we mean how a capability is manifested, e.g. well or bad, slow or fast) seems to be described in UFO by the value of a quality inhering in the capability itself (which, as an UFO-mode can bear UFO-qualities) "a *DispositionType* (such as 'Passenger Carrying Capability') may be

[63] Another minor mismatch is the conceptualization of integral dimensions: in [99] the day, month, and year fields of a date are examples of integral dimensions of the date domain, however, the dependence among these fields is formal and conventional in nature, distinct from the perceptual nature of, say, the dependency between pitch and loudness.

[64] That is, a relation with a structure of their own, such as employments or flight connections. The opposite of 'material' relation is 'formal' relation, e.g. 'one is smaller than two' or the quale-of relation. See [99] for a more detailed discussion.

characterized [i.e. instances of the first type may bear instances of the second type] by a *QualityType* (e.g., 'Passenger Capacity'). This opens us the possibility for representing capability measurement and capability improvement over time" [188]. In this way UFO can indicate the degree of performance of a capability or the fact that the capability changes overs time.

Another important point is that UFO modes inhere contingently in they bearers (e.g. they can be lost or found) while DOLCE qualities, at least in the Wonderweb Deliverable 18 version [177], necessarily has the same temporal (and possibly spatial, for physical qualities) extension of their bearers. However, in a more recent version of DOLCE (OntoCommons Deliverable D2.7 [175]), such a fact has been weakened, and the life of a DOLCE-quality can be strictly smaller than the life of its bearer: in this latter case, if we assume contingency/necessity issues have been reduced to only the temporal modality, then the two approaches are in principle compatible and an entity may lose or gain qualities, as also exemplified in the gear train example.

In conclusion, UFO's moments share strong similarities with DOLCE-qualities and our conceptualization of capabilities could almost be directly translated into UFO. The analogy between the fact that we have founded DOLCE-quality-capabilities on other DOLCE-qualities and the fact that in UFO UFO-disposition-capabilities can bear UFO-qualities (e.g. the example of Passenger Carrying Capability bearing Passenger Capacity) is particularly striking. However, there are various subtle obstacles preventing this, that cannot be resolved by a single, simple modification to either DOLCE, UFO, or this thesis work. For instance, even when assuming that all incompatibilities between DOLCE-qualities and UFO-moments have been amended, one possible, minimal way to modify our approach to fit in UFO by mapping capabilities to externally dependent modes would still require to (i) explain what are founding events for capabilities, (ii) take care of ensuring that contingency of UFO-capabilities and DOLCE-capabilities is aligned (as mentioned above this may be possible if using DOLCE's axiomatization from [175]), (iii) specify what are relational qualities in DOLCE and how they relate to relational qualities in UFO, (iv) take care of also mapping the rest of the formal machinery of this thesis in UFO in such a way as to preserve the most links between capabilities and related entities.

Further analysis of this topics is no easy task. We conclude here for brevity.

### 3.0.10 Reduction to OWL ontology

As mentioned in Chapter 1, we serialize our theory of function, which was developed in this chapter, in the OWL language, for sake of applicability. There are methods to (semi-)automatically produce an OWL-theory from a first-order logic one, which is, in some sense, a good approximation [108]. We start from these, however, since we want to preserve the best the aspects that we consider most important, the OWL version is mostly manually curated.

Note that the only part of the first-order theory developed in the previous sections that is not already in the OWL fragment is constituted by the set of expressions where either ternary relations appear, or at least three variables are necessary. Relevant

examples are the definition of systemic-function-of (df16), the instantiation-founding definition (df11), and the engineering method schema (df24). In these cases we have to decide how to suitably modify the axioms. For instance, we decided to modify the definition of systemic functions in

**ax27**$_{owl}$ $\texttt{SysFunction} \sqsubseteq (\forall \texttt{classifies}.(\texttt{Behavior} \sqcap \exists \texttt{posCausalCon}.\texttt{Goal}) \sqcap \exists \texttt{sp-dependent-on}.\texttt{System})$

"Every systemic function classifies only behaviors that positively causally contribute to some goal; and is specifically dependent on some system."

where `classifies` is the inverse relation of $\texttt{CL}(\cdot,\cdot)$ (constantly classified by, defined below in (df38)). In this way, we highlight and preserve the fact that systemic functions are definition-founded on systems.

Then, all the ternary temporalized relations used in the first-order version, such as $\texttt{CL}(\cdot,\cdot,\cdot)$ and $\texttt{participateAsDoer}(\cdot,\cdot,\cdot)$, have corresponding OWL analogues that have the temporal argument removed. The passage from a temporalized relations to a non-temporalized analogue can be carried out in different ways [239]. One way is to state that the non-temporalized relation holds if and only if the temporalized relation holds whenever one of its arguments exists. Notice that the meaning of the ensuing relation in general depends on which argument is chosen for this definition. For example, in the case of temporalized parthood, the arguments are 'part' and 'whole' and the part is-part-of the whole at some time, whenever the temporalized parthood relation holds; then, we can arrive at two non-temporalized analogues: one holds whenever the part is-part-of the whole life of the part, the other holds whenever the part is-part-of the whole for the whole life of the whole. An example of the latter and not of the former is a heart, which is part of a person from their birth to their death and later is donated to another person; an example of the former and not of the latter is a tree, which is part of a forest from its birth to its death. The former case is also our choice for the parthood (df15), classification, and temporal quale relations; while we chose the latter for participation-like relations (the following axioms are labeled as *meta-rules* because they are formulated in first-order logic but they explain how to link two different languages together):

**df38**$_{meta}$ $\texttt{CL}(x,y) \iff ((\exists t \texttt{PRE}(x,t)) \wedge \forall t(\texttt{PRE}(x,t) \implies \texttt{CL}(x,y,t)))$

"$x$ is constantly classified by $y$ if and only if $x$ is classified by $y$ whenever $x$ exists."

**df39**$_{meta}$ $\texttt{ql}(x,y) \iff ((\exists t \texttt{PRE}(x,t)) \wedge \forall t(\texttt{PRE}(x,t) \implies \texttt{ql}(x,y,t)))$

"$y$ is a constant temporary quale of the quality $x$ if and only if $y$ is a temporary quale of $y$ whenever $x$ exists."

**df40**$_{meta}$ $\texttt{participateAsDoer}(x,y) \iff \exists t(\texttt{PRE}(y,t)) \wedge \forall t(\texttt{PRE}(y,t) \implies \texttt{participateAsDoer}(x,y,t))$

"$x$ constantly participates-as-doer to $y$ if and only if $x$ participates-as-doer to $y$ for the whole of $y$ duration."

There are other ways to obtain a non-temporalized analog in addition to those described above, e.g. one could also state that the non-temporalized relation holds if and only if

the temporalized relation holds at some given time:

**ex32**$_{meta}$ $\texttt{participateAsDoer}(x, y) \iff \exists t\ \texttt{participateAsDoer}(x, y, t)$

> "$x$ participates-at-some-time as doer in $y$ if and only there is some time $t$ such that $x$ participates as doer in $y$ at time $t$."

This does not exhaust the list of methods, and the choice of one over is contextual to the concerns of the given ontology, since this choice affects the intended models. For example, (df40) excludes participation in a process only for a given temporal part, while (ex32) allows it, but in OWL no difference is apparent. Removing the temporal argument diminishes the flexibility of the resulting ontology. For instance, it becomes impossible to directly track dynamic aspects of roles, such as a component that performs one function at a certain time and then switches to another function later. Additionally, it is not possible to express scenarios like a chemical process that is driven by different catalysts in different phases.

Furthermore, we need two additional binary relations that we use to implement and simplify (df24) and redefine (df25):

**df41**$_{meta}$ $\texttt{mainFunctionOf}(\texttt{f}, \texttt{m}) \iff (\texttt{subConceptOf}(\texttt{f}, \texttt{main}^{\texttt{m}}) \wedge \texttt{SysFunction}(\texttt{f}))$

> "$\texttt{f}$ is main-function-of a method $\texttt{m}$ if and only if it is a systemic function specializing the main-function role corresponding to $\texttt{m}$."

**df42**$_{meta}$ $\texttt{subFunctionOf}(\texttt{f}, \texttt{m}) \iff (\texttt{subConceptOf}(\texttt{f}, \texttt{sub}_i^{\texttt{m}}) \wedge \texttt{SysFunction}(\texttt{f}))$

> "$\texttt{f}$ is main-function-of a method $\texttt{m}$ if and only if it is a systemic function specializing any of the sub-function roles corresponding to $\texttt{m}$."

**df43**$_{owl}$ $\texttt{SolvedFunction} \equiv \exists \texttt{MainFunction}.\texttt{EngMethod}$

> "Solved functions are exactly the individuals that are main-function-of some engineering method." Note that an solved function must be a systemic function due to (df41).

Finally, various axioms that require more than two variable are simply suppressed (e.g., (df27) and (ax23)).

DOLCE is a (modal) first-order logic theory, but OWL versions exist, such as DOLCE-lite and DOLCE-ultralite[65]. Any of these could be used to align the OWL version of our ontology with DOLCE: we used the OWL version of DOLCE that was recently submitted as part of the ISO 21838 standard. A serialisation of the resulting theory can be found on GitHub[66]. The ontology is populated by some example instances and its consistency was tested using the Hermit reasoner.

**Brief exemplification** To wrap up this section, we give a picture (3.9) expanding the schema in Figure 3.6 to add a few instances, with the goal of demonstrating how the concepts introduced earlier coalesce to provide a comprehensive description of a functional scenario.

[65] Available at `http://www.loa.istc.cnr.it/index.php/dolce/`.

[66] `https://github.com/kataph/function-method-ontology.git`.

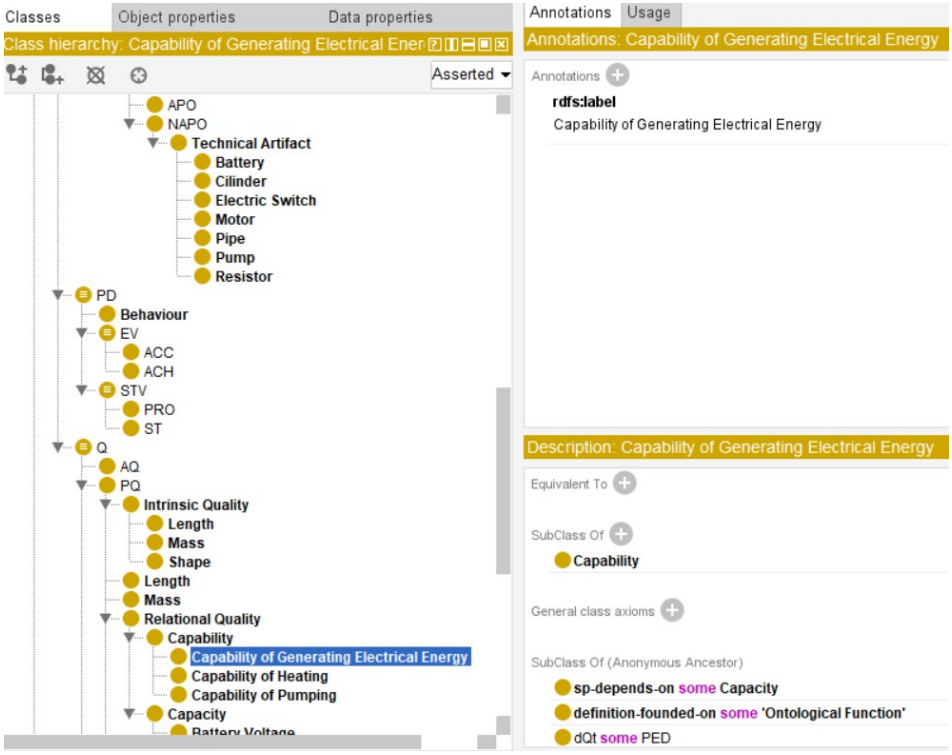



Figure 3.8: The ontology taxonomy in Protegé.

### 3.0.11 Evaluation of FOL and OWL ontologies

Various techniques exist to evaluate the quality of FOL and OWL ontologies. However, in this section we focus on evaluating the OWL version, since the techniques for OWL ontologies are more specialized. However, before coming to that, we shall summarily discuss how our framework compares to the desiderata listed in Section 2.1.6:

- Teleology: it is not always true in this framework that the existence of the function bearer is explained by the function itself, consider the discussion of drug repositioning in Section 3.0.7: a certain drug may bear the function of curing a certain illness, despite having been manufactured to treat a different illness, i.e. it was brought into existence to execute a different function.

- Restriction: the duality between essential and accidental functions has been discussed in Section 3.0.7: cf. definitions (df28) and (df29).

- Normativity and Malfunction: performance of function execution has been discussed in Section 3.0.6, and bad performance at some time does not prevent the corresponding function from being present at that time.

- No epiphenomenalism and Support: this was discussed in Section 3.0.7, in summary the structure of the function bearer is linked to the ascribed function by proxy of the selected capability (cf. Definition (df27)), since the capability itself is grounded in the physical structure of the function bearer (cf. Section 3.0.6).

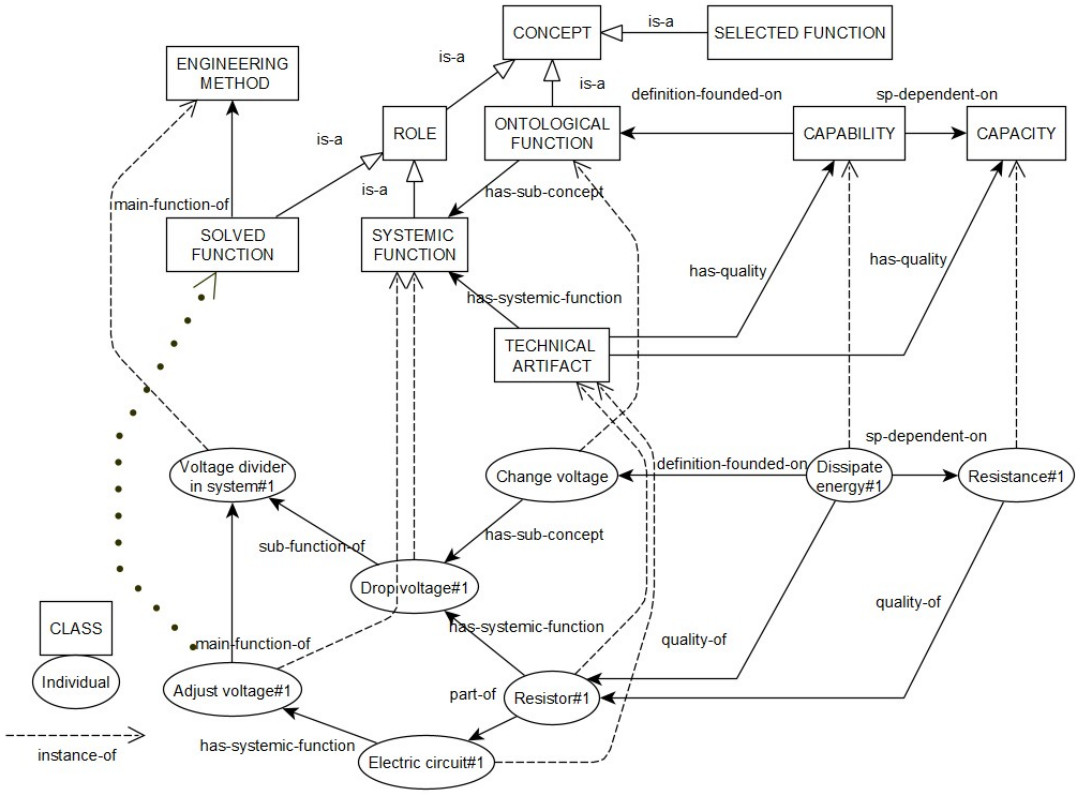
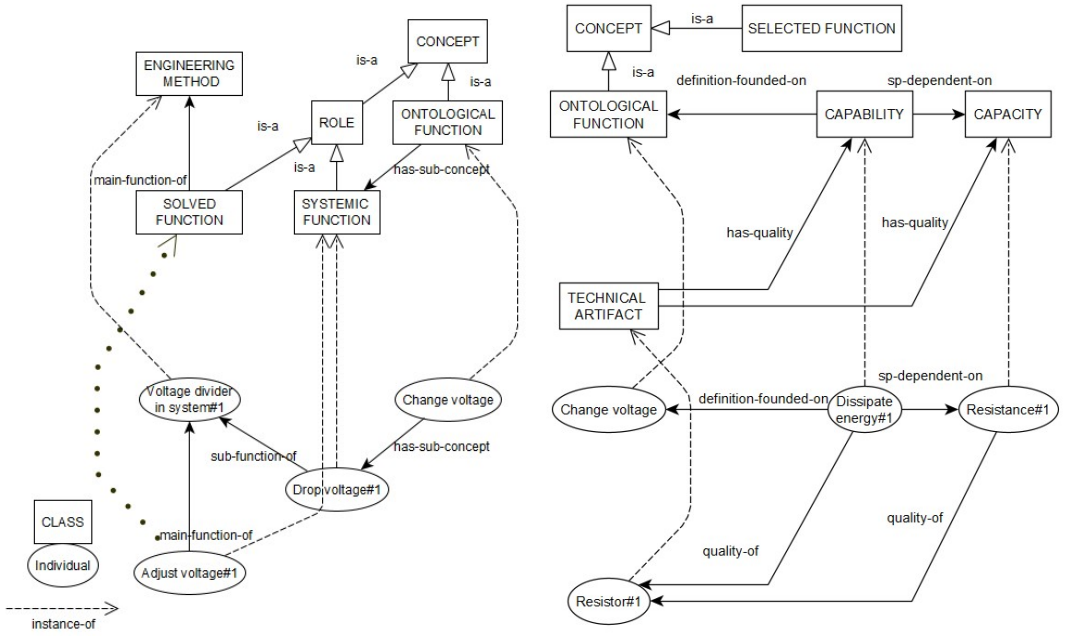



Figure 3.9: The primary concepts and relationships explained in this section. To enhance simplicity and clarity, the diagram is based on the corresponding OWL model, and numerous relations and concepts are omitted. Note that a filled black arrow between individuals indicates that the relevant object property holds between those two individulas, while a filled black arrow between classes indicates that the object property corresponding to the edge's label enjoys an existential restriction axiom between the classes (e.g. any solved function is main-function-of some engineering method). This picture illustrates the underlying OWL ontology and its axiomatization with some example instances, in particular it is not meant as a mandatory modeling pattern. Such minimal patterns would look more like Those in Figure 3.10.
An example instantiation of the schema for a voltage divider is included. The dotted instance-of arrow is inferable, as the associated systemic function serves as the main function of a certain engineering method (cfr. (df43)).




(a) Example of minimal modeling pattern for functional decomposition, extrapolated from Figure 3.9 by maintaining only engineering methods, and systemic functions and their generalizations.

(b) Example of minimal modeling pattern for functional decomposition, extrapolated from Figure 3.9 by leaving only capabilities and capacities and their relation, ontological function the capabilities depend on, and the capability bearer.

Figure 3.10

- Innovation: since functions were separated from ways-of-achievements, innovation happens by adding new ways to achieve functions (which, at a high level of generality can be a fixed, finite amount).
- Continuants: functions and their realizations are differentiated in our framework, the former are concepts/roles the latter are perdurants.
- Domain bridging: we conciously decided to only focus on engineering functions.
- Processes: this was also argued by Burek, however, we opted not to purse this possibility (cf. the end of Section 3.0.7).
- Decomposition: functional decomposition has been accounted for in the dedicated section.

This list also thoroughly answer to Research Question (RQ1).

Coming to the techniques used to evaluate OWL ontologies, many of these are not applicable in this case. For instance, there is no "gold standard" [229] or set of formal ontologies available for comparison using typical ontology metrics[67], nor there is, at

[67]such as those provided by OntoMetrics[68].

this general level, a specific task against which to conduct a performance evaluation. Instead, we will make use of the OntoClean methodology [98], of the Ontology Pitfall Scanner [208], and of the competency questions technique [90] (in which we also include the research questions (RQ1)-(RQ3)).

**OntoClean.** None of the meta-level rules that OntoClean enforces are violated, that is, the ontology is 'clean'. Since we extend DOLCE categories, which are already known to satisfy such rules, OntoClean violations may only be introduced by subsuming a non anti-rigid class in a role-concept extension (i.e., the set of entities that, at a given time, are classified by the role-concept). This does not happen since (systemic, engineering, main-, sub-)functions, as we define them, are indeed anti-rigid (e.g., a heat exchanger, with its behavior, may be used to heat a substance or to cool it, depending on the context).

**Ontology Pitfall Scanner** We evaluated the OWL ontology with the Ontology Pitfall Scanner (OOPS!) [208]. OOPS! is a tool that, given an OWL ontology, searches for matches from a list of 41 common design errors. Some pitfall were found but not discarded, since they were related to imported ontology. Among the remaining pitfalls, some are minor stylistic issues[69], which we also ignored; but some are important: P11 ("Missing domain or range in properties"), P24 ("Using recursive definitions"), and P30 ("Equivalent classes not explicitly declared"). P11 means that some object property has domain or range axioms missing, but this is a deliberate design choice to minimize the ontological commitment of the ontology. P24 arises because OOPS! mistakenly interprets (ax12) as a recursive definition (due to the ontological functions class appearing on both sides of the material implication), although this is not the case. Similarly, P30 occurs because OOPS! incorrectly assumes that the class of resistors (technical artifacts) and the class of resistances (capacities) should be equivalent.

**Research questions and competency questions** The answers to 1-3 are as follows:

- **Answer to (RQ1).** This was answered at the start of this section, however, as a brief summary We modeled functions as DOLCE-concepts that classify certain classes of perdurants and this modelling choice has at least the advantage of explaining the why-how and abstraction-implementation ambiguities in the meaning of 'function': it is possible to separate engineering methods from (proper) functions; and systemic functions from ontological functions, depending on their abstraction level. It also explains how functions can exist even when they are not executed, and various other classical issues and dualities related to functions.

- **Answer to (RQ2).** The functional decomposition of systemic functions is discussed in Section 3.0.3: functions are decomposed into a tree of alternating functions and engineering methods.

[69] Some inverse object-properties are missing, and the naming convention differs between object-properties and classes.

- **Answer to (RQ3).** Capabilities and capacities are discussed in Section 3.0.4. Capabilities are defined by leveraging ontological functions, capacities are roughly considered parameters of capabilities. Moreover, and capabilities are specifically dependent on capacities.

We also consider some related competency questions, to confirm the expressive power of the ontology. In addition, since it is (arguably) self-evident that these questions can be of interest for applications, given the literature review of Chapter 2, the fact of the ontology has the expressive power to formulate these questions constitutes a validation mechanism:

**CQ1** Given a function, which artifacts have the capability of satisfying it? (Here functions will be interpreted here as ontological functions)

**CQ2** How can the given function be implemented? (Here function will be interpreted as ontological function)

**CQ3** What are the capacities that a given capability has? (Where 'has' will be interpreted as 'depends on')

**CQ4** Which parameters explain the performance of some component in a given system? (Interpreted as 'what are the capacities of the capabilities of the component that are relevant in executing its function?')

**CQ5** Which is the functional decomposition of a given system?

**CQ6** Which is the function of the given tag and what is its purpose? (Tags here will be interpreted as systemic functions or identifier thereof)

The ontology, after being suitably populated with an Abox, can answer the questions (CQ1)-(CQ6) by, e.g., means of SPARQL queries. For instance, this is a SPARQL query implementing (CQ1):

```
PREFIX : <https://github.com/kataph/function-method-ontology#>
SELECT ?component
WHERE {
 ?capability :definition-founded-on :<the given function> .
 ?capability :quality-of ?component}
```

While this implements (CQ4):

```
PREFIX : <https://github.com/kataph/function-method-ontology#>
SELECT ?component ?ontologicalFunction ?capacity
WHERE {
        ?systemicFunction :function-of ?component .
        ?systemicFunction :sp-dependent-on :<the given system> .
        ?systemicFunction :sub-concept-of ?ontologicalFunction .
        ?capability :definition-founded-on ?ontologicalFunction .
        ?capability :quality-of ?component .
        ?capability :sp-dependent-on ?capacity .
        ?capacity :quality-of ?component}
```

And this implements (CQ5), by returning the list of all systemic functions of components in a given with their corresponding method and role (main-function or sub-function):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <https://github.com/kataph/function-method-ontology#>
SELECT ?component ?systemicFunction ?role ?method
WHERE {
    ?systemicFunction ?role ?method .
    ?method rdf:type/rdfs:subClassOf* :EngineeringMethod .
    ?systemicFunction :function-of ?component .
    ?component :constant-part-of :<the given system>}
```

The remaining competency questions can be implemented similarly (see also Fig.3.11 for (CQ3); also note that to answer, say, (CQ2) one just needs the pattern 3.10b).

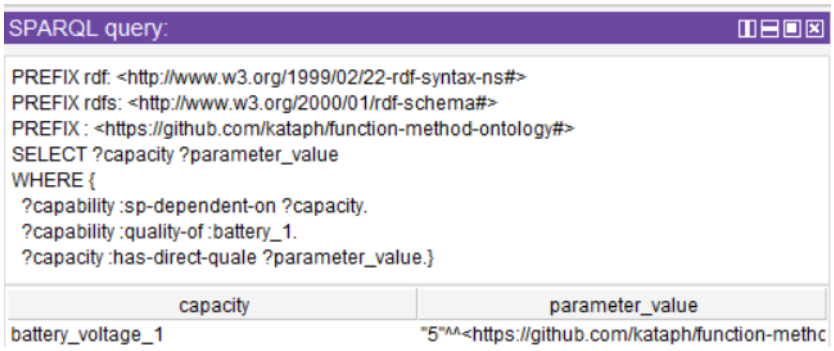



Figure 3.11: The Query (CQ3) implemented in Protegé.

Alternatively, some of the competency questions[70], such as (CQ1) can be formalized as an OWL class expression (see Figure 3.12); in this case, if the question has the shape "Which are all the $X$s such that $P(X)$?" and the instances of the class are the individuals (the $X$s) to be retrieved by the question. This type of queries is called 'DL queries'. The difference is that this latter method necessarily employs reasoning to carry out the classification task, meaning that it has the potential of retrieving instances that are known to satisfy the properties prescribed by the question (the $P(X)$) only implicitly; additionally, this DL queries can execute other reasoning tasks, such as finding subclasses or equivalent classes of the specified OWL class expression. These reasoning tasks correspond to questions roughly of shape "Which are some types of entities that satisfy $P$?" and "Which are some types of entities that exactly satisfy $P$?". SPARQL queries, instead, may or may not employ reasoning, and if they do, they may employ a particular entailment regime developed by a particular RDF-database vendor. However, considering only syntax aspects, SPARQL queries are strictly more expressive than DL queries. Therefore DL and SPARQL queries are not strictly comparable.

[70]Precisely, those that can be formulated in OWL, i.e. either require at most two variables or are constituted by variables chained together by a series of object properties.

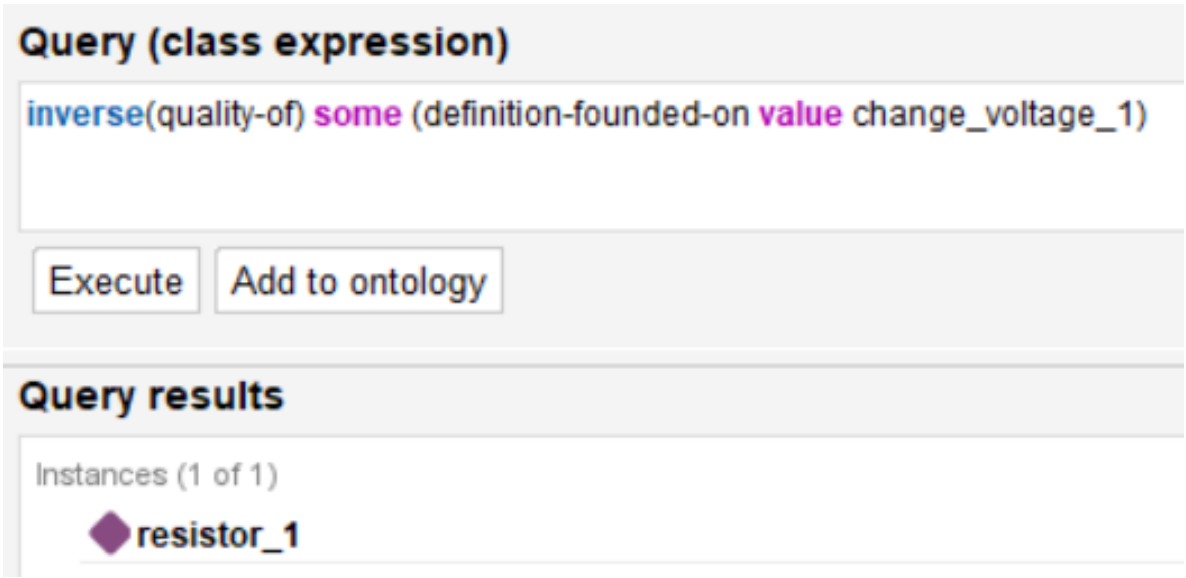



Figure 3.12: Example of competency question formalized as a DL query.

# Chapter 4

# An Ontology of Malfunctions in Engineering Systems

In this chapter, we carry out an ontological analysis of malfunctions based on the review of Section 2.2 as well as the theory of functions built in the previous chapter, with the hope of developing a solid technical foundation or even a unifying framework to talk about malfunctions in engineering. Note that the content of this chapter is roughly based on some already published work ([53]), but said content has been extensively modified and enriched. One motivation for this analysis stems from the recognition that engineering language about malfunctions is often hindered by terminological and conceptual confusion within the domain. Different stakeholders, even though they have expertise in similar or the same areas, might use different terms to describe the same failure phenomenon, as demonstrated in Chapter 2, leading to miscommunication and reduced efficiency in identifying and addressing the root causes. By conducting an ontological analysis and providing clear definitions for key domain concepts, this work aims to establish a common understanding and language for treating engineering malfunctions, failure analysis in particular. This standardized approach could bridge the gap between various engineering disciplines, facilitating interdisciplinary collaboration and knowledge-sharing, and solving interoperability issues. In particular, some key properties of the language introduced in this chapter are:

- it is grounded in a robust ontological analysis of malfunction-related concepts.
- As a formal language, it ensures recorded facts can (i) be validated for compliance with the underlying ontology and (ii) be enhanced through inferencing.
- The language has been serialized using various technologies to facilitate ease of use across different contexts.
- It enables the recording and analysis of failure occurrences as knowledge graphs, aiding systematic knowledge storage and retrieval.
- The language is domain-independent, built on a general ontology of causation. While primarily inspired by engineering, it could also be applied to other fields, such as medical failures.

We also propose an ontology-driven methodology to describe root cause analysis activities and record such descriptions in knowledge graphs conforming to an ontology. The knowledge graph can then be enriched by classification tasks and be ensured compliant with the ontology through the use of a reasoner.

### 4.0.1 Ontological analysis of malfunctions

In the rest of the work, while we work through more precise definitions, we use the term 'malfunction-related happening' as a catch-all term for any occurrence considered unwanted by domain experts. We also take 'unwanted' and 'abnormal' as synonymous primitive terms to mean that there is a mismatch between what is occurring and the ideal mental model of a domain expert.

Given the discussion of malfunction-related happenings in CHapter 2, we claim that there are at least three key aspects that must be highlighted to carry out a successful ontological analysis of these concepts:

- **Ontological categorisation**: malfunction-related happenings are distinguished into the ontological categories of occurrences, namely, event, process, and state.
- **Causal relations**: malfunction-related happenings are defined by the role they take into a causal chain.
- **Granularity**: malfunction-related happenings can be directly decomposed (as in [158]) or associated with a decomposition of an engineering system (e.g., Figure 2.8), so that finding a suitable granularity level for failure analysis or determining how to transition from one granularity level to another is critical.
- **'Internality'**: In addition to granularity, it is important to establish whether a malfunction-related happening is internal to a system or component, or not. This distinction is often essential, as evident in the definition of fault in the IEC vocabulary [124, 192-04-01], or in the differentiation between internal and external causes/faults as discussed in [143, 10].

We believe that 'internality' can be traced back to granularity, for instance, by saying that an occurrence is internal to a system if all its participants are part of that system. For this reason, we will consider the latter two aspects as a single point, on a practical level.

There are other relevant aspects of malfunction-related happenings but we limit our analysis to those listed above. In particular, there exist in the literature two recurrent aspects that we call 'culpability' (that is, distinguishing failures depending on who is responsible for them) and 'observability' (that is, the property of a happening of being easily observable by the technicians working with an engineering system). Those two aspects are surely important, but we exclude them from the current analysis: regarding culpability, causal analysis is a solid basis and necessary precondition to establishing culpability, while the opposite is not true. Regarding observability, it is an epistemic aspect, not ontological, so it is different from what we are studying in this work.

Moreover, in practical settings, malfunction-related happenings are distinguished on the base of the severity of the material, social, and economical damage that they cause. In particular, the countermeasures to prevent them, and the methodologies employed to study small, routine occurrences, and complex and catastrophic ones are drastically different. However, we consider this aspect more a social one, and put it too outside the scope of this analysis.

#### 4.0.1.1 Brief formalisation of general causation

We report now and expand the ontology proposed by Toyoshima, Mizoguchi, and Ikeda in [244] on which we will base the core of this chapter.

We follow a fairly common approach (e.g., see [165, 244]) and assume that causation is a relation that holds between occurrents such as states, processes, or events. Starting from this, we introduce the needed predicates and their taxonomical relations as primitives (the predicates itemized by **pr** are primitives in the logical theory). In particular, the primitives related to concurrent classes are only minimal characterized, since we aim to (i) preserve compatibility of this work to multiple possible upper-level ontologies; and (ii) avoid the genuine ontological problem of characterizing occurrents (which is a work for upper-level ontologies[1]), in accorfance to the design principle expressed at the end of Section 1.1. We still use DOLCE as a guide, again as explained in Section 1.1, therefore, an alignment between DOLCE and the occurrent-primitives will be presented after their introduction. The reason for maintaining the concurrent-primitives is duplex: (i) Toyoshima's causation ontologies uses the same primitives, and (ii) we want to ensure the modularity that the oncoming body of work so that potential users could utilize it independently of the rest of this thesis' work.

**pr1** `Occurrent`(x)

> "$x$ is an occurrent." An occurrent is something that takes place in time and is not (necessarily) wholly present at any time it is present.

**pr2** `State`(x)

> "$x$ is a state." In engineering states are often described by specifying some characteristics (form, value of a process variable, ...) of the entities participating in the state (e.g. 'the shaft is broken', 'high oven temperature', or 'the oven temperature is increasing').

**pr3** `Process`(x)

> "$x$ is a process." In engineering, a typical example of process is a sequence of states (or, equivalently, of state transitions) related to some physical phenomena. For instance corrosion is a sequence of many single chemical reactions that change the arrangement of the relevant atoms.

**pr4** `Event`(x)

[1]The interested reader can consult appropriate reviews, e.g. [217], to see the main ways in which occurrents are classified in various ontological approaches.

"$x$ is an event." Typical examples events are those occurrences that are 'unitary whole[s] with respect to time, representing a distinct individual episode with a definite beginning and end'[2] [217, p.9]. In engineering, important classes of occurrences that are encompassed in the above description, and which in this work we consider as archetypical examples of events, are (a) those occurrences that, being very brief, may be considered instantaneous[3], such as impacts, brittle fractures, crack formations, and (b) those occurrences that have a definite time extension, e.g. 'the ceramic was heated in the oven for two hours'.

Note that the difference between states, processes, and events may be nuanced. For example the difference may be linked to verb tenses (e.g., recognizing 'I am sitting' as a state description and 'I sat for two hours' as an event description), and some authors argue from a multiplicativist point of view that the changing of states or the succession of events *constitute* processes which, in turn, *constitute* events when completed [189]. Additionally, distinct top-level ontologies make the difference more precise but take distinct stances on what such differences are. Following Toyoshima, we postulate that occurrents are partitioned into these three categories:

**ax28** $\texttt{State}(x) \vee \texttt{Process}(x) \vee \texttt{Event}(x) \implies \texttt{Occurrent}(x)$

"States, processes, and events are occurrents."

**ax29** $\texttt{State}(x) \implies \neg(\texttt{Process}(x) \vee \texttt{Event}(x)) \quad \wedge$
$\texttt{Process}(x) \implies \neg\texttt{Event}(x)$

"States, processes, and events are disjoint classes."

In DOLCE upper ontology, as explained in Section 1.1, homeomericity and cumulativity are useful proprieties to decide if an occurrence is or not a state: DOLCE-processes are cumulative (a sum of processes of a certain type is still a process of that type), but not homeomeric (in the classic example of a running process, there are brief temporal parts that are not themselves running-occurrences). DOLCE-states are both cumulative and homeomeric (a part of a sitting state is still a sitting state). DOLCE-events are not cumulative (two individual election-events taken together do not constitute a single election-event). The naive alignment between the occurrence-predicates of Toyoshima's ontology and those of DOLCE in the following:

**mp3** $\texttt{State} \equiv \texttt{S}$[4]

[2]This is actually a description of YAMATO-events. Descriptions of events in other foundational ontologies can be significantly different. The reason we privilege this one, is that it is that it was found to be easy to work with in practice, in modelling scenarios such of those of figures 4.7 and 4.8.

[3]Note that, one of the key ontological assumptions of Toyoshima's causal ontology is that there are no instantaneous occurrents. So, when we say 'instantaneous', we actually refer to an occurrent that is perceived as a unitary whole in time due to its short duration. The key ontological aspect that makes it an event is the 'unitary whole in time', not the 'being very short', see also [217].

[4]We use the symbol $\equiv$ to mean that there are two symbols that are equivalent though they belong to different theories. From a formal logic point of view, we mean that, when discussing the union of the two theories, either there is only one predicate in the signature that can be written using one symbol or the other, interchangeably; or that in the signature there are two equivalent predicates. Both options are fine, since the difference in the theory structure caused by choosing an option over the other are

**mp4** `Process` $\equiv$ `PRO`

**mp5** `Event` $\equiv$ `E`

**mp6** `Occurrent` $\equiv$ `PD`

> "Toyoshimas' states, processes, events, and occurrents correspond to DOLCE's states, processes, and perdurants, respectively."

Since the rest of our framework is based on DOLCE, we commit to this alignment. However, note that this commitment is non-trivial: for instance, associating `Occurrent`s to BFO-occurrents would introduce a considerable amount of ontological baggage quite different from the one entailed by (mp6).

Now, we introduce a binary relation between occurrents, called causal relation (or causation, for short) and written `causeOf`, which we will specialize in the following:

**pr5** `causeOf`(x,y)

> "$x$ causes $y$." In this work, `causeOf` is a binary relation between occurrence-tokens. This means that a statement like 'smoking causes cancer' cannot be modelled. Formula '`causeOf`$(x, y)$' can model only expressions like, e.g., 'John has cancer (state $y$) because smoked for 20 years (event $x$)'.

**ax30** `causeOf`$(x, y) \implies$ `Occurrent`$(x) \wedge$ `Occurrent`$(y)$

> The arguments of the causation relation are both occurrents.

**ax31** $\neg$`causeOf`$(x, x) \quad \wedge$
`causeOf`$(x, y) \implies \neg$`causeOf`$(y, x)$

> Causation is irreflexive and asymmetric.

The latter two axioms are standard, for instance see [244].

We take the general terms 'cause' and 'effect' to be (relational [168]) roles relative to the causal relation `causeOf`. Therefore, whenever `causeOf`$(x, y)$ holds we can say that $x$ is the cause of $y$ and $y$ is the effect of $x$. In this approach, there is a correspondence between the causal relation and the cause and effect roles, so our theory does not need to introduce predicates for causes and effects (e.g. we never write'`Effect`$(x)$'). In the same way, the specialization of the concept of cause, say in root cause analysis, will be formalized through the specialization or generalization of the causation relation.

In [244], a theory of causation is proposed, which distinguishes causation along two axes: causation can be either direct, or indirect. Additionally, causation can be positive or negative. Direct and positive causation is called `achieves` and requires tight coupling between the arguments, in particular the juxtaposition or overlapping of their time extension. Direct and negative causation is called `prevents`. Indirect and positive causation is called `allows`, while indirect negative causation `disallows`. An important aspect of such a theory, which we also adopt, is that it refers to occurrent-instances (cfr Axiom (pr5)).

These relations can hold between actual occurrents or non-actual occurrents. Actual occurrents are those that have happened or will happen, at some time. Non-actual

trivial.

occurrents are those that may never happen, such as impossible events or events described in negative terms (e.g. 'the machine did not work'). For example, a prevented or disallowed occurrent is always non-actual since it did not (and does not) happen.

Below we report the formal theory of causation of [244] except that, for sake of simplicity, we do not use different notation for actual and non-actual events. In particular, the `achieves`, `prevents`, `allows`, and `disallows` relations are specializations of causation and any causal relation can be recognized into one of these four types. Among these four, achieve is the only primitive relation, the others will be defined from it and other auxiliary notions. we also define the subcase of positive causation that we will use in the next section:

**pr6** $\texttt{achieves}(x, y)$

"$x$ achieves $y$."

**ax32** $\texttt{causeOf}(x, y) \Longleftrightarrow \texttt{achieves}(x, y) \vee \texttt{prevents}(x, y) \vee \texttt{allows}(x, y) \vee \texttt{disallows}(x, y)$

"$x$ causes $y$ if and only if it either achieves or allows or prevents or disallows $y$."

**df44** $\texttt{posCausalCon}(x, y) \coloneqq \texttt{achieves}(x, y) \vee \texttt{allows}(x, y) \vee \texttt{facilPrecondFor}(x, y)$

"$x$ positively contribute to cause $y$ if and only if it either achieves or allows or is a facilitative precondition for $y$."

Note that the use of the locution 'positively contribute to cause' in place of a simpler 'positively cause' is due to the former case includes facilitative preconditions, so it not a strict subcase of the `causeOf` relation. Additionally, not all types of occurrents can be related by the `achieves` relation: for example, processes cannot achieve states, and events are never achieved. The reasoning for this is detailed case by case in [244], and we do not report it here[5].

**ax33** $\texttt{achieves}(x, y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x) \vee \texttt{State}(x)) \wedge (\texttt{Process}(y) \vee \texttt{State}(y)) \wedge \neg(\texttt{State}(x) \wedge \texttt{Process}(y))$

Next, `prevents` (the negative counterpart of `achieves`) is defined as achieving a state which is incompatible with the caused occurrence, where incompatible-with is a primitive of the theory that roughly indicates the physical impossibility of coexistence:

**pr7** $\texttt{incompatibleWith}(\mathrm{x},\mathrm{y})$

"$x$ is incompatible with $y$." For example, if we call $x$ the occurrence that 'yesterday I ate exactly two apples' and call $y$ the occurrence that 'yesterday I ate exactly one apple', then we say $\texttt{incompatibleWith}(x, y)$.

**df45** $\texttt{prevents}(x, y) \coloneqq \exists z(\texttt{State}(z) \wedge \texttt{incompatibleWith}(z, y) \wedge \texttt{achieves}(x, z))$

"$x$ prevents $y$ if and only if there is some state $z$ that is incompatible with $y$ and is achieved by $x$."

[5]This is because such a discussion is not strictly necessary to understand the subsequent sections. In addition, it would also be several pages long.

We shall further discuss the `incompatibleWith` primitive when expanding the causation ontology at the end of this section. Additionally, `prevents` has the same argument constraints seen for `achieves`:

**ax34** $\texttt{prevents}(x, y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x) \vee \texttt{State}(x)) \wedge (\texttt{Process}(y) \vee \texttt{State}(y)) \wedge \neg(\texttt{State}(x) \wedge \texttt{Process}(y))$

"If $x$ prevents $y$ then $x$ is either an event, a process, or a state, $y$ is either a process or a state, but $x$ cannot be a state if $y$ is a process."

Then, a primitive is introduced to (roughly) indicate that a state is a necessary or probable (pre-)condition for the happening of an occurrence (`facilPrecondFor`). Similarly, a primitive is used to state that a state is a (pre-)condition preventing or obstacling the happening of an occurrence (`prevPrecondFor`):

**pr8** `facilPrecondFor`(x,y)

"$x$ is a facilitative precondition for $y$."

**pr9** `prevPrecondFor`(x,y)

"$x$ is a preventive precondition for $y$."

**ax35** $\texttt{facilPrecondFor}(z, y) \implies (\texttt{State}(z) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y)))$

"If $z$ is a facilitative precondition for $y$ then $z$ is a state and $y$ is either an event, a process, or a state."

**ax36** $\texttt{prevPrecondFor}(z, y) \implies (\texttt{State}(z) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y)))$

"If $z$ is a preventive precondition for $y$ then $z$ is a state and $y$ is either an event, a process, or a state."

Additionally, it is assumed that, if some state $z$ is incompatible with a preventive precondition $w$ for some occurrence $y$, then $z$ is a facilitative precondition for $y$. For example, if "being properly cleaned" is a preventive precondition for a valve "being clogged", then "being dirty", which is incompatible with "being properly cleaned", is a facilitative precondition for "being clogged". The same holds when facilitative and preventive are switched:

**ax37** $\texttt{incompatibleWith}(z, w) \wedge \texttt{prevPrecondFor}(w, y) \implies \texttt{facilPrecondFor}(z, y)$

"If $z$ is incompatible with $w$ and $w$ is a preventive precondition for $y$, then $z$ is a facilitative precondition for $y$."

**ax38** $\texttt{incompatibleWith}(z, w) \wedge \texttt{facilPrecondFor}(w, y) \implies \texttt{prevPrecondFor}(z, y)$

"If $z$ is incompatible with $w$ and $w$ is a facilitative precondition for $y$, then $z$ is a preventive precondition for $y$."

Coming to indirect causation, `allows` is defined as either (i) achieving or maintaining a facilitative precondition, or (ii) preventing a preventive precondition (and assuming that, at the same time, no additional preventive precondition was achieved). Also, the only argument restriction for `allows` is that states cannot allow other occurrences:

**pr10** `maintains`(x,y)

"$x$ maintains $y$." This is a primitive concepts which is only exemplified in [244], for example, "the process of Suzy's omission to take vitamin C from her diet" *maintains* "the state of the lack of vitamin C in Suzy's body" [244, Example 10][6].

**df46** $\texttt{allows}(x,y) \coloneqq \exists z(((\texttt{achieves}(x,z) \vee \texttt{maintains}(x,z)) \wedge \texttt{facilPrecondFor}(z,y)) \vee (\texttt{prevents}(x,z) \wedge \texttt{prevPrecondFor}(z,y) \wedge \neg\exists w(\texttt{State}(w) \wedge \texttt{achieves}(x,w) \wedge \texttt{prevPrecondFor}(w,y)))$

"$x$ allows $y$ if and only if there exists $z$ such that (i) $x$ achieves or maintains $z$, and $z$ is a facilitative precondition for $y$, or (ii) $x$ prevents $z$ and $z$ is a preventive precondition for $y$ and there is no state $w$, preventive precondition for $y$ achieved by $x$."

**ax39** $\texttt{allows}(x,y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x)) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y))$

"If $x$ allows $y$, $x$ is either an event or a process and $y$ is either an event, a process, or a state."

Finally, `disallows` is the negative counterpart of `allows`, with 'facilitative' and 'preventive' exchanged:

**df47** $\texttt{disallows}(x,y) \coloneqq \exists z(((\texttt{achieves}(x,z) \vee \texttt{maintains}(x,z)) \wedge \texttt{prevPrecondFor}(z,y)) \vee (\texttt{prevents}(x,z) \wedge \texttt{facilPrecondFor}(z,y)))$

"$x$ disallows $y$ if and only if there exists $z$ such that (i) $x$ achieves or maintains $z$, and $z$ is a preventive precondition for $y$, or (ii) $x$ prevents $z$ and $z$ is a facilitative precondition for $y$."

**ax40** $\texttt{disallows}(x,y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x)) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y))$

"If $x$ disallows $y$, $x$ is either an event or a process and $y$ is either an event, a process, or a state."

As a final remark, notice that Toyoshima's theory of causation specifies that, from an ontological viewpoint, each causal relation needs a context. we do not discuss what a context is, but we notice that it contains information required to make causal relation works. For example, a door being closed `prevents` somebody from entering a house only in the context of that house not having other accessible openings. In the following we will always assume the existence of a suitable context which we leave implicit.

**Extensions of the causation ontology** In the causation theory (ax32)-(ax40), one cannot say that an occurrence is internal to another occurrence, where "internal" means that the spatial location of the first occurrence is included in the spatial location of another occurrent or of an object. Hence, we introduce a further primitive:

[6]The choice of involving omissive/negative/non-actual occurrences in causation is a non-trivial ontological commitment taken by Toyoshima's onotlogy, which we shall discuss a little bit more in depth below, after having introduced the (expanded) Toyoshima's ontology.

**pr11** `internalTo`(x,y)

> "$x$ is internal to $y$." For example, consider the case that the right headlight of a car that is not working because the filaments of its light bulb are burned. Then, the occurrence 'the filaments are burned' (in other words, 'the filaments are disconnected') is internal to the occurrence 'the right headlight is not working'. However, note that there are some complications with this predicate, e.g., see (df48*) and (df49*).

For the sake of clarity, we briefly discuss how such a primitive may be defined using a formal ontology supplying both a mereology of objects and an object-occurrence participation relation (e.g., the DOLCE ontology). The first possibility, is to use spatio-temporal inclusion:

**ax41*** $\texttt{internalTo}(x, y) \iff x \subseteq_{ST} y$

> "If $x$ is internal to $y$, $x$ is spatio-temporally included in $y$."

However, DOLCE's spatio-temporal inclusion between perdurants is built on perdurant locations, whose constraction is itself somewhat complex [177]. Then, for the sake of simplicity, we supply a simplified treatement: we say that one occurrence is internal to another when each participant of the first occurrence are proper parts of some participant of the second occurrence. Analogously we define the case that an occurrence is internal to an object. In what follows, logical expressions marked by an asterisk (*) are not taken as part of the ontology, they are added to introduce points of discussion.

**df48*** $\texttt{internalTo}(x, y) \wedge \texttt{Occurrent}(x) \wedge \texttt{Occurrent}(y) \iff$
$\forall z, t(\texttt{participatesIn}(z, x, t) \implies$
$\exists w(\texttt{participatesIn}(w, y, t) \wedge \texttt{partOf}(z, w, t) \wedge z \neq w))$

> "An occurrent $x$ is internal to the occurrent $y$, if and only if, for each participant $z$ of $x$ (at time $t$), there is a participant $w$ of $y$ (at $t$) which has $z$ as proper part (at time $t$)."

**df49*** $\texttt{internalTo}(x, y) \wedge \texttt{Occurrent}(x) \wedge \texttt{Object}(y) \iff$
$\forall z, t(\texttt{participatesIn}(z, x, t) \implies \texttt{partOf}(z, y, t) \wedge z \neq y))$

> "An occurrent $x$ is internal to the object $y$, if and only if each participant $z$ of $x$ (at time $t$), is proper part of $y$ (at time $t$)."

Notice that (df48*) says that the first occurrence of `internalTo` is such that all its participants are proper parts of participants in the second occurrence. Analogously for (df49*). The reason for this particular definition can be seen (at least also) with the following example: assume that, in a chemical plant, a part of a device breaks off and flows downstream, until it arrives in the chamber of a valve, making the valve stuck and causing malfunctioning. Then, by (df48*), the occurrence "there is a foreign object in the valve" is *not* internal to the valve, because the foreign object is not part of the valve. we would have reached the opposite conclusion if we, instead, had defined `internalTo` by, say, building the spatial location of an occurrence from those of their participants, and stated that $x$ is internal to $y$ if the corresponding spatial locations are subset one of the other, since then the location of the foreign object is within the location of the

valve. We want to exclude this interpretation, as, together with the oncoming definition of fault (df55) would entail that the valve itself is at fault, while in that case we would say that it is the broken component that is at fault, while properly speaking the valve is in a external disabled state (df56). Compare the aforementioned example of the burned light bulb filaments. In that case, the occurrence 'the filaments are burned' is internal to the occurrence 'the right headlight is not working' due to the filaments being part of the headlight, so we will say, by Definition (df55), that the headlight is at fault, as needed.

Coming back to the `incompatibleWith` primitive, we interpret it as physical incompatibility between two occurrences. For example, if we call $x$ the occurrence that 'yesterday I ate exactly two apples' and call $y$ the occurrence that 'yesterday I ate exactly one apple', then we say $\texttt{incompatibleWith}(x, y)$. Similarly, if $z$ is 'yesterday I ate less than three apples' then is clear that $x$ and $y$ both 'imply' $z$, meaning that if $x$ or $y$ happened, also $z$ happened. Since this kind of implication (that we call 'physical consequence') is often mentioned in failure analysis (e.g. 'peak vibration is 1.13 times over nominal limit' *causing* the 'fatigue imposed on the axis to be 1.13 times than normal' – taken from [180]), we introduce it explicitly below:

**pr12** $\texttt{physicalConseqOf}(x, y)$

"$y$ is a physical-consequence of $x$."

**ax42** $\texttt{physicalConseqOf}(x, y) \implies \texttt{Occurrent}(x) \wedge \texttt{Occurrent}(y)$

"The argument of the physical-consequence of relation are occurrents."

**ax43** $\texttt{physicalConseqOf}(x, y) \wedge \texttt{physicalConseqOf}(y, z) \implies \texttt{physicalConseqOf}(x, z)$

"physical-consequence is transitive."

Note that physical-consequence is a different type than causal-consequence (e.g., `achieves`). For instance, a key difference is that causal-consequence requires a (possibly extended) spatial and temporal contiguity or overlapping, while physical-consequence is 'immediate'/'synchronous' and does not take place somewhere nor during some time. Another difference is that causal-consequence is antisymmetric and not (generally speaking) transitive, while physical-consequence is transitive and could be bidirectional. Formally, we assume that physical-consequence extends causal-consequence as follows:

**ax44** $X(x, y) \wedge \texttt{physicalConseqOf}(y, z) \implies X(x, z)$ for $X \in \{\texttt{achieves}, \texttt{allows}\}$

"If $x$ achieves $y$, which physically-entails $z$, then $x$ achieves $z$." (Analogous case for 'allows'.)

**ax45** $X(x, z) \wedge \texttt{physicalConseqOf}(y, z) \implies X(x, y)$ for $X \in \{\texttt{prevents}, \texttt{disallows}\}$

"If $x$ prevents $z$, which is physically-entailed by $y$, then $x$ prevents $y$." (Analogous case for 'disallows'.)

For example, if $x$, $y$, $z$ stand for 'the table is under the silverware', 'the fork is falling', and 'some silverware is falling', respectively, then $x$ prevents $z$ and $z$ is a physical-consequence of $y$, so that $x$ prevents $y$ also, by (ax45).

**Dealing with non-actual occurrences.** Above, we have referenced non-actual occurrences. The need for these arises due to negative causation often needed in failure analysis. For instance, if the operation of a pressure relief valve prevents the pressure in a tank from exceeding a certain threshold, the occurrence of the pressure rising is non-actual. However, non-actual occurrences are a complex ontological issue, and some potential ontology users may prefer to avoid dealing with these.

For this reason, we list a few possible strategies to handle this issue:

- Rephrasing to eliminate negative causation: one can rephrase statements to remove instances of negative causation. For example, instead of saying $x$ prevents $y$ because $x$ achieves a $z$ that is incompatible with $y$, we could say $x$ achieves $\sim y$. E.g., changing 'the fracture of the headlight's filament prevents the flowing of current in the filament, since the open failure of the filament is incompatible with a current flow' to 'the filament fractured, achieving the state of no current flow in the filament'. However, this strategy may not always be possible or desirable.

- Replacing negative occurrences with types: one can instead change the definition of negative causation. Instead of stating that, by definition, $x$ prevents $y$ if $x$ achieves a state $z$ that is incompatible with $y$, we state $x$ (an occurrence-token) prevents $Y$ (an occurrence-type) if $x$ achieves a state $z$ that is not compatible with $Y$. Similarly, we can adjust the definition for disallowing. See e.g. [15] for an implementation of this strategy in an UFO-based framework. Such a reference in particular employs the this strategy to define type-level analogous of Toyoshima's incompatible-with and prevents relations. However, in whenever following this approach introduces complexity as the incompatibility of a token with a type cannot be reduced to merely stating that no type-instance occurs while the token occurs. It is necessary to clarify that the non-occurrence is essential; otherwise, the token would be incompatible with any type that happens not to have instances occurring at the same time as the token.

**Time extension of occurrences and causation.** Throughout the axiomatization of Toyoshima's causation ontology, we omitted to formalize relations about the temporal extensions of occurrences and the link between these and the causal relations. A most classical example of relations between temporal extensions would be Allen's algebra [4], which deals with time intervals and defines relations such as 'meet' (two time intervals meet if the end of one coincide with the start of the other) and 'precedes' (if one ends before the other starts), among others. It is possible to extend Toyoshima's ontology with Allen's algebra or similar theories and specify further constraints, such as that an effect cannot occur before a cause. However, Allen's algebra is a well-known ontology and the links between its relations with the causal relations would not have relevant effects on the rest of the work of the thesis, so we omit such formalization for the sake

of brevity and simplicity. An example of a different theory of occurrences that includes both causation and some of Allen's algebra relations is UFO-B [20, 104]: for instance, it is a theorem in that ontology that, if $e_1$ directly causes $e_2$ and $e_2$ directly causes $e_3$, then $e_1$ meets $e_2$, $e_2$ meets $e_3$, and $e_1$ is before $e_3$. In passing, note also that UFO-B is (also) an example of an ontology of causation in which causation is mediated by dispositions: an event ($e_1$) directly causes another ($e_2$) if $e_1$ brings about a situation that triggers (i.e. activates a disposition that manifests in) $e_2$. This is much different from Toyoshima's ontology, but at the same time analogous: in such an ontology it is states that are mediators, since $e_1$ (indirectly) causes $e_2$ if $e_1$ achieves a state that is a facilitative precondition for $e_2$.

#### 4.0.1.2 Taxonomy of malfunction-related happenings

At the beginning of this section, we identified three key aspects of malfunction-related happenings. Using these aspects, we construct a taxonomy. At the top of this taxonomy (excluding 'occurrent') is the term 'malfunction-related happenings', which we have used as a comprehensive term to encompass all occurrences related to a failure.[7]. This taxonomy is depicted in Figure 4.1, a decision diagram for classification of malfunction-related happenings is present in Figure 4.2. Below, we provide formal definitions for the elements of this taxonomy and discuss them in detail.

We begin by introducing the relation `functionIncompatible` to identify malfunction-related occurrences that directly disrupt the normal functionality of artifacts, which we refer to as malfunctions. The purpose of this predicate is to link the previously developed functional theory with the oncoming taxonomy, by defining the `Malfunction` class; it is not a concept included in the taxonomy proper (Figure 4.1). Specifically, the predicate makes use the ascription of a systemic function as defined in (df16). Note that, alternatively, such a predicate could be directly introduced as a primitive, without committing to the rest of the framework presented in this thesis, the remaining definitions would stay unchanged. This ensures the modularity of this portion of the theory, so that a putative user could utilize it independently.

**df50** $\texttt{functionIncompatible}(x, f) \coloneqq \exists o, b, t(\texttt{participatesIn}(o, x, t) \wedge \texttt{systemicFunctionOf}(f, o) \wedge \texttt{CF}(b, f, t) \wedge \texttt{incompatibleWith}(x, b))$

> "$x$ is function-incompatible (or not function-compatible) with $f$ if and only if there exist $o, b, t$ such that $o$ is an object participating in $x$ at time $t$, $f$ is one of $x$'s systemic functions, $b$ is a behavior through which $f$ is realized at time $t$, and $b$ is incompatible with $x$." Note that the $b$ in (df50) must be the behaviour $o$ participates-in-as-doer due to Definition (df16). An occurrent is functional-incompatibile with a function if it is physical-incompatibility with any execution of the function by some object that participates in the current. For instance, if an electric wire's function is to carry electricity, then the "the wire being cut in two" is an occurrence that is physically-incompatible with any execution of the wire's

[7] Ontologically speaking, the class 'malfunction-related happening' is introduced as a subclass of occurrents relevant to this work. This introduction does not imply that these occurrents have a special ontological status.

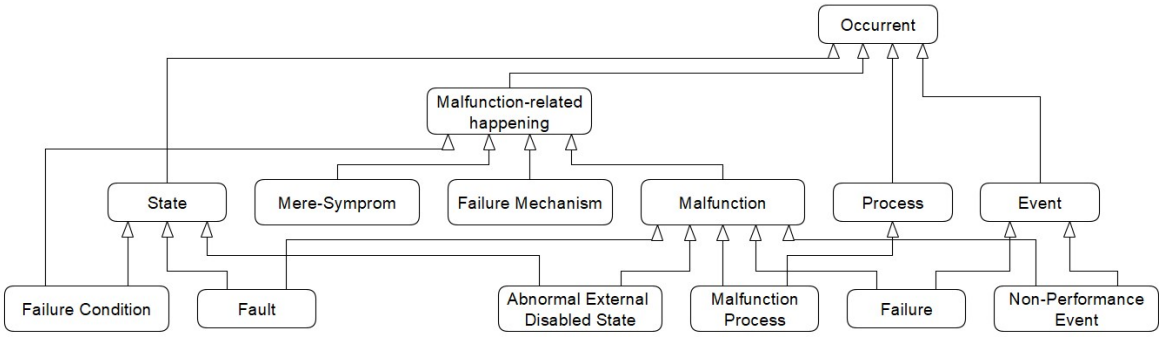



Figure 4.1: The taxonomy introduced in this chapter.

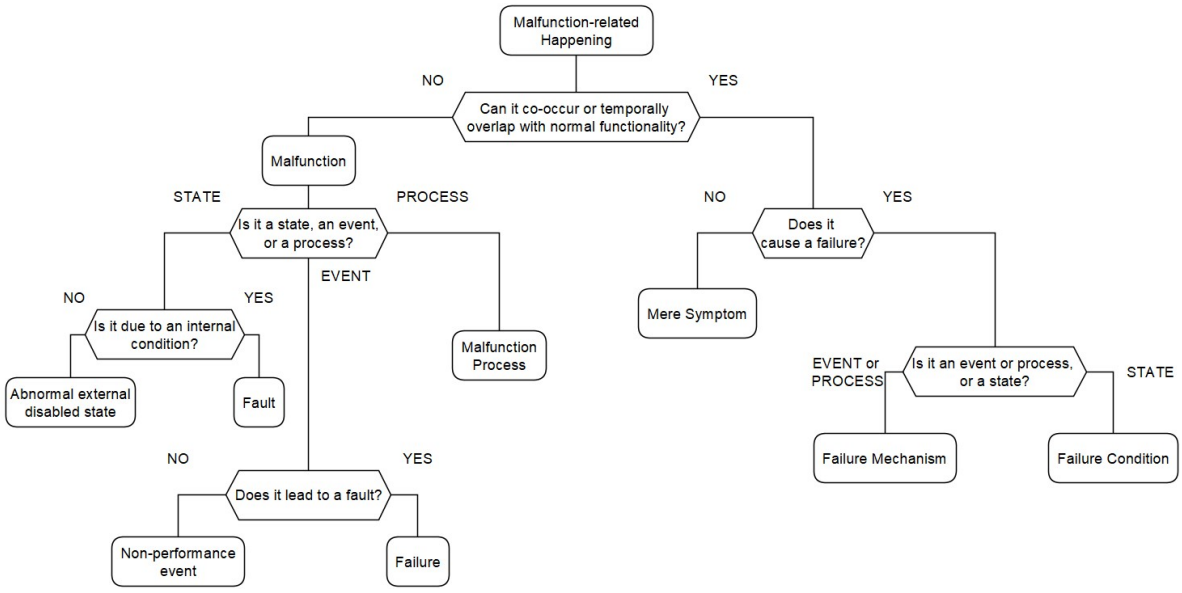



Figure 4.2: A decision tree that, by following top-down the tree and answering the relative questions, helps classify malfunction-related happenings into MALFO's classes. Such questions are only mnemonic heuristics.

function, hence it is `functionIncompatible` with the function itself. Conversely, if a light beam is present in front of the headlights of a car, such an occurrence is compatible with the function of the headlights.
This idea of incompatibility does not consider future effects of the current, e.g. a small crack can exist simultaneously with the normal operation of an axis transmitting torque, even though the such a crack may potentially lead to a fracture in the future, and the fracture itself is incompatible with the axis functionality. Another example: if the function of an electric wire is to carry electricity, and a logically-necessary requirement for the wire of doing that is being physically connected, then 'the wire being cut in two' is `functionIncompatible` with the wire's function.

**df51** $\texttt{Malfunction}(x) \coloneqq \exists f \texttt{functionIncompatible}(x, f)$

"$x$ is a malfunction means that there is some function $f$ which is function incompatible with $x$." Examples are, from the previous examples, the fracture of an axis (incompatible with the axis' function of transmitting torque) or the occurrence that a headlight of a car does not turn on when it should (incompatible with the headlight function of lighting the environment in front of the car when requested). This definition of malfunction is almost a literal interpretation of the term malfunction, except that, in reality, the compatibility could be complete or partial. We ignored this for the sake of simplicity.

**df52** $\texttt{FailureMechanism}(x) \coloneqq ((\texttt{Process}(x) \vee \texttt{Event}(x)) \wedge$
$(\neg\texttt{Malfunction}(x)) \wedge \exists y(\texttt{Failure}(y) \wedge \texttt{posCausalCon}(x, y)^{\dagger}))$

"$x$ is a failure mechanism means that $x$ is not a malfunction, is either a process or an event, and is the positive causal contributor, eventually, of some failure." Examples are common occurrence-types such as corrosion, wear and cracking which are not typically classified as failures themselves but do cause failures in some sense, which is made precise here.

**df53** $\texttt{FailureCondition}(x) \coloneqq (\texttt{State}(x) \wedge (\neg\texttt{Malfunction}(x)) \wedge$
$\exists y(\texttt{Failure}(y) \wedge (\texttt{posCausalCon}(x, y)^{\dagger} \vee$
$\exists z(\texttt{prevPrecondFor}(x, z) \wedge \texttt{disallows}(z, y)))))$

"$x$ is a failure condition means that $x$ is not a malfunction, is a state, and is either the positive causal contributor, eventually, of some failure, or it is the preventive precondition for some occurrence that disallows a failure." Failure conditions are similar to failure mechanisms, only they are states and not processes (e.g. 'constant high temperature' in 'place of rising temperature'). Negative causation is considered explicitly because it can happen that, e.g., a safety valve is in a corroded state and this inhibits its safety function, subsequently leading to a failure.

**df54** $\texttt{MereSymptom}(x) \coloneqq \neg\texttt{Malfunction}(x) \wedge \neg\texttt{FailureMechanism}(x) \wedge$
$\neg\texttt{FailureCondition}(x) \wedge \exists z(\texttt{Malfunction}(z) \wedge \texttt{causeOf}(z, x))$

"$x$ is a mere symptom means that $x$ is not a malfunction, nor a failure mechanism, nor a failure condition, and is caused by a malfunction." A mere symptom does not causally affect a system. For instance, if the right headlight of a car fails

to illuminate, the state of 'there is no light beam from the headlight' is a mere symptom: this state does not lead to the failure of other components[8]. Negative causation (included in `causeOf`) is relevant here because a symptom could be, say, the absence of an occurrence prevented by the malfunction. This is the case of the missing light in the example above.

**df55** $\texttt{Fault}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{State}(x) \wedge \exists z(\texttt{internalTo}(z, x) \wedge \texttt{achieves}(z, x))$

"$x$ is a fault means that $x$ is a malfunction and a state and is achieved by some occurrence internal to itself." We weakened the classic definition of fault "inability to function due to an internal state", by not requiring the $z$ to be a state: we consider relevant the possibility that $z$ may be of another type. For instance, if 'fracture of the main axis' ($z$) causes 'the industrial fan is completely non-operational' ($x$), then the fracture is an event internal to $x$. Since $x$ is a function-incompatible state, the fracture is classified as a fault even though is not a state. Another example, this time in which the $z$ is a state, is the 'being cut into two parts'-state ($z$) of an electrical wire, which is an internal state that causes the state of 'being unable to transmit electricity' ($x$), which is then a fault by definition[9].

**df56** $\texttt{AbExtDisabledSt}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{State}(x) \wedge \neg\texttt{Fault}(x)$

"$x$ is an abnormal external disabled state means that $x$ is a malfunction and state but not a fault." External disabled states are introduced to separate the malfunction states that need corrective actions from those that do not, because the malfunction to be corrected resides somewhere else. For example, when the right headlight of a car is not lit when it should, there are at least two scenarios to consider. The headlight itself may be faulty, say there is a blown lightbulb inside the headlight; in this case the headlight not being lit is a fault that requires repairing the headlight. In the second scenario, another component is faulty, say there is a severed wire, external to the battery, connecting the headlight to the car's battery. Here, the wire must be replaced, and not being lit is an abnormal external disabled state. Note also that the term 'abnormal external disabled state' is very similar to EN 13306-'external disabled states'. The difference is that the latter correspond to the union of the two leftmost leaf classes in Figure 2.7, while MALFO-'abnormal external disabled states' are more specific and correspond to just the 'lacks required external resources' leaf. This is because we have declared that malfunction-related happenings are unwanted occurrences, while planned occurrences are not unwanted.

[8]The absence of light while driving might indeed cause a failure of the car by causing an accident. However, we are excluding this scenario. Imagine, say, that the car is being repaired by a mechanic, rather than being driven.

[9]Notice that, if we were to introduce in our conceptualization performance parameters that allow us to speak about acceptable/unacceptable functional performance, we could talk about 'partial functional incompatibility': when an occurrence is incompatible with the execution of a function within specified nominal bounds. In that case, we could also talk of 'partial faults', for instance, if the wire is not broken but, say, half of its cross-section has been cut away, then the wire has a 'partial' fault. As said before, we are not concerned with these matters in this chapter.

**df57** $\texttt{Failure}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{Event}(x) \wedge \exists y(\texttt{Fault}(y) \wedge \texttt{achieves}(x, y))$

"$x$ is a failure means that $x$ is a malfunction and an event that achieves some fault." This definition follows the common idea of failure as the loss of the ability to perform a function, which brings about a fault. For instance, consider a wire that is gnawed by a rat. When the wire fractures it ceases to be able to execute its function, that is to transmit electricity. In this case, the fracture is a `Failure` and the ensuing state of the wire is a `Fault`[10].

**df58** $\texttt{NonPerformanceEvent}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{Event}(x) \wedge \neg\texttt{Failure}(x)$

"$x$ is a non-performance event means that $x$ is a malfunction and an event, but not a failure." For example, if, when attempting to turn on the right headlight of a car, its lightbulb is broken, 'the right headlight did not turn on' is a non-performance event, because it is a malfunction but does not lead to any additional failures by itself.

**df59** $\texttt{MalfunctionProcess}(x) \coloneqq (\texttt{Malfunction}(x) \wedge \texttt{Process}(x))$

"$x$ is a malfunction process means that $x$ is a malfunction and a process." Malfunction processes complete the partition of malfunctions into subclasses. An example of malfunction process is the 'erroneous calculation process' that Avizienis describes in [10]: a faulty integrated circuit is invoked to calculate some quantity and this causes wrong voltage values at various times during the calculation process and in different circuit locations.

In summary, 'failure', 'fault', and 'failure mechanism' are defined to encapsulate their commonly associated descriptions. Other terms are introduced as variations or in opposition to these concepts: Failure condition describes occurrents that play a similar causal role to failure mechanisms. However, failure conditions are categorized as states, whereas failure mechanisms are typically processes. Non-performance events contrast with failures because they neither constitute a failure nor cause one, despite being incompatible with nominal functionality. Mere symptoms differ from the previous terms in that they do not cause or consist of a malfunction; they are merely effects of such an occurrence. A malfunction is defined as any occurrence that is incompatible with nominal functionality. To categorize malfunctions further, malfunction processes are introduced as a subclass. External disabled states contrast with faults, as they indicate the inability to execute functions not due to an internal state.

All partitions of the taxonomy classes in subclasses are complete, except for {`Failure Mechanism`, `FailureCondition`, `MereSymptom` }, which may fail to cover the complement of the `Malfunction` category. This depends on what malfunction-related happenings are, exactly. We did not give a definition, however, one could stipulate, say, that malfunction-related happenings are exactly the union of all the taxonomy classes

[10]Of course, considering other functions of the wire are considered gives raise to different failure occurrences. For example, considering also the protective function of the insulation, the event of the insulation piercing becomes a failure. Also, considering degraded performance in the function execution, e.g. increase of the electric resistance of the wire due to reduction of cross-sectional area, would naturally give raise to 'partial' faults and failures, which we do not consider for sake of simplicity.

(df60)*, in this case the partition is trivially complete. One could also opt for an intentional definition: a malfunction-related happening is, say, any occurrent that is transitively related to some malfunction by any combination of the causal relations (df61). Then there may exist some malfunction-related happenings that are (non-malfunctions and are) not covered by the taxonomy, since they are related to a malfunction by an 'exotic' causal chain, not considered in the taxonomy definitions.

**df60*** $\texttt{MalfunctionRelHapp}(x) \iff (\texttt{Malfunction}(x) \vee \texttt{MereSymptom}(x) \vee \texttt{FailureMechanism}(x) \vee \texttt{FailureCondition}(x))$

"$x$ is a malfunction-related happening if and only if $x$ is either a mere symptom, a malfunction, a failure mechanism, or a failure condition."

**df61*** $\texttt{MalfunctionRelHapp}(x) \iff \exists y(\texttt{functionIncompatible}(x, y) \wedge ((\texttt{causeOf}(x, y)^{\dagger}) \vee (\texttt{causeOf}(y, x)^{\dagger})))$

"$x$ is a malfunction-related happening if and only if there is a function-incompatible occurrence that is related with $x$ by the transitive closure ($\texttt{causeOf}^{\dagger}$) of the `causeOf` relation plus the `facilPrecondFor` relation (which has to added so that `posCausalCon` is a subcase of $\texttt{causeOf}^{\dagger}$)."

Axioms (df51) to (df59) constitute our formal taxonomy. Failure-mode is not part of the taxonomy because in the literature it is a polysemous term with (at least) the following meanings:

- domain-level sub-*type* of malfunction-related happenings ('mode', or 'manner', in the sense of 'type'). So that 'rupture', 'overheating', 'wear', 'electrical corrosion' would be failure modes. These types are also (possibly) related to a system component, in the sense that, given a certain component in a certain system, if that component participates in a malfunction-related happening, that happening must belong to a set of occurrence-types, which are the (typical) failure-modes of the component. Given that, in this case, the name of a failure mode is the name of an occurrence type that an underlying system or component has can participate in, this sense of 'failure mode' is very similar to dispositions: consider e.g. the example of the disposition to break and the disposition to crack of a vase by Barton et al. [18].

- Any *attribute* of Table 2.1, or a similar one ('mode', in the sense of 'attribute'). Persistency, criticality, etc. would be failure-modes.

- A specific *role* in a causal chain of malfunction-related happenings ('mode' as a causal, relational role intermediate between cause and effect). This is the apparent meaning in e.g. figures 2.6a and 2.6b, and [25]. Therefore, any malfunction-related happening could be a failure-mode, if considered in appropriate causal chain and granularity level.

The first and second of these meanings are not (fully) treatable with the three aspects that we used to create the taxonomy (occurrence-category, causal role, and internality). The third is, but we claim to already have a taxonomy that covers effectively all

malfunction-related happenings using these three aspects, so the third meaning can be ignored; similarly, while the component-to-failure-mode relation of the first meaning cannot be treated, each of those failure-mode is still a sub-type of a category of our taxonomy.

Importantly, note that the meaning of failure modes as types of malfunction-related happenings is likely the meaning employed in FMEA. Since that meaning is at type-level more than token-level, it is difficult to treat with the tools used in this chapter. Additionally, a FMEA table collecting failure modes of components is also an epistemic artefact, in the sense that it collects the *known* failure modes of components, not all possible failure modes: even considering only failure modes of a certain relevance, and assuming that such failure modes are only a finite number, an engineer will never be sure that a FMEA table contains all modes, even after an extensive refinement from, say, a FRACAS methodology. The following formula, where $\Phi$ is a type of occurrents, tries to capture this idea of failure modes:

**ex33** $\texttt{failureModeOf}(\Phi, \texttt{f}, o) \Longleftrightarrow \texttt{systemicFunctionOf}(\texttt{f}, o) \wedge \forall e \Phi(e) \implies \exists e, f, t$
$((\texttt{Fault}(e) \vee \texttt{Failure}(e)) \wedge \texttt{PC}(o, e, t) \wedge \texttt{functionIncompatible}(e, \texttt{f}))$

"A type of occurrents is a failure mode of a systemic function ($f$) of some object ($o$) if and only if it instantiates only occurrents that are either faults or failures, (ii) are participated by the object at some time, and (iii) are incompatible with a function of said object." Note that the choice for faults or failures is due these two classes are the ones most linked with an internal condition of the object.

However, due to (ex33) containing a second-order relation and important epistemic issue of knowing about the occurrent $e$ in (ex33), we keep this definition outside of MALFO proper.

As a consequence of our claim that the taxonomy covers effectively malfunction-related happenings, we should be able to classify the terms of malfunction-related happenings found in the literature within our taxonomy, and also to map the various failure-modes taxonomies within ours. We show this in Table 4.1 for an important part of the terminology used in the literature. The classification should be taken as an approximation in a couple of cases since sometimes terms are introduced only informally in the literature. We also discuss, in the next section 4.1.1, informal mappings of our theory with a few taxonomies present in the literature.

| TERM | CLASSIFICATION | ORIGINAL CLASSIFICATION | SOURCE |
|---|---|---|---|
| Abnormal instrument reading | Mere symptom | Failure mode [False alarm, faulty instrument indication] | [127] |
| Plugged/chocked | Ab. External disabled state | Failure mechanism [External influence – Flow restricted/blocked due to fouling, contamination, icing, flow assurance (hydrates), etc.] | [127] |
| Breakage | Failure | Failure mechanism [Material failure – Fracture, breach, crack] | [127] |

| | | | |
|---|---|---|---|
| Breakdown | Malfunction-related happening [anything 'high severity'?] | Failure mode [Breakdown, serious damage (seizure, breakage), and/or major process fluid leak] | [127] |
| Burst | Failure [explosion is DOLCE-event] | Failure mechanism [Item burst, blown, exploded, imploded, etc.] | [127] |
| Cavitation | Failure mechanism | Failure mechanism [eroded, worn out, worn] | [127] |
| Clearance/ alignment failure | Failure mechanism [can operate while misaligned] | Failure mechanism [Failure caused by faulty clearance or alignment] | [127] |
| Contamination | Failure mechanism | Failure mechanism [Contaminated fluid/gas/-surface, e.g. lubrication oil contaminated, gas-detector head contaminated] | [127] |
| Control/ signal failure | Failure | Failure mode [No, or faulty monitoring or regulation failure to transmit or receive command or data, or failure to actuate function] | [127] |
| Corrosion | Failure mechanism | Failure mechanism [All types of corrosion, both wet (electrochemical) and dry (chemical)] | [127] |
| Corrosion | Failure mechanism | Failure mode | [245] |
| Corrosion | Failure mechanism | Failure mode | [129] |
| Crack initiation | Failure mechanism | Event | [180] |
| Deformation | Failure mechanism [a deformation is a fault, but creeping etc. are fail. mech.] | Failure mechanism [Distortion, bending, buckling, denting, yielding, shrinking, blistering, creeping, etc] | [127] |
| Deformation, plastic | Failure mechanism | Failure mode [creep, buckling, . . . ] | [245] |
| Deformation, plastic | Fault | Failure mode [permanent deformation] | [129] |
| Delayed operation | Non-performance event | Failure mode [Delayed response to commands] | [127] |
| Earth/ isolation fault | Fault [low electrical resistence state] | Failure mechanism [Earth fault, low electrical resistance] | [127] |
| Electrical failure - general | Malfunction* | Failure mechanism [Failures related to the supply and transmission of electrical power, but where no further details are known] | [127] |
| Erosion | Failure mechanism | Failure mechanism [Erosive wear] | [127] |
| Erosion, electrical | Failure mechanism | Failure mode | [129] |
| Erratic output | Non-performance event [intermittent] | Failure mode [cillating, hunting, instability] | [127] |
| Failure to start on demand | Non-performance event | Failure mode [Doesn't start on demand] | [127] |

| | | | |
|---|---|---|---|
| Failure to stop on demand | Non-performance event | Failure mode [Doesn't stop on demand] | [127] |
| Fan fails | Failure | Event | [180] |
| Fatigue | Failure mechanism | Failure mechanism | [127] |
| Fatigue | Failure mechanism | Failure mode | [245] |
| High/cycle fatigue conditions | Failure mechanism | Failure mechanism | [180] |
| High output | Non-performance event | Failure mode [Overspeed/output above acceptance] | [127] |
| Instrument failure - general | Mere symptom | Failure mechanism [Failure related to instrumentation but no details known] | [127] |
| Insufficent power | Abnormal external disabled state | Failure mode [Lack of or too low power supply] | [127] |
| Leakage | Failure condition or fault* | Failure mechanism [External and internal leakage, either liquids or gases] | [127] |
| Leakage, external - fuel | Failure condition or fault* | Failure mode [External leakage of supplied fuel/gas] | [127] |
| Leakage, internal | Failure condition or fault* | Failure mode [Leakage internally of process or utility fluids] | [127] |
| Looseness | Failure mechanism | Failure mechanism [Disconnection, loose items] | [127] |
| Low output | Non-performance event | Failure mode | [127] |
| Minor in-service problems | Malfunction-related happening [anything low criticality?] | Failure mode [Loose items, discoloration, dirt] | [127] |
| No power/ voltage | External disabled state | Failure mechanism [Missing or insufficient electrical power supply] | [127] |
| No signal/indication/alarm | Mere symptom** | Failure mechanism [No signal/indication/alarm when expected] | [127] |
| Noise | Mere symptom | Failure mode [Abnormal noise] | [127] |
| Normal operation of the fan | Failure mechanism*** | Contributory factor | [180] |
| Open circuit | Fault | Failure mechanism [Disconnection, interruption, broken wire/cable] | [127] |
| Out of adjustment | Fault | Failure mechanism [Calibration error, parameter drift] | [127] |
| Overheating | Fault | Failure mechanism [Material damage due to overheating/burning] | [127] |
| Overheating | Malfunction mechanism | Failure mode [Machine parts, exhaust, cooling water] | [127] |
| Principal shear stresses exist | Failure condition | Failure mechanism | [180] |

| | | | |
|---|---|---|---|
| Rupture | Failure | Failure-mode | [245] |
| Short circuiting | Fault | Failure mechanism | [127] |
| Spurious stop | Non-performance event | Failure mode [Unexpected shutdown] | [127] |
| Sticking | Non-performance event | Failure mechanism [Sticking, seizure, jamming due to reasons other than deformation or clearance/alignment failures] | [127] |
| Stress concentration at keyway corners | Failure condition | Failure mechanism | [180] |
| Structural deficiency | Fault | Failure-mode [Material damages (cracks, wear, fracture, corrosion)] | [127] |
| Vibration | Failure mechanism or mere symptom* | Failure mechanism [Abnormal vibration] | [127] |
| Vibration | Failure mechanism or mere symptom* | Failure mode [Abnormal vibrations] | [127] |
| Vibration | Failure mechanism | Contributory factor | [180] |
| Wear | Failure mechanism | Failure mechanism [Abrasive and adhesive wear, e.g. scoring, galling, scuffing, fretting] | [127] |
| Wear | Failure mechanism | Failure-mode | [245] |
| Wear | Failure mechanism | Failure-mode | [129] |
| Wear | Failure mechanism | Contributory factor | [180] |

Table 4.1: A list of some terms taken from the engineering literature, associated with their original classification and a putative one from our ontology (items in the first column are to be intended as subclasses of the corresponding items in the second column). Notice that the classification may use non-leaf terms from our taxonomy (*) or it may be a partially arbitrary choice due to the term's meaning being under-specified in its original context (**). Square brackets in the second column refer to the attributes of Table 2.1.

#### 4.0.1.3 Finer causal roles

In the previous section, we introduced concepts such as `Failure` based on their causal relationships with other occurrences. These concepts do not require comparison with other causes. In this section, however, we delve into causes that can be compared to other causes.

In Section 2.2.1, we briefly mentioned immediate causes, defining them as occurrences directly related to an effect through the `causeOf` relation. Here, 'immediate' contrasts with causes situated further along a causal chain, where non-immediate causes are related to an effect through the transitive closure of the `causeOf` relation. This exploration is valuable when discussing cause-types that lie between immediate-cause and transitive-cause, as the latter is too broad and the former too specific.

Root causes exemplify such intermediate cause-types. Defining this concept poses

challenges. We cannot rely on Del Frate's forward-looking perspective, which involves hypothetical scenarios of amending the root cause, as Toyoshima's causal ontology focuses solely on actual occurrences. Thus, we adopt a backward-looking perspective, aiming to discern the relative contributions of different causes to an effect.

**Effective sufficient causes** Determining what constitutes the 'true' cause of an effect is ontologically complex. Consider a force pushing an object, resulting in its movement. We might intuitively attribute the cause of the movement to the application of the force. However, while the force's application is necessary (if we mean to obtain the object's acceleration), it is not sufficient on its own: the presence of a wall or excessive static friction could prevent movement. The absence of these obstructions is necessary but not sufficient, suggesting an equivalence with the force's application, which is counterintuitive.

To address this issue, Mizoguchi and collaborators recently proposed a development in the causation ontology[11]. They suggest that genuine sufficient causation, interpreted as the `achieves` predicate, occurs only between an event and a state or between two states. For instance, an event like 'the deer was killed' achieves the state 'the deer being dead', or a state like 'the valve being half-open' achieves 'the flow rate being half of the capacity'.

In other cases, what seems like genuine sufficient causation is often a form of indirect causation (necessary-causes or 'allowments'). For example, the application of force is not genuinely sufficient to move a body because its effectiveness relies on a context where multiple states set up the conditions for movement. These states negate preventive preconditions and include all facilitative preconditions for body movement except for one: the application of some strong enough force. When an event achieves this final precondition the body moves. It appears that the event 'the body started to move' was achieved by the force application, but the force application merely allowed the movement. Hence, the event causing movement should be termed an 'effective sufficient cause', which effectively achieves the effect. The determination of which causes are effective depends on the context, but ontologically, only an accumulation of `allows` exists in the effective-causation scenario[12]. Consider e.g. 4.3, a sequence of events achieves corresponding states such that each event allows the body movement. These events do not happen at the same time, but some happen before and some later. Assume that the body is very heavy and, even though various preparations took place, even pushing on it does not budge, but, when John start helping to push the body, it finally starts to move. Then, 'John started pushing the body' is the effective sufficient cause for the body movement.

We do not explore this further here; related discussions are available in [190, p. 11 and onwards]. In this chapter, we do not explore this complex issue further, limiting ourselves to notice that it is often the case that, to cause an occurrent, multiple conditions must 'accumulate' together, as it is the case in the examples of Section 4.1.2, where often multiple occurrences concur to the causation of the same effect.

[11] Riichiro Mizoguchi, personal communication, March 2023.

[12] See also https://arxiv.org/abs/2307.07517.

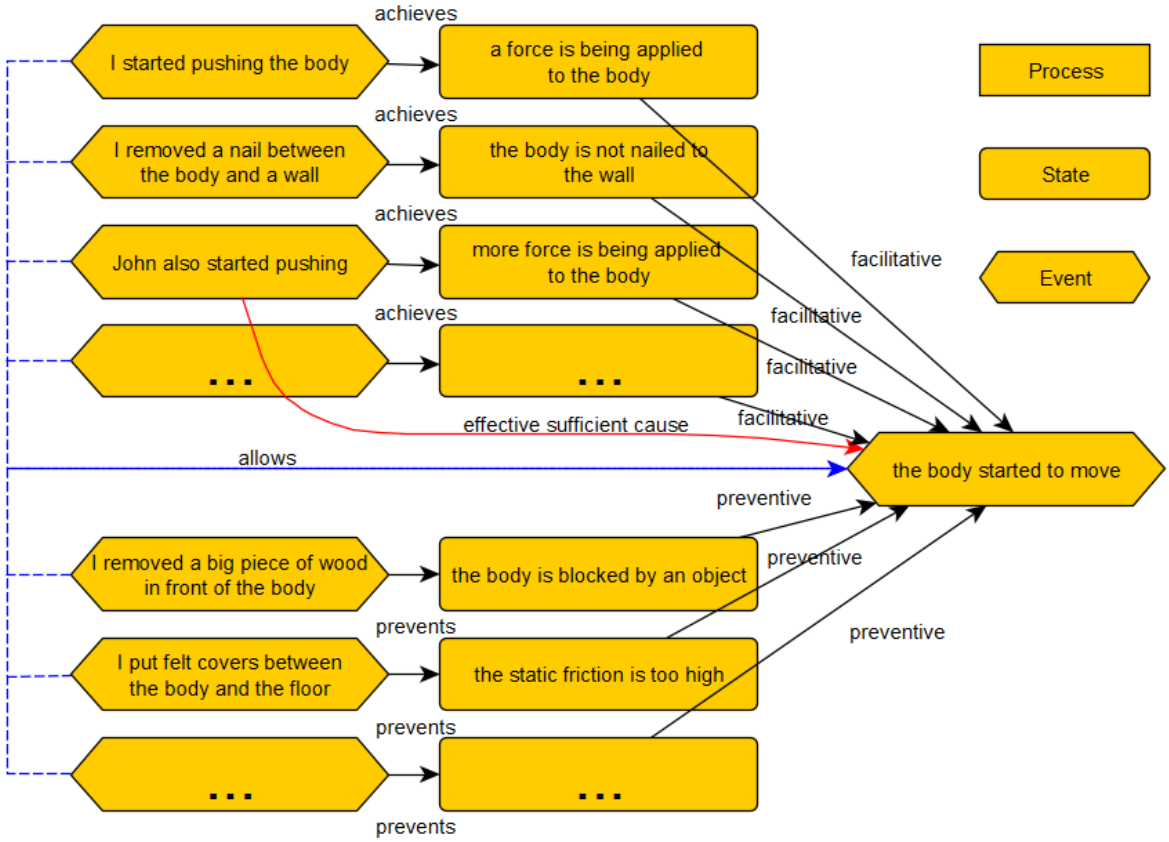



Figure 4.3: Exemplification of the sufficient cause mechanism.

The examples of Section 4.1.2 also show a possible way to determine (backwards-view) root causes heuristically: see Example 4.1.2.2. We remark that such heuristics may very well be useful in applications, but are not intended to constitute an ontological analysis of the root cause concept. In particular, root cause is not assumed to be an ontological category, instead is taken to be a label attributed to an occurrence depending on the perspectives of the person carrying out the causal analysis.

## 4.1 Evaluation of the MALFO ontology

In this section, we follow a series of standard steps frequently used in the literature to demonstrate the quality of an ontology

Our first claim is that MALFO's taxonomy effectively captures key aspects of the language used in failure analysis. Specifically, that it is possible to classify malfunction-related terms from the literature within our taxonomy and align them with various existing failure-mode taxonomies. Table 4.1 demonstrated this mapping[13] for a selection of relevant terms extracted from the literature. This is esplored further in the following section, MALFO is aligned with other ontologies from the literature. Both these efforts serve to showcase how MALFO can help address the challenge of semantic heterogeneity and to demonstrate its high-level of terminological and conceptual coverage.

We now address the most fundamental requirement from a formal logic perspective: ensuring the consistency of the ontology. This has been verified for both the first-order theory, using an automated theorem prover, and for the OWL approximation, using an OWL reasoner. The first-order consistency proof is available in our GitHub[14] repository, while OWL consistency can be independently verified by running any standard OWL reasoner on MALFO-OWL within Protegé. In fact, within ontology engineering, one often aims for a stronger notion of consistency: that the ontology remains consistent even when all predicates are instantiated simultaneously. In the OWL setting, this can be demonstrated by populating the ABox with appropriate individuals and checking consistency with a reasoner. To support this, we have included in the GitHub repository a set of sample models represented in a readable diagrammatic format, along with a Python script that converts these into OWL. We argue that the same level of consistency holds for the first-order version of the ontology, as it is evident that the diagrammatic models also satisfy the axioms of the first-order theory.[15]

From a logical standpoint, it is noteworthy that MALFO constitutes a definitional extension of Toyoshima's causal ontology (as expanded in the previous section), with

[13]The classification shown in the table was performed manually by 'reverse engineering' the meanings of terms from their usage in the texts. Inevitably, this process involves a degree of subjectivity, as the meanings of terms are often under-specified. Nonetheless it is a good example of how the mapping process can be carried out.

[14]`https://github.com/kataph/MALFO`, also note that therein a version of MALFO with an alignment to the work of Chapter 3 is present: `MALFO-plusDOLCE-functions.owl`.

[15]One could attempt to confirm this by using first-order provers to test the consistency of the base theory augmented with existential axioms that guarantee non-emptiness of MALFO classes. However, such attempts typically fail due to timeouts. This was confirmed using the SystemOnTPTP platform (`https://tptp.org/cgi-bin/SystemOnTPTP`) with several provers, including Vampire.

the addition of the `functionIncompatible` predicate.[16] In other words, MALFO introduces only defined terms that build upon pre-existing symbols, as outlined in definitions (df51)-(df59). Constructing an ontology in this way (by extending a well-founded base through definitions) is a methodologically robust approach, as it ensures that the extended ontology preserves both the ontological soundness and consistency of the original framework.

Another evaluation step involves using the OOPS! tool[17] [208], which analyzes an OWL ontology and produces a checklist highlighting common pitfalls in ontology development. When applied to MALFO-OWL, the tool identified only three non-trivial issues: (i) The object properties `functionIncompatible` and $\texttt{FunctionOf}_{\texttt{Sys}}$ lack explicit domain and/or range axioms. This omission is intentional, as these properties relate to functions, which fall outside the intended scope of MALFO. They are included solely to facilitate future mappings with function-oriented ontologies. (ii) Certain SWRL[18] classes are flagged as RDFS classes rather than OWL classes. However, this is an expected characteristic of the SWRL language and does not indicate an error. (iii) The class `State` is flagged due to recursive definitions. In MALFO-OWL, we specify that states can only achieve or prevent other states. This means that if $x$ is a state and $x$ achieves $y$, then $y$ is also a state, a valid and harmless inference. Since no further significant issues were detected by OOPS!, we consider this evaluation successfully passed.

OntoClean [98] is a widely recognized methodology for evaluating the ontological soundness of taxonomies that we have already used in the previous chapter. In the case of MALFO, the class `Occurrent` and its subclasses (`Event`, `Process`, and `State`) are categorized as OntoClean "categories", characterized by the metaproperties -O, -I, +R (i.e., they do not supply or inherit identity criteria, but are rigid).[19] All other classes, such as `Malfunction`, are considered "formal roles", bearing the metaproperties -O, -I, ∼R, +D (i.e., they do not supply or inherit identity criteria, are anti-rigid, and are generically dependent on external entities).[20] MALFO thus contains three types of subsumption relationships: category-to-category, role-to-category, and role-to-role. All of these are permitted within the OntoClean framework, indicating that MALFO respects OntoClean's constraints.

Another common aspect of ontology evaluation involves the use of competency questions. These are typically formulated in natural language and are intended to be translated into formal SPARQL queries using the vocabulary of the OWL ontology, as we did in Chapter 3. The underlying idea is that an ontology demonstrates its value by being expressive enough to support the translation of questions that are particularly

[16] This predicate is intended to facilitate potential alignments between MALFO and ontologies that address functional concepts. In such cases, MALFO can be seen as a definitional extension of Toyoshima's ontology together with the selected function ontology.

[17] `https://oops.linkeddata.es/`.

[18] SWRL: `https://www.w3.org/submissions/SWRL/`, is a rule language for OWL. We used it to encode the sufficient condition in the definition of `Fault` (df55), which cannot be fully expressed in standard OWL.

[19] -O -I +R.

[20] -O -I ∼R +D.

relevant to its intended users. Originally, however, competency questions were meant to be translated into reasoning tasks that rely on first-order inference over the ontology's axioms [88]. In contrast, basic SPARQL does not utilize OWL axioms in its reasoning process.[21] Even within this more reasoning-oriented view, competency questions come in a range of forms and complexities [136]. In the context of MALFO, we believe the most meaningful approach is to employ reasoning to fully determine a partially specified interpretation of the ontology. Specifically, consider the following reasoning task:

> **MALFO Interpretation Expansion Problem:** Given a (finite) model $\mathfrak{M}$ of Toyoshima's ontology in which each occurrence is labeled as either function-incompatible or not, determine the unique[22] MALFO model that extends $\mathfrak{M}$ by assigning interpretations to all predicates in MALFO's signature.

An example input to this problem is the directed graph shown in Figure 4.7 (excluding the information within angle brackets). The output consists of determining which nodes correspond to which MALFO classes, if any.

This reasoning task corresponds to a broad set of competency questions (e.g., "What are the failure mechanisms in this scenario?", "Is X a fault or a failure?", "Does X cause a failure directly or indirectly?"). All of these questions can, in principle, be answered once the reasoning task is resolved. While MALFO does, in theory, solve this problem, since the expanded model exists and is unique, this does not necessarily imply the existence of an effective or automated procedure for computing the expansion in practice.

Before addressing the practical solvability of the expansion problem, we first consider a more foundational question: Does solving this problem provide evidence of MALFO's quality? We argue that it does, to the extent that the classification of occurrents under MALFO's taxonomy is itself of interest. Supporting the relevance of such classification is the broader claim, presented in the introduction, that a clear, precise, and ontologically well-founded vocabulary enhances semantic interoperability in the context of failure analysis. Moreover, the interpretation expansion problem is not specific to the current state of MALFO; it generalizes to any definitional extension of the ontology. Thus, its significance could increase as more conceptually rich notions are integrated into MALFO. For instance, at the end of Section 4.1.1, we introduce definitions for Kitamura and Mizoguchi's notions of ultimate causes and absolute causes [143]. These and similar future extensions would enrich MALFO in ways that are intrinsically valuable, particularly due to their relevance for root cause analysis.

Turning to the issue of actually solving the expansion problem: we have successfully translated the first-order theory into OWL, including equivalence axioms for all MALFO classes,[23] and this translation allows the use of OWL reasoners for classification. One might therefore expect OWL reasoners to effectively solve the expansion problem by classifying individuals into the appropriate ontology classes. However, a

[21] Although it is possible to manually craft SPARQL queries that emulate some inference steps, this approach can become unwieldy and may not be ideal.

[22] Uniqueness is guaranteed since MALFO is a definitional extension of Toyoshima's ontology.

[23] With the exception of `Fault`, which requires SWRL rules to represent its sufficient conditions.

limitation arises due to the necessity of using negation in the definitions of five specific classes[24] during the translation. Due to OWL's open-world assumption, even after populating the ABox with a complete model, a reasoner will always assume the possibility of missing information and hence cannot confirm that certain conditions do not hold. This prevents accurate classification into those five classes. To address this issue, we developed a custom script that implements a reasoning mechanism based on negation-as-failure semantics (available in the repository). This allows for correct and unrestricted classification of instances under all MALFO classes. Additionally, we translated MALFO into DLV System's disjunctive datalog language, which also supports negation-as-failure and benefits from DLV's efficient reasoning algorithms. This makes the expansion problem effectively solvable in that setting as well. As for first-order logic, although the expansion problem is evidently decidable, general-purpose automated model provers may not yield practical results due to the semi-decidability of first-order logic. While a custom first-order prover could in principle be constructed to handle this task efficiently, we consider the OWL-based solution, combined with the auxiliary tools described above, satisfactory for our current goals.

Finally, a thorough evaluation of MALFO would involve assessing its impact on engineering practices and knowledge sharing in failure analysis. This, however, is left for future work, as the current paper focuses on presenting MALFO as a formal theory.

### 4.1.1 Aligning concepts in the literature with our ontology

In this section, we discuss how our ontology can be linked to several malfunction-related concepts present in the literature. Except for the material in [143], we describe such links only informally since the concepts in the literature are given only informally.

**Del Frate's definitions** Our definition of failure, as presented in (df57) fits well with the function-based definition of Del Frate. The main difference between ours and Del Frate's definition is that in the former we speak of bringing about the inability to execute a required function, while the latter mentions the termination of the ability to execute a required function. It could be argued that such phrases are equivalent (one is a sort of double negation of the other). In any case, MALFO does not contain an 'ability to execute a required function'-concept since it focuses on abnormal conditions only. Instead, such a concept seems to be naturally interpreted as a capability. Following this lead, together with the discussion in Section 3.0.6 and our definition of slected functions, a more faithful representation of Del Frate's failures is as follows (where `Life` is functional symbol indicating the time extension of a perdurant for introduced for the sake of simplicity):

**mp7** $*Failure(x) \implies \texttt{Event}(x) \wedge \exists y, f, q, t_a, t_b(\texttt{PC}(y, x, \texttt{Life}(x)) \wedge \texttt{hasFunction}(y, f) \wedge \texttt{defFoundedOn}(q, f) \wedge \texttt{PRE}(q, t_a) \wedge \neg\texttt{PRE}(q, t_b) \wedge t_a < \texttt{Life}(x) < t_b$

[24]Namely: mere symptoms, non-performance events, failure mechanisms, failure conditions, and down-states.

> "If $x$ is a Del Frate-failure, then $x$ is an event and there exist a participant (for the whole of $x$ extension) that has a selected capability (in the sense of Definition (df27)) that is lost during the time extension of $x$."

Note that this definition is only partial (only a necessary condition): a full definition would need to specify the particular role of the failed object and to specify the causal link between the failure event $x$ and the loss of the selected capability $q$. However, this is a very complex ontological endeavor. For this reason, we opted for a 'capability-free' definition in MALFO mediated by the incompatible-with predicate between perdurants.

Coming to specification-based failures, they suggest that an engineered device is tied to a designer's plan for its intended use, detailing how, under what conditions, and in what environment the device should operate. This definition aims to shift culpability for a failure from the equipment manufacturer to the user if the user operates the equipment outside its specified conditions. Since our ontology does not address culpability or specifications/plans, this concept of failure differs from (df57) and cannot be formulated within our framework.

Material-based failures can be divided into two scenarios based on their definition: (i) if the change in material properties makes the item unable to perform, it falls under a function-based failure. In our ontology, this corresponds to specific subtypes of the `Failure` class, such as 'rupture', or 'deformation[-event]'. (ii) When the change in material properties increases the likelihood of a failure, we cannot accurately represent this scenario, as we do not consider stochastic relationships, only causation between individual occurrences. Instead, we can represent cases where a change in material properties directly causes a failure.

**Jansen's definitions** Jensen's discussed varied expressions related to malfunctions [133, 218]. In particular, he distinguished among, at least, 'having a malfunction', 'being malfunctioning', and 'having no function'. He apparently did not determine what a malfunction is, however.

Jansen's definition of being-malfunctioning ("a material object x is *malfunctioning* at t with respect to the function to F-in-asituation-S, if and only if, at t, x is in situation S but does not F") looks very similar to stating participation in a MALFO non performance event. In fact, naively, one would translate it to participation in a MALFO-malfunction, however, MALFO-malfunctions would include MALFO-failure events, such as the fracture of an axis. Now, suppose that such an event happened while the axis was not functioning (say a forklift hit it). Then Jansen's definition does not apply because the axis is not in the situation to execute the function. Restricting the participated occurrence type to non performance events should ameliorate such this issue.

Jansen's definition of having-a-malfunction ("a material object x has a *malfunction* at t with respect to the function to F-insituation-S, if and only if x has the function to F-in-situation-S but would not F in S, because, at t, it does not have the disposition to F-in-situation-S") looks similar to participation in a MALFO-fault. Alternatively, we could use the capability framework of Chapter 3 to replace the disposition and paraphrase to "a material object x has a malfunction at t with respect to the function

to F if and only if (i) F is a function attributed to x (either one of its systemic or selected functions), (ii) the corresponding capability of x is, at time t, either completely lost or with a quale located in a low/no performance region". For example, part of the above prosaic definition can be translated into symbols into the following partial formal definition:

**ex34*** $^*hasMalfunction(x, f, t) \Longleftarrow \texttt{hasFunction}(x, f) \wedge \exists q \texttt{qt}(q, x) \wedge \texttt{defFoundedOn}(q, f) \wedge \neg \texttt{PRE}(q, t)$

> "The object $x$ has a malfunction at time $t$ with respect to the function $f$ if $f$ is a function of $x$ and there is a capability $q$ of $x$ founded on $f$ that is not present at $t$."

Note that our two alternative definitions are very similar, since the degradation or loss of the capability will likely have some cause internal to the object, and, therefore, this cause will be a likely candidate for the $z$ in Definition (df55), entailing that the condition of degradation or loss of the capability is a MALFO-fault.

Jensen insists also on differentiating the previous expressions from the condition of not having a function. In our framework, not having a function correspond to not having attributed systemic or selected functions, which is clearly different from the previous cases.

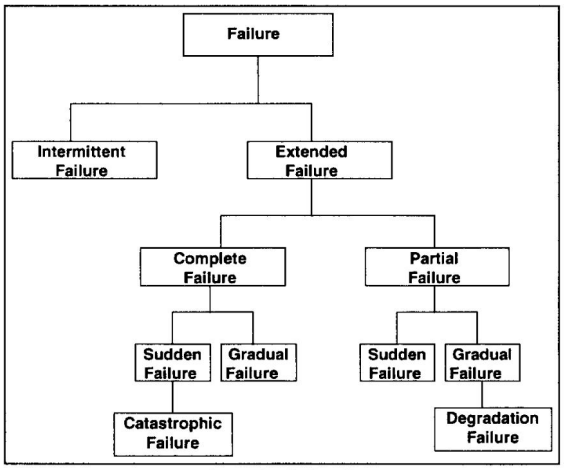
Finally, since having a malfunction and being malfunctioning have been mapped to participation to either non performance events or faults, it is natural to map the concept of Jansen-malfunction with the union of `Fault` and `NonPerformanceEvent`.

**Blache's failure taxonomy** The taxonomy of (specification-based[25]) failures built by Blache [25] utilizes the attributes of Table 2.1 (see 4.4a). In Blache's taxonomy, failures are initially divided into two main categories: extended failures and intermittent failures. Extended failures are then further classified based on their severity and suddenness.

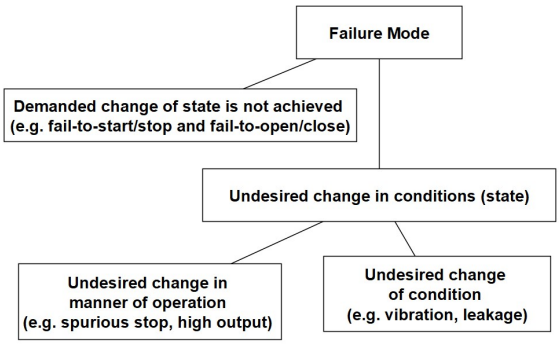
Any ontology that includes the first three attributes from Table 2.1 and the specification based failure concept could define Blache's taxonomy classes. However, since these failures are specification-based and cannot be associated with our concept of failure, as discussed in the previous section, our ontology does not qualify. In passing, we note that persistence and suddenness are not exclusive to the failure category; these attributes can also be attributed to other malfunction-related happenings (e.g., 'intermittent noise') and even to more general occurrences.

**OREDA failure mode taxonomy** The OREDA project handbook [225] describes another subdivision of failure modes. We pictorially represent this taxonomy in Figure 4.4b. Failure modes are described as *observed effects of failures*, indicating that 'ob-

[25]Blache's definition is "An event when machinery/equipment is not available to produce parts at specified conditions when scheduled or is not capable of producing parts or perform scheduled operations to specification. For every failure, an action is required" – the emphasis on the production of parts is due to the focus on manufacturing equipment.




(a) Blache's failure taxonomy [25].




(b) Taxonomy of failure modes, described in [225, p. 41]. The source remarks that, about the bottom-right category, "[t]his category does not affect the function immediately, but may do so if not attended to within a reasonable time".

Figure 4.4

servability' is a key aspect of the top node of the taxonomy. Since our ontology does not consider observability, we cannot align with such a definition.

Considering the taxonomy's leaf nodes, and especially their examples, the category 'demanded change of state is not achieved' can arguably be mapped as a subclass of `NonPerformanceEvent` ('fail to start' is an example of an occurrence-type common to both classes). For the category 'undesired change in manner of operation', which is said to be a state but exemplified with occurrences looking like events (e.g. 'spurious stop'), we propose a division for mapping purposes. This category could be split into 'alteration in the manner of operation' (e.g., 'high output'), considered a subclass of malfunction states (i.e., either faults or abnormal external disabled states), and 'interruption of operation' (e.g., 'spurious stop'), which could be mapped to `NonPerformanceEvent`.

Finally, the category 'undesired change of condition' should be considered a subclass of occurrences that are states and function-compatible (i.e., not function-incompatible). While our ontology cannot express the *potential* impact of such occurrences on the underlying function if not attended to, an *actual* instantiation of this scenario could be represented in a model of the theory. Additionally, we could state that addressing the undesired condition `prevents` a failure from occurring.

**Avizienis' fault-error-failure chain** As previously mentioned, Avizienis [10] states that faults cause failures via errors. This may appear incompatible with our definitions, which follow Del Frate, and Rausand and Øien in stating the opposite: that failures cause faults. However, this is not a contradiction. In our ontology, it is still possible for a fault to cause a failure. Similarly, Avizienis acknowledges that failures may cause

faults [10]. Therefore, while Avizienis' characterizations of failures and faults differ from ours, they can be further specialized to achieve compatibility.

Another distinction between Avizienis' conceptualization and ours is his specification that faults can be either dormant or active: an active fault produces an error, while a dormant fault does not. In our ontology, this can be approximated as follows: an active fault is the part of a `Fault` that occurs during (or immediately before) the causation of an error, while a dormant fault is the remaining part. Another interpretation, based on an ontology that admits dispositions, could view a fault as a disposition, active when realized and dormant when not. This latter interpretation aligns with [119] and with [67], with the difference that, in that work, 'malfunctions' are dispositions that result in 'failure events' when active.

Another critical difference is the presence of errors. We did not define this concept in our ontology, partially because Avizienis' definition and examples of errors, as well as his statement that "error propagation within a given component [...] is caused by the computation process" [10], suggest that errors are specific to the software domain. However, we assert that any Avizienis-style causal chain can be reformulated into a MALFO-ontology causal chain: errors can be mapped to some kind of malfunction, depending on the occurrence type (event, state, or process) the error is recognized as. Example 4.1.2.1 demonstrates this mapping.

For Avizienis, errors propagate within a component, resulting in other errors, and only when an error reaches the boundary of a system (e.g., when a second system requests a service from the first), a failure occurs. Thus, Avizienis' distinction between failures and errors also hinges on what we termed observability in Section 4.0.1: the difference between errors and failures is that failures are observed by an external system. Our theory does not consider observability, therefore it cannot explain this latter difference.

**IOF modular maintenance ontology** The IOF's maintenance ontology (IOF-maint) [120] provides another relevant source of malfunction-related concepts. Since this ontology is based on the Basic Formal Ontology (BFO), a comprehensive mapping between MALFO and BFO would be required for proper integration. However, such an extensive mapping is outside the scope of this work. Instead, we offer a preliminary, partial analysis here. Some of the most relevant terms in IOF-maint are 'failure event', 'failure process', 'failure effect', and 'failed state', with formal definitions available in the IOF's documentation[26]. These definitions are complex, and we reproduce them verbatim (some predicates are starred to avoid homonymy with our predicates). Then we try emphasize key aspects (the interested reader can consult IOF documentation for an in-depth treatment), this is essential as the IOF-maint is still in provisional status, meaning that while details may change, the key aspects are likely to remain consistent.

**df62*** $\texttt{FailureEvent}(e) \iff {}^*\texttt{Event}(e) \wedge \exists o(\texttt{FailedState}(o) \wedge \texttt{initiates}(e, o)) \wedge$
$\exists i, f, p1(\texttt{MaintainableMaterialItem}(i) \wedge \texttt{hasParticipantAtAllTimes}(e, i) \wedge$

[26] Accessible at `https://github.com/iofoundry/ontology/blob/master/maintenance/Maintenance.rdf`, we specifically refer to commit 3467e52.

$\texttt{PrimaryFunction}(f) \wedge \texttt{hasFunction}(i, f) \wedge \texttt{FunctioningProcess}(p1) \wedge$
$\texttt{realizes}(p1, f) \wedge \texttt{precedes}(p1, e) \wedge \neg\exists p2(\texttt{FunctioningProcess}(p2) \wedge$
$\texttt{realizes}(p2, f) \wedge \texttt{precedes}(p1, p2) \wedge \texttt{precedes}(p2, e)))$

**df63*** $\texttt{FailedState}(o1) \Longleftrightarrow \texttt{MaintenanceState}(o1) \wedge$
$\exists i(\texttt{MaintainableMaterialItem}(i) \wedge \texttt{hasParticipantAtAllTimes}(o1, i) \wedge$
$\exists e(\texttt{FailureEvent}(e) \wedge \texttt{initiates}(e, o1)) \wedge \exists o2((\texttt{DegradedState}(o2) \vee$
$\texttt{OperatingState}(o2) \wedge \texttt{hasParticipantAtAllTimes}(o2, i)) \wedge \texttt{precedes}(o2, o1)) \wedge$
$\neg\exists o3((\texttt{DegradedState}(o3) \vee \texttt{OperatingState}(o3)) \wedge$
$\texttt{hasParticipantAtAllTimes}(o3, i) \wedge \texttt{precedes}(o2, o3) \wedge \texttt{precedes}(o3, o1)))$[27]

**df64*** $\texttt{FailureEffect}(x) \Longrightarrow$
$^*\texttt{Process}(x) \wedge \exists f((\texttt{FailureEvent}(f) \vee \texttt{FailureProcess}(f)) \wedge \texttt{precededBy}(x, f))$

**df65*** $\texttt{FailureProcess}(x) \Longleftrightarrow$
$^*\texttt{Process}(x) \wedge \exists y(\texttt{DispositionToFail}(y) \wedge \texttt{realizes}(x, y))$

First, note that the starred $^*\texttt{Event}$ and $^*\texttt{Process}$ belong to BFO, and as such do not need to be the same as MALFO's. Indeed, since `FailedState` and all other IOF-maint's states are subclasses of $^*\texttt{Process}$, MALFO's `Process` and `State` must be different from the homonymous BFO concepts. Analysis of BFO's predicates definitions suggests that the following mapping is the most appropriate:

**mp8** $(\texttt{State}(x) \vee \texttt{Process}(x) \vee \texttt{Event}(x)) \Longrightarrow (^*\texttt{Process}(x) \wedge {}^*\texttt{Event}(x))$

This is because BFO-events that are also BFO-processes are those BFO-processes that are relevant for some agent, and the latter class appears to be the smallest BFO class (defined with the predicates from IOF-maint) that contains MALFO's occurrents.

More importantly, the IOF-maint has just one relation that suggests a causal link: `initiates`. Defined as the "comes before an event or process in time and results in beginning or creation of the event or process", this looks like direct positive causation from Toyoshima's ontology, but `initiates` definition is too general to make precise mapping statements. Third, the IOF-maint strategy to define failure events and failed states is to assert that the failed component (a "maintainable material item") functioned one last time before failing. This entails that failure events and failed states in IOF-maint are significantly different from MALFO-failures and MALFO-faults, despite their terminological similarities. In MALFO, malfunctions (including faults and failures) are explicitly stated to be incompatible with normal functionality, not merely that they are preceded by a single instance of functioning. In IOF-maint's failed states are intended to indicate that the underlying item cannot perform its function (with the heuristic definition stating "state of an item being unable to perform a required function due to a failure event"), but is not explicitly made clear in the formal definition. Thus, a failed state (or failure event) in IOF-maint is not necessarily equivalent to a MALFO-fault (or MALFO-failure). Conversely, the inclusion in the opposite direction also does not hold, as IOF-maint requires that failed items must have functioned at least once, while MALFO does not impose such a requirement. The misalignment between MALFO and

[27] Note the circularity between this definition and (df62). We assume that at least one of the is hence intended to be an axiom and not proper definition.

IOF-maint regarding failed events implies that no direct alignment can be established with failure effects either.

Fourth, IOF-maint's use of dispositions, derived from BFO, to describe failure processes as realizations of dispositions is a concept that MALFO cannot express. Dispositions could fit nicely with the first meaning of failure mode we mentioned at the end of our taxonomy, as they could provide a link between system components and types of malfunction-related happenings. This is indeed done in IOF-maint, where the definition of the term 'failure mode codes' entails that failure modes are 'undesirable dispositions'. However, it is important to note that 'failure mode' is not a formal term within the ontology itself.

**Kitamura and Mizoguchi ontological analysis of faults** Kitamura and Mizoguchi's work contains a nuanced classification of faults and their causes, some of which can be mapped to MALFO: in Kitamura and Mizoguchi's classification, causes are either internal or external to a given system. This distinction can be mapped to MALFO using the `internalTo` predicate. Specifically, an internal cause in Kitamura and Mizoguchi's terminology would be a cause in MALFO that is `internalTo` a given system. Second, Kitamura and Mizoguchi's "abnormal states" (which are abnormal "only because of the propagation of abnormal input") and "faults" (irreversible changes in the internal state of a component) mirror MALFO's abnormal external disabled states and faults, respectively. Third, Kitamura and Mizoguchi divide causes into "relative", "absolute", and "ultimate" causes: a fault may start a chain of "propagation events", each causing an abnormal state, which is considered as just superficial symptom of the fault. In this context, the first fault in the chain is the absolute cause of the last abnormal state, and the second-before-last abnormal state is the relative cause of the last abnormal state. The fault itself may have been caused by and preceded by multiple made from various faults and abnormal states. In such a case, the initial cause originating outside the analyzed system is considered the ultimate cause of all subsequent occurrences.

Given these descriptions, we deduce the following definitions of Kitamura and Mizoguchi's concepts mentioned above (labeled with a star $*$) in terms of those of our ontology[28]:

**df66*** $^*\texttt{AbnormalState}(x) \iff \texttt{AbExtDisabledSt}(x)$

> "Kitamura and Mizoguchi's abnormal states are equivalent to our abnormal external disabled states."

**df67*** $^*\texttt{Fault}(x) \iff \texttt{Fault}(x)$

> "Kitamura and Mizoguchi's faults are equivalent to our faults."

**df68*** $^*\texttt{relativeCause}(x, y) \iff \texttt{posCausalCon}(x, y)$

> "Kitamura and Mizoguchi's relative causation is equivalent to the positive causal contribution relation."

[28]Note that (df69*) and (df70*) are second-order logic formulas.

**df69*** $^*\texttt{absoluteCause}(x, y) \iff$
$(\exists z(\texttt{Fault}(z) \wedge \texttt{posCausalCon}(x, z)) \wedge$
$\exists x_1, x_2, \ldots, x_n(x_1 = z \wedge x_n = y \wedge$
$\forall j = 2, 3 \ldots, n-1\ (\texttt{posCausalCon}(x_{j-1}, x_j) \wedge \neg\texttt{Fault}(x_j))))$

"$x$ is a Kitamura and Mizoguchi's absolute cause of $y$ if and only if $x$ positively causally contributes to some fault $z$ and there is a (positive) causal chain of occurrences $x_1, x_2, \ldots, x_n$, none of which is a fault except possibly the first and the last, which starts with $z$ and ends with $y$."

**df70*** $^*\texttt{ultimateCause}(x, y, s) \iff (\texttt{externalTo}(x, s) \wedge \exists z(\texttt{Fault}(z) \wedge$
$\texttt{posCausalCon}(x, z) \wedge \exists x_1, x_2, \ldots, x_n(x_1 = z \wedge x_n = y \wedge \forall j = 2, 3 \ldots, n$
$(\texttt{posCausalCon}(x_{j-1}, x_j) \wedge \texttt{internalTo}(x_j, s)))))$

"$x$ is a Kitamura and Mizoguchi's ultimate cause of $y$ with respect to the system $s$ if and only if $x$ is external to $s$ and positively causally contributes to some fault $z$ and there is a (positive) causal chain of occurrences $x_1, x_2, \ldots, x_n$, all internal to $s$ except for the first, which starts with $z$ and ends with $y$."

## 4.1.2 Examples of application

Here we describe possible applications of MALFO regarding causal analysis of malfunction-related happenings.

### 4.1.2.1 Errors and failures in an integrated circuit

We reformulate this example, taken from Avizienis [10], using our ontology to demonstrate both a mapping with an alternative conceptualization of malfunction-related happenings and to explore the connection between causation and the temporal duration of occurrences.

Avizienis' example consists of a short circuit occurring in an integrated circuit. The short circuit is an Avizienis-failure[29] that causes a connection of the circuit being stuck at a boolean value, the latter being an Avizienis-fault. The Avizienis-fault remains dormant until the circuit is invoked, at which point it becomes active and generates an error. This error then propagates, causing additional errors. If these errors adversely affect the service provided by the circuit, a subsequent failure occurs.

This example is illustrated in Figure 4.5: the upper chain, with ellipses as nodes, represents the fault-error-failure conceptualization according to Avizienis. The lower chain depicts the corresponding model using Toyoshima's causal ontology. Dashed lines connect corresponding occurrences between the two chains, while dotted lines at the bottom link each occurrence in the lower chain to its time extension, represented by a gray bar along the time axis. Specifically, let's assume the short circuit involves a particular failure event: the breakdown of some insulation (1). This breakdown achieves the condition of being stuck at a certain boolean value (3), and the time extensions of (1) and (3) are juxtaposed, enabling the tight causal interaction required by `achieves`.

[29] Avizienis' concept of failures may be different from MALFO's `Failure`.

Occurrence (3) is also a facilitative precondition for the erroneous calculations (5), but this alone is not sufficient to cause (5). There is a time interval during which (3) persists without (5) commencing, allowing us to say that the facilitative precondition is 'dormant' during this period. To trigger (5), several other occurrences must transpire (e.g., the IC must be powered, its data connections must be at least partially linked to adjacent components, etc.). For simplicity, let's assume the only trigger required to start the calculation process is the circuit's invocation, such as by sending the correct signal to a connection (2). At that moment, the calculation process begins, characterized by a complex sequence of states (identified by various voltage levels at different circuit points) that are tightly coupled both temporally and causally. We isolate two of these states (4, 6), which are part of the entire process and occur at different times. These states are not the only ones in the process; there is also a sequence of additional states between (4) and (6). The final state in this chain involves an incorrect value at the circuit's output (7), which acts as a facilitative precondition for the erroneous service delivery (8). Occurrence (7) is only a precondition for (8), as multiple conditions (not discussed in the example) must accumulate for (8) to occur.

MALFO axioms entail that (1) is a `Failure`, (8) is a `NonPerformanceEvent`, (3) is a `Fault` (assuming (1) is internal to (3)). Additionally, (4), (6), and (7) are abnormal external disabled states, because they are not achieved by some internal occurrence; and (5) is a `Malfunction` `Process`. This classification may be done manually or it can be also automated as it will be discussed in Section 4.1.2.3. To begin the classification process, it is necessary to specify the causal arrows, the basic ontological categories of the occurrences (`Event`, `Process`, `State`), and their function-compatibility. Some causal relationships can be inferred; for instance, the `allows` relation between (1) and (5) in Figure 4.5 can be deduced from the arrows connecting (1) to (3) and (3) to (5).

#### 4.1.2.2 Explaining a root cause analysis of a material failure

With this example, we aim to show how our ontology-based approach can be exploited to enhance the precision of root cause analysis. Consider Matthew's case study on the failure of the main drive shaft of a radial fan [180]: the fan's shaft broke, leading to the catastrophic failure of the entire machine. Matthews' analysis identifies multiple contributing factors, such as the impeller's low balance grade, high vibration levels, bearing wear, and specific design details of the shaft that increased stress concentration near the keyway. Importantly, Matthews distinguishes the varying levels of contribution from these causes. He identifies the primary cause, that he terms "proximate [in effect]", as the shaft design, while categorizing the other factors, like the impeller's balance, vibrations, and bearing wear, as "contributory factors", entailing that they played a lesser role.

Matthew summarizes his analysis in the pictures of Figure 4.6, and the complete analysis can be found in [180]. Here, we will present some re-interpretations of his analysis through the lenses of our approach. It's important to note that, despite our best efforts, our interpretation may not perfectly align with Matthews' original analysis. Nonetheless, our objective is to demonstrate the application of our ontology in a

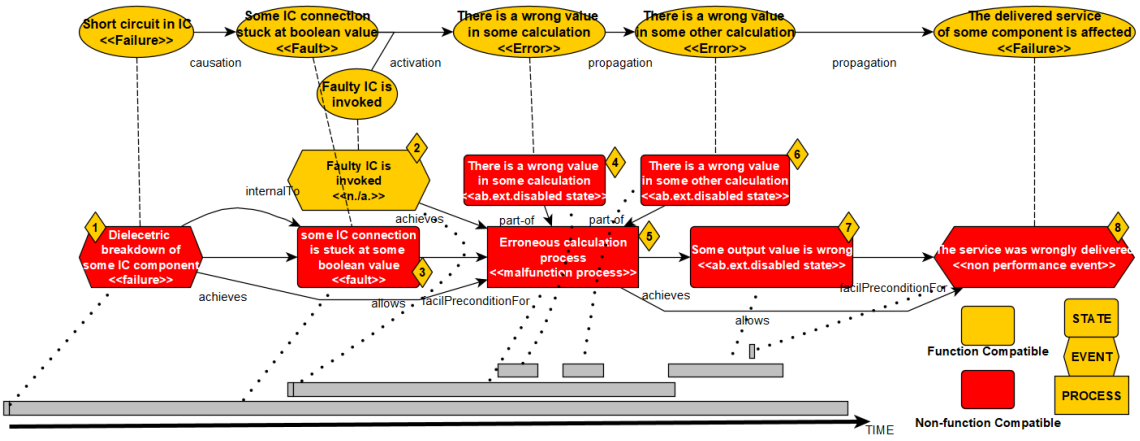



Figure 4.5: Avizienis' example of a failure in an integrated circuit. The top row is Avizienis' original conceptualization, the bottom row is a revised conceptualization using Toyoshima's ontology. Dashed lines between the two conceptualizations link corresponding occurrences.Red nodes are function-incompatible occurrences. Since MALFO is a definitional extensions of (an adequate extension of) Toyoshima's ontology, the bottom row is also a model of MALFO and the MALFO's class of an occurrence is put between double angle quotation marks ("«" and "»").

practical use case, rather than to precisely replicate Matthews' analysis.

Our interpretations are ontology models, and, since they contain only asymmetric binary relations they can be depicted as labelled directed graphs. The first interpretation is shown in Figure 4.7. Individual occurrences are represented as nodes, and their labels provide descriptions serving as unique identifiers. Each event's MALFO class is denoted between double-angle quotation marks[30]. The nodes shape indicates if the occurrence is an event, process, or state. The arrows colors help to (i) Disambiguate how a (dis-)allows arrow can be deduced (for example, `achieves` + `facilitative`/ `preventive` entails `allows`/ `disallows`), as seen between nodes (16), (17), and (18). (ii) Show that multiple arrows contribute to the causation of the same occurrence. For instance, in nodes (21), (22), and (24), `achieves` causation is supported by an `allows` causation of the same color. This is a typical scenario in Toyoshima's ontology, where `achieves` causation is assisted by `allows` causation pointing to the same occurrence.

[30]Note that some classifications depend on the implicit context. For instance, the shaft design (1) could be viewed as a failure concerning the function of the design team or process. Consequently, the rough surface finish (4) would be a fault regarding the shaft *as a design object*, of, say, preventing crack formations). Also 17 is not a malfunction-related happening.

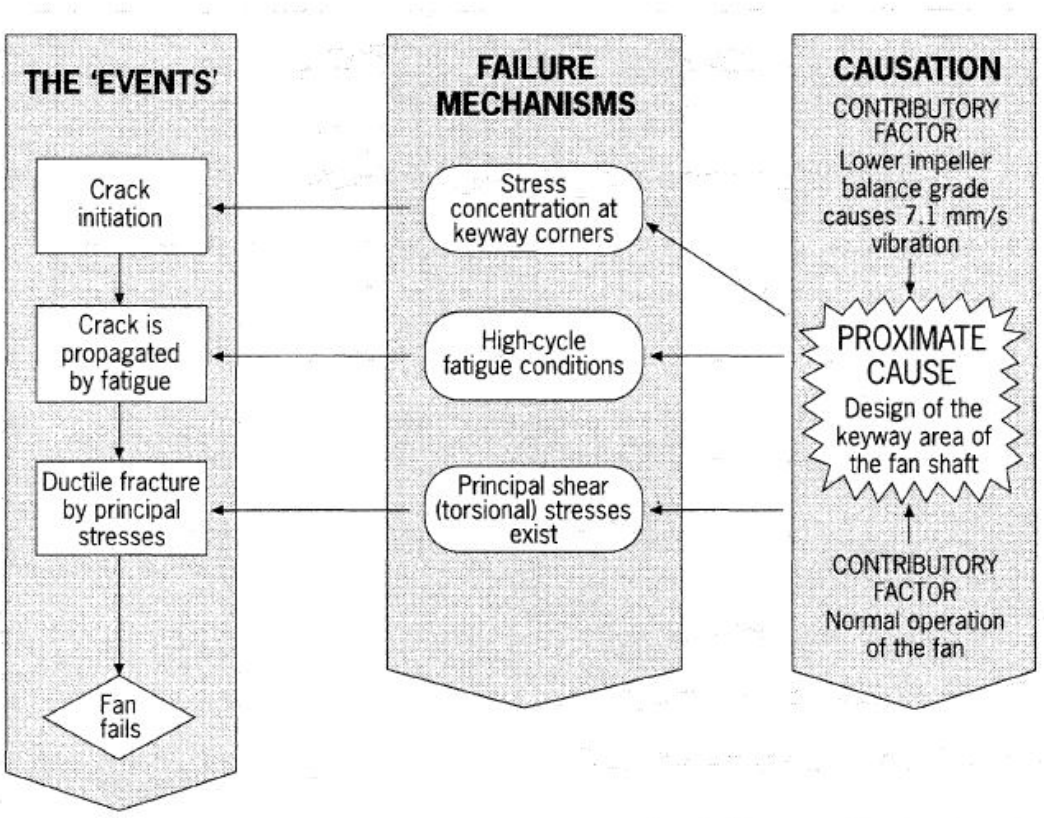



Figure 4.6: Example of material failure root cause analysis from [180]. Notice that here the fracture is called "ductile", but on page 193 is called "brittle". We use the latter term in Figures 4.7, 4.8.

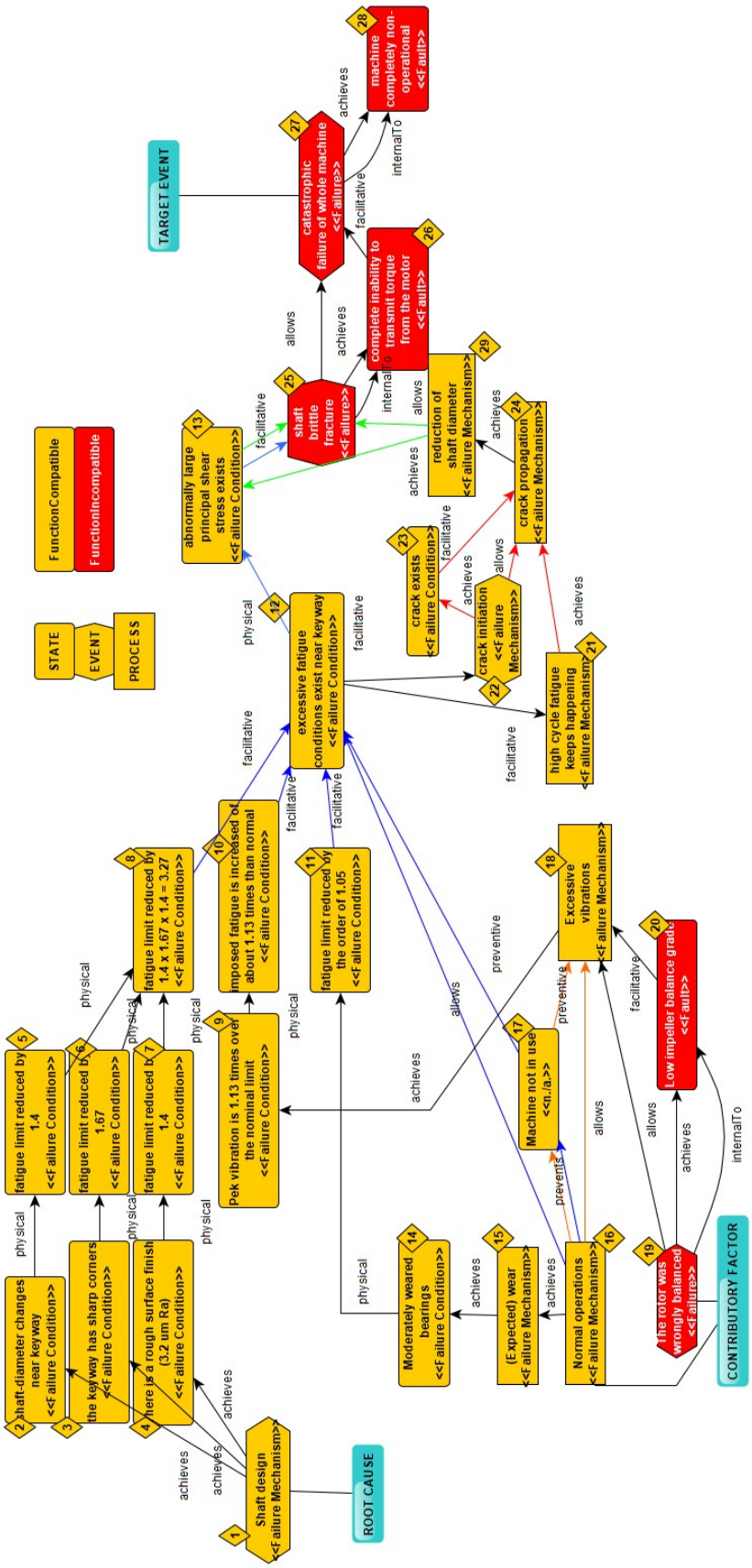



Figure 4.7: Causal graph for the fracture of the main fan drive shaft described in [180]. The graph is a model of Toyoshima's ontology enriched with the information if an occurrence is function-compatible or not (yellow ans red nodes respectively). The MALFO class of an occurrence node is between angular brackets. Blue rectangles serve only to help with the discussion in the text. The numbers in the small diamonds are labels that are used to reference the nodes in the main text. Some arrows are colored with different colors to highlight their role in reasoning. For instance, the green 'allows' arrow from 16 to 18 can be deduced using the two green arrows from 16 to 17, and from 17 to 18. In addition, when multiple arrows of the same color point to the same node this is to highlight that they concur in the causation of said node.

The graph can be summarized as follows: the target event being analyzed is the catastrophic failure of the whole machine (27), caused by the shaft rupture (25). This rupture is a result of the excessive fatigue conditions near the keyway (12) and the reduction of the shaft diameter (29) due to the propagation of a previously initiated crack (22). Fatigue conditions also lead to both the initiation and propagation of the crack. The fatigue conditions are determined by multiple occurrences, divided into three subgraphs: The first subgraph (1-8) is initiated by the shaft design (1). The second subgraph (10-9-18-20-19) begins with the erroneous balancing of the rotor (19). The third subgraph (11-14-15-16-17) is initiated by the normal operations of the machine (16).

**Analyzing a RCA using a MALFO model.** Examining the graph reveals nodes 25 and 12 as crucial because their removal would result in disconnected (non-singleton) parts of the graph. There exists a single causal chain mediated by states that links Node 25 to the target event (27). Therefore, it can be argued that the root cause problem for the target event can be traced back to the same problem for Node 25. Undoubtedly, it was the shaft rupture that directly caused the catastrophic failure.

The rupture is caused by the fatigue conditions, as depicted in the subgraph between nodes 12 and 25. Essentially, abnormal stresses lead to the formation of a crack in the shaft, which subsequently propagates due to high-cycle fatigue, ultimately resulting in the brittle fracture of the remaining portion of the shaft. Note that the fatigue conditions contribute to both the initiation and propagation of the crack, as well as to the final rupture itself.

The key issue now shifts to identifying the cause of Node 12. Bearing wear, vibrations, and the shaft design collectively contribute to the emergence of this node through the three subgraphs mentioned earlier. Specifically, they lead to three distinct yet similar conditions (8, 10, 11) that share comparable quantitative parameters. Upon comparing these parameters, it becomes evident that the segment of the graph initiated by the shaft design plays the most significant role in causing the critical Node 12.

In the graph, there are exactly three nodes positioned leftmost from the others: the shaft design (1), normal operations (16), and the erroneous balancing (19). We these nodes as potential candidates for identifying the root cause. Based on the earlier discussion, we designate the shaft design as the 'root cause', and nodes 16 and 19 as 'contributory factors'. In particular, normal operations, although not inherently abnormal, are still considered a contributing factor. In another context, normal operations could even be the main responsible for the shaft rupture: this happens, say, in the case that the catastrophic failure is due to expected wear due to aging.

The result of this analysis is an agreement with Matthews' as it is indeed based on his analysis, but notice that, while in our case the critical point of the analysis is clear: determining (through the comparison of some relevant parameters) the most important causal chain that brought about Node 12; in Matthews' case the critical point is the same but is only implicitly stated. In fact, this very important point is reduced to a few sentences in Matthews' analysis. In detail, he states:

In our analysis, we align with Matthews' conclusions. This is unsurprising, since

we are rephrasing his work. However, there is a noticeable difference: the critical point in the RCA analysis is identifying the primary causal chain leading to Node 12 by comparing relevant parameters. We explicitly highlight this point, while Matthews only implicitly touches on it in a few sentences:

- "If there had not been a stress concentration (caused by the design) to start with, then the fatigue alone would not have resulted in this particular failure." [180, p. 188]
- "The relatively small deviations from impeller balance grade and maximum vibration limits tell us that the imposed fatigue condition was mild, and therefore acted as a contributory factor only." [180, p. 189]
- "The combination of sharp corners and rough surface finish of the key slot - and its proximity to the small radius change of shaft section - produced high stress concentrations. These were sufficient to result in crack initiation under fatigue conditions." [180, p. 189]

There are some concerns with the first and third sentences: the first would still apply if stress concentration and fatigue were swapped (fatigue is essential since stress concentration alone wouldn't cause failure). The third sentence describes stress concentration as sufficient, but the exact meaning of sufficient is not clarified. It cannot simply mean physical sufficiency because external factors could prevent fracture even with stress concentration present. For instance, the fan operated for a significant period with stress concentration but without failure, and under different conditions or with a different shaft material, the same stress concentration might not lead to failure at all. The second sentence implies a comparison of relevant parameters, a point that we emphasized in the causal graph.

**RCA based on graph-topology.** Next, we present an alternative approach to assessing the relative contributions of causes to a failure. We reinterpret the graph shown in Figure 4.7 as Figure 4.8 with a different causal structure: here, we posit that high-cycle fatigue (21 – from now on all nodes refer to Figure 4.8 unless otherwise specified) primarily propagates the crack, while the initiation of the crack (22) is attributed to excessive stresses near the keyway (3). The preference for either the causal structure of 4.7 or that of 4.8 or another will depend on the specific material theory adopted regarding fractures. Additionally, we partition the incorrect design of the shaft into an event and a state, which we assume are functionally incompatible with the intended function of the design itself.

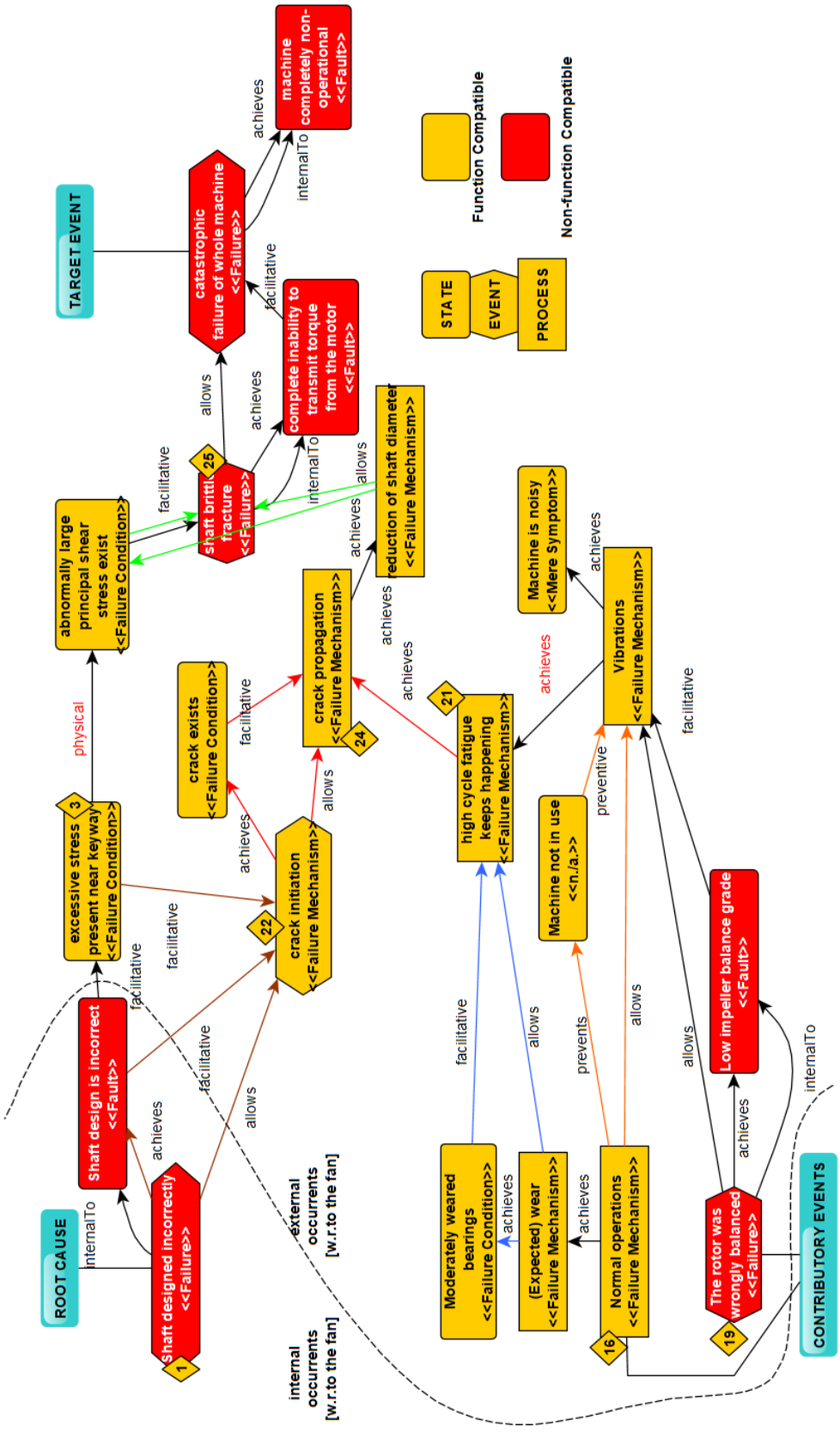



Figure 4.8: Revised causation structure. A dashed line distinguishes occurrences internal to the fan system from those that are external. The two red arrow labels, which are discussed in the main text, denote potential errors and their respective corrections facilitated by the reasoning process. The numbers in the small diamonds are labels that are used to reference the nodes when discussing the picture in the main text. Some arrows are colored in the same way as in Figure 4.7.

In this revised structure, nodes 21 and 24, when removed, partition the graph into two disconnected parts. The first part, initiated by nodes 16 and 19 contributes solely to the propagation of the crack without initiating it. Conversely, the part initiated by Node 1 contributes in various ways to both the initiation and propagation of the crack. This critical distinction, coupled with the observation that the incorrect balancing of the rotor (19) holds a comparable causal role in the graph to normal operations (16), which are not abnormal, suggests once more that nodes 16 and 19 play a minor role in causing the target event compared to the shaft design (1). Once again, we align with Matthew's analysis, this time based on the topology of the graph rather than a quantitative comparison.

**RCA based on formal ontology concepts.** This time we make use of the concepts of `absoluteCause` and `relativeCause` to help the analysis.

For example, in Figure 4.8, Node (19) is the only absolute cause of (25); while Node (1), assuming that all nodes to the left/right of the dashed line are external/internal to the fan system[31]), is the only ultimate cause of (25).

Then, to find the root cause of an event, one could just focus on the absolute and ultimate causes of that event, filtering out any different causal role. In particular, searching for `ultimateCause`s is a formalization of the process of seeking root causes across different organizational contexts, as seen in models like HFACS. In the context of Figure 4.8, the ultimate cause of (25), situated within the context of machine operations, is node (1), situated in the context of the design process.

A more targeted approach may focus solely on specific categories of malfunction-related occurrences when searching for root cause(s) (e.g., `Failure`, `Fault`, `Failure Condition`, and `FailureMechanism`), while disregarding others (`MereSymptom`, `Malfunction Process`, `AbExtDisabledSt`, `NonPerformanceEvent`).

#### 4.1.2.3 Using automated reasoning and queries

Inferencing may be used to enhance the use of the ontology, as it will show in the following.

The inferencing process must begin with some established facts. In the context of intended instantiations of our ontology, as shown in Figures 4.5, 4.7, and 4.8, these facts include:

- the classification of each occurrent as either `State`, `Event`, or `Process`,
- each occurrent is classified as either `functionIncompatible` (i.e. a malfunction) or not,
- all the primitive ontology relations (`achieves`, `facilitative`, `preventive`, `has PhysicalConseq`, and `internalTo`)[32] need to be asserted. Non-primitive causal

[31] Then nodes (1) and "Shaft design is incorrect" must be thought as referring to e.g. the shaft design as a design-object which is not internal to any given physical fan system.

[32] In Toyoshima's causal ontology there exist additional primitive relations. For simplicity, we exclude those.

relations (`allows`, `disallows`, `prevents`) may also be asserted in addition or as a replacement for the assertion of a set of primitives.

These facts need to be provided a priori, and we do not investigate, in this chapter, the methods for their collection. Nonetheless, these facts are not mere arbitrary assertions: for example, in Figure 4.8 Node (19) is `internalTo` Node (20), therefore every object participating in (19) is part of some object participating in (20). If we assume that the objects participating in these occurrents are exactly those mentioned in the occurrent descriptions (i.e. 'rotor' in (19) and 'impeller' in (20)), then we the rotor is part of the impeller. This is true if, for instance, rotor and impeller indicate the same entity ('impeller' being the specific type of 'rotor' used in the fan), since part-of is reflexive (anything is part-of itself). But it is false if, for instance, the rotor is an assembly containing all rotating components of the fan, which are not limited to the impeller. Similar observations apply to all other types of initial facts. The key point is that, for each asserted fact, the ontology provides guidelines to constrain its meaning, even if these guidelines are not implemented through a formal/automatic process.

Another requirement for a set of asserted facts is consistency, which can be formally verified using the ontology axioms. After verifying consistency, one can use the ontology axioms to infer (i) the presence of non-primitive causal relations, (ii) the classification of an occurrent into one of the classes of MALFO's taxonomy, and (iii) the presence of finer causal relations such as absolute and ultimate causes. While this consistency checking and inferencing process can be performed manually, it is impractical and unscalable. Fortunately, numerous methods exist to automate the inferencing process:

- First-order theorem provers, of which many are available online, such as Vampire[33] or Prover9[34] and many others. Most of these theorem provers are also available as a service from the TPTP library[35]). These provers reason using first-order logic and as such their typical application is restricted to academic problems due to complexity and non-decidability issues (i.e. the reasoning process may not return a solution once started). Nonetheless, we have developed serializations of MALFO in prover9 and TPTP formats[36] that can be utilized with these tools.

- Logic programming. A portion of MALFO's first-order logic axioms can be straightforwardly implemented as rules using logic programming. Models, such as Figure 4.7, can be stored in any database system that supports inferencing in a logic-programming style, such as DLV[37]. We provide an intensional database written as a disjunctive Datalog program, roughly equivalent to first-order MALFO.

- OWL reasoners. MALFO has also a version as an OWL-ontology. Models can

[33] `https://vprover.github.io/`.

[34] `https://www.cs.unm.edu/~mccune/prover9/`.

[35] `https://www.tptp.org/cgi-bin/SystemOnTPTP`.

[36] These, as any oncoming serialization is available in the GitHub repository `https://github.com/kataph/MALFO`, also note that therein a version of MALFO with an alignment to the work of Chapter 3 is present: `MALFO-plusDOLCE-functions.owl`.

[37] `https://www.dlvsystem.it/dlvsite/dlv-user-manual/#`.

be stored as knowledge graphs (RDF(S) graphs) using the ontology vocabulary, enabling application of OWL-reasoners.

- Ad-hoc reasoning on knowledge graphs. We provide a library of SPARQL rules that implement the described reasoning tasks. This extends MALFO capabilities to RDF triplestores that may have limited or no built-in reasoning support.

MALFO is fundamentally a first-order theory (with the addition of transitivity closure of predicates), meaning that any translation into decidable languages will only partially preserve MALFO's axioms. Additionally, OWL operates under the open world assumption, making it challenging, if not impossible, to assert negative facts about entities.

For example, the primary distinction between first-order MALFO and its OWL version is that OWL lacks the capability to classify an instance as a predicate defined through negation. In MALFO, these correspond to `MereSymptom`, `NonPerformanceEvent`, and `AbExtDisabledSt`; for instance, a functionally compatible occurrence is considered a mere symptom if it does *not* cause any failure. Therefore, instances belonging to these classes cannot be recognized by an OWL reasoner.

Using a functional-programming-style language one could infer such predicates, assuming that the language used allows for negated statements. In functional programming there are different types of negation, the one that we intend to be used for MALFO reasoning task is negation-as-failure: a fact is assumed false if it cannot be deduced from the available facts. However, using negation-as-failure can lead to incorrect inferences if a negated fact is actually true but has been omitted from the initial set of facts for any reason. The ad-hoc inference rules we encoded as queries also implement negation-as-failure, using the "FILTER NOT EXISTS" SPARQL construct. These queries materialize the rule heads into the graph whenever the pattern specified in the rule-body appears. It's crucial to execute these rules in a specific order since some predicates are dependent on others, a script that takes care of this is also provided on GitHub.

Lastly, note that absolute causes and ultimate causes cannot be defined in OWL. However, they can be identified programmatically. For instance, a SPARQL query (supplied on GitHub) can select all `absoluteCause`s. Similarly, the sufficient conditions for `FailureCondition` and for `Fault` cannot be expressed in OWL and have been implemented as SWRL rules instead.

To illustrate the inference process described previously and in the preceding sections, we provide the GraphML files that were used to generate the images 4.5, 4.7, and 4.8. Additionally, a script is supplied that converts these GraphML files into RDF knowledge graphs. Any method described above can then be used to test the inference process.

OWL reasoning, despite its limitations, offers benefits such as consistency checking and providing explanations. For example, consider Figure 4.8. If, erroneously, instead of a `hasPhysicalConseq` arrow from Node (3) an `achieves` arrow was inserted, then the reasoner would reveal the error: given the 'facilitative' arrow into Node (25), one would erroneously conclude that (3) `allows` (25). However, (25) being a `State`, is outside the domain of `allows`, resulting in a contradiction. This inconsistency can be highlighted and explained through OWL reasoning, as illustrated in Figure 4.9.

Similarly, a `facilitative` arrow instead of an `achieves` pointing to (21) would also lead to inconsistency, explained by the fact that a process is outside the domain of `facilitative`.

Additionally, OWL reasoning can explain correct inferences. For instance, an explanation of why Node "Vibrations" is a `FailureMechanism` is found in 4.10.

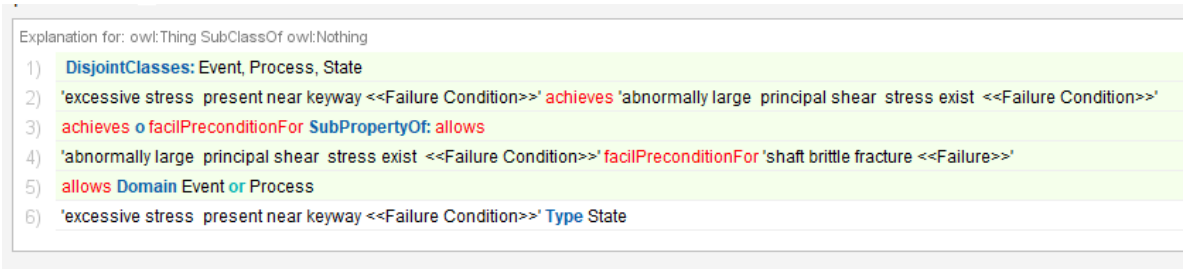



Figure 4.9: Explanation of why an achieves departing from Node (3) in Figure 4.8 leads to an inconsistency, as identified by the Hermit reasoner (version 1.4.3.456).

Summarizing, the proposed methodology for the use of MALFO consists of the following steps:

1. gather an initial set of facts and represent them as a model of the ontology using the ontology's language (refer to Section 4.1.2.3).

2. Utilize reasoning techniques to verify adherence to the ontology and derive further facts (refer to Section 4.1.2.3).

3. Leverage the structure of the model and the ontological knowledge to elucidate the fundamental cause of a specific target event, akin to what was done in Example 4.1.2.2.

4. Record the outcomes of the analysis in a knowledge graph.

Note that we do not provide specific instructions on identifying the root cause, as we believe that no universally applicable instructions exists. The methodology's strength lies in a formal language for causation, where terms have precise and constrained meanings, rather than using purely descriptive textual documents. After all, "[p]eople solve failures; the methodology or process is just a tool to find the root causes"[214, p. 21].

Additionally, note that certain issues present in existing methodologies, as discussed in Section 2.2.2, such as lack of terminological and conceptual clarity, systematic approach, comprehensive treatment of causation, and efficient storage of knowledge in readily accessible databases, are addressed by our methodology.

### 4.1.3 Main limitations

MALFO's language enables the description of causation patterns in a flexible yet structured manner. For example, a common causation pattern observed in failure scenarios is causation by the accumulation of factors. In MALFO, this can be depicted by multiple

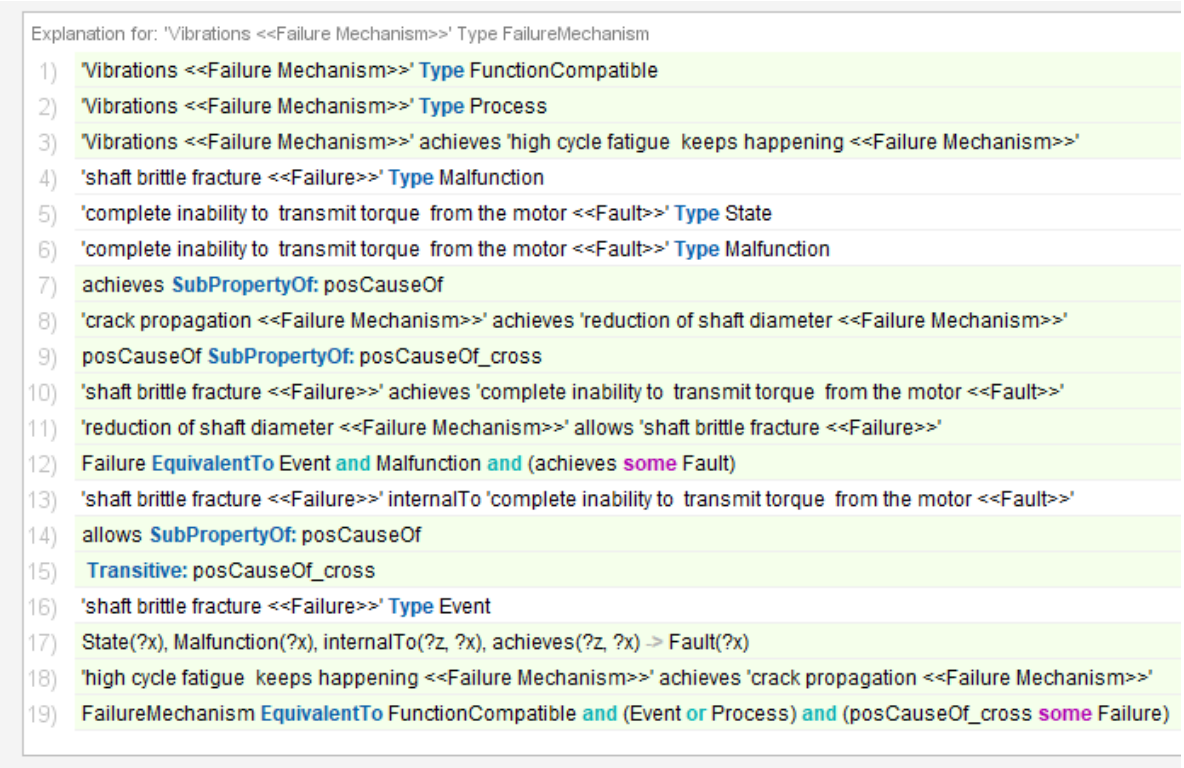



Figure 4.10: Explanation of why "Vibrations" in Figure 4.8 is classified as a failure mechanism, as determined by the Hermit reasoner (version 1.4.3.456). Note the SWRL rule in step 17.

`allows` or `facilitative` arrows pointing to the same occurrence. This happens e.g. in Figure 4.7 for nodes (12) and (24). Upon closer examination, this fact reveals some of MALFO's limitations, since MALFO cannot assert 'collective causation'. For instance, We would like to express collective causation of (8) by occurrences (5), (6), and (7), possibly differentiating the degree of causal contribution (e.g., (5)'s contribution to (8) is 1.4 and (6)'s is 1.67, etc.). Semantic Web Stack technologies restrict prevent the achievement of these goals, as they only support classes and binary relations, not the n-ary statements that would be needed. Reification could solve this and indeed it is a possibility to consider. Similarly, expressing '(21) achieves (24) due to (22) facilitating (24)' requires statements about statements, which cannot be done for the same reasons.

Another limitation is the impossibility of making general assertions like 'smoking causes lung cancer overall', since the causation is always at token-level. Similarly, expressing probabilistic statements, such as 'smoking increases the probability of lung cancer by 30 times', is problematic due to the need for both type-level causation and quantitative weights for causal relationships.

### 4.1.4 Causal relations between functions

It is finally time to discuss relations of causal nature between functions, completing the discussion about started in Section 3.0.8. First of all, it is clear that we can lift all causal relations from occurrences to functions, similarly to what has been done for other relations holding between perdurants. More precisely, we can establish new relations at the concept level that reflect the corresponding relations between behaviors associated with these functions. It remains to consider when this lifting process is of engineering interest. Causal relations of interest between functions already exist in the literature, Kitamura and Mizoguchi metafunctions [142] are probably the most prominent example. In particulr, they introduce 8 types of metafunctions, defined in base of the type of causal contribution and the types of flows involved in the two functions. For instance, a function (f) has the 'toDrive' metafunction w.r.to another function (f') if it is a mandatory prerequisite for that function, it has a proportional effect on f' performance, and it provides an energy flow that f' consumes. After careful consideration, we can state the following:

**mp9** $(\texttt{provides}(\texttt{f},\texttt{f'}) \vee \texttt{drives}(\texttt{f},\texttt{f'}) \vee \texttt{enables}(\texttt{f},\texttt{f'}) \iff \forall b'(\texttt{CL}(b',\texttt{f'}) \implies \exists b(\texttt{CL}(b,\texttt{f}) \wedge \texttt{posCausalCon}(b,b'))))$ where $\texttt{f},\texttt{f'}$ are functions.

> "f provides for or drives or enables f' if and only if for all behaviors $b'$ classified by f' there is a behavior $b$ classified by f such that $b$ positively causally contribute to $b'$."

**mp10** $(\texttt{controls}(\texttt{f},\texttt{f'}) \vee \texttt{improves}(\texttt{f},\texttt{f'}) \vee \texttt{enhances}(\texttt{f},\texttt{f'}) \iff \forall b(\texttt{CL}(b,\texttt{f}) \implies \exists b'(\texttt{CL}(b',\texttt{f'}) \wedge \texttt{posCausalCon}(b,b'))))$ where $\texttt{f},\texttt{f'}$ are functions.

> "f controls or improves or enhances f' if and only if for all behaviors $b$ classified by f there is a behavior $b'$ classified by f' such that $b$ positively causally contribute to $b'$."

The distinction between the disjuncted relations in the defined has to be carried out by specializing the positive causal contribution relation, and by explaining the relation between the functions flows. Consider e.g. the toDrive metafunction example above, or the toControl metafunction, which holds if the first function realization maintain the realizations of second function to stay within given performance parameters (the first function 'regularizes' the second). Since some of these specialization processes are not currently expressible in our framework, we do not refine alignments (mp9)-(mp10) further.

There are two final metafunctions: toPrevent and toAllow[38]. They consist of preventing a malfunction of or directly positively causing a negative side effect of the target function. We define only the former, since our framework does not (immediately) have the 'side effect' concept required for the latter:

**mp11** $\texttt{prevents}(\texttt{f},\texttt{f'}) \iff \forall b(\texttt{CL}(b,\texttt{f}) \implies \exists b', m(\texttt{Malfunction}(m) \wedge \texttt{functionIncompatible}(m,\texttt{f'}) \wedge \texttt{prevents}(b,m) \wedge \texttt{CL}(b',\texttt{f'}) \wedge \texttt{allows}(b',m)))$ where $\texttt{f},\texttt{f'}$ are functions.

[38]Note the homonymy with the corresponding negative causal relations.

> "`f` prevents `f'` if and only if for all behaviors $b$ classified by `f` there exist a behavior $b'$ classified by `f'` and a malfunction $m$ such that $m$ is functionally incompatible with `f'`, prevented by $b$, and allowed by $b'$."

Another relevant author is Burek [39], who defines two relations relevant for this section: triggers and improves. Burek-improves is different from the metafunction toImprove: the latter increases the performance, the former reduces side effects. For the same reasons that we skipped alignment with toAllow above, we skip the alignment with Burek-improves. Regarding the triggers relation, it is described as providing "the necessary and sufficient condition for the function realization. In this sense a trigger can be understood as a direct cause of the function realization", therefore it is a subcase of (mp9), where the positive causation is either an `achieves` or an effective sufficient cause.

# Chapter 5

# Some Applications

After the two previous chapters concerned with formal matters about developing an overarching framework for engineering systems functions and malfunctions, in this chapter we expand our results into methodologies aimed to industrial applications. The applicative turn we take takes advantage of the theory developed earlier and showcasing its potentiality through two semantic-technology-based tools.

## 5.1 Domain Terms Glossary

Domain-specific glossaries[1] can be employed to increase data quality, reduce semantical heterogeneity and ease communication, facilitate translation efforts, and standardize enterprise documentation, both for internal and customer use. The development of such glossaries is not trivial, as, in addition to the standard difficulties of building a vocabulary, specific technical expertise is required.

For manufacturing companies, technical terms vocabularies rotate around products their components and related processes, including operational, manufacturing, and maintenance processes (and also anomalous conditions). Adige S.p.A. falls in this case, and extrapolation of technical terms from technical manuals indeed revealed a prominence of nouns related to product and component, especially from the electromechanical domain.

In this section we discuss a possible way to use ontologies to help to build a high-quality glossary in the specific case of Adige, but the outlined procedure is relevant more in general, e.g. in the manufacturing domain.

**Brief review of industrial vocabularies** Vocabularies related to domains of interest for some industrial settings (industrial vocabularies for short) present the same

[1] As a terminological note, we use 'vocabulary' in two senses: either as some part of the natural language specific to some domain, or as a document describing such a part of the language. In the latter sense vocabulary is synonymous with dictionary. We call a dictionary 'glossary' to emphasize either that it is limited to a specific domain, or that it contains a list of terms with short explicatory definitions and/or comments, i.e. their glosses. Finally, we do not distinguish between sense, meaning, and concept.

variety that can be found in general settings: they may be constituted by simple alphabetically ordered lists of terms, such as Oracle's glossary of networking terms[2] and IEEE's glossary for software engineering terms[6]; or they may extend to comprehensive dictionaries, such as the Oxford Dictionary of Electronics and Electrical Engineering[7].

Glossaries of non-small size will often contain homonymous or polysemous[8] terms. This may be managed by either allowing multiple entries for the same term (one entry per meaning) and identifying each entry with a unique key (such is the case for SAP terms and IEV), or by using one entry per term and listing the multiple meanings in the gloss of the term (as in IEEE's glossary). Bigger glossaries will also be divided into subsections grouping terms by (sub-)domain.

Note that important semantic relations, such as hyponymy/hypernymy and meronymy/holonymy are not usually made explicit, but are only implicit in the definitions (e.g., 'an actuator is a special case of a transducer' [41]).

**Development of an ontology-based vocabulary** For Adige, early approaches to the development of a glossary revealed some critical issues. Among those, and in addition to the classical issues of dictionaries such as managing synonyms etc., the main ones are:

- determine how to represent the relationship between components and user interface software elements, as well as that between machine components and machine cycles.
- Manage terms with ambiguous meanings (e.g., 'CAD' can refer to different software products depending on the context).
- Manage hierarchies among terms (e.g. a 'valve' can be a 'pressure regulator valve', a 'check valve' etc.).

We divide these issues into two categories: (G1) determine and manage the relevant semantic relations between terms, and (G2) determine and manage semantic ambiguity between terms.

As a solution fit to solve the above-mentioned issues, after an analysis of the technical language employed by the company, a prototype hand-curated glossary of several

[2] `https://docs.oracle.com/cd/E53394_01/pdf/E54755.pdf` or the vocabulary of ISO 14224[3]; multilingual thesaurus containing acronyms, abbreviations, synonyms, antonyms, deprecated or obsolete terms, associated terms, and explicatory notes, in addition to the glosses, such as the IEV[4], SAP terminological database[5], last accessed September 2024.

[6] Institute of Electrical and Electronics Engineers, `https://ieeexplore.ieee.org/document/159342`, last accessed September 2024.

[7] `https://www.oxfordreference.com/display/10.1093/acref/9780198725725.001.0001/acref-9780198725725`, last accessed September 2024.

[8] Two terms are homonymous if they have different grammatical and/or semantic roots but the same lexical expression (e.g., 'I *set* up the tent' and 'the *set* of all prime numbers' are homonymous). A term is polysemous if it is linked to multiple meanings stemmed from the same semantic root (e.g., 'argument' may a variable in an arithmetic equation, or it may be an input variable for a function in a programming language).

hundred terms was developed. The glossary is ontology-based in the sense that it is a sort of WordNet of terms, containing various semantical and lexical relations, and the semantic relations are taken from the ontology developed in the previous chapters.

Now, to solve issue (G1), an analysis of the relevant semantic relations between domain terms revealed, as is to be expected, that subsumption and parthood are the pivotal semantic relations. Participation-like relations are also common to relate components to processes. Moreover, function attribution ('X does Y', 'the function of X is Y', etc.), together with 'failure mode attribution' ('the breaking of X', 'problems in Y'), is a semantic relation that is especially important in this domain, much more than for the non-technical language. In summary, we claim that the use of the semantic relations taken from a top-level ontology (DOLCE in our case) augmented by those relations studied in this thesis, are sufficient to answer (G1).

Regarding (G2), it is a multi-faceted issue. On one hand, there is the issue of recording and describing the multiple senses of polysemous words, and collecting the multiple synonymous words that can be used to indicate a sense. On the other hand, there is the issue of retrieving the correct sense of a term given a context. All of these various facets can be treated by dictionaries. In particular, semantic dictionaries[9], such as WordNet, can detail the meaning of a term by relating it to other terms through the appropriate semantic relations. Sense disambiguation of a term can then happen by analysis of the semantic relations of that term in a local context. Of course, this procedure can be either manual or automatic. Automated determination of the correct meaning of a polysemous word given some of its context is called Word Sense Disambiguation (WSD), a classical task in natural language processing. Various classes of approaches to WSD exist, but, for the limited scope of this section, we just mention that some of these approaches are knowledge-based: those that are based on a knowledge base, such as a glossary [198]. Additionally, though those methods have been outperformed by LLM-based methods, they may still be relevant for the cheap data and energy consumption and for the good performance on tasks over specific domains with little data available [57]. Moreover, even current SotA high-performance LLM-based algorithms make use of semantic dictionaries (e.g. WordNet) to feed high quality data to ML-pipelines [269]. In summary, any semantic dictionary, in particular, any dictionary constituted by the lexical content of an ontology can be an answer to (G2), even to the point of driving automated NLP applications (the development of such applications is, however, outside the scope of this thesis).

In addition to these semantic issues, technical language is also peculiar at the lexical level. For instance, abbreviations and acronyms are quite common (e.g., CAD[10], QBH[11], ...), as the use of numeric or alphanumeric codes to identify specific components or processes.

[9]By semantic dictionary, or semantic lexicon, we mean a dictionary that makes heavy use of semantic relations between terms, while a thesaurus uses at most hyponymy, synonymy, and general association. In this sense, the lexical content of an ontology is a semantic dictionary.

[10]Computer Assisted Design.

[11]Quick Buttons and Holder. This term refers to a device used specifically to connect optical fibers to optical systems, however.

For all these reasons, the glossary was created as the lexical content of a systematically annotated ontology. The ontology is constituted by (a part of) DOLCE and (a part of) the ontological analysis developed in the previous chapters. In particular, the preferred terms for the relevant top-level classes of the ontology are:

- device: called engineering artefact in [32], the definition of this concept is outside the scope of this thesis (a marginal discussion is present in Section 3.0.7), and an in-depth discussion can be found in e.g. [32]. In the glossary the preferred word is 'device' due to such a term being preferred in use in technical language. The gloss of the word is also simplified to 'an artifact performing one or more functions'.
- Function: this term is used as an umbrella for the various concepts of engineering function analyzed in this thesis.
- Operand: this term is used to indicate function operands, synonym to flow from the systematic design literature.
- Engineering method: the way-of-achievement of functions relevant for functional decomposition.

While the main top-level relations between these concepts are:

- subsumption: to build taxonomical tree.
- parthood: to build partonomies. The archetypical use is for the bill of materials (BoMs) of products. Note that on the class level the parthood relation is lifted from the instance level using existential quantification (always and only from the whole to the part): $X$ has $Y$ as part means:

  **ex35** $X \sqsubseteq \exists \texttt{has-part}.Y$
  "Each $X$ has as part some $Y$."

- functional ascription: functional ascription was discusses in the previous chapter and is intended to be either `hasFunction` or `systemicFunctionOf` (for the sake of simplicity, they are collapsed to the same relation. Formally we are talking about the union of these two relations). It links a device to (one of) its function(s). Again, on a class level the relation is lifted from the instance level using existential quantification: $X$ has function $Y$ becomes

  **ex36** $X \sqsubseteq \exists \texttt{has-function}.Y$
  "Each $X$ has as function some $Y$."

- method-function relation: an individual method solves an individual function, and an individual function is solved by an individual method. At a class level a method X solves only a class Y of functions:

  **ex37** $X \sqsubseteq \forall \texttt{solves}.Y$
  "Each $X$ solves some $Y$."

- operand-function relation: an individual function can have as input or as output an individual operand. At a class level these relations are lifted using universal quantification:

  **ex38** $X \sqsubseteq \forall \texttt{has-input}.Y$

  "If an $X$ has an input the input is a $Y$."

  **ex39** $X \sqsubseteq \forall \texttt{has-output}.Z$

  "If an $X$ has an output the output is a $Z$."

Note that taxonomical trees of devices were built making use of the other semantic relations, for instance, devices are first classified by type of ascribed function, then by type of implemented engineering method, then by other characteristics (e.g., shape). For instance, torque-transmission devices are devices whose function is to transmit torque, gears are those torque-transmission devices that use coupling of toothed components to execute their function, cogwheels are wheel-shaped gears, and sprockets are cogwheels whose teeth are designed to bite a chain or other transmission member coupled to a pinion. Thus, the following subsumption chain is produced: device $>$ transmission device $>$ gear $>$ cogwheel $>$ sprocket.

Note that the description of a device in functional terms takes into account the discussion about the No-function-in-structure (NOFIS) principle from the engineering literature: as argued by Keuneke and Allemang it is impossible to describe device functions strictly adhering to the NOFIS principle; the description of the function of a device may therefore reference other classes of devices known to experts and intended to function in some way, without the assumption that those devices function correctly [139].

Coming to functions, their taxonomy was obtained by producing various ontological function types in the manner discussed in Section 3.1 (see Figure 5.1). Then, an attentive analysis of Adige's documentation was carried out to extract a complete set of functional verbs (e.g., move, translate, control, etc.) to ensure that the function taxonomy covers all verbs used in Adige's documents. Of course, the assumption is that a functional verb corresponds to a unique (modulo synonyms) ontological function concept.

In addition to the above semantic relations, a set of annotation properties was selected to carry the lexical information. We decided to use SKOS[12] since it is a well-known standard that supplies annotations for definitions, notes, preferred labels, alternative labels, deprecated labels, etc. (`skos : definition`, `skos : note`, `skos : prefLabel`, `skos : altLabel`, `skos : hiddenLabel`, etc.). In particular, each concept is linked to a set of synonymous terms (some valid, using `skos : altLabel` and some deprecated, using `skos : hiddenLabel`) and one term is selected to be the preferred term for that concept. The preferred term is then used, once concatenated with an appropriate namespace, as the unique[13] identifier for the concept. Note that this is essentially a WordNet synset

[12] `https://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html`.

[13] Uniqness requires paying attention that preferred terms are unique within a given namespace.

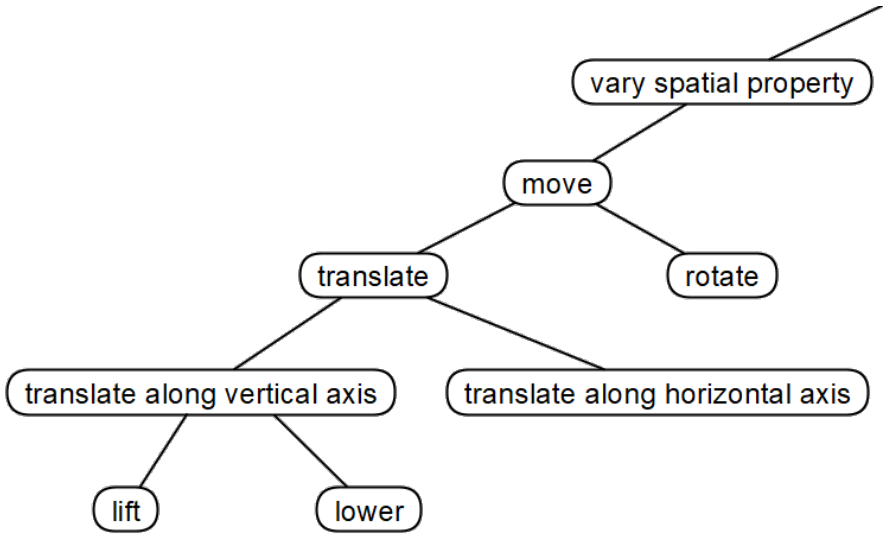



Figure 5.1: Excerpt of the taxonomy of functional terms.

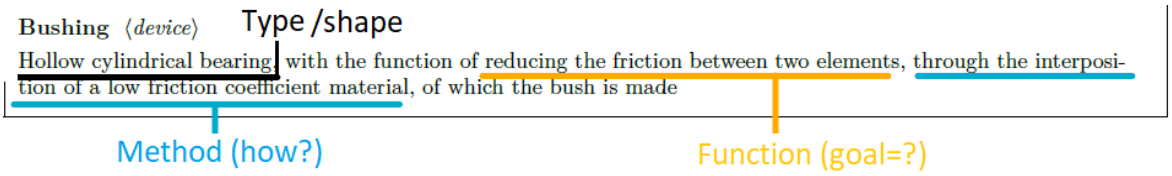



Figure 5.2: Example of a natural language definition of the term 'bush'.

structure, albeit differently encoded. Additionally, the glossary was developed in parallel in two languages, Italian and English. Terms of different languages are annotated with corresponding language tags.

A remarkable feature of the glossary is that natural language glosses were semi-automatically built from the semantic network neighbourhood of the corresponding term. See e.g. Figure 5.2, in which a gloss for 'bush' is present that is mainly constituted by sentences automatically generated using hypernyms and associated functions and engineering methods.

## 5.2 Semantic Search of Documents for Customer Service

Another need for Adige is storing, sharing, and retrieving maintenance-related information. In particular, a search application to retrieve relevant documents (e.g. service manuals, maintenance field reports, technicians' notes, etc.) was considered critically helpful in this context.

Some of the main requirements collected for the application search are:

- The necessity of finding relevant items within the first few (2-5) results.

- The 'smartness' of the search application (i.e. some kind of semantic awareness from part of the application).
- The need to be able to filter search results with a variety of "logical" (precise, punctual) filters, in particular component codes (the EMS keys for the product types), especially when arranged in a relevant bill of materials, should be pivotal for the search and retrieval of the documents[14].
- The need for gen-ai augmentation techniques (such as summarization of search results).
- The need to deal with documents in different languages.

After requirement collection various search applications were reviewed and a prototype was developed using the Elasticsearch engine. This choice was motivated by the fact that Elasticsearch is well-known in enterprise environments, is (albeit partially) open source, and has capabilities sufficient to implement all the desired requirements.

The data and knowledge flow in the application is schematically represented in Figure 5.3: the documents of interest are heterogeneous and contained in different applications (1), these are indexed (2) by the Elasticsearch engine (5), using both ontology(4)-enriched keywords and vector embeddings. When a user (6) queries the application, results are returned by searching the indexed data. The search makes use of keywords, and categorical and numerical filters. The filters requires information coming from other databases (3) of the EMS, for instance for the product types codes and bill of materials.

The ensuing search can be considered 'smart' or 'semantic' for various reasons: the ontology-based enrichment consists of an expansion of keyword indices using (some of) the semantic and lexical relations of the glossary discussed in the previous section, allowing the retrieval of semantically related terms from the search keywords. The vector embeddings in the indices also have the same function (vector embeddings are more flexible and work-free, however they are more computationally expansive and it is not trivial to inject domain knowledge in them, the opposite holds for the keyword expansion). Finally, an LLM is prompted with an additional query (e.g. for summarization) and the search results, implementing a basic RAG (retrieval augmented generation). Compare Figure 5.4.

In this context the ontology has three roles: as said above it enriches the document indices. Additionally, it supplies (some of) the application filters, and, moreover, it provides semantic integration capabilities.

Regarding the application filters, they were derived by reviewing similar contemporary applications, classical related engineering methodologies (mostly FRACAS, cf. Section 2.2.2), derivation from MALFO ontology, and discussions with domain experts.

[14]Notice that the use of precise filters requires that the documents to be retrieved be conspicuously annotated with relevant metadata. Doing these annotations has a cost that must be delegated to manual labor or NLP knowledge-extraction tools, either during data acquisition or to the historical document databases. In any case, this is an important issue, but it is outside the scope of the search application.

In particular, the key, pivotal ontological distinctions in this context appear to be the trifecta *component-defect-cause* and duality problem-issue. Using MALFO, defects and causes should be interpreted as individual occurrences of some of the categories of MALFO taxonomy (of course the component is the bearer of the function attacked by the malfunction). In particular, the component-defect relation is participation, and the cause-defect relation is some flavor of positive causation, as intended in Toyoshima's ontology. The precise flavor of causation can be determined only through failure analysis of the precise happening (in practice causations of disparate types can, and are, collected), however this is most often not necessary. If a throughout root cause analysis is required, then the originating happening (recorded as an 'issue' in the indexed documents) is escalated to start the assignation of the resources necessary for the analysis. An escalated issue is usually termed a 'problem', and multiple issues can be escalated to the same problem. Formally, a (product-defect-cause) issue can be formalized as a triple $(p, d, c)$, where $p$ is a technical artefact (`TechArtifact`); and $d$ and $c$ instantiate two (possibly different) categories of MALFO; $p$ participates in $d$ (for the whole of $d$'s life), and $\texttt{posCausalCon}(c, d)^{\dagger}$ holds; finally, $p$ must be the bearer of the function with whom $d$ is incompatible with (cf. Definition (df51)). On the other hand, a reasonable interpretation of 'problems' is any individual MALFO's malfunction that is able to cause multiple issues, of such quantity and quality to be a problem for the enterprise (e.g., a design error causing excessive stresses on system components, thus systematically causing (gear-fracture-fatigue) issues is a 'problem').

Regarding the integration capabilities, they are necessary to (at least) interoperate between different data presentations. For instance, BoM used for manufacturing processes are generally too complex for document retrieval tasks. For this reason, a simpler, (partially) function-based, ontologized[15] BoM was developed for some of Adige's machines. Mappings between these two BoM types are necessary (also) for the search tasks (cf. the bottom-left of Figure 5.4), and they can be realized with ontology-based methods.

Confront with service technicians gave a positive outlook on the ability of the application described above of satisfying the requirements listed at the start of this section. More precisely, the interaction with the company's personnel in regard to this document search application was as follow: a first step consisted of an extensive series of interviews[16] to service technicians and managers carried out in order to establish the main characteristics (especially weak points and desiderata) of the knowledge management strategy of the service department. This set of interviews was conducted in the context of broader efforts to better Adige's knowledge management and an extensive list of requirements and desiderata was derived, a subset of these directly related to a hypothetical document-search application. Later, a more specific set of interviews was conducted with a reduced set of technicians and managers, in order to better pinpoint the requirements for the search application. As a result the requirement list at the start of this section was produced, together with a first serialized prototype of the application of Figures 5.3 and 5.4. Finally, an iterative process was carried out in which three

[15]This ontologization is based on, and aligned with, the ontologies of chapters 3 and 4.

[16]Most of this first set of interviews were directed by external consultants.

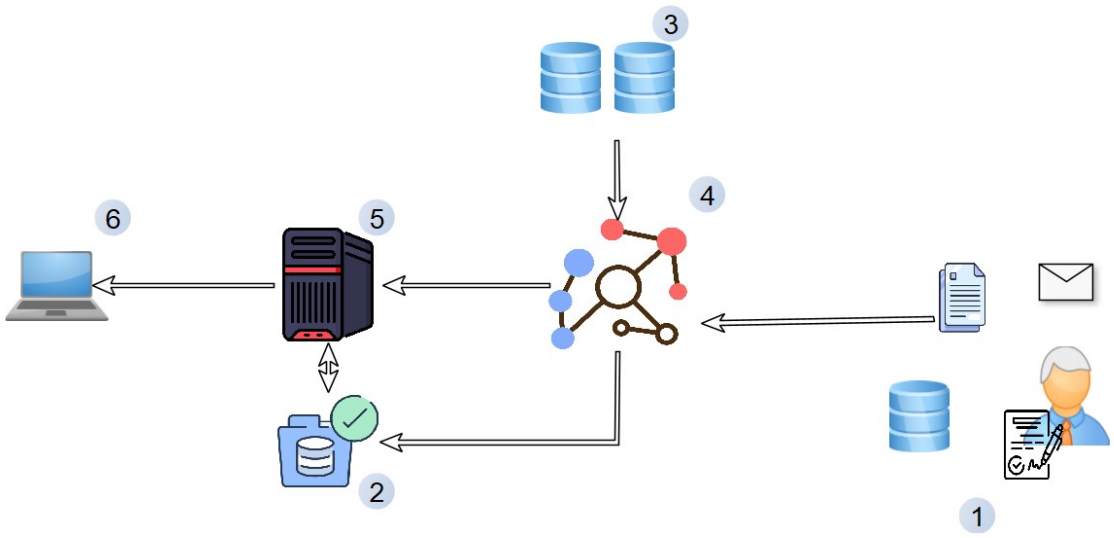



Figure 5.3: 'Smart' document search application architecture: (1) the documents of interest, which are heterogeneous and contained in different applications; (2) the Elasticsearch engine index; (3) the EMS, containing e.g. the machine BOMs; (4) the ontology; (5) the Elasticsearch engine proper; (6) the user.

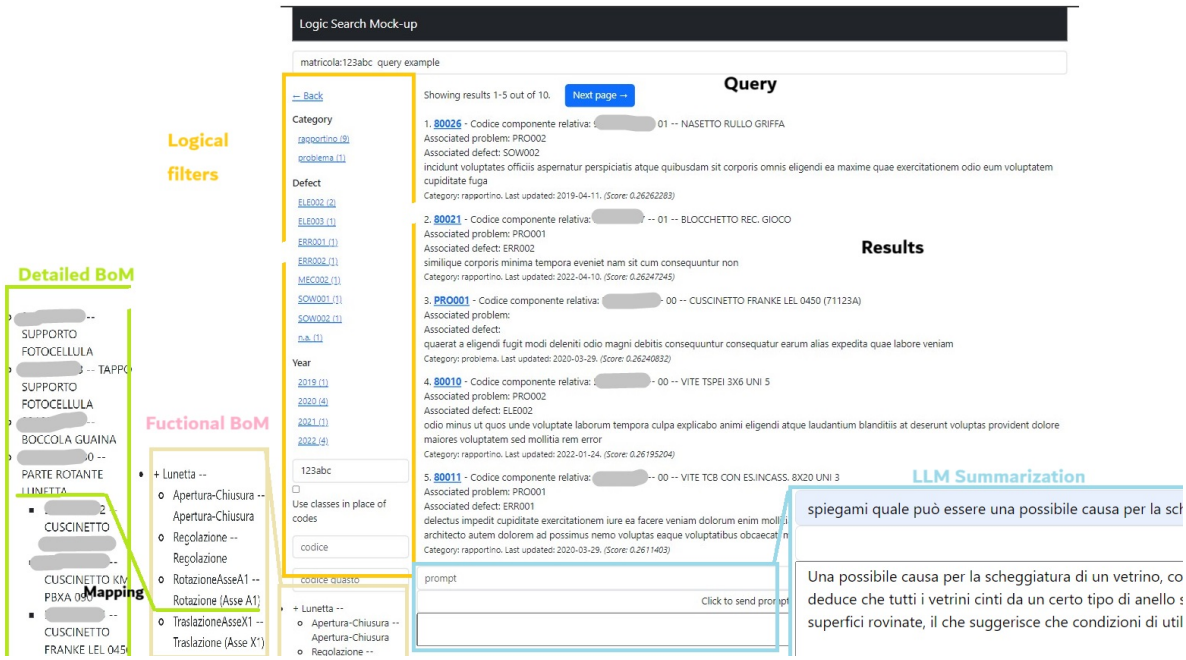



Figure 5.4: View of the document search application graphic user interface.

service technicians gave feedback about the prototype that was used for its betterment. The result of this process was that the feedback of the service technicians about the ability of the prototype to satisfy its requirements was mainly[17] positive.

As a final remark, note that the core challenge of this documental search task is data integration, on multiple levels: for collecting heterogeneous documents in a standardized manner, mapping between different data presentations, providing a unique interpretation of the labels attached to issue reports. Note also that the ontological efforts of the previous sections have a positive impact on such data integration.

## 5.3 Ontology-Driven Root Cause Analysis

For Adige, as well as for all manufacturing companies, equipment and product maintenance is essential. Maintenance itself is a broad discipline, and, since our main target use-case is breakdown maintenance (i.e., something breaks or otherwise malfunctions and must be repaired), we mainly focused, in Chapter 4, on the analysis of malfunctions. There we also introduced our MALFunction Ontology (MALFO) and discussed how it can be used to enrich root cause analysis efforts. Since the details have been already discusses there, here we briefly outline how a MALFO-based methodology could be used in an enterprise environment.

First, it must be noted that it is typical, in practical settings, to distinguish between simple, routine malfunctions that cause limited material and economical damage, and extensive, catastrophic ones, which cause more extended damage. This is reflected also in commercial products[18]. This may lead to different workflows and applications being used to manage malfunctions depending on their damage. Additionally, in this context, an enterprise may also have available various FMEA tables (or otherwise results from similar methodologies, such as RCA reports). The data from each of these sources is likely to be very semantically complex and difficult to integrate together with the data coming from the other sources. Hence, we highlight the usefulness of using an ontology, such as MALFO, for describing the metadata to attach to the original documents, or to revision and restructure the original documents in an ontologized way.

More specifically, data collected from malfunctions of limited damage will likely contain causal information such as cause, failure mode, and effect of the incident, and the vocabulary of MALFO could be used to formally define and relate such concepts. Likewise, RCA documents could be enriched with MALFO-based tagging, or being replaced directly by MALFO models, as described in Chapter 4. Advantages of this approach are the use of an overarching conceptual model for the data, and its serialization in a queryable way. However, this proposal must be refined by further work before arriving to an operative level and the work in Chapter 4 is mainly of theoretical nature.

[17]The main exception here consists of the technicians' doubts about the difficult to make operative the information flow from heterogenoeous documents (item (1) in Figure 5.3) into the search database. This concern is correct and on point, but is outside the scope of the scope of this search application, as discussed also in Note 14.

[18]Such as in the difference between incidents and problems in Relyance's FRACAS application `https://relyence.com/help/user-guide/introduction-to-fracas.html`.

# Final Considerations

This thesis addressed some of the challenges in information modeling and management commonly found in the industry and especially manufacturing companies, with a focus on Adige S.p.A.. The main objectives were to resolve issues related to heterogeneity in information (I1), rigid information structures (I2), and absence of systematic data integration (I3). To tackle these issues, the thesis employed Applied Ontology, a field that leverages logical and philosophical methods to address practical problems, especially in conceptual modeling and knowledge engineering.

Using this approach, the thesis focused on the modeling of functions and malfunctions in engineering systems (RQ1), which are pivotal topics for information modeling in manufacturing and maintenance, and especially for Adige. This was done to facilitate (RQ2) and to exemplify (RQ3) the adoption of semantic technologies, which, in turn, facilitate semantic data integration and tackle the issues I1, I2, and I3.

More precisely, the thesis opened with a thorough introduction to Applied Ontology, the Semantic Web and, more generally, semantic technologies (Chapter 1), explaining both foundational and practical aspects, and providing information about their historical development. In particular the upper ontology DOLCE was introduced and the critical role of semantic technologies for data interoperability and integration was explained. The second chapter (2) analyzed the history, use, and conceptualization of functions, behaviors, capabilities, and of malfunctions, failures, and faults in engineering, finding key conceptual issues and informing the development of subsequent chapters. The third chapter (3) developed an overarching ontological DOLCE-based formal theory encompassing various specialized aspects of engineering function, and addressing topics such as resource modeling and functional decomposition. The fourth chapter (4) linked and disambiguated a host of malfunctions-related terms and analyzing various connected engineering methodologies. Again, an overarching formal theory was proposed, which was linked to the efforts of Chapter 3 and tackled the causal aspects of failure analysis. Note that in both the third and fourth chapter the research results there obtained, which include formal ontologies serialized in both first-order logic and OWL, were compared with various other pre-existing approaches taken from the literature. Finally, Chapter 5 exploited chapters 3 and 4 to exemplify how ontologies can have a pivotal role in practical enterprise knowledge-management applications: a plan to use the malfunction ontology to enrich failure data collection and analysis was detailed, and the development of an actual ontology-based glossary and ontology-driven smart document search application were described.

The research questions can now be answered as follows:

- `RQ1` What are the main fundamental issues in knowledge representation in the notions of function, failure and malfunction in engineering, and how can applied ontology contribute to their solution?

  There are many fundamental issues in knowledge representation of functions and malfunctions of engineering systems, many of which are discussed in depth in the literature reviews of Chapter 2 and are otherwise too long to be detailed here. However, some honorable mentions are, regarding function:

  - the duality between generally valid functions and function-means/way-of-achievements: what is the difference between *what* a function does with respect to *how* a function does it, is there a difference between e.g. a 'divide' event and a 'cut' event?
  - The ontological categorization of functions and behaviors: what are engineering behaviors and what are functions? Is function a polysemous term with meanings corresponding to a single ontological category or does it have multiple heterogeneous meanings?
  - The difference between functions and capabilities: sometimes 'function' and 'capability' are treated as synonyms, are there genuine ontological differences that can differentiate them?
  - Capability can be compared and change in time: which are ways to model this ontologically?

  And, regarding malfunctions:

  - causality is an essential aspect of malfunction-related happenings: which are the main causal relations between these?
  - How can malfunction-related happenings be differentiated: can we build a taxonomy?
  - Engineering methodologies in failure analysis have to deal with difficult ontological issues. Can the development of a well-founded relevant ontology help such methodologies, in particular with abductive engineering methodologies?

  For all these issues, applied ontology helps to drive conceptual modeling and provide an overarching conceptual framework that can be used to explain key concepts and realize mappings with and between other frameworks. We elaborate on said framework in the next research question.

- `RQ2` How can the analysis of the fundamental knowledge representation issues of `RQ1` enable applied ontology and semantic technologies to represent knowledge about functions and malfunctions in a well-founded way?

  Paraphrasing Studer [237], Applied Ontology can lead conceptual modeling and produce formal, explicit, and shared specifications, called ontologies, of conceptualizations of domains contained in manufacturing and maintenance. This is the

main fruit of our work and was done in chapters 3 and 4. The ontologies developed there model important pieces of engineering knowledge, including information about functional modeling, functional decomposition, resource modeling, FMEA, and root cause analysis. In this way, the fact that the ontologies are well-founded (from an ontological point of view) ensures their high quality, which, in turn, facilitates interoperability in the way described in Section 1.2.

Going more in detail, unsurprisingly, we can divide the framework developed in the thesis into two main modules: one ontology for functions[19] and one for malfunctions[20]. The ontology of functions individuates and disambiguates different meanings of the term 'functions': *systemic functions* are roles contextualized by a system and represent the causal contribution of a part of the system to the whole of the system's behavior. *Ontological functions* are essentially event types representing a change between initial and final states of engineering interest. *Engineering methods* are social objects (as e.g. descriptions and plans) that describe a way a certain ontological function can be executed (e.g. laser-cutting is an engineering method for the ontological function to-divide). Engineering methods contextualize certain roles, in particular main-functions, which we call *Solved functions* and classify the individual function decomposed by the method, and sub-functions, which are the functions required for the method's implementation. Moreover, engineering functional decomposition is explained through engineering methods. Finally functions can also be ascripted to a bearer by means of an appropriate selection event, in that case they are essential properties of the bearer and are called *selected functions*. These are all extrinsic entities w.r.to the function's bearer (except for selected functions). Relevant intrinsic entities are instead *capabilities*, which are similar to dispositions (if not outright a subclass of) and explain the ability of their bearer to execute a function of a certain type. Capabilities can be analytically and intrinsically explained by simpler qualities on which they depend on. In particular, *capacities* are the most relevant category of said simpler qualities.

This ontology arguably does have some virtues, among which, in particular:

- the punctual disambiguation of various related but different ontological entities that are often indicated by engineers with the same term: 'function'. As stated in the thesis, we espouse a descriptional approach [45], since it appears that such a multiplicity of meanings is strategically useful for engineers [253].
- The topic of functional decomposition is discussed at length in the ontology: ontological modelling of functional decomposition is described and exemplified. Additionally, the issue of parthood between functions (which is related but somewhat different from decomposition) is also tackled.

[19]Serialized in OWL and other formats in `https://github.com/kataph/function-method-ontology`.

[20]Serialized in OWL and other formats in `https://github.com/kataph/MALFO`.

  - Analogously, an ontologically-founded intrinsical analysis of capabilities is outlined, which explains their performance. In particular, the issue of capability quality spaces is discussed, together with the dependence of capabilities on other categories of quality.
  - Comparison and mapping between this ontology of functions and some other approaches present in the literature is discussed extensively.

  The ontology of malfunctions (called MALFO for short), instead mainly does three things: (i) it supplies an ontological framework for causation, especially in the context of malfunction-related happenings. To do this, we mainly exploit the causation ontology of Toyoshima, although with some modifications and an attention to important details such as the treatment of non-actual occurrents and of synchronous relations between occurrences. And (ii) it contains a formal taxonomy of malfunction-related happenings. We do not mention all its elements, but, in particular, *malfunctions* are conceptualized as occurrences incompatible with functions, *failures* are malfunctions that are events resulting in *faults*, which are states that are malfunctions due to some internal conditions, *failure mechanisms* are processes that eventually cause failures, and *mere symptoms* are malfunction-related happenings that are not malfunctions and do not cause faults. External disabled states, non-performance events, and failure conditions exhaust MALFO's taxonomy.

  MALFO also (arguably) has some virtues, among which:

  - it has at its core a formal taxonomy of malfunction-related happenings, that disambiguate and clarify the meaning of many terms used in failure analysis.
  - It also contains an ontology of causation, which can be used to model causation between occurrences and explains the main relations between malfunction-related happenings.
  - The causation ontology and the formal taxonomy of MALFO can be used to describe malfunctions in industrial cases in an ontologically-sound way. Moreover this is discussed and exemplified in detail.
  - As for the ontology of functions, comparison and mapping between MALFO and some other approaches is discussed extensively.

- `RQ3` What are some possible ways to use ontologies (and semantic technologies more in general) to drive examples of relevant enterprise applications?

  On a general level, this question is addressed in Section 1.2, which discusses briefly the usefulness of semantic technologies in data integration and data interoperability. However and moreover, this question was answered by examples: this is the scope of Chapter 5, which showcases some semantic-technology-based tools.

  In particular, an ontology-based domain glossary is described, together with guidelines on how to enrich typical workflows related to failure analysis and reporting. The glossary is based on the ontologies discussed in `RQ2` and, in particular, a

faceted taxonomy of devices is obtained by using (also) selected functions and engineering methods as facets. Moreover an application to facilitate access to enterprise knowledge is discussed: it uses ontologies to enrich document indices, supply metadata for filtering, and integrate different data presentations imported from different silos of the enterprise management system. Particular issues that this approach mitigates are the complexity and heterogeneity of bills of materials, the need to map between different (views of) bills of materials, the need to be able to have complex logical filters (e.g. by BOM codes).

**Future work.** As a final remark, further work is both needed and planned, not least due to the various limits of this thesis. In particular, the current functional theory should be further developed, especially along the following directions:

- more practical tools for functional decomposition should be developed. The analysis of this thesis is mainly carried out at a formal level, so it would be most useful to delve further into (executions of) engineering methodologies, examples, comparative analyses, and design of supporting toolsets.
- In the same spirit, methodologies to exploit the developed ontologies for failure analysis and reporting, for which only the theoretical foundations have been solidly established, should be developed in detail (also addressing the limitations discussed in Section 4.1.3). Furthermore, we have only scratched the surface of what is possible from this point of view: we have focused on a-posteriori descriptive methodologies, but engineering has plenty of methodologies to analyze ontologically (FMEA, FRACAS, HFACS, FTA, ...). Field studies would be especially interesting.
- Various foundational ontological issues should also be better developed. These are many, but, in particular, we mention relational qualities, which have only received a passing treatment in this thesis (we used them to characterize capabilities and capacities, but described them only briefly). Occurrents are protagonists in this thesis, but we have been more concerned with their properties at a domain level, so to speak, (sometimes) evading or touching only briefly important foundational issues such as identity and identification criteria, and unit criteria. Therefore a more in-depth analysis would be useful. Dispositions constitute another topic that has been evaded, albeit not completely, in this thesis. They should also receive detailed analysis (e.g. what are dispositions? Are capabilities dispositions? What are the causal relations between dispositions and dispositions or other entities? Etc.). Finally, while function mereology has been discussed, functional-object mereology has been only mentioned. This is another possible line of research, in addition to a more in-depth analysis of the criteria that a set of functions should have to be able to constitute a decomposition of another function. Note that the reasons that all these topics did not receive full attention in this thesis include scope costraints and, moreover, the design choice of taking minimal ontological commitments in the thesis.

- Additionally, the discussion on the nature of capabilities and capacities quality spaces and the causal relation between them strongly needs further analysis: as of now, we have only started delving into it, and a more systematic treatment should be devised. In particular, it should also be investigated the correspondence, if any, between functional decomposition and hierarchical decomposition of capabilities into simpler qualities.
- The work developed in the thesis would benefit from using it to build a systematic showcase of modeling patterns tackling common knowledge representation issues in manufacturing and maintenance.
- Finally, while we made considerable efforts to compare to and establish mappings with as many other relevant framework as possible, the result is still patchworked and could be made more systematic. In particular (i) a systematic comparison of the most established upper ontologies w.r.to selected topics of this thesis could be realized, to schematically determine which ontological commitments and modelling patterns are possible and how commitments effect patterns, (ii) a systematic mapping of industrial standards concepts and terms into this thesis' framework could also be realized.

# Acknowledgments

**AI-generated content statement.** With this paragraph, I recognize explicitly that I have used generative artificial intelligence when writing this thesis. Such a use has been limited to rephrasing short English texts written by me, to gain suggestions on how to write with a better style. Each AI-generated sentence was later read and possibly rejected or modified to fit my style of writing. The whole process was sometimes repeated multiple times for the same sentence. Note that I did not use AI for developing the software artefacts mentioned in the thesis, nor for the development of the logical theories, nor for any planning or design related to this thesis content or structure.

# Bibliography


[1] Maintenance - Maintenance terminology, Standard EN 13306. Cen-en, 2017.

[2] Polleres A. The role of logics and logic programming in semantic web standards (owl2, rif, sparql1.1), 2010.

[3] Singhal A. Introducing the knowledge graph: Things, not strings. *Google Blog*, May 2012.

[4] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[5] Renzo Angles. A comparison of current graph database models. pages 171–177, 04 2012.

[6] Anonymous. `https://news.ycombinator.com/item?id=8510970` 2014. Accessed: March 2024.

[7] Robert Arp and Smith Barry. Function, role and disposition in Basic Formal Ontology. In *Nature Precedings*, 2008.

[8] Charles Arthur. Tech giants may be huge, but nothing matches big data. The Guardian, ISSN 0261-3077. Accessed: March 2024.

[9] Marc Artiga. Re-organizing organizational accounts of function. *Applied Ontology*, 6(2):105–124, 2011.

[10] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, January 2004.

[11] Francisco J. Ayala. Teleological explanations in evolutionary biology. *Philosophy of Science*, March 1970.

[12] Carlos L.B. Azevedo, Maria-Eugenia Iacob, João Paulo A. Almeida, Marten van Sinderen, Luís Ferreira Pires, and Giancarlo Guizzardi. Modeling resources and capabilities in enterprise architecture: A well-founded ontology-based proposal for archimate. *Information Systems*, 54:235–262, 2015.

[13] Lynne Rudder Baker and Philosophy Documentation Center. The Metaphysics of Malfunction. *Techné: Research in Philosophy and Technology*, 13(2):82–92, 2009.

[14] René Ronald Bakker. *KNOWLEDGE GRAPHS: representation and structuring of scientific knowledge*. PhD thesis, Universiteit Twente, 1987.

[15] Riccardo Baratella, Mattia Fumagalli, Ítalo Oliveira, and Giancarlo Guizzardi. Understanding and modeling prevention. 04 2022.

[16] Smith Barry and alt. Basic Formal Ontology 2.0, June 2015.

[17] Matthew A Barsalou. *Root cause analysis: A step-by-step guide to using the right tool at the right time*. CRC Press, 2014.

[18] A. Barton, L. Jansen, and J.-F. Ethier. A taxonomy of disposition-parthood. In *Proceedings of the Workshop on Foundational Ontology in Joint Ontology Workshops: JOWO 2017*, volume 2050, pages 1–10. CEUR-WS, 2017.

[19] Lucas Bechberger and Margit Scheibel. Representing complex shapes with conceptual spaces. In *Second International Workshop "Concepts in Action: Representation, Learning, and Application"*, 2020.

[20] Alessander Benevides, Jean-Rémi Bourguet, Giancarlo Guizzardi, Rafael Peñaloza, and João Almeida. Representing a reference foundational ontology of events in sroiq. *Applied Ontology*, 14:1–42, 07 2019.

[21] Mike Bergman. Sources and Classification of Semantic Heterogeneities, June 2006.

[22] Mike Bergman. Big Structure and Data Interoperability, August 2014.

[23] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[24] Thomas Bittner, Maureen Donnelly, and Stephan Winter. Ontology and Semantic Interoperability. In *Large-scale 3D data integration*. CRC Press, January 2006.

[25] K.M. Blache and A.B. Shrivastava. Defining failure of manufacturing machinery and equipment. In *Proceedings of Annual Reliability and Maintainability Symposium (RAMS)*, pages 69–75, 1994.

[26] Stefano Borgo, Massimiliano Carrara, Pawel Garbacz, and Pieter Vermaas. Towards the Ontological Representation of Functional Basis in dolce. In *Interdisciplinary Ontology Vol. 2, Proceedings of the 2nd Interdisciplinary Ontology Meeting*, page 13, Tokyo, Japan, February 2009.

[27] Stefano Borgo, Massimiliano Carrara, Pawel Garbacz, and Pieter E. Vermaas. A formal ontological perspective on the behaviors and functions of technical artifacts. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1):3–21, February 2009.

[28] Stefano Borgo, Massimiliano Carrara, Pawel Garbacz, and Pieter E. Vermaas. A Formalization of Functions as Operations on Flows. *Journal of Computing and Information Science in Engineering*, 11(3):031007, September 2011.

[29] Stefano Borgo, Amedeo Cesta, Andrea Orlandini, and Alessandro Umbrico. Knowledge-based adaptive agents for manufacturing domains. *Engineering with Computers*, 35(3):755–779, July 2019.

[30] Stefano Borgo, Francesco Compagno, Nicola Guarino, Claudio Masolo, and Emilio M Sanfilippo. An overview of some ontological challenges in engineering maintenance. In *Domain Ontologies for Research Data Management in Industry Commons of Materials and Manufacturing (DORIC-MM 2021)*, page 79, 2021.

[31] Stefano Borgo, Roberta Ferrario, Aldo Gangemi, Nicola Guarino, Claudio Masolo, Daniele Porello, Emilio M. Sanfilippo, and Laure Vieu. DOLCE: A descriptive ontology for linguistic and cognitive engineering. *Applied Ontology, to appear*, 17(1):45–69, March 2022.

[32] Stefano Borgo, Maarten Franssen, Paweł Garbacz, Yoshinobu Kitamura, Riichiro Mizoguchi, and Pieter E Vermaas. Technical Artifacts: An Integrated Perspective. *Applied Ontology*, page 18, 2017.

[33] Stefano Borgo, Riichiro Mizoguchi, and Barry Smith. On the Ontology of Functions. In *Applied Ontology*, volume 6, pages 99–104. Elsevier, 2011.

[34] Stefano Borgo, Emilio M Sanfilippo, and Walter Terkaj. Capabilities, Capacities, and Functionalities of Resources in Industrial Engineering. In *CEUR Workshop Peoceedings*, page 12, Bolzano, Italy, 2021.

[35] Stefano Borgo and Laure Vieu. Artefacts in Formal Ontology. In *Philosophy of Technology and Engineering Sciences*, pages 273–307. Elsevier, 2009.

[36] David C. Brown and Lucienne Blessing. The Relationship Between Function and Affordance. In *Volume 5a: 17th International Conference on Design Theory and Methodology*, pages 155–160, Long Beach, California, USA, January 2005. ASMEDC.

[37] Mikkel Haggren Brynildsen, Maja Miličić Brandt, Johan Wilhelm Klüwer, Caitlin Woods, and Melinda R Hodkiewicz. Automated verification of measurement precision for internet-of-things equipment. In *22nd International Semantic Web Conference on Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice, ISWC-Posters-Demos-Industry 2023*. Central Europe Workshop Proceedings (CEUR-WS), 2023.

[38] P. Burek, R. Hoehndorf, F. Loebe, J. Visagie, H. Herre, and J. Kelso. A top-level ontology of functions and its application in the Open Biomedical Ontologies. *Bioinformatics*, 22(14):e66–e73, July 2006.

[39] Patryk Burek. *Ontology of Functions: A Domain-independent Framework for Modeling Functions*. PhD thesis, Universität Leipzig, 2006.

[40] Patryk Burek, Frank Loebe, and Heinrich Herre. Overview of GFO 2.0 Functions: An ontology module for representing teleological knowledge. *Procedia Computer Science*, 192:1021–1030, 2021.

[41] Andrew Butterfield and John Szymanski. actuator, 2018.

[42] Jacek Błażewicz, Wieslaw Kubiak, Tadeusz Morzy, and Marek Rusinkiewicz, editors. *Handbook on Data Management in Information Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[43] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Ontology-Based Data Access and Integration. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 1–7. Springer New York, New York, NY, 2017.

[44] K. Campbell. *Abstract Particulars*. Basil Blackwell, Oxford, 1990.

[45] Massimiliano Carrara, Pawel Garbacz, and Pieter E. Vermaas. If engineering function is a family resemblance concept: Assessing three formalization strategies. *Applied Ontology*, 6(2):141–163, 2011.

[46] B. Chandrasekaran. Functional Representation and Causal Processes. In *Advances in Computers*, volume 38, pages 73–143. Elsevier, 1994.

[47] B. Chandrasekaran, A.K. Goel, and Y. Iwasaki. Functional representation as design rationale. *Computer*, 26(1):48–56, January 1993.

[48] B Chandrasekaran and John R Josephson. Representing Function as Effect. In *Proceedings of the Fifth International Workshop on Advances in Functional Modeling of Complex Technical Systems*, page 13, Paris, France, 1997.

[49] B. Chandrasekaran and John R. Josephson. Function in Device Representation. *Engineering with Computers*, 16(3-4):162–177, December 2000.

[50] B. Chandrasekaran and Sanjay Mittal. Deep versus compiled knowledge approaches to diagnostic problem-solving. *International Journal of Man-Machine Studies*, 19(5):425–436, November 1983.

[51] J. A. Collins, B. T. Hagan, and H. M. Bratt. The Failure-Experience Matrix—A Useful Design Tool. *Journal of Engineering for Industry*, 98(3):1074–1079, August 1976.

[52] Francesco Compagno and Stefano Borgo. Towards a formal ontology of engineering functions, behaviours, and capabilities. *Semantic Web Journal*, Special Issue on Semantic Web for Industrial Engineering: Research and Applications, 2022.

[53] Francesco Compagno and Stefano Borgo. Ontological analysis of malfunctions: Some formal considerations. In *Formal Ontology in Information Systems*, pages 149–162. IOS Press, 2024.

[54] Francesco Compagno and Stefano Borgo. Functional Modelling in Engineering: A First Comparison of Approaches on Two Problems. In *Proceedings of the 12th International Workshop on Formal Ontologies Meet Industry (FOMI 2022)*, volume 3240 of *CEUR Workshop Proceedings*, page 12, Tarbes, France, September 12-15, 2022.

[55] Francesco Compagno, Stefano Borgo, Claudio Masolo, and Emilio M Sanfilippo. Comparing Ontological Alternatives to Model Engineering Systems and Components. In *Proceedings of the Joint Ontology Workshops 2021*, page 12, Bolzano, Italy, September 2021. CEUR.

[56] Compagno, Francesco, Borgo, Stefano, Sanfilippo, Emilio M., and Terkaj, Walter. Manufacturing Resources, Capabilities, and Engineering Functions: Towards an Ontology-based Integration. In *Formal Ontology in Information Systems Conference for 2023*, Sherbrooke, Canada, July 2023. Accepted for publication.

[57] Sofia I. R. Conceição, Diana F. Sousa, Pedro Silvestre, and Francisco M Couto. lasigeBioTM at SemEval-2023 task 7: Improving natural language inference baseline systems with domain ontologies. In Atul Kr. Ojha, A. Seza Doğruöz, Giovanni Da San Martino, Harish Tayyar Madabushi, Ritesh Kumar, and Elisa Sartori, editors, *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 10–15, Toronto, Canada, July 2023. Association for Computational Linguistics.

[58] E Cramer. *Chaos Und Ordnung*. Deutsche Verlagsanstalt, Stuttgart, 1988.

[59] Robert Cummins. Functional Analysis. *The Journal of Philosophy*, 72(20):741–765, November 1975.

[60] Johan De Kleer. How circuits work. *Artificial Intelligence*, 24(1-3):205–280, December 1984.

[61] Johan De Kleer and John Seely Brown. Mental models of physical mechanisms and their acquisition. In J. Anderson, editor, *Cognitive skills and their acquisition*, pages 285–309. Psychology Press, 1982.

[62] L Del Frate. Preliminaries to a formal ontology of failure of engineering artifacts. In M Donnelly and G Guizzardi, editors, *Formal Ontology in Information Systems: Proceedings of the Seventh International Conference (FOIS 2012)*, pages 117–130, Netherlands, 2012. IOS Press. null ; Conference date: 24-07-2012 Through 27-07-2012.

[63] L. Del Frate, M. Franssen, and P.E. Vermaas. Towards a trans-disciplinary concept of failure for integrated product development. *International Journal of Product Development*, 14(1-4):72–95, 2011.

[64] Luca Del Frate, Sjoerd D. Zwart, and Peter A. Kroes. Root cause as a u-turn. *Engineering Failure Analysis*, 18(2):747–758, 2011. The Fourth International Conference on Engineering Failure Analysis Part 1.

[65] M. Dimanzo, E. Trucco, F. Giunchiglia, and F. Ricci. Fur: Understanding functional reasoning. *Int. J. Intell. Syst.*, 4(4):431–457, December 1989.

[66] Cory Doctorow. metacrap. `https://people.well.com/user/doctorow/metacrap.htm` 2001. Accessed: March 2024.

[67] Bruno Borlini Duarte, Ricardo de Almeida Falbo, Giancarlo Guizzardi, Renata Guizzardi, and Vítor E. Silva Souza. An ontological analysis of software system anomalies and their associated risks. *Data & Knowledge Engineering*, 134:101892, July 2021.

[68] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16)*, 09 2016.

[69] MILITARY HANDBOOK: ELECTRONIC RELIABILITY DESIGN HANDBOOK, October 1998.

[70] M.S. Erden, H. Komoto, T.J. van Beek, V. D'Amelio, E. Echavarria, and T. Tomiyama. A review of function modeling: Approaches and applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22(2):147–169, 2008.

[71] EUSTIS. USAAMRDL TECHNICAL REPORT 71-74 HELICOPTER DEVELOPMENT RELIABILITY TEST REQUIREMENTS. Technical report, EUSTIS directorate, 1972.

[72] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. Mit Press, 1998.

[73] Claudenir M. Fonseca, Daniele Porello, Giancarlo Guizzardi, João Paulo A. Almeida, and Nicola Guarino. Relations in Ontology-Driven Conceptual Modeling. In Alberto H. F. Laender, Barbara Pernici, Ee-Peng Lim, and José Palazzo M. de Oliveira, editors, *Conceptual Modeling*, volume 11788, pages 28–42. Springer International Publishing, Cham, 2019.

[74] Philippe Fournier-Viger. The semantic web and why it failed. `https://data-mining.philippe-fournier-viger.com/the-semantic-web-and-why-it-failed/` 2018. Accessed: March 2024.

[75] Christian Fürber. *Data Quality Management with Semantic Technologies*. Springer Fachmedien Wiesbaden, Wiesbaden, 2016.

[76] Pawel Garbacz, Stefano Borgo, Massimiliano Carrara, and Pieter E. Vermaas. Two ontology-driven formalisations of functions and their comparison. *Journal of Engineering Design*, 22(11-12):733–764, December 2011.

[77] Peter Gärdenfors. *Conceptual spaces - the geometry of thought*. Bradford Books, 2000.

[78] Peter Gärdenfors and Simone Löhndorf. What is a domain? dimensional structures versus meronomic relations. *Cognitive Linguistics*, 24, 2013.

[79] Wolfgang Beitz Gerhard Pahl. *Konstruktionslehre: Handbuch für Studium und Praxis*. Springer, 1977.

[80] J. S. Gero. Categorising Technological Knowledge From a Design Methodological Perspective. In *International Conference on Technological Knowledge: Philosophical Reflections*, Boxmeer, the Netherlands, June 2002.

[81] John S. Gero and Udo Kannengiesser. The situated function–behaviour–structure framework. *Design Studies*, 25(4):373–391, July 2004.

[82] James J. Gibson. The Theory of Affordances. In *The Ecological Approach to Visual Perception.* Houghton Mifflin, Boston, 1979.

[83] M.A. Giese and M. Lappe. Measurement of generalization fields for the recognition of biological motion. *Vision Research*, 42(15):1847–1858, 2002.

[84] Amaninder Singh Gill and Chiradeep Sen. Logic Rules for Automated Synthesis of Function Models Using Evolutionary Algorithms. In *Volume 2: 41st Computers and Information in Engineering Conference (CIE)*, Virtual, Online, August 2021. American Society of Mechanical Engineers.

[85] Ashok K. Goel and Sambasiva R. Bhatta. Use of design patterns in analogy-based design. *Advanced Engineering Informatics*, 18(2):85–94, April 2004.

[86] Ashok K. Goel, Spencer Rugaber, and Swaroop Vattam. Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1):23–35, February 2009.

[87] Dale L. Goodhue, Michael D. Wybo, and Laurie J. Kirsch. The Impact of Data Integration on the Costs and Benefits of Information Systems. *MIS Quarterly*, 16(3):293–311, 1992. Publisher: Management Information Systems Research Center, University of Minnesota.

[88] Michael Grüninger and Mark S Fox. The role of competency questions in enterprise engineering. In *Benchmarking—Theory and practice*, pages 22–31. Springer, 1995.

[89] M. Grüninger. *Using the PSL ontology*, pages 423–443. Springer, 2009.

[90] Michael Grüninger and Mark Fox. Methodology for the design and evaluation of ontologies. In *Proc. IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing.*, 07 1995.

[91] Zhenzhen Gu, Francesco Corcoglioniti, Davide Lanti, Alessandro Mosca, Guohui Xiao, Jing Xiong, and Diego Calvanese. A systematic overview of data federation systems. *Semantic Web*, 15(1):107–165, January 2024.

[92] Nicola Guarino. Bfo and dolce: So far, so close? *Cosmos + Taxis*, 4(4):10–18, 2017.

[93] Nicola Guarino, Riccardo Baratella, and Giancarlo Guizzardi. Events, their names, and their synchronic structure. *Applied Ontology*, 17(2):249–283, January 2022.

[94] Nicola Guarino and Giancarlo Guizzardi. "we need to discuss the relationship": revisiting relationships as modeling constructs. In *Advanced Information Systems Engineering: 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings 27*, pages 279–294. Springer, 2015.

[95] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology? In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Springer, Berlin, Heidelberg, 2009.

[96] Nicola Guarino, Tiago Prince Sales, and Giancarlo Guizzardi. Reification and truthmaking patterns. 09 2018.

[97] Nicola Guarino and Christopher Welty†. A Formal Ontology of Properties. In G. Goos, J. Hartmanis, J. van Leeuwen, Rose Dieng, and Olivier Corby, editors, *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, volume 1937, pages 97–112. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[98] Nicola Guarino and Christopher A. Welty. An Overview of OntoClean. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 201–220. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[99] G Guizzardi. *Ontological Foundations for Structural Conceptual Models*. PhD thesis, Centre for Telematics and Information Technology ; Telematica Instituut, Enschede; Enschede, 2005.

[100] Giancarlo Guizzardi. The problem of transitivity of part-whole relations in conceptual modeling revisited. pages 94–109, 06 2009.

[101] Giancarlo Guizzardi, Alessander Benevides, Claudenir Fonseca, Daniele Porello, João Almeida, and Tiago Prince Sales. UFO: Unified Foundational Ontology. *Applied Ontology*, January 2022.

[102] Giancarlo Guizzardi, Claudenir M. Fonseca, Alessander Botti Benevides, João Paulo A. Almeida, Daniele Porello, and Tiago Prince Sales. Endurant types in ontology-driven conceptual modeling: Towards ontouml 2.0. In *Conceptual Modeling*, pages 136–150, Cham, 2018. Springer International Publishing.

[103] Giancarlo Guizzardi, Claudio Masolo, and Stefano Borgo. In defense of a trope-based ontology for conceptual modeling: An example with the foundations of attributes, weak entities and datatypes. In David W. Embley, Antoni Olivé, and Sudha Ram, editors, *Conceptual Modeling - ER 2006*, pages 112–125, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[104] Giancarlo Guizzardi, Gerd Wagner, Ricardo de Almeida Falbo, Renata SS Guizzardi, and João Paulo A Almeida. Towards ontological foundations for the conceptual modeling of events. In *Conceptual Modeling: 32th International Conference, ER 2013, Hong-Kong, China, November 11-13, 2013. Proceedings 32*, pages 327–341. Springer, 2013.

[105] Claudio Gutierrez and Juan F. Sequeda. Knowledge graphs. *Communications of the ACM*, 64(3):96–104, March 2021.

[106] Peter Gärdenfors. *Representing actions and functional properties in conceptual spaces*, pages 167–196. De Gruyter Mouton, Berlin, New York, 2007.

[107] Peter Gärdenfors and Massimo Warglien. Using conceptual spaces to model actions and events. *Journal of Semantics*, 29:487–519, 11 2012.

[108] Torsten Hahmann and Robert W. Powell II. Automatically extracting owl versions of fol ontologies. In *The Semantic Web – ISWC 2021*, pages 252–269, Cham, 2021. Springer International Publishing.

[109] Alon Halevy. Why Your Data Won't Mix: New tools and techniques can help ease the pain of reconciling schemas. *Queue*, 3(8):50–58, 2005.

[110] Arthur Hall D. *A Methodology for Systems Engineering*. D. Van Nostrand Company, 1962.

[111] E.J. Hallquist and Thomas Schick. Best practices for a fracas implementation. *Annual Symposium Reliability and Maintainability, 2004 - RAMS*, pages 663–667, 2004.

[112] John R. Hauser and Don Clausing. The house of quality. *Harward Business Review*, May 1988.

[113] James Hendler. Wither owl. `https://www.slideshare.net/jahendler/wither-owl?from_search=0` 2016.

[114] Heinrich Herre. *General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling*, pages 297–345. 09 2010.

[115] Heinrich Herre, Barbara Heller, Patryk Burek, Robert Hoehndorf, Frank Loebe, and Hannes Michalek. General Formal Ontology (GFO) - A Foundational Ontology Integrating Objects and Processes [Version 1.0]. Technical Report 8, Institute of Medical Informatics, Statistics and Epidemiology (IMISE), July 2006.

[116] Julie Hirtz, Robert B. Stone, Daniel A. McAdams, Simon Szykman, and Kristin L. Wood. A functional basis for engineering design: Reconciling and evolving previous efforts. *Research in Engineering Design*, 13(2):65–82, March 2002.

[117] Two-Bit History. `https://twobithistory.org/2018/05/27/semantic-web.html` 2018. Accessed: March 2024.

[118] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[119] Melinda Hodkiewicz, Johan W. Klüwer, Caitlin Woods, Thomas Smoker, and Emily Low. An ontology for reasoning over engineering textual data stored in FMEA spreadsheet tables. *Computers in Industry*, 131:103496, October 2021.

[120] Melinda Hodkiewicz, Caitlin Woods, Matt Selway, and Markus Stumptner. Iofmaint–modular maintenance ontology. *arXiv preprint arXiv:2404.05224*, 2024.

[121] W. Houkes and P. E. Vermaas. *Technical Functions: On the Use and Design of Artefacts*. Springer, 2010.

[122] Wybo Houkes and P.E. Vermaas. *Technical Functions*, volume 1 of *Philosophy of Engineering and Technology*. Springer Netherlands, Dordrecht, 2010.

[123] Thomas J. Howard and Mogens Myrup Andreasen. Mind-sets of functional reasoning in engineering design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 27(3):233–240, 2013.

[124] IEC. International Electrotechnical Vocabulary (IEV) - Part 192: Dependability. Standard, International Electrotechnical Commission, Geneva, CH, February 2015.

[125] Kaoru Ishikawa. *Guide to quality control*. Asian Productivity Organization, Tokyo, Japan, 2 edition, December 1986.

[126] ISO. ISO 15531-31: Industrial automation systems and integration - Industrial manufacturing management data - Part 31: Resource information model. Industrial Manufacturing Management Data, ISO, 2004.

[127] ISO. ISO 14224: Petroleum, petrochemical and natural gas industries — Collection and exchange of reliability and maintenance data for equipment. Technical report, ISO, 2016.

[128] ISO. ISO/CD 23726-3 Automation systems and integration — Ontology based interoperability, Part 3: Industrial data ontology. Technical report, ISO, 2023.

[129] ISO 15243: ISO 15243:2017 Rolling bearings — Damage and failures — Terms, characteristics and causes, 2017.

[130] Yumi Iwasaki and B Chandrasekaran. Design Verification Through Function- and Behavior-Oriented Representations. In J. S. Gero and Fay Sudweeks, editors, *Artificial Intelligence in Design '92*, page 20. Springer Netherlands, July 1992.

[131] Yumi Iwasaki, Richard Fikes, Marcos Vescovi, and B Chandrasekaran. How Things Are Intended to Work: Capturing Functional Knowledge in Device Design. In *International Joint Conference on Artificial Intelligence*, page 8, 1993.

[132] Krzysztof Janowicz and Martin Raubal. Affordance-based similarity measurement for entity types. In Stephan Winter, Matt Duckham, Lars Kulik, and Ben Kuipers, editors, *Spatial Information Theory*, pages 133–151, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[133] Ludger Jansen. Functions, Malfunctioning, and Negative Causation. In Alexander Christian, David Hommen, Nina Retzlaff, and Gerhard Schurz, editors, *Philosophy of Science*, volume 9, pages 117–135. Springer International Publishing, Cham, 2018.

[134] Eeva Järvenpää, Niko Siltala, Otto Hylli, and Minna Lanz. The development of an ontology for describing the capabilities of manufacturing resources. *Journal of Intelligent Manufacturing*, 30(2):959–978, February 2019.

[135] Jean-Pierre Jourdan, Ronan Bureau, Christophe Rochais, and Patrick Dallemagne. Drug repositioning: a brief overview. *Journal of Pharmacy and Pharmacology*, 72(9):1145–1151, September 2020.

[136] Megan Katsumi and Michael Grüninger. Automated reasoning support for ontology development. In Ana Fred, Jan L. G. Dietz, Kecheng Liu, and Joaquim Filipe, editors, *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 208–225, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[137] A. Keuneke and D. Allemang. Exploring the No-Function-In-Structure principle. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 78–89, 1989.

[138] A.M. Keuneke. Device representation-the significance of functional knowledge. *IEEE Expert*, 6(2):22–25, April 1991.

[139] Anne Keuneke and Dean Allemang. Exploring the No-Function-In-Structure principle. *Journal of Experimental & Theoretical Artificial Intelligence*, 1(1):79–89, January 1989.

[140] Jwan Khisro. Understanding the relation between interoperability and data quality: a study of data hub development in Swedish electricity market. *International Journal of Public Information Systems*, 14(1), 2020.

[141] Yoshinobu Kitamura, Yusuke Koji, and Riichiro Mizoguchi. An ontological model of device function: Industrial deployment and lessons learned. *Applied Ontology*, 1(3-4):237–262, January 2006.

[142] Yoshinobu Kitamura and Riichiro Mizoguchi. Meta-Functions of Artifacts. In *Proc. of The Thirteenth International Workshop on Qualitative Reasoning (QR-99)*, pages 136–145, Loch Awe, Scotland, 1999.

[143] Yoshinobu Kitamura and Riichiro Mizoguchi. An Ontological Analysis of Faults. In *Proceedings of tenth international workshop on principles of diagnosis (DX-99)*, page 8, 1999.

[144] Yoshinobu Kitamura and Riichiro Mizoguchi. Ontology-based description of functional design knowledge and its use in a functional way server. *Expert Systems with Applications*, 24(2):153–166, February 2003.

[145] Yoshinobu Kitamura and Riichiro Mizoguchi. Ontology-Based Functional-Knowledge Modeling Methodology and Its Deployment. In Enrico Motta, Nigel R Shadbolt, Arthur Stutt, and Nick Gibbins, editors, *Engineering Knowledge in the Age of the Semantic Web*, volume 3257, pages 99–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[146] Yoshinobu Kitamura and Riichiro Mizoguchi. Ontology-based systematization of functional knowledge. *Journal of Engineering Design*, 15(4):327–351, August 2004.

[147] Yoshinobu Kitamura and Riichiro Mizoguchi. Characterizing functions based on ontological models from an engineering point of view. *Frontiers in Artificial Intelligence and Applications*, 209:301–314, 01 2010.

[148] Yoshinobu Kitamura and Riichiro Mizoguchi. Characterizing functions based on phase- and evolution-oriented models. *Applied Ontology*, 8(2):73–94, 2013.

[149] Yoshinobu Kitamura and Riichiro Mizoguchi. Characterizing functions based on phase- and evolution-oriented models. *Applied Ontology*, 8(2):73–94, 2013.

[150] Yoshinobu Kitamura, Toshinobu Sano, Kouji Namba, and Riichiro Mizoguchi. A functional concept ontology and its application to automatic identification of functional structures. *Advanced Engineering Informatics*, 16(2):145–163, April 2002.

[151] Yoshinobu Kitamura, Sho Segawa, Munehiko Sasajima, Shinya Tarumi, and Riichiro Mizoguchi. Deep Semantic Mapping between Functional Taxonomies for

Interoperable Semantic Search. In John Domingue and Chutiporn Anutariya, editors, *The Semantic Web*, volume 5367, pages 137–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[152] Yoshinobu Kitamura, Sunao Takafuji, and Riichiro Mizoguchi. Towards a reference ontology for functional knowledge interoperability. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 01 2007.

[153] Johan De Kleer and John Seely Brown. A Qualitative Physics Based on Confluences. *Artificial Intelligence*, page 78, 1984.

[154] Johan W. Klüwer, Francisco Martin-Recuerda, Arild Waaler, Daniel Lupp, Maja M. Brandt, Stephan Grimm, Aneta Koleva, Mesbah Khan, Lillian Hella, and Nils Sandsmark. ISO 15926-14:2020(E). Working draft, READI, September 2020.

[155] Peter Maloy Koch. *A CAPABILITIES-BASED ACCOUNT OF PATIENT WELFARE*. PhD thesis, Buffalo State University, August 2016.

[156] Aljosha Köcher, Alexander Belyaev, Jesko Hermann, Jürgen Bock, Kristof Meixner, Magnus Volkmann, Michael Winter, Patrick Zimmermann, Stephan Grimm, and Christian Diedrich. A Reference Model for Common Understanding of Capabilities and Skills in Manufacturing, September 2022.

[157] Aljosha Kocher, Constantin Hildebrandt, Luis Miguel Vieira da Silva, and Alexander Fay. A Formal Capability and Skill Model for Use in Plug and Produce Scenarios. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1663–1670, Vienna, Austria, September 2020. IEEE.

[158] Yusuke Koji, Yoshinobu Kitamura, and Riichiro Mizoguchi. TOWARDS MODELING DESIGN RATIONAL OF SUPPLEMENTARY FUNCTIONS IN CONCEPTUAL DESIGN. In *Proceedings of the TMCE 2004*, page 14, 2004.

[159] Susan Toth Korda. *Using Functional Representation for Smart Simulation of Devices*. PhD thesis, The Ohio State University, 1993.

[160] Peter Krumhauer. *Rechnerunterstuetzung Für Die Konzeptphase Der Konstrktion*. PhD thesis, TU Berlin, 1974.

[161] Tolga Kurtoglu, Albert Swantner, and Matthew I. Campbell. Automating the conceptual design process: "From black box to component selection". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 24(1):49–62, February 2010.

[162] Tolga Kurtoglu and Irem Y. Tumer. A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems. *Journal of Mechanical Design*, 130(5):051401, May 2008.

[163] José Emilio Labra Gayo, Eric Prud'hommeaux, Iovka Boneva, and Dimitris Kontokostas. *Validating RDF data*. Number 16 in Synthesis lectures on the semantic web: theory and technology. Morgan & Claypool Publishers, San Rafael, Californien, 2018.

[164] Jan Eric Larsson. Diagnosis based on explicit means-end models. *Artificial Intelligence*, 80(1):29–93, January 1996.

[165] J. Lehman, S. Borgo, C. Masolo, and A. Gangemi. Causality and causation in dolce. In *Proceedings of the 3th International Conference of Formal Ontology in Information Systems (FOIS 2004)*, pages 273–284. IOS Press, 2004.

[166] J. Lehman, C. Yung, N. Tomlin, D. Conklin, and M. Stephens. Carbon nanotube-based black coatings. *Applied physics reviews*, 5(1), 2018.

[167] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, S. Richter, T. Chipman, M. Weise, and J. Wix. Industry foundation classes ifc4 official release, 2013.

[168] Frank Loebe. Abstract vs. social roles – Towards a general theoretical account of roles. *Applied Ontology*, 2:127–158, 2007.

[169] A. C. Love. Darwin's functional reasoning and homology. In M. Wheeler, editor, *150 Years of Evolution: Darwin's Impact on Contemporary Thought & Culture*, pages 49–67. SDSU Press, 2011.

[170] Jesus Hiram Lugo Calles, Vishal Ramadoss, Matteo Zoppi, G. Cannata, and Rezia Molfino. Modeling of a cable-based revolute joint using biphasic media variable stiffness actuation. In *Proceedings of the Third IEEE International Conference on Robotic Computing (IRC)*, 2019.

[171] Jonathan R. A. Maier and Georges M. Fadel. Affordance based design: A relational theory for design. *Research in Engineering Design*, 20(1):13–27, March 2009.

[172] Jonathan RA Maier and Georges M Fadel. Affordance: the fundamental concept in engineering design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 80258, pages 177–186. American Society of Mechanical Engineers, 2001.

[173] Wolfgang Malzkorn. Defining disposition concepts: A brief history of the problem. *Studies in History and Philosophy of Science Part A*, 32(2):335–353, 2001.

[174] Dan Marshall and Brian Weatherson. Intrinsic vs. Extrinsic Properties. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2018 edition, 2018.

[175] Claudio Masolo. Ontocommons report d2.7: Tro final release. Technical report, Horizon 2020, grant agreement 958371, 2023.

[176] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. DOLCE: A descriptive ontology for linguistic and cognitive engineering, 2003.

[177] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. WonderWeb Deliverable D18, 2003.

[178] Claudio Masolo, Giancarlo Guizzardi, Laure Vieu, Emanuele Bottazzi, and Roberta Ferrario. Relational Roles and Qua-individuals. page 10.

[179] Claudio Masolo, Laure Vieu, Emanuele Bottazzi, Carola Catenacci, Roberta Ferrario, Aldo Gangemi, and Nicola Guarino. Social Roles and their Descriptions. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, page 11, Whistler, Canada, June 2004.

[180] Clifford Matthews. *A Practical Guide to Engineering Failure Investigation: Matthews/A Practical Guide to Engineering Failure Investigation*. John Wiley & Sons, Ltd, Chichester, UK, January 1998.

[181] Colin McGinn. The Ontology of Energy. In *Basic Structures of Reality*. Oxford University Press, New York, 2012.

[182] Gavin Mendel-Gleason. The semantic web is dead – long live the semantic web! `https://terminusdb.com/blog/the-semantic-web-is-dead/` 2022. Accessed: March 2024.

[183] Eric C. Merrel, David G. Limbaugh, Peter M. Koch, and Barry Smith. Capabilities.

[184] Peter Mika. What happened to the semantic web? `https://ht.acm.org/ht2017/images/MikaPeterACM%20Hypertext%202017-WhatHappenedSemanticWeb.pdf` 2017. Accessed: March 2024.

[185] MIL-HDBK-2155: FAILURE REPORTING, ANALYSIS AND CORRECTIVE ACTION TAKEN, December 1995.

[186] MIL-STD-1629A: Procedures for performing a failure mode, effects and criticality analysis, 24 November1980.

[187] Ruth Garrett Millikan. *Language, thought, and other biological categories: New foundations for realism*. MIT press, 1987.

[188] Gabriel Miranda, João Almeida, Giancarlo Guizzardi, and C.L.B. Azevedo. Foundational choices in enterprise architecture: The case of capability in defense frameworks. 08 2019.

[189] Riichiro Mizoguchi. Causation: Revisited. In *JOWO*, 2020.

[190] Riichiro Mizoguchi. Causing is achieving – a solution to the problem of causation, 2023.

[191] Riichiro Mizoguchi and Stefano Borgo. The Role of the Systemic View in Foundational Ontologies. In *The Joint Ontology Workshops (JOWO)*, page 11, 2021.

[192] Riichiro Mizoguchi, Yoshinobu Kitamura, and Stefano Borgo. Towards A Unified Definition of Function. *Frontiers in Artificial Intelligence and Applications*, page 15, January 2012.

[193] Riichiro Mizoguchi, Yoshinobu Kitamura, and Stefano Borgo. A unifying definition for artifact and biological functions. *Applied Ontology*, 11(2):129–154, June 2016.

[194] Riichiro Mizoguchi, Yoshinobu Kitamura, and The Hegeler Institute. A Functional Ontology of Artifacts:. *Monist*, 92(3):387–402, 2009.

[195] Riichiro Mizoguchi and Fumiaki Toyoshima. YAMATO: Yet Another More Advanced Top-level Ontology with Analysis of Five Examples of Change. In *Proceedings of FOUST II: 2nd Workshop on Foundational Ontology*, page 13, Bozen-Bolzano, Italy, September 2017. CEUR-WS.org.

[196] Riichiro Mizoguchi and Fumiaki Toyoshima. Yamato: Yet another more advanced top-level ontology with analysis of five examples of change. *Applied Ontology*, 2017.

[197] Sergio Mota. Dispensing with the theory (and philosophy) of affordances. *Theory & Psychology*, 31(4):533–551, August 2021.

[198] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), feb 2009.

[199] Fabian Neuhaus. On the Definition of 'Ontology'. In *Proceedings of The Joint Ontology Workshops (JOWO)*, 2017.

[200] P. Oliveira, F. Rodrigues, P. Henriques, and H. Galhardas. A Taxonomy of Data Quality Problems. In *2nd Int. Workshop on Data and Information Quality,*, 2005.

[201] Jack E. Olson. *Data quality: the accuracy dimension.* Morgan Kaufmann, San Francisco, 2003.

[202] Lassila Ora. Semantic web and ai: Can we finally realize the vision?

[203] G. Pahl and W. Beitz. *Konstruktionslehre: Handbuch Für Studium Und Praxis.* Springer Verlag, Berlin, Heidelberg, 1977.

[204] G. Pahl, W. Beitz, J. Feldhusen, and K.H. Grote. *Engineering Design: A Systematic Approach.* Springer, London, 3rd ed edition, 2007.

[205] G. Pahl, Ken Wallace, Luciënne Blessing, and G. Pahl, editors. *Engineering design: a systematic approach*. Springer, London, 3rd ed edition, 2007.

[206] Sean B. Palmer. What happened to the semantic web? `https://lists.w3.org/Archives/Public/public-lod/2017Oct/0003.html` 2017. Accessed: March 2024.

[207] Mohammad Farhad Peerally, Susan Carr, Justin Waring, and Mary Dixon-Woods. The problem with root cause analysis. *BMJ quality & safety*, 2016.

[208] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):7–34, 2014.

[209] Lena Qian and John S. Gero. Function–behavior–structure paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10(4):289–312, September 1996.

[210] Marvin Rausand, Anne Barros, and Arnljot Høyland. *System Reliability Theory: Models, Statistical Methods, and Applications*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc, Hoboken, NJ, third edition edition, 2021.

[211] Marvin Rausand and Knut Øien. The basic concepts of failure analysis. *Reliability Engineering & System Safety*, 53(1):73–83, July 1996.

[212] ReCaM — Rapid Reconfiguration of Flexible Production Systems through Capabilitybased Adaptation, Autoconfiguration and Integrated tools for Production Planning, D3.1: Formal capability model and resource description, October 2016.

[213] Failure Mode/Mechanism Distributions 1991, 1991.

[214] Randy Riddell. *Practical Root Cause Failure Analysis: Key Elements, Case Studies, and Common Equipment Failures*. CRC Press, Boca Raton, 1 edition, May 2022.

[215] Jonathan Rochkind. `https://bibwild.wordpress.com/2014/10/28/is-the-semantic-web-still-a-thing/` 2014. Accessed: March 2024.

[216] Wolf Georg Rodenacker. *Methodisches Konstruieren: Grundlagen, Methodik, Praktische Beispiele*. Springer, Berlin, 1970.

[217] Fabrício Rodrigues and Mara Abel. What to consider about events: A survey on the ontology of occurrents. *Applied Ontology*, 14:1–36, 08 2019.

[218] Johannes Röhl and Ludger Jansen. Why functions are not special dispositions: An improved classification of realizables for top-level ontologies. *Journal of Biomedical Semantics*, 5(1):27, 2014.

[219] Douglas T. Ross and Kenneth E. Schoman. Structured analysis for requirements definition. *IEEE Transactions on Software Engineering*, SE-3:6–15, 1977.

[220] Karlheinz Roth. *Konstruieren Mit Konstruktionskatalogen*. Springer-Verlag, New York, 1982.

[221] Emilio M. Sanfilippo, Walter Terkaj, and Stefano Borgo. Ontological modeling of manufacturing resources. *Applied Ontology*, 16(1):87–109, January 2021.

[222] Arkopaul Sarkar and Dušan Šormaz. Ontology Model for Process Level Capabilities of Manufacturing Resources. *Procedia Manufacturing*, 39:1889–1898, 2019.

[223] Munehiko Sasajima, Yoshinobu Kitamura, Mitsuru Ikeda, and Riichiro Mizoguchi. FBRL: A function and behavior representation language. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, volume 2, Montreal, Quebec, Canada, August 1995.

[224] Munehiko Sasajima, Yoshinobu Kitamurat, Mitsuru Ikeda, Shinji Yoshikawa, Akira Endou, and Riichiro Mizoguchi. An Investigation on Domain Ontology to Represent Functional Models. *Proceedings of Eighth International Workshop on Qualitative Reasoning about Physical Systems (QR 94)*, page 10, 1994.

[225] Selskapet for Industriell og Teknisk Forskning, editor. *OREDA: Offshore Reliability Data Handbook*. Norske Veritas, Høvik, 3. ed edition, 1997.

[226] V Sembugamoorthy and B Chandrasekaran. Functional representation of devices and compilation of diagnostic problem-solving systems. *Experience, memory and Reasoning*, pages 47–73, 1986.

[227] Juan Sequeda and Ora Lassila. *Designing and Building Enterprise Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Springer International Publishing, Cham, 2021.

[228] Olivier Serrat. *The Five Whys Technique*, pages 307–310. Springer Singapore, Singapore, 2017.

[229] Hela Sfar, Anja Habacha Chaibi, Amel Bouzeghoub, and Henda Ben Ghezala. Gold standard based evaluation of ontology learning techniques. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 339–346, Pisa Italy, April 2016. ACM.

[230] Scott A Shappell and Douglas A Wiegmann. The human factors analysis and classification system–hfacs. Technical report, FAA Civil Aeromedical Institute, 2000.

[231] Lorenzo Solano, Pedro Rosado, and Fernando Romero. Knowledge representation for product and processes development planning in collaborative environments. *International Journal of Computer Integrated Manufacturing*, 27(8):787–801, August 2014.

[232] D. Andrew Spear, Ceuster Werner, and Smith Barry. Functions in Basic Formal Ontology. *Applied Ontology*, 11(2):103–128, June 2016.

[233] Prasanna Sridharan and Matthew Campbell. A grammar for function structures. *Proceedings of the ASME Design Engineering Technical Conference*, 3, 01 2004.

[234] Prasanna Sridharan and Matthew I. Campbell. A study on the grammatical construction of function structures. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 19(3):139–160, August 2005.

[235] Robert B. Stone and Kristin L. Wood. Development of a Functional Basis for Design. *Journal of Mechanical Design*, 122(4):359–370, December 2000.

[236] Michael Stonebraker. What goes around comes around. In *Readings in Database Systems*. Mit press, 2005.

[237] Rudi Studer, V.Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161–197, March 1998.

[238] Martin Sudmanns, Dirk Tiede, Stefan Lang, and Andrea Baraldi. Semantic and syntactic interoperability in online processing of big Earth observation data. *International Journal of Digital Earth*, 11(1):95–112, January 2018.

[239] Walter Terkaj, Stefano Borgo, and Emilio M Sanfilippo. Ontology for Industrial Engineering: A DOLCE Compliant Approach. In *FOMI 2022: 12th International Workshop on Formal Ontologies Meet Industry*, volume 3240 of *CEUR Workshop Proceedings*, page 13, Tarbes, Fran, 2022.

[240] Walter Terkaj, Tullio Tolio, and Anna Valente. *Design of Focused Flexibility Manufacturing Systems (FFMSs)*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[241] Fumiaki Toyoshima, Adrien Barton, and Jean-François Ethier. Affordances and their ontological core. *Applied ontology*, 17(2):285–320, 2022.

[242] Fumiaki Toyoshima, Adrien Barton, and Jean-François Ethier. Functional parthood: A dispositional perspective. In *Workshop on Foundational Ontology (FOUST IV). The Joint Ontology Workshops 2020 (JOWO 2020)*, number 2708. CEUR Workshop proceedings, 2020.

[243] Fumiaki Toyoshima, Adrien Barton, and Jean-François Ethier. Investigating functions in bfo from the viewpoint of extrinsic dispositions. In *6th Workshop on Foundational Ontology (FOUST 2022)@ The 8th Joint Ontology Workshops (JOWO 2022)*, pages paper–5. CEUR-WS. org, 2022.

[244] Fumiaki Toyoshima, Riichiro Mizoguchi, and Mitsuru Ikeda. Causation: A functional perspective. *Applied Ontology*, 14(1):43–78, February 2019.

[245] Irem Y Tumer, Robert B Stone, and David G Bell. REQUIREMENTS FOR A FAILURE MODE TAXONOMY FOR USE IN CONCEPTUAL DESIGN. In *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN*, page 11, Stockholm, August 2003.

[246] Yasushi Umeda, H. Takeda, T. Tomiyama, and H. Yoshikawa. Function, behaviour, and structure. *Applications of artificial intelligence in engineering V*, 1(Design):177–193, 1990.

[247] Marcello Urgo and Walter Terkaj. Formal modelling of release control policies as a plug-in for performance evaluation of manufacturing systems. *CIRP Annals*, 69(1):377–380, 2020.

[248] Michael Uschold. Ontology and database schema: What's the difference? *Applied Ontology*, 10(3-4):243–258, December 2015.

[249] Michael Uschold. *Demystifying OWL for the Enterprise*. Synthesis Lectures on Data, Semantics, and Knowledge. Springer International Publishing, Cham, 2018.

[250] Michaël Verdonck. *Ontology-driven conceptual modeling: Model comprehension, ontology selection, and method complexity*. PhD thesis, Ghent University, 2018.

[251] Michaël Verdonck and Frederik Gailly. Insights on the use and application of ontology and conceptual modeling languages in ontology-driven conceptual modeling. pages 83–97, 11 2016.

[252] Verein Deutscher Ingenieure. VDI 2860: Assembly and handling; handling functions, handling units; terminology, definitions and symbols. Technical report, Verein Deutscher Ingenieure, 1990.

[253] Pieter Vermaas, Dingmar Eck, and Peter Kroes. The Conceptual Elusiveness of Engineering Functions:. *Philosophy & Technology*, 26, June 2012.

[254] Pieter Vermaas and Pawel Garbacz. Functional Decomposition and Mereology in Engineering. In *Philosophy of Technology and Engineering Sciences*, pages 235–271. Elsevier, 2009.

[255] Pieter Vermaas and Pawel Garbacz. Functional Decomposition and Mereology in Engineering. In *Philosophy of Technology and Engineering Sciences*, pages 235–271. Elsevier, 2009.

[256] Pieter E. Vermaas. On the formal impossibility of analysing subfunctions as parts of functions in design methodology. *Research in Engineering Design*, 24(1):19–32, January 2013.

[257] Pieter E. Vermaas and Kees Dorst. On the conceptual framework of John Gero's FBS-model and the prescriptive aims of design methodology. *Design Studies*, 28(2):133–157, March 2007.

[258] Pieter E. Vermaas and Wybo Houkes. Ascribing Functions to Technical Artefacts: A Challenge to Etiological Accounts of Functions. *The British Journal for the Philosophy of Science*, 54(2):261–289, June 2003.

[259] Marcos Vescovi, Yumi Iwasaki, and Richard Fikes. CFRL: A Language for Specifying the Causal Functionality of Engineered evices. In *AAAI-93 Proceedings*, page 8, 1994.

[260] W. E. Vesely, F. Goldberg, N. H. Roberts, and D. Haasl. *Fault Tree Handbook*. U.S. Nuclear Regulatory Commission, 1987.

[261] Laure Vieu, Stefano Borgo, and Claudio Masolo. Artefacts and Roles: Modelling Strategies in a Multiplicative Ontology. In *Frontiers in Artificial Intelligence and Applications*, page 15, January 2008.

[262] Carl Friedrich Weizsäcker von. *Die Einheit Der Natur*. Carl Hanser Verlag, München, 1971.

[263] Matthew West. *Developing High Quality Data Models*. Morgan Kaufmann, Burlington, MA, 2011.

[264] Jeff White. `https://www.forbes.com/sites/forbestechcouncil/2023/04/17/why-high-quality-and-relevant-data-is-essential-in-todays-business-landscape/` 2023. Accessed: March 2024.

[265] Caitlin Woods, Matt Selway, Melinda Hodkiewicz, Farhad Ameri, Markus Stumptner, and William Sobel. On the Notion of Maintenance State for Industrial Assets. In *CEUR Workshop Proceedings, Joint Ontology Workshops 2021 Episode VII: The Bolzano Summer of Knowledge, JOWO 2021*, volume 2969, page 14, 2021.

[266] Seung-Cheol Yang, Lalit Patil, and Debasish Dutta. Function Semantic Representation (FSR): A Rule-Based Ontology for Product Functions. *Journal of Computing and Information Science in Engineering*, 10(3):031001, September 2010.

[267] A. Young. *Der Kreative Kosmos*. Kösel-Verlag, München, 1987.

[268] Yahia Zare Mehrjerdi. Quality function deployment and its extensions. *International Journal of Quality & Reliability Management*, 27(6):616–640, 2010.

[269] Xudong Zhang, Tiange Zhen, Jing Zhang, Yujin Wang, and Song Liu. SRCB at SemEval-2023 task 1: Prompt based and cross-modal retrieval enhanced visual word sense disambiguation. In Atul Kr. Ojha, A. Seza Doğruöz, Giovanni Da San Martino, Harish Tayyar Madabushi, Ritesh Kumar, and Elisa Sartori, editors, *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 439–446, Toronto, Canada, July 2023. Association for Computational Linguistics.

[270] Meng Zhao, Yong Chen, Linfeng Chen, and Youbai Xie. A state–behavior–function model for functional modeling of multi-state systems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233(7):2302–2317, April 2019.

# Appendix A

# Collected Axioms

In this appendix we copy all the axioms of the framework developed in this thesis, to have them collected in the same location. This set of axioms is intended to be a middle-level ontology extending DOLCE (therefore formulas belonging to DOLCE are not reported), with focus on technical functions and malfunctions (plus a rephrasing of Toyoshima's causal ontology). Note that formulas belonging to the 'base' DOLCE theory, formulas belonging to theories of other authors that were considered only for sake of mapping and discussion, and OWL formulas are not reported here. Also note that consistency of the theory is trivial, since the theory (including DOLCE) can be empty. Stronger consistency results where some predicates are instantiated were obtained by explicitly showing models for the OWL versions of some portions[1] of the whole theory, as was documented in chapters 3 and 4 and on Github[2][3].

High-level DOLCE extension.

**df9** $\texttt{externalTo}(x, y) \iff \neg(\texttt{qt}(x, y) \vee \texttt{qt}(y, x) \vee \exists t(\texttt{K}(x, y, t) \vee \texttt{K}(y, x, t) \vee \texttt{O}(x, y, t)))$

**df11** $\texttt{instantiationFoundedOn}(x, y) \iff (\texttt{CN}(x) \wedge \texttt{CN}(y) \wedge \exists z, t(\texttt{CL}(z, x, t)) \wedge \forall z, t(\texttt{CL}(z, x, t) \implies \exists w(\texttt{externalTo}(z, w) \wedge \texttt{CL}(w, y, t))))$

**ax1** $\texttt{CL}(x, y, t) \implies (\texttt{ED}(x) \vee \texttt{PD}(x))$

**ax2** $\texttt{CL}(x, y) \implies \texttt{PD}(x) \wedge \forall t(\texttt{PRE}(x, t) \implies \texttt{CL}(x, y, t))$

**ax3** $\texttt{RL}(x) \implies \exists y\ \texttt{defFoundedOn}(x, y)$

**df13** $\texttt{subConceptOf}(x, y) \iff (\texttt{CN}(x) \wedge \texttt{CN}(y) \wedge \exists z, t(\texttt{CL}(z, x, t)) \wedge \forall z, t(\texttt{CL}(z, x, t) \implies \texttt{CL}(z, y, t)))$

[1] In any case, the consistency of the whole FOL theory with all predicates instantiated is almost self-evident (most of the theory is a definitional extension of DOLCE, which is consistent). However, the most direct way to prove this would be to explicitly show a model, and, since the model would have to instantiate all the predicates it would be exceedingly big (thousands upon thousands of assertions, depending on how they are written); additionally, one would need to check that it actually satisfies the hundreds of axioms of DOLCE plus the theory of this appendix. For these reasons we do not carry out such effort.

[2] `https://github.com/kataph/MALFO`.

[3] `https://github.com/kataph/function-method-ontology.git`.

**ax4** $\texttt{RelationalQuality}(x) \implies \texttt{Q}(x)$

**df14** $\texttt{IntrinsicQuality}(x) \iff (\texttt{Q}(x) \wedge \neg\texttt{RelationalQuality}(x))$

**ax5** $\texttt{TechArtifact}(x) \implies \texttt{POB}(x)$

DOLCE extention with high-level function-related concepts.

**ax7** $\texttt{participateAsDoer}(x, y, t) \implies (\texttt{TechArtifact}(x) \vee \texttt{Agent}(x)) \wedge \texttt{Behavior}(y) \wedge \texttt{PC}(x, y, t)$

**ax8** $\texttt{participateAsFlow}(x, y, t) \implies \texttt{Behavior}(y) \wedge \texttt{PC}(x, y, t)$

**ax9** $\texttt{Behavior}(x) \implies \exists y, z, t(\texttt{participateAsDoer}(y, x, t) \wedge \texttt{participateAsFlow}(z, x, t) \wedge y \neq z)$

**ax10** $\texttt{Goal}(x) \implies \texttt{CN}(x) \wedge \forall y, t(\texttt{CL}(y, x, t) \rightarrow \texttt{STV}(y))$

Systemic functions, ontological functions.

**df16** $\texttt{systemicFunctionOf}(x, y) \iff (\texttt{RL}(x) \wedge \exists z, g, b, t(\texttt{System}(z) \wedge \texttt{goalOf}(g, z) \wedge \texttt{CL}(b, x, t) \wedge \texttt{P}(y, z, t) \wedge \forall b', t'(\texttt{CL}(b', x, t') \implies (\texttt{Behavior}(b') \wedge \texttt{participateAsDoer}(y, b', t') \wedge \texttt{P}(y, z, t') \wedge \texttt{posCausalCon}(b', g)))))$

**df17** $\texttt{SystemicFunction}(x) \iff \exists y\ \texttt{systemicFunctionOf}(x, y)$

**ax11** $\texttt{SystemicFunction}(x) \implies \exists y(\texttt{System}(y) \wedge \texttt{defFoundedOn}(x, y))$

**ax12** $\texttt{OntologicalFunction}(x) \implies (\exists y(\texttt{subConceptOf}(y, x) \wedge \texttt{SystemicFunction}(y)) \wedge \forall y(\texttt{subConceptOf}(y, x) \wedge \neg\texttt{OntologicalFunction}(y) \implies \texttt{SystemicFunction}(y)))$

Engineering methods and functional decomposition.

**ax13** $\texttt{EngMethod}(x) \implies \texttt{NASO}(x)$

**ax14** $(\texttt{MainFunction}(x) \vee \texttt{SubFunction}(x)) \implies (\texttt{RL}(x) \wedge \exists \texttt{m}\ (\texttt{EngMethod}(\texttt{m}) \wedge \texttt{instantiationFoundedOn}(x, \texttt{m})))$

**ax15** $(\texttt{MainFunction}(x) \vee \texttt{SubFunction}(x)) \implies \exists y(\texttt{OntologicalFunction}(y) \wedge \texttt{subConceptOf}(x, y))$

**ax16** $\texttt{EngMethod}(\texttt{m}) \implies \exists! n\ \texttt{engMethodNum}(\texttt{m}, n),$

**ax17** $\texttt{engMethodNum}(\texttt{m}, n) \iff \exists! \texttt{main}, \texttt{sub}_1, \ldots, \texttt{sub}_n(\texttt{MainFunction}(\texttt{main}) \wedge \texttt{SubFunction}(\texttt{sub}_1) \wedge \ldots \wedge \texttt{SubFunction}(\texttt{sub}_n) \wedge \texttt{instantiationFoundedOn}(\texttt{main}, \texttt{m}) \wedge \texttt{instantiationFoundedOn}(\texttt{sub}_1, \texttt{m}) \wedge \ldots \wedge \texttt{instantiationFoundedOn}(\texttt{sub}_n, \texttt{m}))$

**df24** $\texttt{decom}(\texttt{f}; \texttt{f}_1, \texttt{f}_2, \ldots, \texttt{f}_n) \iff (\texttt{SystemicFunction}(\texttt{f}) \wedge \texttt{SystemicFunction}(\texttt{f}_1) \wedge \ldots \wedge \texttt{SystemicFunction}(\texttt{f}_n) \wedge \exists \texttt{m}(\texttt{EngMethod}(\texttt{m}) \wedge \texttt{subConceptOf}(\texttt{f}, \texttt{main}^{\texttt{m}}) \wedge \texttt{subConceptOf}(\texttt{f}_1, \texttt{sub}_1^{\texttt{m}}) \wedge \ldots \wedge \texttt{subConceptOf}(\texttt{f}_n, \texttt{sub}_n^{\texttt{m}})))$

**df25** $\texttt{SolvedFunction}(x) \Longleftrightarrow \texttt{MainFunction}(x)$

Capacities and capabilities

**ax18** $\texttt{Capacity}(x) \implies \texttt{RelationalQuality}(x) \wedge$
$\exists y(\texttt{SD}(x,y) \wedge \texttt{IntrinsicQuality}(y) \wedge (\texttt{CP}(\texttt{topBearerOf}(y), \texttt{topBearerOf}(x)) \vee \texttt{CK}(\texttt{topBearerOf}($

**ax19** $\texttt{Capacity}(x) \implies \exists y(\texttt{Capability}(y) \wedge \texttt{SD}(y,x))$

**df26** $\texttt{topBearerOf}(x) = \imath\, y(\texttt{qt}(x,y) \wedge (\texttt{PD}(y) \vee \texttt{ED}(y)))$

**ax20** $\texttt{Capability}(x) \implies \texttt{RelationalQuality}(x) \wedge$
$\exists y(\texttt{OntologicalFunction}(y) \wedge \texttt{defFoundedOn}(x,y))$

**ax21** $\texttt{Capability}(x) \implies \exists y(\texttt{Capacity}(y) \wedge \texttt{SD}(x,y))$

**ax22** $\texttt{Capability}(x) \implies \exists y(\texttt{EngMethod}(y) \wedge \texttt{defFoundedOn}(x,y))$

Relations that relates functions to functions or other entities. Note that there is some overloading of symbols depending on the type of their arguments.

**df27** $\texttt{hasFunction}(x,f) \Longleftrightarrow \exists a,q \texttt{intentionalSel}(a,e,x,q) \wedge \texttt{defFoundedOn}(q,f)$
$\wedge\, \texttt{Capability}(q) \wedge \texttt{qt}(q,x)$

**df28** $\texttt{SelectedFunction}(f) \Longleftrightarrow \exists x(\texttt{hasFunction}(x,f))$

**df29** $\texttt{systemicFunctionOf}(\texttt{f},x) \wedge \texttt{hasFunction}(x,\texttt{f'}) \implies \texttt{subConceptOf}(\texttt{f},\texttt{f'})$

**df30** $\texttt{precedes}(x,y) \Longleftrightarrow \exists t,t'(\texttt{ql}(t,x)_T \wedge \texttt{ql}(t',y)_T \wedge t \leq t')$

**df31** $\texttt{follows}(x,y) \Longleftrightarrow \texttt{precedes}(y,x)$

**ax23** $\texttt{CL}(x,\texttt{main}^{\texttt{m}}) \implies \exists y(\texttt{CL}(y,\texttt{m}) \wedge x \approx_{ST} y)$

**df32** $\texttt{precedes}(\Phi,\Psi) \Longleftrightarrow \forall x(\Psi(x) \implies \exists y(\Phi(y) \wedge \texttt{precedes}(y,x)))$

**df33** $\texttt{follows}(\Phi,\Psi) \Longleftrightarrow \forall x(\Psi(x) \implies \exists y(\Phi(y) \wedge \texttt{follows}(y,x)))$

**df34** $\texttt{precedes}(\texttt{c},\texttt{d}) \Longleftrightarrow \forall x(\texttt{CL}(x,\texttt{d}) \implies \exists y(\texttt{CL}(y,\texttt{c}) \wedge \texttt{precedes}(y,x)))$

**df35** $\texttt{follows}(\texttt{c},\texttt{d}) \Longleftrightarrow \forall x(\texttt{CL}(x,\texttt{d}) \implies \exists y(\texttt{CL}(y,\texttt{c}) \wedge \texttt{follows}(y,x)))$

Mappings of Toyoshima's ontology occurrent categories with DOLCE perdurant categories.

**mp3** $\texttt{State} \equiv \texttt{S}$

**mp4** $\texttt{Process} \equiv \texttt{PRO}$

**mp5** $\texttt{Event} \equiv \texttt{E}$

**mp6** $\texttt{Occurrent} \equiv \texttt{PD}$

Toyoshima's ontology.

**ax28** $\texttt{State}(x) \vee \texttt{Process}(x) \vee \texttt{Event}(x) \implies \texttt{Occurrent}(x)$

**ax29** $\texttt{State}(x) \implies \neg(\texttt{Process}(x) \vee \texttt{Event}(x)) \quad \wedge$
$\texttt{Process}(x) \implies \neg\texttt{Event}(x)$

**ax30** $\texttt{causeOf}(x, y) \implies \texttt{Occurrent}(x) \wedge \texttt{Occurrent}(y)$

**ax31** $\neg\texttt{causeOf}(x, x) \quad \wedge$
$\texttt{causeOf}(x, y) \implies \neg\texttt{causeOf}(y, x)$

**ax32** $\texttt{causeOf}(x, y) \iff \texttt{achieves}(x, y) \vee \texttt{prevents}(x, y) \vee \texttt{allows}(x, y)$
$\vee\, \texttt{disallows}(x, y)$

**df44** $\texttt{posCausalCon}(x, y) \coloneqq \texttt{achieves}(x, y) \vee \texttt{allows}(x, y) \vee \texttt{facilPrecondFor}(x, y)$

**ax33** $\texttt{achieves}(x, y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x) \vee \texttt{State}(x)) \wedge (\texttt{Process}(y) \vee \texttt{State}(y)) \wedge$
$\neg(\texttt{State}(x) \wedge \texttt{Process}(y))$

**df45** $\texttt{prevents}(x, y) \coloneqq \exists z(\texttt{State}(z) \wedge \texttt{incompatibleWith}(z, y) \wedge \texttt{achieves}(x, z))$

**ax34** $\texttt{prevents}(x, y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x) \vee \texttt{State}(x)) \wedge (\texttt{Process}(y) \vee \texttt{State}(y)) \wedge$
$\neg(\texttt{State}(x) \wedge \texttt{Process}(y))$

**ax35** $\texttt{facilPrecondFor}(z, y) \implies (\texttt{State}(z) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y)))$

**ax36** $\texttt{prevPrecondFor}(z, y) \implies (\texttt{State}(z) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y)))$

**ax37** $\texttt{incompatibleWith}(z, w) \wedge \texttt{prevPrecondFor}(w, y) \implies$
$\texttt{facilPrecondFor}(z, y)$

**ax38** $\texttt{incompatibleWith}(z, w) \wedge \texttt{facilPrecondFor}(w, y) \implies$
$\texttt{prevPrecondFor}(z, y)$

**df46** $\texttt{allows}(x, y) \coloneqq \exists z(((\texttt{achieves}(x, z) \vee \texttt{maintains}(x, z)) \wedge$
$\texttt{facilPrecondFor}(z, y)) \vee (\texttt{prevents}(x, z) \wedge$
$\texttt{prevPrecondFor}(z, y) \wedge$
$\neg\exists w(\texttt{State}(w) \wedge \texttt{achieves}(x, w) \wedge \texttt{prevPrecondFor}(w, y)))$

**ax39** $\texttt{allows}(x, y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x)) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y))$

**df47** $\texttt{disallows}(x, y) \coloneqq \exists z(((\texttt{achieves}(x, z) \vee \texttt{maintains}(x, z)) \wedge$
$\texttt{prevPrecondFor}(z, y)) \vee$
$(\texttt{prevents}(x, z) \wedge \texttt{facilPrecondFor}(z, y)))$

**ax40** $\texttt{disallows}(x, y) \implies (\texttt{Event}(x) \vee \texttt{Process}(x)) \wedge (\texttt{Event}(y) \vee \texttt{Process}(y) \vee \texttt{State}(y))$

**ax41** $\texttt{internalTo}(x, y) \iff x \subseteq_{ST} y$

**df48** $\texttt{internalTo}(x, y) \wedge \texttt{Occurrent}(x) \wedge \texttt{Occurrent}(y) \iff$
$\forall z, t(\texttt{participatesIn}(z, x, t) \implies$
$\exists w(\texttt{participatesIn}(w, y, t) \wedge \texttt{partOf}(z, w, t) \wedge z \neq w))$

**df49** $\texttt{internalTo}(x, y) \wedge \texttt{Occurrent}(x) \wedge \texttt{Object}(y) \iff$
$\forall z, t(\texttt{participatesIn}(z, x, t) \implies \texttt{partOf}(z, y, t) \wedge z \neq y))$

**ax42** $\texttt{physicalConseqOf}(x, y) \implies \texttt{Occurrent}(x) \wedge \texttt{Occurrent}(y)$

**ax43** $\texttt{physicalConseqOf}(x, y) \wedge \texttt{physicalConseqOf}(y, z) \implies$
$\texttt{physicalConseqOf}(x, z)$

**ax44** $\texttt{Pred}(x, y) \wedge \texttt{physicalConseqOf}(y, z) \implies \texttt{Pred}(x, z)$

**ax45** $\texttt{Pred}(x, z) \wedge \texttt{physicalConseqOf}(y, z) \implies \texttt{Pred}(x, y)$

Link between MALFO and functional theory

**df50** $\texttt{functionIncompatible}(x, f) \coloneqq \exists o, b, t(\texttt{participatesIn}(o, x, t) \wedge \texttt{systemicFunctionOf}(f, o) \wedge \texttt{CF}(b, f, t) \wedge \texttt{incompatibleWith}(x, b))$

MALFO Axioms

**df51** $\texttt{Malfunction}(x) \coloneqq \exists f\texttt{functionIncompatible}(x, f)$

**df52** $\texttt{FailureMechanism}(x) \coloneqq ((\texttt{Process}(x) \vee \texttt{Event}(x)) \wedge (\neg\texttt{Malfunction}(x)) \wedge \exists y(\texttt{Failure}(y) \wedge \texttt{posCausalCon}(x, y)^{\dagger}))$

**df53** $\texttt{FailureCondition}(x) \coloneqq (\texttt{State}(x) \wedge (\neg\texttt{Malfunction}(x)) \wedge \exists y(\texttt{Failure}(y) \wedge (\texttt{posCausalCon}(x, y)^{\dagger} \vee \exists z(\texttt{prevPrecondFor}(x, z) \wedge \texttt{disallows}(z, y)))))$

**df54** $\texttt{MereSymptom}(x) \coloneqq \neg\texttt{Malfunction}(x) \wedge \neg\texttt{FailureMechanism}(x) \wedge \neg\texttt{FailureCondition}(x) \wedge \exists z(\texttt{Malfunction}(z) \wedge \texttt{causeOf}(z, x))$

**df55** $\texttt{Fault}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{State}(x) \wedge \exists z(\texttt{internalTo}(z, x) \wedge \texttt{achieves}(z, x))$

**df56** $\texttt{AbExtDisabledSt}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{State}(x) \wedge \neg\texttt{Fault}(x)$

**df57** $\texttt{Failure}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{Event}(x) \wedge \exists y(\texttt{Fault}(y) \wedge \texttt{achieves}(x, y))$

**df58** $\texttt{NonPerformanceEvent}(x) \coloneqq \texttt{Malfunction}(x) \wedge \texttt{Event}(x) \wedge \neg\texttt{Failure}(x)$

**df59** $\texttt{MalfunctionProcess}(x) \coloneqq (\texttt{Malfunction}(x) \wedge \texttt{Process}(x))$

Debatable or informal axioms, or axioms given for the sake of exemplifications, which we did not commit to. Note that some of these axioms are couples of alternatives.

**df18** $\texttt{Connect}(x) \iff (\texttt{CL}(y, x) \iff \texttt{E}(y) \wedge \exists a, b, c, s_i, s_f, t_i, t_f\ (\texttt{sState}(y, s_i, t_i) \wedge \texttt{eState}(y, s_f, t_f) \wedge \texttt{Goal}(s_f) \wedge \texttt{POB}(a) \wedge \texttt{POB}(b) \wedge \texttt{POB}(c) \wedge \texttt{PC}(a, s_i, t_i) \wedge \texttt{PC}(a, s_f, t_f) \wedge \texttt{PC}(b, s_i, t_i) \wedge \texttt{PC}(b, s_f, t_f) \wedge \texttt{PC}(c, s_f, t_f) \wedge \texttt{P}(a, c, t_f) \wedge \texttt{P}(b, c, t_f)) \wedge \neg\exists c'(\texttt{POB}(c') \wedge \texttt{PC}(c', s_i, t_i) \wedge \texttt{P}(a, c', t_i) \wedge \texttt{P}(b, c', t_i))))$

**df19** $\texttt{Convert}(x) \iff (\texttt{CL}(y, x) \iff \texttt{E}(y) \wedge \exists s_i, s_f, t_i, t_f, a, \Phi, \Psi\ (\texttt{sState}(y, s_i, t_i) \wedge \texttt{eState}(y, s_f, t_f) \wedge \texttt{Goal}(s_f) \wedge \Phi \in \mathcal{NR} \wedge \Psi \in \mathcal{NR} \wedge \Psi \cap \Phi = \emptyset \wedge \texttt{PC}(a, s_i, t_i) \wedge \texttt{PC}(a, s_f, t_f) \wedge \Phi(a, t_i) \wedge \Psi(a, t_f)))$

**df20** $\texttt{ChangeQualityValue}(x) \iff (\texttt{CL}(y, x) \iff \texttt{E}(y) \wedge \exists s_i, s_f, t_i, t_f, a, q, v_i, v_f\ (\texttt{sState}(y, s_i, t_i) \wedge \texttt{Goal}(s_f) \wedge \texttt{eState}(y, s_f, t_f) \wedge \texttt{qt}(q, a)[t_i] \wedge \texttt{qt}(q, a)[t_f] \wedge v_i \neq v_f \wedge \texttt{PC}(a, s_i, t_i) \wedge \texttt{PC}(a, s_f, t_f) \wedge \texttt{ql}(q, v_i, t_i) \wedge \texttt{ql}(q, v_f, t_f)))$

**df21** $\texttt{ToGroove}(x) \Longleftrightarrow (\texttt{CL}(y,x) \Longleftrightarrow \texttt{E}(y) \wedge \exists s_i, s_f, t_i, t_f, a, f$
$(\texttt{sState}(y,s_i,t_i) \wedge \texttt{Goal}(s_f) \wedge \texttt{eState}(y,s_f,t_f) \wedge \texttt{Groove}(f) \wedge \texttt{SD}(f,a) \wedge \texttt{PC}(a,s_i,t_i) \wedge$
$\texttt{PC}(a,s_f,t_f) \wedge \neg\texttt{PRE}(f,t_i) \wedge \texttt{PRE}(f,t_f)))$

**df22** $\texttt{OntologicalFunction}(x) \Longleftrightarrow (\texttt{Connect}(x) \vee \texttt{Divide}(x) \vee \texttt{Convert}(x) \vee \dots)$

**df23** "A concept is an ontological function if and only if it classifies exactly those events consisting in a transition such that the changes (or the absence thereof) between the initial and final states is characterized in terms of a formula expressed in the language of an upper ontology."

**ax23** $\texttt{P}(\texttt{f},\texttt{f}') \wedge \texttt{F}(\texttt{f}) \wedge \texttt{F}'(\texttt{f}') \implies \texttt{P}(\texttt{F},\texttt{F}')$

**ax24** $\texttt{decom}(\texttt{f};\texttt{f}_1,\texttt{f}_2) \implies \texttt{P}(\texttt{f}_1,\texttt{f}) \wedge \texttt{P}(\texttt{f}_2,\texttt{f})$

**ex16** "Capabilities are those physical qualities that (i) are relational qualities, (ii) signify the ability of their bearer to participate in a certain way in processes of a certain type, which specializes some ontological-function type, (iii) their corresponding quality space is built (in the way described above) upon the product of multiple, separable domains."

**ex17** "Capacities are those physical qualities that: (i) are relational qualities, (ii) characterize some aspect(s) of how their bearer is able to participate in a certain way in a process of a certain type, which specializes some ontological-function type, (iii) their corresponding quality space is built from a single domain."

**ex18** "Every capacity is causally dependent on some intrinsic quality."

**ex19** " Every capability is causally dependent on some capacity."

**df60** $\texttt{MalfunctionRelHapp}(x) \Longleftrightarrow (\texttt{Malfunction}(x) \vee \texttt{MereSymptom}(x) \vee$
$\texttt{FailureMechanism}(x) \vee \texttt{FailureCondition}(x))$

**df61** $\texttt{MalfunctionRelHapp}(x) \Longleftrightarrow \exists y(\texttt{functionIncompatible}(x,y) \wedge$
$((\texttt{causeOf}(x,y)^{\dagger}) \vee (\texttt{causeOf}(y,x)^{\dagger})))$