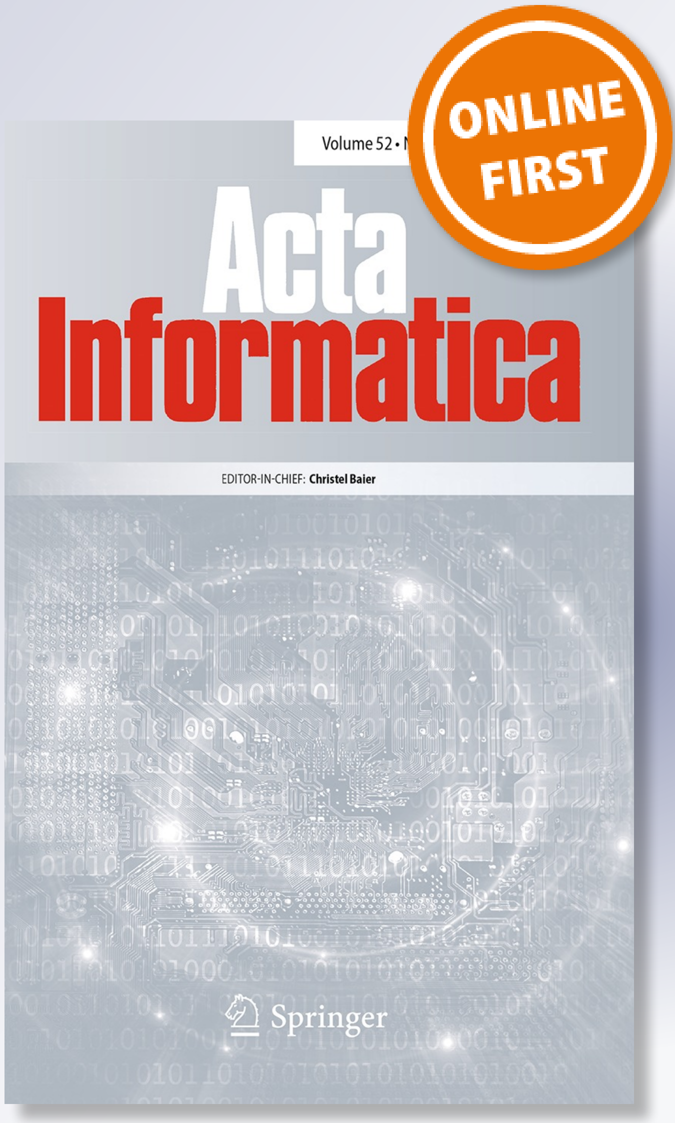
Dynamic controllability via Timed Game Automata

Alessandro Cimatti, Luke Hunsberger, Andrea Micheli, Roberto Posenato & Marco Roveri

Acta Informatica

ISSN 0001-5903

Acta Informatica
DOI 10.1007/s00236-016-0257-2



Acta Informatica
DOI 10.1007/s00236-016-0257-2

ORIGINAL ARTICLE

Dynamic controllability via Timed Game Automata

Alessandro Cimatti[1] · Luke Hunsberger[2] · Andrea Micheli[1] · Roberto Posenato[3] · Marco Roveri[1]

Received: 26 January 2015 / Accepted: 19 January 2016

Abstract Temporal networks are data structures for representing and reasoning about temporal constraints on activities. Many kinds of temporal networks have been defined in the literature, differing in their expressiveness. The simplest kinds of networks have polynomial algorithms for determining their temporal consistency or different levels of controllability, but corresponding algorithms for more expressive networks (e.g., those that include observation nodes or disjunctive constraints) have so far been unavailable. This paper introduces a new approach to determine the dynamic controllability of a very expressive class of temporal networks that accommodates observation nodes and disjunctive constraints. The approach is based on encoding the dynamic controllability problem into a reachability game for Timed Game Automata (TGAs). This is the first sound and complete approach for determining the dynamic controllability of such networks. The encoding also highlights the theoretical relationships between various kinds of temporal networks and TGAs. The new algorithms have immediate applications in the design and analysis of workflow models being developed to automate business processes, including workflows in the health-care domain.

This paper is an extended version of two earlier papers [9, 10].

✉ Andrea Micheli
amicheli@fbk.eu

Alessandro Cimatti
cimatti@fbk.eu

Luke Hunsberger
hunsberg@cs.vassar.edu

Roberto Posenato
roberto.posenato@univr.it

Marco Roveri
roveri@fbk.eu

1 Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy

2 Vassar College, 124 Raymond Ave., Box 444, Poughkeepsie, NY, USA

3 University of Verona, via le Grazie 15, 37134 Verona, Italy

Published online: 01 February 2016

Keywords Dynamic controllability · Temporal networks · Timed Game Automata

1 Introduction

Constraint-based temporal reasoning has been widely used in many different applications across many different domains. Over the years, different formalisms have been presented to address specific requirements that frequently arise in real-world applications. The most commonly used formalism is probably the Simple Temporal Network (STN), in which a set of real-valued variables, called time-points, are subject to convex, binary difference constraints [19]. Recently, a significant amount of research has focused on temporal reasoning in the presence of uncertainty. Temporal uncertainty arises, for example, in AI planning when the durations of some activities (i.e., the durations of some temporal intervals) are not controlled by the plan executor (or agent), but instead are only observed in real time as the activities complete. In such settings, the executor seeks a dynamic *strategy* for executing the controllable time-points such that all relevant constraints will necessarily be satisfied no matter how the uncertain durations turn out. To accommodate this kind of uncertainty, STNs have been augmented to include *contingent links*, where each contingent link represents an interval whose duration is bounded but uncontrollable; the resulting network is called a *Simple Temporal Network with Uncertainty* (STNU) [46].

An STNU can be viewed as a data structure for representing certain kinds of knowledge about a situation. Given some STNU, different kinds of questions can be asked. In the literature, three types of *controllability* have been identified—strong, dynamic and weak—that make different assumptions about when the durations of the contingent links become known to the executor [46].

This paper focuses on *dynamic controllability* (DC)—that is, whether there exists a strategy for executing the controllable time-points that depends only on past observations of the outcomes of uncontrollable durations, and that guarantees that all relevant constraints will be satisfied no matter how the durations of the contingent links turn out. Polynomial algorithms for checking the dynamic controllability of STNUs [25,33,34,36] and run-time algorithms for generating an execution strategy in real-time [23,24] have been presented in the literature.

Although STNUs have been successful in some domains, many other domains require a richer set of constraints and features. For example, in the health-care domain, where workflow management systems are being developed to automate medical-treatment processes, medical tests for any given patient frequently generate information in real time that can affect which treatment pathway the patient will follow [13]. The system must guarantee that any possible execution of the workflow strictly satisfies all specified temporal constraints no matter which test outcomes are observed. The Conditional Simple Temporal Network with Uncertainty (CSTNU) has been introduced to represent the temporal features of workflows, and the dynamic controllability property—which captures the temporal *safety* of workflows—has been defined for CSTNUs [26]. Although some progress has been made toward a DC-checking algorithm for arbitrary CSTNUs [15], a *sound-and-complete* DC-checking algorithm for CSTNUs has not yet been found.

Disjunctive constraints also arise in workflow management systems, for example, when two tests cannot be done simultaneously, but can be done in either order. Strong, dynamic and weak controllability have been defined for Disjunctive Temporal Networks with Uncertainty (DTNUs) [38,42], and algorithms for checking the strong and weak controllability properties for DTNUs have been proposed [11,12,38]; however, a dynamic controllability algorithm has only been presented for a subclass of DTNUs [42].

This paper makes the following contributions. First, it summarizes the characteristics of STNUs, CSTNUs and DTNUs and then it combines them within a single, unifying formalism called *Conditional Disjunctive Temporal Network with Uncertainty* (CDTNU). Second, it presents a novel approach for checking the dynamic controllability of such networks whereby any given CDTNU is translated into a Timed Game Automaton (TGA) [32] such that the dynamic controllability problem for the CDTNU is equivalent to a reachability game for the generated TGA. The reachability game is then solved using off-the-shelf software that is able to synthesize a viable execution strategy or determine that no such strategy exists [5]. This paper presents different instantiations of this approach that address the contingent links and observation time-points of CSTNUs, and the disjunctive constraints of DTNUs. The result is the first *sound-and-complete* DC-checking algorithm for temporal networks having contingent links, disjunctive constraints and observation time-points in any combination. Finally, the encoding of such networks into TGAs highlights important theoretical relationships between the different kinds of temporal reasoning frameworks and the TGA framework.

1.1 Related work

Starting from the seminal paper of Dechter et al. [19] describing the Simple Temporal Problem (STP) and the Temporal Constraint Satisfaction Problem (TCSP), researchers have explored a variety of techniques for solving temporal problems in the face of uncertainty.[1] Vidal et al. [45–48] introduced the Simple Temporal Problem under Uncertainty (STPU), and defined three different kinds of controllability: weak, dynamic and strong. The dynamic controllability (DC) problem for STNUs, which is the most relevant to real-world applications, has been widely studied, yielding a variety of DC-checking algorithms and techniques for managing the execution of DC networks [22–25,33,35,36]. Venable and colleagues have defined extensions of the STPU that include disjunction and preferences [38,39,42,43]. Cimatti et al. [11,12] focus on strong and weak controllability, while also allowing disjunctive constraints.

Tsamardinos et al. [41] introduced the Conditional Temporal Problem (CTP), an extension of the STP that includes *observation nodes*. Each observation node has a corresponding Boolean propositional variable. The execution of the observation node determines the truth value of its corresponding propositional variable. In addition, time-points can be labeled with arbitrary conjunctions of (positive or negative) propositional variables; time-points need only be executed in scenarios where the corresponding Boolean label is true. The authors presented a formal semantics for the *dynamic consistency* problem: to determine if there exists a strategy for executing the time-points in the network that guarantees that all of the constraints will be satisfied no matter how the observations turn out. They also showed how to convert the semantic constraints from the definition of dynamic consistency into a DTP, which enabled them to solve the dynamic consistency problem using an off-the-shelf DTP solver, albeit in exponential time.

Hunsberger et al. [26] defined a Conditional Simple Temporal Network with Uncertainty (CSTNU) that combines the contingent links from STNUs with the observation nodes from

[1] Some authors choose to distinguish the network data structure from the problem being solved, giving rise to parallel notations such as STN versus STP, CSTN versus CTP, STNU versus STPU, DTN versus DTP, and so on. This is useful given that one can pose a variety of problems for a given network structure (e.g., strong controllability vs. weak controllability for STNUs). This paper primarily uses the "N" notation, the major exceptions being when describing the work of other authors who have tended to use the "P" notation.

the CTP. They defined the dynamic controllability property for CSTNUs in a way that generalizes both the dynamic controllability of STNUs and the dynamic consistency of the CTP. Combi et al. [15] introduced a sound-but-not-complete DC-checking algorithm for CSTNUs based on a variety of constraint-propagation rules. Preliminary empirical results suggest that that algorithm can be practically efficient, even if its time complexity is exponential in the worst case. A complete DC-checking algorithm for CSTNUs following that approach has not yet been presented.

The work that is most closely related to ours is due to Vidal [44]. In that work, Timed Game Automata (TGAs) are used to check the dynamic controllability of a variant of STNUs called *Contingent Temporal Constraint Networks* (CTCNs). The algorithm incrementally constructs a TGA, interleaving checks for winning TGA strategies along the way. The most significant drawback is that the resulting TGA has exponential size, compared to the linear-sized TGAs generated by our approach. Moreover, the approach presented in this paper goes beyond the STNU formalism by accommodating both disjunctive constraints and observation nodes.

Orlandini and colleagues [7,37] have used TGAs to validate timeline-based plans. In that work, each plan is encoded as a TGA that includes an uncontrollable observer that plays the role of the environment. The observer checks the controllability of the plan and synthesizes a controller. Their plan-to-TGA approach deviates from the standard definition of dynamic controllability by allowing a free time-point to be scheduled *instantaneously* upon the observation of an uncontrollable execution event. In addition, their approach is limited to non-disjunctive, non-conditional temporal constraints [37].

Recently, Cheikhrouhou et al. [8] proposed the use of TGAs for analyzing temporal constraints in business processes represented as an extended version of the Business Process Model Notation (BPMN). Their proposal does not consider uncertainty and uses TGAs only for verifying a subset of possible temporal constraints—the duration of activities (i.e. the duration constraints) and the time between events (i.e. the temporal dependency constraints)—in sequential or parallel branches.

Abdeddaim et al. [1] use STNUs to represent strategies for a subclass of TGAs—the exact opposite of our approach. In their work, an executor needs to be able to solve the DC-checking problem (e.g., using an on-line algorithm) to generate a TGA strategy.

As already mentioned, many researchers have addressed the DC decision problem [33, 36] and the problem of managing the execution of DC networks using on-line reasoning algorithms [24,33]. However, none of them have addressed the problem of synthesizing directly executable strategies.

Recently, Morris [34] presented an algorithm that not only checks the dynamic controllability property for STNUs in $O(N^3)$ time, but also can be used to generate a dispatchable network from any dynamically controllable network. The dispatchable network can be executed with minimal constraint propagation using a greedy dispatcher.

1.2 Paper structure

The paper is structured as follows. Section 2 introduces the healthcare workflows that motivate this work and provides a running example for the rest of the paper. Section 3 formally presents four classes of temporal networks that accommodate different kinds of temporal uncertainty: STNU, CSTNU, DTNU and CDTNU. For each type of network, the dynamic controllability problem is addressed. Section 4 analyzes the different features of the temporal networks under analysis and for each of them presents a formal encoding of the dynamic controllability problem as a TGA reachability game. Finally, Sect. 5 discusses the features and limitations of the presented approach and highlights promising lines of research for the future.

2 Motivating example: healthcare workflows

A workflow is an abstract model for representing, coordinating and controlling complex processes. A *workflow management system* is a software suite that supports the automatic execution of workflows [21]. Although workflows are being applied to a variety of businesses, the research presented in this paper has been motivated by the use of workflows to automate medical-treatment processes in the healthcare domain [4]. The workflow technology may help to plan and manage the executions of medical-treatment processes that can be very different due the presence of many possibilities and combinations of events, even in situations having a general pattern to be followed. Moreover, planning in advance and constantly monitoring the process in an automatic way may help identifying previously unforeseen courses of development.

In a workflow management system, the management of temporal aspects is critical. The literature contains many proposals for extending workflow models to represent and manage the most important kinds of temporal constraints that arise in various domains [14,17,20]. This paper focuses on the conceptual model proposed by Combi et al. [16], where a workflow is specified by a *workflow schema*: a directed graph where nodes represent activities, and arcs represent control flows that define dependencies among activities, including constraints on the order of execution. Figure 1 illustrates a small portion of a workflow schema (or graph).

There can be two types of activities in a workflow graph: *tasks* and *connectors*. Tasks represent elementary work units to be executed by external agents (e.g., doctors); connectors represent internal activities executed by the workflow management system to coordinate the execution of tasks. In the graph, each task is represented by a box containing a name (e.g., Neurological Evaluation) and a range (e.g., [5, 10]) that constrains the task's duration during execution. Each connector is represented by a diamond that, similarly, may contain a temporal range constraining its duration. Additional information associated with a connector depends on its type—*split* or *join*—as discussed below.

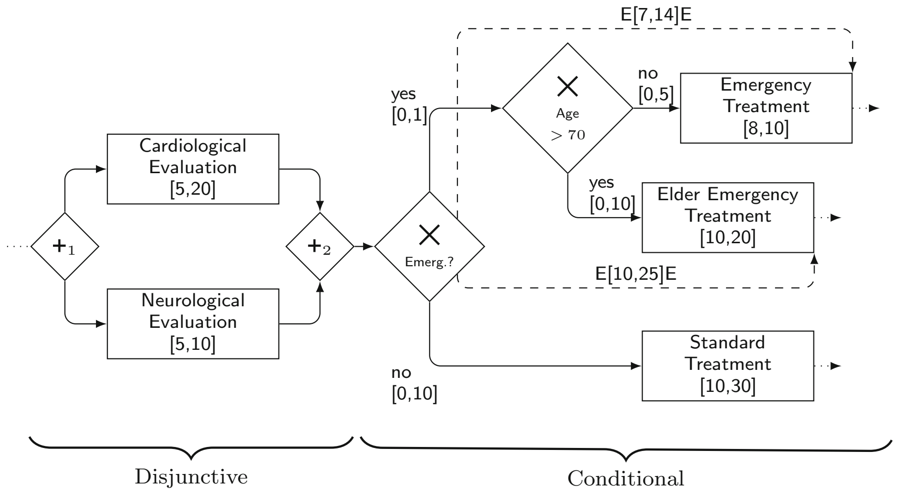

E[7,14]E
yes
[0,1]
Age
> 70
no
[0,5]
Emergency Treatment [8,10]
Cardiological Evaluation [5,20]
yes
[0,10]
Elder Emergency Treatment [10,20]
$+_1$
$+_2$
Emerg.?
E[10,25]E
Neurological Evaluation [5,10]
Standard Treatment [10,30]
no
[0,10]
Disjunctive
Conditional

Fig. 1 An excerpt of a simplified triage workflow schema. The example is composed of two parts, labeled as "Disjunctive" and "Conditional" with braces, that will be referenced in the paper

The arcs in a workflow graph are labeled directed edges that impose ordering constraints among nodes and, optionally, temporal delays. For example, an arc from a predecessor node N_1, to a successor node N_2, with the label [5,10], specifies that N_2 must start between 5 and 10 time units after the completion of N_1. A *split* connector has one incoming arc and multiple outgoing arcs. After the execution of the split connector node, one or more successor nodes must be considered for execution, depending on whether the split connector type is *parallel, alternative* or *conditional. Join* connectors are the dual of *split* connectors, having multiple incoming arcs but only one outgoing arc. Join connectors effectively close branching paths opened by split connectors.

The workflow in Fig. 1 represents a simplified *triage* process for a hospital emergency room. In the figure, all temporal ranges are in minutes; parallel connectors are identified by $+$ and conditional connectors by $\times$; and each connector is presumed to have a temporal range of [0,0]. According to this workflow, an incoming patient is first evaluated from cardiological and neurological points of view: two parallel tasks that can be done in either order, but cannot overlap. The subsequent treatment path depends on the observation of two Boolean conditions: (1) whether it is an emergency (Emerg.?); and (2) whether the patient is old (Age > 70?). In the non-emergency case, the patient is given a Standard Treatment. But for an emergency, depending on the patient's age, the patient is given either Emergency Treatment or Elder Emergency Treatment. In other words, the conditional connectors, $\times_{\text{Emerg.?}}$ and $\times_{(\text{Age}>70)}$, each split the flow into alternative pathways based on the observation of their corresponding Boolean conditions.

Finally, some activities may be subject to important timing conditions, therefore workflows also include dashed edges that represent temporal constraints. Such constraints may relate the starting or ending times of the source and target nodes in any combination. For example, the dashed edge labeled by E[7,14]E specifies that the end of the Emergency Treatment task must occur between 7 and 14 min after the end of the $\times_{\text{Emerg.?}}$ connector. The other dashed edge similarly specifies that the end of the Elder Emergency Treatment task must occur between 10 and 25 min after the end of the $\times_{\text{Emerg.?}}$ connector.

Once a workflow schema is defined, many questions can arise regarding its possible executions: is there sufficient time for executing it? Which resources are necessary for executing it?, etc. Among all, one question seems to be the fundamental one: is there a strategy for executing the possible instances of the schema that guarantees that all structural, temporal, and resource constraints will not be violated? Even considering only the temporal constraints described above, the number of possible instances of a schema can be very large, depending on both the conditional connectors (each of which splits a flow into at least two alternative flows) and the temporal durations of the tasks. Indeed, if the duration of a task cannot be fixed prior to execution, but only observed after its completion (e.g., as in the case of medical tests), then there may be many different instances of a single schema corresponding to the different possible task durations.

In order to answer the fundamental question, a careful analysis of the workflow schema is needed. In fact, we must determine whether all the possible instances can be successfully executed despite the uncertainty associated with conditional connectors and task durations.

3 The dynamic controllability of temporal networks

A workflow can be viewed as a constraint system that involves a rich variety of temporal constraints and features. Over the years, a number of formalisms have been presented in

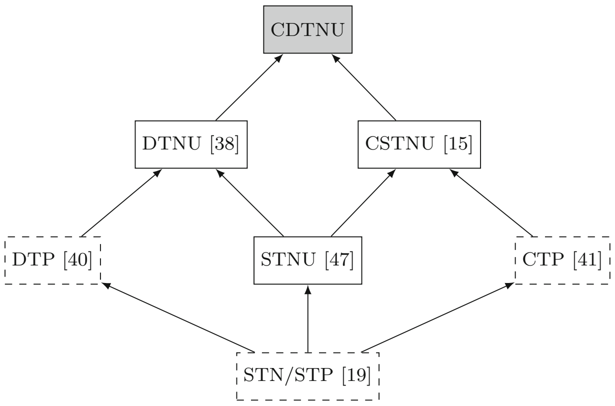


Fig. 2 An overview of the different kinds of temporal networks, with corresponding citations. *Arrows* represent expressiveness subsumption. The CDTNU box is highlighted since it constitutes an original contribution of this paper. *Dashed boxes* indicate networks that are not discussed in this paper

the literature to address different kinds of temporal information. Most of those formalisms are constraint networks in which real-valued variables called time-points are subject to various kinds of constraints. This section presents the relevant background for several kinds of temporal networks along with the corresponding dynamic controllability problems. It then introduces a new kind of network, called a Conditional Disjunctive Temporal Network with Uncertainty (CDTNU), that subsumes all of the features of the preceding networks. The CDTNU formalism enables a unified view of a substantial portion of the temporal problems that have been addressed in the literature. Figure 2 previews the temporal networks that are relevant to this paper.

Each of the temporal networks presented in this section includes a set of real-valued variables called time-points, and a set of binary difference constraints on those variables. In a Simple Temporal Network (STN), the scheduler (or agent) is presumed to control the execution of all of the time-points [19]. Thus, the most important property of an STN is whether it is *consistent* (i.e., whether there exists an assignment to the time-points that satisfies all of the constraints). Thus, the Simple Temporal Problem (STP) is a kind of constraint satisfaction problem. Against that background, this paper addresses uncertainty in temporal networks that arises from two sources.

Tasks with uncertain durations

One source of uncertainty arises when the duration of a task is not under the control of the scheduler, although that duration may have known bounds. Such tasks are represented by *contingent links* [46]. Each contingent link specifies bounds on the duration of a temporal interval between a starting time-point and an ending time-point. While the scheduler may control the execution of the starting time-point, it does not control the duration of the interval; thus, it does not control the execution of the ending time-point. To accommodate this difference, the time-points in a network with contingent links are partitioned into two classes: *free* and *uncontrollable*. Typically, the starting time-point of a contingent link is free, while the ending time-point is uncontrollable.[2] The name for a network accommodating this kind of uncertainty is typically given the suffix *with Uncertainty* [e.g., Simple Temporal Network *with Uncertainty* (STNU) or Disjunctive Temporal Network *with Uncertainty* (DTNU)].

[2] Contingent links may also form chains or trees, in which case only the starting time-point for the entire chain or tree is free, while the rest of the time-points are uncontrollable.

For networks with this kind of uncertainty, the most important property is not consistency, but *controllability.* In particular, is there a strategy for executing the free (i.e., controllable) time-points such that all of the constraints in the network will necessarily be satisfied no matter how the uncertain durations turn out? Three levels of controllability have been defined: weak, strong and dynamic [46]. They differ according to when the scheduler becomes aware of the durations of the contingent links.

A network is *strongly controllable* if there is a fixed, unconditioned, non-reactive assignment for the free time-points that will satisfy all of the constraints in the network, regardless of how the uncontrollable durations of the contingent links subsequently turn out. In effect, the scheduler must choose all execution times before learning the duration of any contingent link. Such a solution corresponds to a time-triggered program, where activities are started at fixed times that are determined in advance of execution.

In sharp contrast, a network is *weakly controllable* if there is a *strategy* that assigns values to the free time-points as a function of the uncontrollable durations of all contingent links. Although the values for the uncontrollable durations need not be known when generating the strategy, this version of controllability presumes that *all* durations are provided to the executor *in advance of execution.*

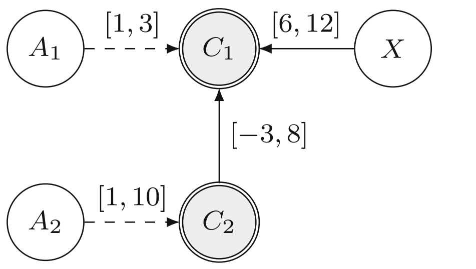
This paper focuses on *dynamic controllability,* which is widely viewed as the most relevant version of controllability for most real-world applications. A network is dynamically controllable if it has a *dynamic execution strategy* that can react, in real time, to contingent durations—but only after some positive delay. In other words, the values that the execution strategy assigns to the free time-points may depend on uncontrollable events—namely, the execution of contingent time-points—but only if that information has already been *observed* in real time. It cannot depend on advance knowledge of future uncontrollables.

Observations with uncertain outcomes

Another source of uncertainty arises from actions that generate information. For example, a doctor measuring the blood pressure of a patient only discovers whether the patient has high blood pressure after the measurement is taken. Temporal networks accommodate this kind of uncertainty by including *observation time-points* [41]. Each observation time-point has a corresponding Boolean propositional letter. The execution of an observation time-point generates a truth value for the corresponding propositional letter, in real time. Thus, an observation time-point represents a Boolean condition that can be observed at run time. For this reason, networks modeling this kind of uncertainty are called *Conditional* Temporal Networks [e.g., Conditional Simple Temporal Networks (CSTNs) or Conditional Simple Temporal Networks with Uncertainty (CSTNUs)]. In Conditional Temporal Networks, some time-points and constraints may be applicable only in certain scenarios (e.g., if a patient has high blood pressure and is in critical condition).

For Conditional Temporal Networks, the most important property has been called *dynamic consistency* [41]. Intuitively, such a network is dynamically consistent if there exists a dynamic strategy for executing its time-points such that all relevant constraints will be satisfied no matter which combination of outcomes is observed in real time. The execution decisions made by such a strategy may depend on the outcomes of observation time-points that have occurred in the past, but not on advance knowledge of such outcomes.

Some of the networks in this section accommodate both kinds of uncertainty described above (i.e., contingent links and observation time-points). For such networks, the property of dynamic controllability is defined in a way that subsumes the dynamic controllability of networks with contingent links and the dynamic consistency of networks with observation time-points. This version of dynamic controllability intuitively corresponds to the fundamental query presented earlier for workflows. In particular, the existence of a dynamic strategy for



$$\begin{array}{lcl}\mathcal{T}_f & = & \{A_1, A_2, X\};\quad \mathcal{T}_u \;=\; \{C_1, C_2\} \\ \mathcal{C} & = & \{C_1 - C_2 \in [-3, 8],\;\; C_1 - X \in [6, 12]\} \\ \mathcal{L} & = & \{(A_1, 1, 3, C_1), (A_2, 1, 10, C_2)\}\end{array}$$

Fig. 3 An STNU and its graphical representation. Contingent links are indicated by *dashed arrows*. Note that the terminus of any contingent link is an uncontrollable (i.e., contingent) time-point, indicated with *doubly-circled solid nodes*

scheduling the free time-points corresponds to the existence of a feasible tactic for executing the tasks of the workflow regardless of which paths are taken through the flow and which durations are subsequently observed. This parallel has already been analyzed elsewhere [26]. On top of that, the new CDTNU formalism also accommodates disjunctive constraints. The section concludes by defining the dynamic controllability problem for CDTNUs.

3.1 Simple Temporal Networks with Uncertainty

A Simple Temporal Network with Uncertainty (STNU) is a data structure for representing and reasoning about temporal knowledge in domains where some time-points are controlled by the executor (or agent) while others are controlled by the environment.[3] All temporal constraints in an STNU are *simple* (i.e., binary and convex difference constraints).

Definition 1 A Simple Temporal Network with Uncertainty (STNU) is a tuple $(\mathcal{T}, \mathcal{C}, \mathcal{L})$ where:

1. $\mathcal{T}$ is a set of real-valued variables, called *time-points,* that is partitioned into the sets, $\mathcal{T}_f$ and $\mathcal{T}_u$, of *free* and *uncontrollable* time-points;
2. $\mathcal{C}$ is a set of simple temporal constraints, each of the form, $Y - X \leq \delta$, for some $X, Y \in \mathcal{T}$ and $\delta \in \mathbb{R}$; and
3. $\mathcal{L}$ is a set of *contingent links,* each of the form, (A, ℓ, u, C), where $A \in \mathcal{T}$, $C \in \mathcal{T}_u$, and $0 < \ell < u < \infty$.

An expression, $Y - X \in [a, b]$, abbreviates the pair of constraints, $Y - X \leq b$ and $X - Y \leq -a$. A contingent link, (A, ℓ, u, C), represents a temporal interval from A to C whose duration is uncontrollable, but bounded by $C - A \in [\ell, u]$. A is called the *activation* time-point; C is called the *contingent* time-point. Figure 3 shows a sample STNU.

Dynamic Controllability of STNUs. Informally, an STNU is dynamically controllable (DC) if there exists a strategy for executing the free (or controllable) time-points such that all constraints in the network will be satisfied *no matter what durations the environment "chooses" for the contingent links—within their specified bounds.* The decisions that constitute such a strategy can depend only on execution events that occurred in the past; however, the strategy can be dynamic in that it may react—*after a positive delay*—to observations of contingent

[3] The agent and environment are not part of the formal semantics for STNUs; they are used here for expository convenience.

time-points executing. Such strategies are called *dynamic execution strategies* [35]. Thus, for an STNU, $(\mathcal{T}, \mathcal{C}, \mathcal{L})$, the agent seeks a dynamic execution strategy for executing the *free* time-points in $\mathcal{T}_f \subseteq \mathcal{T}$ such that all constraints in $\mathcal{C}$ will necessarily be satisfied no matter how the durations of the contingent links in $\mathcal{L}$ turn out—within their specified bounds.

An agent's execution strategy can be compactly defined in terms of *real-time execution decisions* (RTEDs), where each RTED has one of two forms: `wait` or (T, χ_f) [22]. A `wait` decision can be glossed as "wait until some contingent time-point happens to execute." A (T, χ_f) decision can be glossed as "if nothing happens *before* time T (i.e., if no contingent time-point happens to execute before time T), then I shall execute the (free) time-points in the set χ_f at time T." The *outcomes* for an RTED specify the range of execution events that could happen *next,* given the limited information available to the agent. For example, one outcome of a (T, χ_f) decision might be that a contingent time-point happens to execute at some time $\rho < T$. In such a case, the agent might choose to react by adopting a new decision. Another outcome might be that no contingent time-points execute before time T, in which case the time-points in χ_f would be executed at time T.

In the case of the STNU in Fig. 3, the agent seeks a strategy for executing the free time-points, A_1, A_2 and X, that will guarantee that the constraints among C_2, C_1 and X are satisfied, no matter what durations the environment happens to "choose" for the contingent links, $(A_1, 1, 3, C_1)$ and $(A_2, 1, 10, C_2)$. For example, the agent might decide to execute A_2 at time 0, and X at time 1, and then wait. Should the environment happen to "choose" a duration of 5 for the contingent link, $(A_2, 1, 10, C_2)$, the agent would observe, at time 5, the execution of C_2. The agent might then react—after some positive delay—by, for example, deciding to execute A_1 at time 7. Later, the agent might observe the environment choosing to execute C_1 at 9. In this example, after all time-points have executed, $C_1 - C_2 = 9 - 5 = 4 \in [-3, 8]$ and $C_1 - X = 9 - 1 = 8 \in [6, 12]$; thus, all constraints in $\mathcal{C}$ are satisfied and the agent has succeeded. It can be checked that this STNU is *dynamically controllable* (i.e., there exists a strategy for the agent that ensures success no matter how the environment behaves). The formal semantics for the dynamic controllability of STNUs is given in "Appendix 1".

3.2 Conditional Simple Temporal Networks with Uncertainty

A Conditional Simple Temporal Network with Uncertainty (CSTNU) augments an STNU to include *observation time-points,* each of which has a corresponding propositional letter. The propositional letters represent Boolean conditions whose truth values are observed in real time, during execution.[4] In particular, the execution of an observation time-point generates a truth value for its corresponding propositional letter. A *scenario* represents one possible set of truth values for all of the propositional letters. A *partial scenario* specifies the truth values of the propositional letters that have been observed so far.

Propositional *labels* comprising conjunctions of (positive or negative) propositional letters can be attached to time-points and temporal constraints in a CSTNU. A time-point with a propositional label ℓ is only executed in scenarios where ℓ is true. Similarly, a constraint labeled by ℓ only applies in scenarios where ℓ is true.

Definition 2 (*Labels* [41]) Given a set P of propositional letters, a *label* is any (possibly empty) conjunction of (positive or negative) literals from P. The *label universe* of P, denoted by P^*, is the set of all labels with literals drawn from P.

[4] The "C" in CSTNU stands for "Conditional", as in the *Conditional Temporal Problem* (CTP) introduced by Tsamardinos et al. [41]. A CSTNU extends both Conditional Simple Temporal Networks (CSTNs) and STNUs.

Definition 3 (*CSTNU* [26]) A *Conditional Simple Temporal Network with Uncertainty* (CSTNU) is a tuple, $(\mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P, \mathcal{L})$, where:

1. P is a set of propositional letters;
2. $\mathcal{T}$ is a set of time-points;
3. $\mathcal{OT} \subseteq \mathcal{T}$ is a set of observation time-points;
4. $\mathcal{O} : P \to \mathcal{OT}$ is a bijection between observation time-points and propositional letters;
5. $L : \mathcal{T} \to P^*$ is a function assigning labels to time-points;
6. $\mathcal{C}$ is a set of *labeled* temporal constraints of the form, $\langle Y - X \leq \delta,\ \ell\rangle$, where $X, Y \in \mathcal{T}$, $\delta \in \mathbb{R}$, and $\ell \in P^*$; and
7. Ignoring any labels, $(\mathcal{T}, \mathcal{C}, \mathcal{L})$ is an STNU.[5]

For the subclass of workflow schemata having no disjunctive constraints (like the two parallel, non-overlapping tasks of Fig. 1), Hunsberger et al. [26] presented a method of encoding the temporal information from each workflow schema into a CSTNU with the aim of rigorously analyzing and validating the temporal safety of the workflow. In particular, the CSTNU for a given workflow schema is obtained as follows. First, each task is represented by a contingent link. Second, each connector is represented by a pair of (starting and ending) time-points, linked by a duration constraint. Third, each arc is represented by a duration constraint. Fourth, the ending time-point for each conditional split connector is represented by an observation time-point for a proposition whose possible values correspond to the different branching decisions of that connector.[6] Fifth, the propositional label for each time-point is obtained by accumulating the propositional literals along the relevant pathway. Sixth, the propositional label for each temporal constraint is obtained by conjoining the labels on the associated time-points. Finally, if the duration range of a connector is [0, 0], then the connector can be represented by a single time-point.

Figure 4 shows the CSTNU obtained in this way for the portion of the workflow from Fig. 1 that excludes the disjunctive Cardiological Evaluation and Neurological Evaluation tasks. To facilitate comparison with the original workflow, the contingent links derived from the three workflow tasks have been highlighted in rounded gray rectangles. The observation time-point E yields the proposition p (i.e., Emerg?); and A yields the proposition q (i.e., Age > 70?). The contingent link $(ET_s, 8, 10, ET_e)$ corresponds to the Emergency Treatment task; $(EET_s, 10, 20, EET_e)$ corresponds to Elder Emergency Treatment; and $(ST_s, 10, 30, ST_e)$ corresponds to Standard Treatment. The CSTNU formalism is useful when the modeled situation has multiple possible evolutions depending on observations made at runtime. While STNUs are suitable for representing temporal plans, CSTNUs can be used to model conditional temporal plans.

Dynamic Controllability of CSTNUs. Hunsberger et al. [26] define the critical property of dynamic controllability for CSTNUs, generalizing both the dynamic controllability of STNUs [35] and the *dynamic consistency* of a CTP [41]. In brief, a CSTNU is dynamically controllable if there exists a strategy for executing the free time-points in the network such that all constraints in $\mathcal{C}$ are guaranteed to be satisfied no matter how the durations of the contingent links turn out, and no matter how the observations of the various propositions turn out, in real time—with the caveat that in any given scenario, only the time-points whose labels are true in that scenario need to be executed, and only the constraints whose labels

[5] There are some additional "well-definedness" conditions on CSTNUs that are omitted for expository convenience [26].

[6] Without loss of generality, this paper considers only binary branching in CSTNUs.

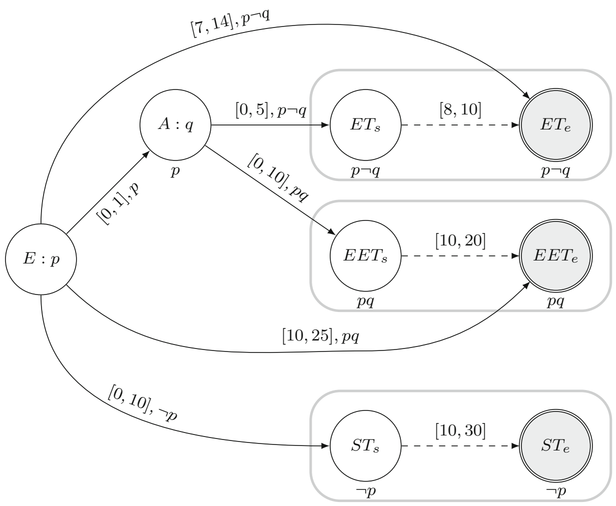


Fig. 4 The CSTNU obtained from the "Conditional" portion of the workflow from Fig. 1. *Circles* represent time-points (or nodes); each observation node includes a `time-point:proposition` indicator (e.g., $A : q$); propositional labels for time-points are written below the time-points; and *rounded rectangles* are used to highlight the contingent links derived from workflow tasks

are true in that scenario need to be satisfied. Subsequent work yielded a variety of constraint propagation rules for CSTNUs, which led to a sound-*but-not-complete* DC-checking algorithm for CSTNUs [15]. Since the temporal safety of a workflow schema corresponds directly to the dynamic controllability of the underlying CSTNU, providing a sound-and-complete DC-checking algorithm for CSTNUs remained an important open problem.

3.3 Disjunctive Temporal Networks with Uncertainty

Another important way of extending STNUs is to include disjunctive constraints. Disjunctions frequently arise in practice. For example, workflows in the healthcare domain frequently involve activities whose executions cannot overlap due to conflicting requirements, as illustrated in Fig. 1, where the Cardiological Evaluation and Neurological Evaluation tasks can be executed in any order, but cannot overlap. To retain maximal flexibility, it is desirable to constrain these tasks to be non-overlapping without imposing any a priori order on them. An STNU cannot accommodate such disjunctive constraints. However, similarly to the Disjunctive Temporal Problem (DTP) [40], *Disjunctive Temporal Networks with Uncertainty* (DTNUs) [43] can accommodate arbitrary disjunctions in the free constraints and binary disjunctions in the contingent constraints.

Definition 4 (*DTNU*) A DTNU is a triple, $(\mathcal{T}, \mathcal{C}, \mathcal{L})$, where:

1. $\mathcal{T}$ is a set of real-valued variables called *time-points,* partitioned into the sets, $\mathcal{T}_f$ and $\mathcal{T}_u$, of *free* and *uncontrollable* time-points;
2. $\mathcal{C}$ is a set of constraints, each obtained as the arbitrary Boolean combination of atoms in the form, $Y - X \leq \delta$, for some $X, Y \in \mathcal{T}$ and $\delta \in \mathbb{R}$; and
3. $\mathcal{L}$ is a set of *contingent links*, each of the form, $(A, \mathcal{B}, C)$, where $A \in \mathcal{T}$, $C \in \mathcal{T}_u$, and $\mathcal{B}$ is a finite set of pairs (ℓ, u) such that $0 < \ell < u < \infty$; and for any distinct pairs, (ℓ_i, u_i) and (ℓ_j, u_j) in $\mathcal{B}$, either $\ell_i > u_j$ or $u_i < \ell_j$.

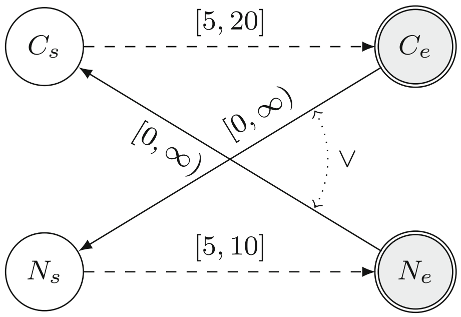


Fig. 5 The DTNU obtained from the "Disjunctive" portion of the workflow from Fig. 1. The two constraints ($N_e - C_s \leq 0$) and ($C_e - N_s \leq 0$) are part of a single disjunctive free constraint

Generalizing the contingent links in an STNU, a contingent link $(A, \mathcal{B}, C)$ in a DTNU represents a temporal interval from A to C whose duration, $C - A$, is uncontrollable, but guaranteed to lie within a union of disjoint intervals. In particular, if $\mathcal{B} = \{(\ell_1, u_1), \cdots, (\ell_n, u_n)\}$, then $C - A$ is guaranteed to fall somewhere within the set $[\ell_1, u_1] \cup \cdots \cup [\ell_n, u_n]$. Although contingent durations in a DTNU can be disjunctive in this way, the execution semantics ensures that the choices made by the environment for distinct contingent durations are independent.

This is useful for modeling periodic activities whose windows of opportunity have certain degrees of uncertainty. An STNU is the particular case of a DTNU in which all of the $\mathcal{B}$ sets are singletons, and conjunction is the only allowed Boolean operator in the constraints belonging to $\mathcal{C}$.

The example of the parallel Cardiological Evaluation and Neurological Evaluation tasks can be encoded as a DTNU $(\mathcal{T}_f \cup \mathcal{T}_u, \mathcal{C}, \mathcal{L})$ as depicted in Fig. 5. First, let C_s, C_e, N_s and N_e be the starting and ending times for these two tasks; then, $\mathcal{T}_f \doteq \{C_s, N_s\}$ and $\mathcal{T}_u \doteq \{C_e, N_e\}$. The constraint set $\mathcal{C}$ contains just one disjunctive constraint:

$$\mathcal{C} \doteq \{(N_e - C_s \leq 0) \vee (C_e - N_s \leq 0)\}.$$

Finally the set of contingent links is given by:

$$\mathcal{L} \doteq \{(C_s, \{[5, 20]\}, C_e), (N_s, \{[5, 10]\}, N_e)\}.$$

Dynamic Controllability of DTNUs. The dynamic controllability problem for DTNUs is defined analogously to the STNU case [38,42]. Disjunctive free constraints simply give more freedom to the agent, while disjunctions in contingent constraints allow the environment to choose from among a set of intervals. However, these extensions do not dramatically change the semantics of dynamic controllability.

3.4 Conditional Disjunctive Temporal Networks with Uncertainty

The CSTNU and DTNU formalisms extend STNUs in different directions. On the one hand, a CSTNU includes observation time-points to model discrete observations at run-time; on the other hand, a DTNU allows disjunctive constraints that can model non-convex durations, non-overlapping durations, and all of the other temporal relations from Allen's *interval algebra* [2]. These two extensions are both interesting and useful from a practical standpoint, as illustrated by Fig. 1. Therefore, this paper introduces a new formalism that accommodates all of the features from both CSTNUs and DTNUs and, thus, is expressive enough to faithfully represent all of the temporal information from the workflow depicted in Fig. 1.

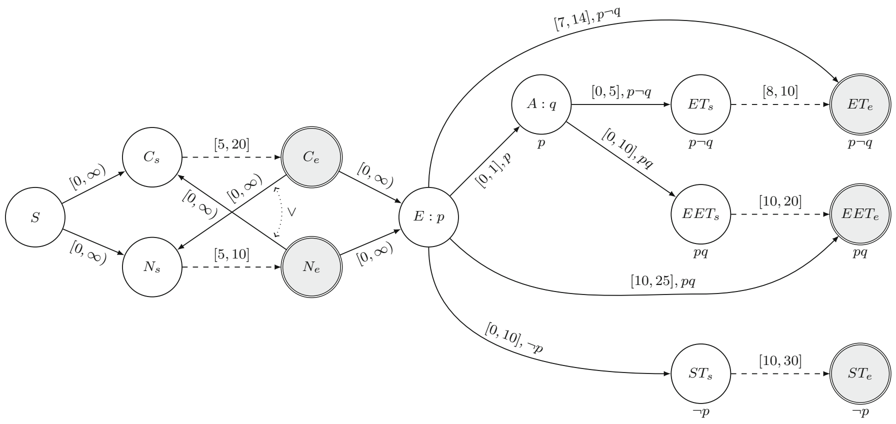


Fig. 6 The CDTNU obtained from the complete workflow from Fig. 1. The newly introduced node *S* represents the start of the workflow

The new formalism is called a Conditional Disjunctive Temporal Network with Uncertainty (CDTNU), defined below. It is devised in such a way that it directly subsumes both CSTNUs and DTNUs. Thus, every CSTNU is a CDTNU, and every DTNU is a CDTNU.

Definition 5 (*CDTNU*) A *Conditional Disjunctive Temporal Network with Uncertainty* (CDTNU) is a tuple, $(\mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P, \mathcal{L})$, where:

1. P is a set of propositional letters;
2. $\mathcal{T}$ is a set of time-points;
3. $\mathcal{OT} \subseteq \mathcal{T}$ is a set of observation time-points;
4. $\mathcal{O} : P \to \mathcal{OT}$ is a bijection between observation time-points and propositional letters;
5. $L : \mathcal{T} \to P^*$ is a function that assigns propositional labels to time-points;
6. $\mathcal{C}$ is a set of *labeled*, *disjunctive* temporal constraints of the form, $\langle \phi,\ \ell \rangle$, where $\ell \in P^*$, and ϕ is an arbitrary Boolean combination of atoms, each of the form, $Y - X \leq \delta$, for some $X, Y \in \mathcal{T}$ and $\delta \in \mathbb{R}$; and
7. Ignoring any labels, $(\mathcal{T}, \mathcal{C}, \mathcal{L})$ is a DTNU.

Figure 6 depicts a CDTNU that effectively models the workflow from Fig. 1. Note that edges with no label are required to hold in every scenario; and disjunctive constraints can be labeled, although the sample CDTNU does not show this possibility.

Dynamic Controllability of CDTNUs. The dynamic controllability problem for CDTNUs is analogous to the DC problem for CSTNUs, with the important difference that a CDTNU can have non-convex binary disjunctions in the contingent links (as in a DTNU) that need to be considered for dynamic execution strategies. A complete formalization of the DC problem for CDTNUs is given in "Appendix 2".

4 Reducing dynamic controllability to TGA reachability

This section presents a general approach to solve the dynamic controllability problem for all of the temporal networks discussed above. The basic idea is to reduce the DC-checking problem for a temporal network to a *reachability game* for a *Timed Game Automaton* (TGA) obtained via a *linear* encoding procedure. Since the reachability problem for TGAs is decidable and algorithms have been developed to solve it, this reduction constitutes a viable and novel solution approach for the open problems of determining the dynamic controllability of CSTNUs, DTNUs and CDTNUs.

This section begins by introducing the relevant background from the TGA literature. It then presents, for each kind of temporal network, an encoding of that network into a TGA such that the temporal network is dynamically controllable if and only if the reachability game for the corresponding TGA is solvable.

4.1 Timed Game Automata

A *finite automaton* [31] comprises a finite set of states (or locations) and a finite set of labeled transitions (or actions). One of the states is called the initial (or starting) state; a distinguished subset of states comprise the *final* (or *accepting*) states. Each labeled transition specifies a legal move from one state to another.

A *Timed Automaton* (TA) [3] augments a finite automaton to include real-valued *clocks.* Each transition in a TA may include temporal constraints, called *guards,* that disable the transition if the current clock values do not satisfy those constraints. Each transition may also include *clock resets* that cause specified clocks to be reset to 0 whenever the transition is taken. Finally, each location may include an *invariant*—that is, a constraint specifying the conditions under which the automaton may stay in that location. Definition 6 formalizes this structure.

Definition 6 (*Timed Automaton*) A *Timed Automaton* (TA) is a tuple, $A = (L, l_0, Act, \mathcal{X}, E, Inv)$, where:

1. L is a finite set of locations;
2. $l_0 \in L$ is the initial location;
3. Act is a set of actions;
4. $\mathcal{X}$ is a finite set of real-valued clocks;
5. $E \subseteq L \times \mathcal{H}_k^{\cap}(\mathcal{X}) \times Act \times 2^{\mathcal{X}} \times L$ is a finite set of transitions; and
6. $Inv : L \to \mathcal{H}_k^{\cap}(\mathcal{X})$ associates an *invariant* to each location.

Elements in $\mathcal{H}_k^{\cap}(\mathcal{X})$ are conjunctions of constraints of the form, $x \bowtie k$ or $y - x \bowtie k$, where $x, y \in \mathcal{X}$, k is an integer, and $\bowtie$ is one of $<, \leq, =, >$ or $\geq$.

Figure 7 shows a sample TA. The TA has one clock, `cC`. The entering arrow with no predecessor node indicates that X is the initial location. X's invariant is $\texttt{cC} \leq 3$. Each transition has a label, $\langle G;\ N;\ R\rangle$, where G is the guard, N is a name for the transition, and R is the set of clocks it resets. A *run* starts in the initial location, X, with $\texttt{cC} = 0$. X's invariant, $\texttt{cC} \leq 3$, and the guard, $\texttt{cC} \geq 1$, on the `pass` transition, together ensure that the TA must take the transition from X to Y at some time when $1 \leq \texttt{cC} \leq 3$. When taken, that transition resets `cC` to 0. Afterward, the `gain` transition, whose guard is $\texttt{cC} \geq 5$, could be

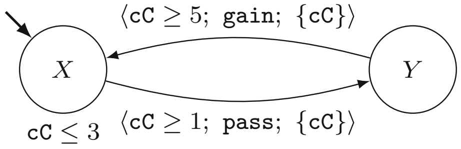


Fig. 7 A sample timed automaton

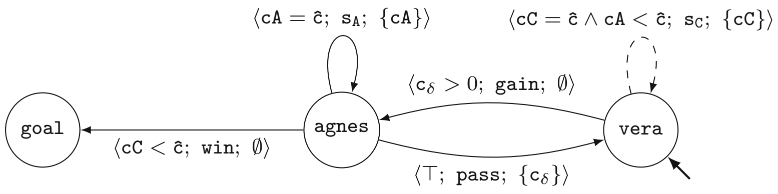


Fig. 8 A sample Timed Game Automaton. Controllable transitions (belonging to Act_c) are *solid*, while uncontrollable transitions (belonging to Act_u) are *dashed*

taken back to X at any time for which $\texttt{cC} \geq 5$. If taken, the gain transition also resets cC to 0. However, since Y has no invariant, the TA could instead remain at Y forever.

A *Timed Game Automaton* (TGA) in turn generalizes a TA by partitioning the set of transitions into *controllable* and *uncontrollable* transitions. A TGA can be used to model a two-player game between an *agent* and the *environment*, where the agent controls the *controllable* transitions, and the environment controls the *uncontrollable* transitions. TGAs are formally defined in Definition 7.

Definition 7 (*Timed Game Automaton*) A Timed Game Automaton (TGA) is a Timed Automaton whose set of actions, Act, is partitioned into controllable (Act_c) and uncontrollable (Act_u) actions.

Figure 8 shows a TGA with three locations: agnes, vera and goal, where vera is the initial location. It has four clocks: cA, cC, $\hat{\texttt{c}}$ and $\texttt{c}_\delta$. The solid arrows represent controllable transitions; the dashed arrow represents the one uncontrollable transition. For example, the transition from agnes to itself has the label, $\langle \texttt{cA} = \hat{\texttt{c}};\ \texttt{s}_\texttt{A};\ \{\texttt{cA}\}\rangle$, which specifies that it can only be taken if cA and $\hat{\texttt{c}}$ have the same value; and that taking this transition resets cA to 0. Consider the following possible run of this TGA. It begins at the initial location vera, with all clocks set to 0. Five units of time later, when all clocks read 5, the agent takes the gain transition to agnes. (The guard is satisfied; and no clocks are reset.) Then, at time 6, the agent takes the $\texttt{s}_\texttt{A}$ transition, which causes cA to be reset to 0. Then, at time 7, the agent takes the pass transition back to vera, which resets $\texttt{c}_\delta$ back to 0. At this point, $\texttt{c}_\delta = 0$; $\texttt{cA} = 1$; and $\texttt{cC} = \hat{\texttt{c}} = 7$. Thus, the environment can take the $\texttt{s}_\texttt{C}$ transition from vera to itself, resetting cC to 0. Then, at time 10, the agent takes the gain transition back to agnes, and at 11 the win transition to the goal state.

In what follows, the common practice of labeling certain locations *urgent* is used. An urgent location is one in which players are prevented from *waiting*. Making a location ℓ urgent is equivalent to: (1) introducing a new clock c that is reset by every transition entering ℓ; and (2) conjoining a new invariant, $c \leq 0$, to ℓ.

For any TGA, different kinds of games can be modeled [6]. In a reachability game, the controller (or agent) seeks to move the TGA into one of the *winning locations* within a finite amount of time. In the avoidance game, the controller seeks to prevent the TGA from entering a certain set of locations. In this paper we use *memory-less* strategies, since they have been shown to be sufficient for reachability and avoidance games [6,32]. Intuitively, a memory-less strategy associates a state of the system to either an action to be executed or a special symbol λ that stands for "wait": the controller shall not take any controllable transition, it just needs to wait the opponent move (i.e., do nothing, wait until an uncontrollable transition is taken by the opponent).

Definition 8 For a TGA, $(L, l_0, Act, \mathcal{X}, E, Inv)$, a *memory-less strategy* is a mapping $f : L \times \mathbb{R}^{\mathcal{X}}_{>=0} \rightarrow Act_c \cup \lambda$.

Further details on the semantics for TGAs are available from Maler et al. [32].

4.2 TGA encodings

This section presents a series of TGA encodings, one for each class of temporal network introduced in Sect. 3.

4.2.1 STNU-to-TGA encoding

Given any STNU $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$, the goal is to generate a corresponding TGA $T_{\mathcal{S}} = (L, l_0, Act, \mathcal{X}, E, Inv)$, and a winning condition ϕ, such that the STNU $\mathcal{S}$ is dynamically controllable if and only if the TGA $T_{\mathcal{S}}$ admits a *counter-strategy* for ϕ. An important—and unexpected—part of this STNU-to-TGA encoding is that *uncontrollable* TGA transitions are used to model the execution of the *free* time-points in $\mathcal{S}$, and *controllable* TGA transitions are used to model the execution of the *uncontrollable* time-points in $\mathcal{S}$. Thus, the traditional use of TGAs where the environment is associated with uncontrollable transitions has been inverted. (That is why a *counter-strategy* is sought.) The underlying reason is that according to the STNU semantics, when both players attempt to make transitions at the same time, Agnes (the agent) must play before Vera (the environment), whereas in the TGA semantics, the uncontrollable transition would go first.

For this encoding, the set of locations is: $L \doteq \{\mathtt{agnes}, \mathtt{vera}, \mathtt{goal}\}$, where $\mathtt{agnes}$ is marked *urgent.* Note that L has only three locations, regardless of the number of time-points in the STNU. Intuitively, $\mathtt{agnes}$ represents a state in which Agnes can execute one or more free time-points; $\mathtt{vera}$ represents a state in which Vera can execute one or more contingent time-points; and $\mathtt{goal}$ represents a state in which all of the constraints have been satisfied and the game is over (and $\mathtt{agnes}$ wins, having successfully scheduled all of the time-points, while satisfying all of the constraints). The initial location of the TGA is $\mathtt{vera}$ (i.e., $l_0 \doteq \mathtt{vera}$).

The set of clocks is: $\mathcal{X} \doteq \{\hat{\mathtt{c}}, \mathtt{c}_\delta\} \cup \{\mathtt{cX} \mid X \in \mathcal{T}\}$. All clocks start at 0. The clock $\hat{\mathtt{c}}$ is never reset; it simply measures global time. The clock $\mathtt{c}_\delta$ is used to ensure that there will always be a positive delay between the execution of any contingent time-point (by Vera) and any *reaction* by Agnes, which is crucial for capturing the STNU semantics. Finally, for each time-point $X \in \mathcal{T}$, there is a corresponding clock $\mathtt{cX}$. That clock is reset at most once each run, at the instant X is executed. It follows that any time-point X has been executed if and only if $\mathtt{cX} < \hat{\mathtt{c}}$. (Since the initial state is $\mathtt{vera}$, no time-point can be executed at 0.) Also, after being executed, the execution time for X is forever equal to $\hat{\mathtt{c}} - \mathtt{cX}$.

The sets of controllable and uncontrollable actions are defined as follows. First, the controllable actions (for Vera) consist of one action for each contingent time-point in $\mathcal{S}$, as follows: $Act_c \doteq \{\mathtt{ex_X} \mid X \in \mathcal{T}_u\}$. Each action in this set represents the execution of the corresponding time-point. The uncontrollable actions (for Agnes) include more options: $Act_u \doteq \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$, where:

$$\begin{aligned}
\mathcal{A}_1 &= \{\mathtt{ex_X} \mid X \in \mathcal{T}_f\};\\
\mathcal{A}_2 &= \{\mathtt{cv_C} \mid (A, \ell, u, C) \in \mathcal{L}\};\ \text{and}\\
\mathcal{A}_3 &= \{\mathtt{gain}, \mathtt{pass}, \mathtt{win}\}.
\end{aligned}$$

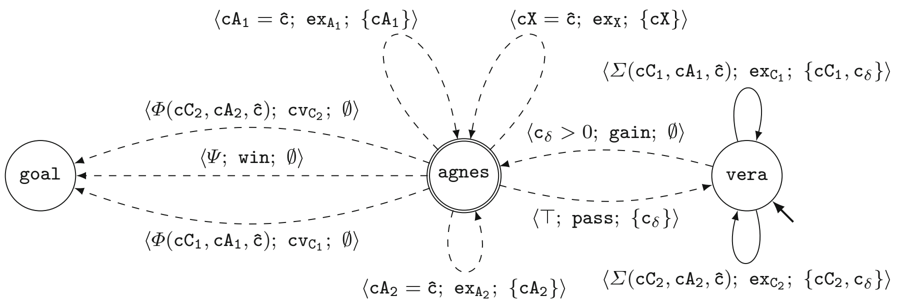

$\langle cA_1 = \hat{c};\ ex_{A_1};\ \{cA_1\}\rangle$
$\langle cX = \hat{c};\ ex_X;\ \{cX\}\rangle$
$\langle \Sigma(cC_1, cA_1, \hat{c});\ ex_{C_1};\ \{cC_1, c_\delta\}\rangle$
$\langle \Phi(cC_2, cA_2, \hat{c});\ cv_{C_2};\ \emptyset\rangle$
$\langle c_\delta > 0;\ gain;\ \emptyset\rangle$
$\langle \Psi;\ win;\ \emptyset\rangle$
goal
agnes
vera
$\langle \top;\ pass;\ \{c_\delta\}\rangle$
$\langle \Phi(cC_1, cA_1, \hat{c});\ cv_{C_1};\ \emptyset\rangle$
$\langle cA_2 = \hat{c};\ ex_{A_2};\ \{cA_2\}\rangle$
$\langle \Sigma(cC_2, cA_2, \hat{c});\ ex_{C_2};\ \{cC_2, c_\delta\}\rangle$

Fig. 9 Encoding the STNU from Fig. 3 into a TGA. *Solid arrows* represent controllable transitions (for Vera); *dashed arrows* uncontrollable transitions (for Agnes). The *doubly-circled* agnes location is *urgent;* the initial location is vera

$\mathcal{A}_1$ contains one execution action for each free time-point. $\mathcal{A}_2$ contains one action for each contingent link; these actions are only enabled if Vera violates the bounds on any of her contingent links. gain and pass model the interplay between the execution of time-points by Agnes and Vera; win is used at the end when all time-points have been executed and all constraints have been satisfied.

The transition relation, E, for the TGA encoding of an STNU is demonstrated in Fig. 9, using the sample STNU from Fig. 3. For each free time-point X, there is a transition from agnes to agnes labeled by $\langle cX = \hat{c};\ ex_X;\ \{cX\}\rangle$, which represents the execution of X by Agnes. The *guard,* $cX = \hat{c}$ (i.e., X not yet executed), ensures that this transition will be taken at most once per run. The set, $\{cX\}$, stipulates that the clock cX will be reset by this transition, signalling that X has been executed. Similarly, for each contingent link, (A, ℓ, u, C), there is a transition from vera to vera labeled by $\langle \Sigma(cC, cA, \hat{c});\ ex_C;\ \{cC, c_\delta\}\rangle$, which represents the execution of C by Vera. The guard, $\Sigma(cC, cA, \hat{c}) \doteq (cA < \hat{c}) \wedge (cC = \hat{c}) \wedge (cA \geq \ell) \wedge (cA \leq u)$, ensures that this transition can only be taken when the link is currently activated and its duration would fall within $[\ell, u]$. In addition, for each contingent link, (A, ℓ, u, C), there is a transition from agnes to goal labeled by $\langle \Phi_C(cA, cC, \hat{c});\ cv_C;\ \emptyset\rangle$, enabling Agnes to move to goal should Vera ever violate the bounds on that link by failing to execute C. Its guard is: $\Phi_C(cA, cC, \hat{c}) \doteq (cA < \hat{c}) \wedge (cA > u) \wedge (cC = \hat{c})$. Next, if $\mathbf{t}$ is the vector of clocks cX such that $X \in \mathcal{T}$, the transition from agnes to goal labeled by $\langle \Psi(\mathbf{t}, \hat{c});\ win;\ \emptyset\rangle$ signals the end of the game. $\Psi(\mathbf{t}, \hat{c})$ models that all time-points have been executed and all constraints are satisfied: $\Psi(\mathbf{t}, \hat{c}) \doteq \bigwedge_{x \in \mathcal{T}} (cX < \hat{c}) \wedge \bigwedge_{Y - X \leq k} (cX - cY \leq k)$. Last, to model the interplay between the players, there are two more transitions. The transition from vera to agnes labeled by $\langle c_\delta > 0;\ gain;\ \emptyset\rangle$ enables Agnes to gain control for the purpose of executing some free time-points—but only after some positive delay since Vera last executed a contingent time-point. The transition from agnes to vera labeled by $\langle \top;\ pass;\ \{c_\delta\}\rangle$ enables Agnes to immediately pass back to vera, once she has finished executing her chosen time-points. Crucially, no time elapses from the instant Agnes leaves vera to the instant she returns, because agnes is an urgent state. From Vera's perspective, the winning condition ϕ of the (safety) game is to avoid the goal state. A counter-strategy for Agnes foils Vera by ensuring that goal can be reached.

The correctness of the STNU-to-TGA encoding is proven in "Appendix 3".

4.2.2 CSTNU-to-TGA encoding

We now consider CSTNUs and extend the STNU encoding to accommodate observations and Boolean propositions. We retain the structure of the STNU encoding, by modifying only the way free constraints are represented and by adding a suitable encoding for the Boolean propositions that are decided by Vera. A CSTNU instance can be seen as an STNU in which some parts can be disabled depending on the observations that become visible during the execution. As described in Sect. 3.2, an observation node is a time-point whose execution generates a truth value for an associated proposition (equivalently, a Boolean variable). For example, let X be an observation time-point whose execution establishes the truth value of the proposition p. Note that, before X is executed (i.e., while $\mathtt{cX} = \hat{\mathtt{c}}$), the truth value of p is unknown, but after p is executed, its truth value must be either true or false. This feature can be accommodated in a TGA by introducing a new clock $\mathtt{bP}$ (which stands for "Boolean p"), whose value is meaningful only after X has been executed. After X has executed, if $\mathtt{bP} = \hat{\mathtt{c}}$, then p shall be interpreted as being true; but if $\mathtt{bP} < \hat{\mathtt{c}}$ (i.e., if $\mathtt{bP}$ has been reset), then p shall be interpreted as being false.

As shown in Fig. 10, at the instant X is executed by Agnes, the guard on the `resetP` transition (a loop at `vera`) gives Vera precisely one opportunity to reset `bP` (i.e., to make p false). The guard on `resetP` is:

$$\Upsilon(\mathtt{cX}, \hat{\mathtt{c}}, \mathtt{bP}) \doteq (\mathtt{cX} < \hat{\mathtt{c}}) \wedge (\mathtt{cX} = 0) \wedge (\mathtt{bP} = \hat{\mathtt{c}})$$

which represents that X has been executed *now,* but `bP` has not yet been reset. If Vera does not take this opportunity, then `bP` shall forever be equal to $\hat{\mathtt{c}}$ (i.e., p shall forever be true).

Next, the `win` transition that represents the "all time-points executed and all constraints satisfied" condition is changed so that it emanates not from the `agnes` location, but from the `vera` location—and with a guard that includes the constraint $\mathtt{c}_\delta > 0$. This is done to ensure that Vera always gets her opportunity to set the truth value of any proposition corresponding to a just executed observation time-point. (Otherwise, Agnes might surreptitiously execute an observation time-point and then take the `win` transition before Vera has a chance to set the truth value of the corresponding proposition.)

Finally, since the nodes and constraints in a CSTNU can be labeled by conjunctions of (positive or negative) propositional letters, the "all time-points executed and all constraints satisfied" condition must be represented in a new way. To see this, consider the following example. Suppose that the time-points, R and S, and the constraint, $S - R \leq 5$, are labeled

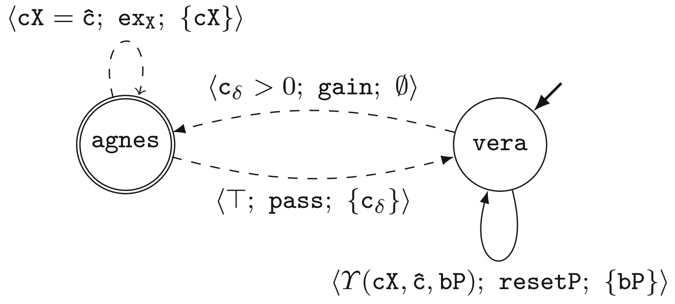


Fig. 10 An observation node X whose associated proposition is p. The loop in `agnes` executes the time point X, while the one in `vera` sets p to false. It is up to the environment to decide whether to take the loop in `vera` or not

by $p\neg q$ (i.e., $p \wedge (\neg q)$). Suppose further that X and Y are the observation time-points for p and q, respectively. Then, in any scenario in which X and Y are both executed, and p is true, and q is false, success requires that both R and S be executed, and that $S - R \leq 5$ be satisfied. This can be represented by the following *conditional* constraint:

$$((\texttt{cX} < \hat{\texttt{c}}) \wedge (\texttt{cY} < \hat{\texttt{c}}) \wedge (\texttt{bP} = \hat{\texttt{c}}) \wedge (\texttt{bQ} < \hat{\texttt{c}}))$$
$$\rightarrow ((\texttt{cR} < \hat{\texttt{c}}) \wedge (\texttt{cS} < \hat{\texttt{c}}) \wedge (\texttt{cR} - \texttt{cS} \leq 5))$$

which is equivalent to:

$$\neg(\texttt{cX} < \hat{\texttt{c}}) \vee \neg(\texttt{cY} < \hat{\texttt{c}}) \vee \neg(\texttt{bP} = \hat{\texttt{c}}) \vee$$
$$\neg(\texttt{bQ} < \hat{\texttt{c}}) \vee ((\texttt{cR} < \hat{\texttt{c}}) \wedge (\texttt{cS} < \hat{\texttt{c}}) \wedge (\texttt{cR} - \texttt{cS} \leq 5))$$

Given that no clock's value can ever exceed that of the global clock $\hat{\texttt{c}}$, this further simplifies to:

$$(\texttt{cX} = \hat{\texttt{c}}) \vee (\texttt{cY} = \hat{\texttt{c}}) \vee (\texttt{bP} < \hat{\texttt{c}}) \vee (\texttt{bQ} = \hat{\texttt{c}}) \vee \ ((\texttt{cR} < \hat{\texttt{c}}) \wedge (\texttt{cS} < \hat{\texttt{c}}) \wedge (\texttt{cR} - \texttt{cS} \leq 5))$$

Since the guards on TGA transitions do not allow disjunctions, the above condition can be represented using five separate transitions, one for each disjunct, as illustrated in Fig. 11.

In general, suppose that $\ell_1, \ell_2, \ldots, \ell_M$ are the M distinct labels that appear in some CSTNU. (Typically, M is much smaller than the number of labels in the label universe, P^*.) For each i, let τ_i be the set of time-points labeled by ℓ_i, and $\mathcal{C}_i$ the set of constraints labeled by ℓ_i. In addition, for each i, let θ_i be the conditional constraint that can be glossed as: "In scenarios where ℓ_i is true, all of the time-points in τ_i must be executed, and all of the constraints in $\mathcal{C}_i$ must be satisfied," as discussed in the preceding example. The desired "all time-points executed and all constraints satisfied" condition is then the conjunction: $\wedge_{i=1}^{M} \theta_i$. This conjunction of conditional constraints can be effectively accommodated in the TGA using a sequence of $(M+1)$ *urgent* locations starting from $L_0 = \texttt{vera}$, and ending at $L_M = \texttt{goal}$, as follows:

$$(\texttt{vera} = L_0) \ \rightsquigarrow \ L_1 \ \rightsquigarrow \ L_2 \ \rightsquigarrow \ \ldots \ \rightsquigarrow \ (L_M = \texttt{goal})$$

For each i, there is a set of transitions from L_{i-1} to L_i that together represent the conditional constraint θ_i, as illustrated in Fig. 12. If, in some scenario, Agnes can follow a path through this network of transitions from `vera` to `goal`, then all of the relevant time-points must have been executed, and all of the relevant constraints must have been satisfied.

Figure 13 illustrates the TGA that is obtained in this way from the sample CSTNU seen earlier in Fig. 4. In the figure, the sequence of locations, L_i, have been renamed to show the associated labels, as follows:

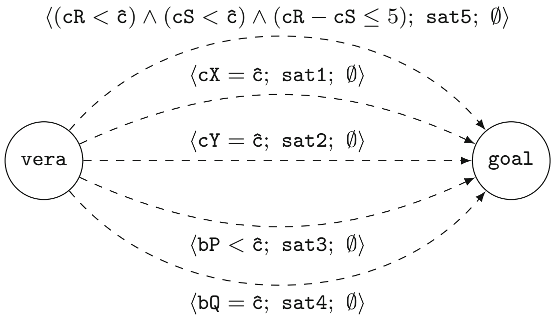


Fig. 11 The transitions capturing the sample "all time-points executed and all constraints satisfied" constraint in the scenario $p\neg q$, discussed in the text

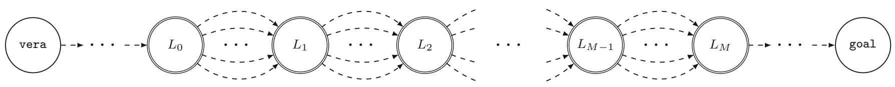


Fig. 12 Using a sequence of locations in a TGA to accommodate the "all time-points executed and all constraints satisfied" condition for a CSTNU

$$\texttt{vera} \rightsquigarrow L_{\emptyset} \rightsquigarrow L_p \rightsquigarrow L_{p\neg q} \rightsquigarrow L_{pq} \rightsquigarrow (L_{\neg p} = \texttt{goal})$$

In addition, to reduce the clutter, only the guards are shown on the transitions in this sequence.

4.2.3 DTNU-to-TGA encoding

DTNUs generalize STNUs in two different dimensions. First, the durations of contingent links can be constrained to lie within a union of disjoint intervals. Second, the free constraints can comprise arbitrary Boolean combinations of difference constraints. This section shows how a DTNU $(\mathcal{T}, \mathcal{C}, \mathcal{L})$ can be translated into an equivalent TGA by making two modifications to the STNU-to-TGA translation presented in Sect. 4.2.1. First, if the duration for a contingent link is constrained to lie within one of n disjoint intervals, then there will be n corresponding loops at the vera location, where the guard for each loop effectively specifies one of the allowed intervals for that contingent duration. Second, the "all time-points executed and all constraints satisfied" transition to the goal location is represented by alternative pathways through a sequence of locations from vera to goal, using a technique that generalizes that shown for CSTNUs in the preceding section.

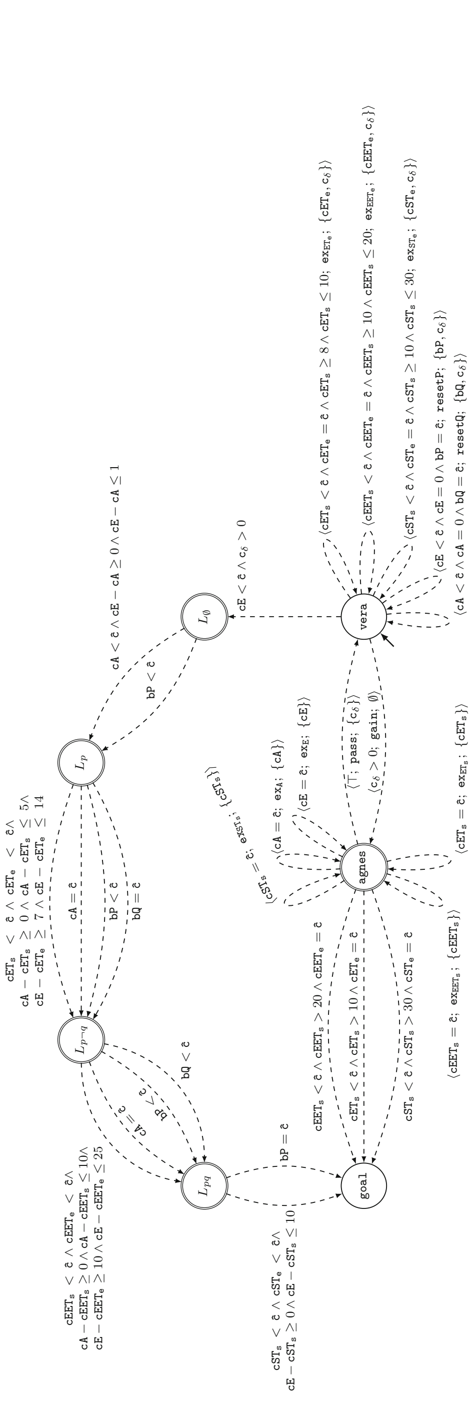
To begin, as for an STNU, each free time-point X will have a corresponding transition, $(\texttt{agnes}, \texttt{cX} = \texttt{cG}, \texttt{ex}_{\texttt{X}}, \{\texttt{cX}\}, \texttt{agnes})$, that represents the execution of X by Agnes. However, for each disjunctive contingent link, $(A, \mathcal{B}, C)$, where $\mathcal{B} \doteq \{(\ell_1, u_1), \cdots, (\ell_n, u_n)\}$, there are n loop transitions: $(\texttt{vera}, \varSigma(\texttt{cC}, \texttt{cA}, \texttt{cG}, l_i, u_i), \texttt{ex}_{\texttt{C}}, \{\texttt{cC}, \texttt{c}_{\delta}\}, \texttt{vera})$, for $i \in [1, n]$. They represent the possible executions of C by Vera. The respective guards,

$$\varSigma(\texttt{cC}, \texttt{cA}, \texttt{cG}, l_i, u_i) \doteq (\texttt{cA} < \texttt{cG}) \wedge (\texttt{cC} = \texttt{cG}) \wedge (\texttt{cA} \geq l_i) \wedge (\texttt{cA} \leq u_i)$$

ensure that (one of) these transitions can be taken only when the link is currently activated and its duration would fall within one of the allowed intervals of $\mathcal{B}$. In addition, for each contingent constraint, there is a transition, $(\texttt{agnes}, \varPhi_C(\texttt{cA}, \texttt{cC}, \texttt{cG}, max_i(u_i)), \texttt{cv}_{\texttt{C}}, \emptyset, \texttt{goal})$ that allows Agnes to win the game if Vera refuses to schedule an uncontrollable time-point within the maximum allowed bound, $max_i(u_i)$. The guard is expressed by $\varPhi_C(\texttt{cA}, \texttt{cC}, \texttt{cG}, u) \doteq (\texttt{cA} < \texttt{cG}) \wedge (\texttt{cA} > u) \wedge (\texttt{cC} = \texttt{cG})$, as in the STNU case. The interplay between the players, governed by the pass and gain transitions, is identical to the STNU case.

Next, the TGA must accommodate the arbitrary Boolean combinations of constraints in $\mathcal{C}$. In principle, we would like to have a transition, $(\texttt{agnes}, \varOmega(\mathbf{c}, \texttt{cG}), \texttt{win}, \emptyset, \texttt{goal})$ that signals the end of the game, where $\varOmega(\mathbf{c}, \texttt{cG})$ encodes the fact that all time-points have been executed and all constraints satisfied, and $\mathbf{c}$ is the set of clocks associated with the time-points.

First, let $\varLambda \doteq \bigwedge_{i=1}^{|\mathcal{C}|} \mathcal{C}_i$ be the first-order logic formula encoding all the constraints in $\mathcal{C}$. Then, let $\varOmega(\mathbf{c}, \texttt{cG})$ be the formula that results from replacing each atomic constraint,



vera
agnes
goal
$L_\emptyset$
L_p
$L_{p\neg q}$
L_{pq}
$\langle \mathtt{cET_s} < \hat{c} \wedge \mathtt{cET_e} = \hat{c} \wedge \mathtt{cET_s} \geq 8 \wedge \mathtt{cET_s} \leq 10;\ \mathtt{ex_{ET_e}};\ \{\mathtt{cET_e}, \mathtt{c_\delta}\}\rangle$
$\langle \mathtt{cEET_s} < \hat{c} \wedge \mathtt{cEET_e} = \hat{c} \wedge \mathtt{cEET_s} \geq 10 \wedge \mathtt{cEET_s} \leq 20;\ \mathtt{ex_{EET_e}};\ \{\mathtt{cEET_e}, \mathtt{c_\delta}\}\rangle$
$\langle \mathtt{cST_s} < \hat{c} \wedge \mathtt{cST_e} = \hat{c} \wedge \mathtt{cST_s} \geq 10 \wedge \mathtt{cST_s} \leq 30;\ \mathtt{ex_{ST_e}};\ \{\mathtt{cST_e}, \mathtt{c_\delta}\}\rangle$
$\langle \mathtt{cE} < \hat{c} \wedge \mathtt{cE} = 0 \wedge \mathtt{bP} = \hat{c};\ \mathtt{resetP};\ \{\mathtt{bP}, \mathtt{c_\delta}\}\rangle$
$\langle \mathtt{cA} < \hat{c} \wedge \mathtt{cA} = 0 \wedge \mathtt{bQ} = \hat{c};\ \mathtt{resetQ};\ \{\mathtt{bQ}, \mathtt{c_\delta}\}\rangle$
$\mathtt{cE} < \hat{c} \wedge \mathtt{c_\delta} > 0$
$\mathtt{cA} < \hat{c} \wedge \mathtt{cE} - \mathtt{cA} \geq 0 \wedge \mathtt{cE} - \mathtt{cA} \leq 1$
$\mathtt{bP} < \hat{c}$
$\langle \top;\ \mathtt{pass};\ \{\mathtt{c_\delta}\}\rangle$
$\langle \mathtt{c_\delta} > 0;\ \mathtt{gain};\ \emptyset\rangle$
$\langle \mathtt{cE} = \hat{c};\ \mathtt{ex_E};\ \{\mathtt{cE}\}\rangle$
$\langle \mathtt{cA} = \hat{c};\ \mathtt{ex_A};\ \{\mathtt{cA}\}\rangle$
$\langle \mathtt{cST_s} = \hat{c};\ \mathtt{ex_{ST_s}};\ \{\mathtt{cST_s}\}\rangle$
$\langle \mathtt{cET_s} = \hat{c};\ \mathtt{ex_{ET_s}};\ \{\mathtt{cET_s}\}\rangle$
$\langle \mathtt{cEET_s} = \hat{c};\ \mathtt{ex_{EET_s}};\ \{\mathtt{cEET_s}\}\rangle$
$\mathtt{cET_s} < \hat{c} \wedge \mathtt{cET_e} < \hat{c} \wedge$
$\mathtt{cA} - \mathtt{cET_s} \geq 0 \wedge \mathtt{cA} - \mathtt{cET_s} \leq 5 \wedge$
$\mathtt{cE} - \mathtt{cET_e} \geq 7 \wedge \mathtt{cE} - \mathtt{cET_e} \leq 14$
$\mathtt{cA} = \hat{c}$
$\mathtt{bP} < \hat{c}$
$\mathtt{bQ} = \hat{c}$
$\mathtt{bQ} < \hat{c}$
$\mathtt{bP} < \hat{c}$
$\mathtt{cA} = \hat{c}$
$\mathtt{cEET_s} < \hat{c} \wedge \mathtt{cEET_e} < \hat{c} \wedge$
$\mathtt{cA} - \mathtt{cEET_s} \geq 0 \wedge \mathtt{cA} - \mathtt{cEET_s} \leq 10 \wedge$
$\mathtt{cE} - \mathtt{cEET_e} \geq 10 \wedge \mathtt{cE} - \mathtt{cEET_e} \leq 25$
$\mathtt{bP} = \hat{c}$
$\mathtt{cST_s} < \hat{c} \wedge \mathtt{cST_e} < \hat{c} \wedge$
$\mathtt{cE} - \mathtt{cST_s} \geq 0 \wedge \mathtt{cE} - \mathtt{cST_s} \leq 10$
$\mathtt{cEET_s} < \hat{c} \wedge \mathtt{cEET_s} > 20 \wedge \mathtt{cEET_e} = \hat{c}$
$\mathtt{cET_s} < \hat{c} \wedge \mathtt{cET_s} > 10 \wedge \mathtt{cET_e} = \hat{c}$
$\mathtt{cST_s} < \hat{c} \wedge \mathtt{cST_s} > 30 \wedge \mathtt{cST_e} = \hat{c}$

Fig. 13 The TGA derived from the sample CSTNU from Fig. 4. In transitions leading from `vera` to `goal`, and from `agnes` to `goal`, the names and clock resets have been omitted

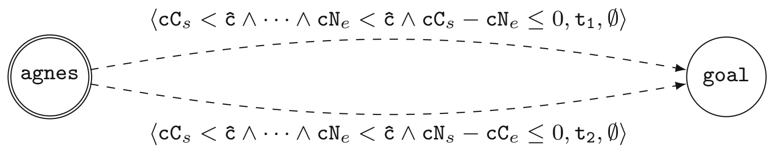

$\langle \mathtt{cC}_s < \hat{\mathtt{c}} \wedge \cdots \wedge \mathtt{cN}_e < \hat{\mathtt{c}} \wedge \mathtt{cC}_s - \mathtt{cN}_e \leq 0, \mathtt{t}_1, \emptyset \rangle$
agnes
goal
$\langle \mathtt{cC}_s < \hat{\mathtt{c}} \wedge \cdots \wedge \mathtt{cN}_e < \hat{\mathtt{c}} \wedge \mathtt{cN}_s - \mathtt{cC}_e \leq 0, \mathtt{t}_2, \emptyset \rangle$

Fig. 14 The DNF encoding of the guards on constraints from `agnes` to `goal` for the sample DTNU in Fig. 5

$X - Y \leq \delta$, with the equivalent clock constraint, $\mathtt{cY} - \mathtt{cX} \leq \delta$, while preserving the Boolean structure of the formula:[7]

$$\Omega(\mathbf{c}, \mathtt{cG}) \doteq \Lambda[(X - Y \leq \delta)/(\mathtt{cY} - \mathtt{cX} \leq \delta)].$$

For the DTNU from Fig. 1 that represents the non-overlapping cardiological and neurological evaluation tasks, Λ consists solely of the one constraint in $\mathcal{C}$:

$$\Lambda \doteq (N_e - C_s \leq 0) \vee (C_e - N_s \leq 0).$$

Therefore, $\Omega(\mathtt{cC}_s, \mathtt{cC}_e, \mathtt{cN}_s, \mathtt{cN}_e, \mathtt{cG})$ is equal to:

$$(\mathtt{cC}_s - \mathtt{cN}_e \leq 0) \vee (\mathtt{cN}_s - \mathtt{cC}_e \leq 0).$$

However, it is not always possible to directly use the formula $\Omega(\mathbf{c}, \mathtt{cG})$ as a guard for the transition from `agnes` to `goal` because the definition of a TGA restricts the language of the guards to be purely conjunctive. For this reason, we aim at building a piece of automaton—possibly adding new locations—that connects `agnes` to `goal` in such a way that the free constraints are equivalent to the disjunction of the conjunction of the guards along each path from `agnes` to `goal`. There are several ways in which this can be done.

Disjunctive Normal Form. Similarly to what has been done for CSTNUs, we can create a set of transitions from `agnes` to `goal` such that each pathway from `agnes` to `goal` can be taken if and only if $\Omega(\mathbf{c}, \mathtt{cG})$ is satisfied. This is always possible, since alternative transitions emanating from a single location are equivalent to a single transition with a disjunctive guard. Thus, all we have to do is convert $\Omega(\mathbf{c}, \mathtt{cG})$ into Disjunctive Normal Form (DNF) and create a separate transition from `agnes` to `goal` for every disjunct. In this setting, negation of atomic constraints is not a problem because $\neg(\mathtt{cY} - \mathtt{cX} \leq \delta)$ is equivalent to $\mathtt{cY} - \mathtt{cX} > \delta$, which is allowed in the guards of a TGA. As for the names of the actions, we assign to each action a unique new name. It is easy to see that there exists a path from `agnes` to `goal` if and only if the free constraints are satisfied, because one disjunct of the DNF is satisfied. The main drawback of this technique is that, for a general formula, the number of disjuncts in the DNF is exponential, and thus the encoding is exponential. Nevertheless, this constitutes a sound-and-complete encoding for (constructively) deciding the dynamic controllability of DTNU.

Considering again the running example, the encoding of the constraints between `agnes` and `goal` is composed of only two transitions, as Ω is already in DNF. The transitions are depicted in Fig. 14.

[7] Here we use the classical notation $\phi[x/y]$ for the substitution of the term x for the term y in the formula ϕ. However, we slightly abuse the notation by assuming that the substitution is applied to all atoms of the formula.

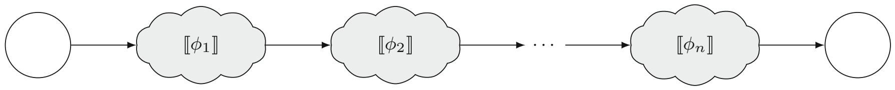
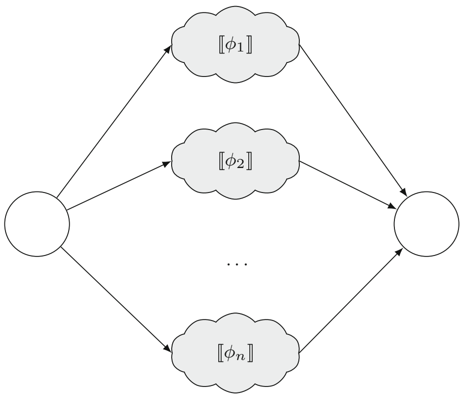


Fig. 15 Encoding the conjunction $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$. $[\![\phi]\!]$ stands for the recursive encoding of ϕ

Fig. 16 Encoding the conjunction $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$. $[\![\phi]\!]$ stands for the recursive encoding of ϕ



Negative Normal Form. If we allow for the introduction of new (urgent) locations in the TGA, we can encode $\Omega(\mathbf{c}, \mathtt{cG})$ linearly, thus obtaining a linear size of the overall DTNU-to-TGA encoding. The idea comes from the following observation. Suppose we have a piece of automaton that encodes a formula ϕ_1 in such a way that it is possible to move from location L_1^s to L_1^e if and only if ϕ_1 is satisfied, and suppose that we have an analogous encoding for another formula ϕ_2 with starting and ending locations L_2^s to L_2^e. We can encode the formula $\phi_1 \wedge \phi_2$ by "concatenating" the two automata. That is, we introduce a transition from L_1^e to L_2^s with the tautological guard $\top$. Now, in order to move from L_1^s to L_2^e the formula $\phi_1 \wedge \phi_2$ must be satisfied.[8] Similarly, if we consider the formula $\phi_1 \vee \phi_2$ we can introduce two extra locations $L_\vee^s$ and $L_\vee^e$ and introduce four transitions with the guard $\top$: one from $L_\vee^s$ to L_1^s, one from $L_\vee^s$ to L_2^s, one from L_1^e to $L_\vee^e$, and one from L_2^e to $L_\vee^e$. In this way, we create a "diamond" with two paths from $L_\vee^s$ to $L_\vee^e$; one path encodes ϕ_1, the other encodes ϕ_2. This construction is simple and correct, even though it introduces many unneeded locations. In fact it is also possible to compress this encoding by merging locations instead of linking them with tautological transitions and it is possible to merge sequences of guards in a single conjunctive guard. However, we decided to explain the simplest version for clarity.

Given a rewriting of $\Omega(\mathbf{c}, \mathtt{cG})$ that only has disjunctions and conjunctions (but no negations) we can recursively create a piece of automaton that encodes the formula. This is done by constructing an automaton in which disjunctions and conjunctions are recursively encoded as shown in Figs. 15 and 16. It is well known [27] that we can syntactically and linearly transform $\Omega(\mathbf{c}, \mathtt{cG})$ into Negative Normal Form (NNF) and transform the negations of the atoms into positive atoms as before, by exploiting the fact that $\neg(\mathtt{cY} - \mathtt{cX} \leq \delta)$ is equivalent to $\mathtt{cY} - \mathtt{cX} > \delta$.

Figure 17 depicts the running example encoded using the NNF decomposition of the free constraints optimized by compressing the conjunction of the $\mathtt{cX} < \mathtt{cG}$ in a single guard and by avoiding the unneeded tautological guards. In the running example, Ω is already in NNF as there are no negations.

[8] We are assuming that all the locations of the automata pieces are urgent, so the clocks are frozen and no time can elapse.

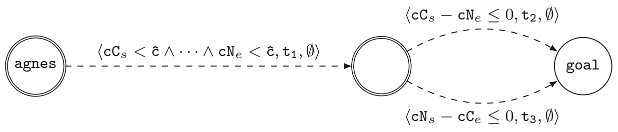

agnes
⟨cC_s < ĉ ∧ ··· ∧ cN_e < ĉ, t_1, ∅⟩
⟨cC_s − cN_e ≤ 0, t_2, ∅⟩
⟨cN_s − cC_e ≤ 0, t_3, ∅⟩
goal

Fig. 17 NNF constraints between `agnes` and `goal` for the sample DTNU in Fig. 5

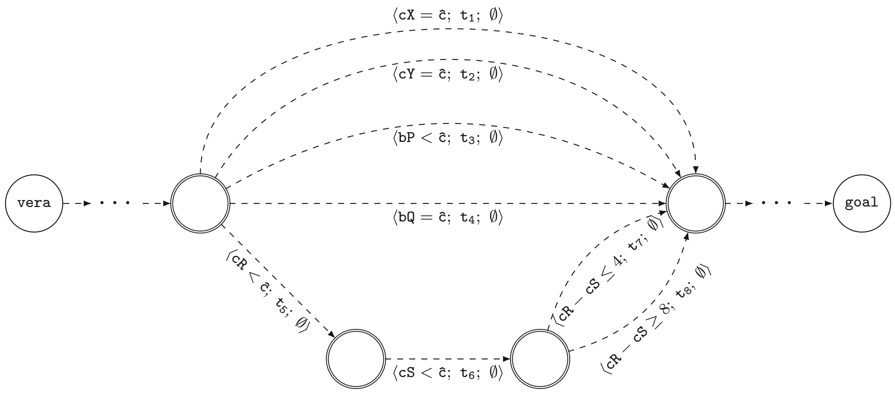


Fig. 18 Encoding of the example disjunctive labeled free constraint in TGA

Even though the NNF decomposition is always more succinct than the DNF, the effort of dealing with disjunctions is moved from the encoding to the TGA solver, so we are in a trade-off condition.

4.2.4 CDTNU-to-TGA encoding

The encoding of a CDTNU into a TGA is accomplished by combining the techniques used in the previous encodings. The main observation is that the encodings of contingent constraints and observation time-points can be combined without any modifications; however, there needs to be a way of constructing a path leading to the `goal` location such that that path can be traversed if and only if all of the free constraints are satisfied and all of the relevant time-points have been executed.

To do so, the features from the CSTNU and DTNU encodings are combined, as follows. First, each propositional letter p is given a dedicated clock `bP`. Next, note that Boolean labels are essentially logical implications for constraints. In particular, if a constraint ϕ is labeled with $\ell_1, \ell_2 \cdots, \ell_n$ from P^*, then that constraint can be expressed as a single implication $\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n \rightarrow \phi$. This follows from the semantics of dynamic controllability for CSTNUs: a labeled constraint need only be satisfied in scenarios in which its label is true. Given this observation, an appropriate piece of automaton can be built using the NNF technique described above, where implications are rewritten as disjunctions (e.g., $A \rightarrow B \equiv \neg A \vee B$), each positive literal p is encoded as $\texttt{bP} = \hat{\texttt{c}}$, and each negative literal $\neg p$ is encoded as $\texttt{bP} < \hat{\texttt{c}}$.

For example, consider a disjunctive constraint, $(S - R \leq 4) \vee (S - R \geq 8)$ labeled by $p\neg q$, where the observation time-point X generates a truth value for p, and Y generates a truth value for q. That constraint is logically equivalent to the following implication:

$$(p \wedge \neg q) \rightarrow ((S - R \leq 4) \vee (S - R \geq 8)).$$

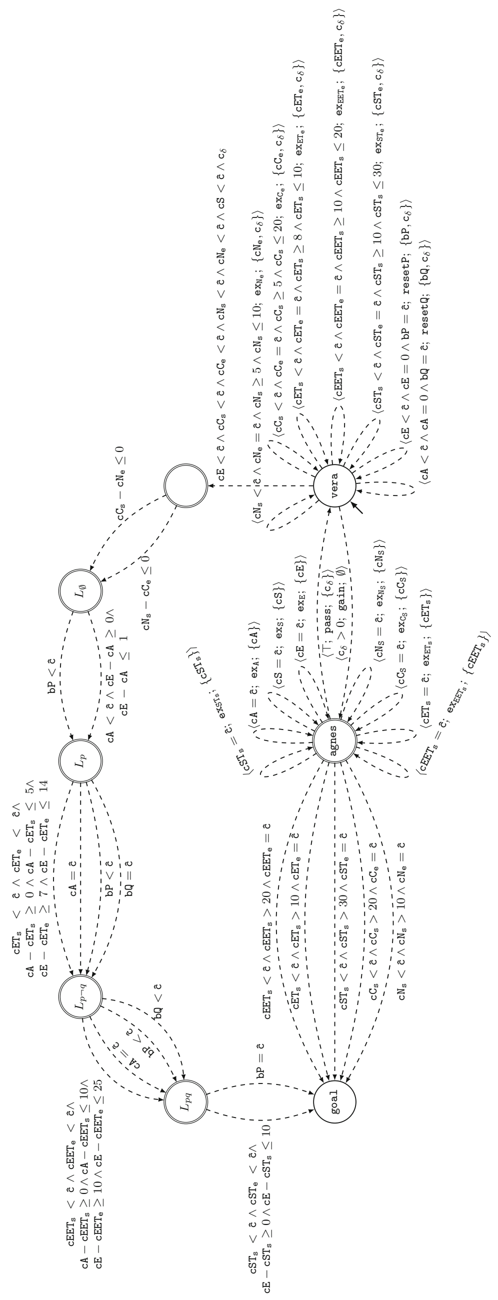


Fig. 19 TGA encoding of the CDTNU in Fig. 6. In transitions leading from vera to goal and from agnes to goal the names and resets have been omitted

This formula can translated into a TGA using the NNF technique, as follows.

$$((\mathtt{cX} = \hat{\mathtt{c}}) \vee (\mathtt{cY} = \hat{\mathtt{c}}) \vee (\mathtt{bP} < \hat{\mathtt{c}}) \vee (\mathtt{bQ} = \hat{\mathtt{c}})) \vee$$
$$((\mathtt{cR} < \hat{\mathtt{c}}) \wedge (\mathtt{cS} < \hat{\mathtt{c}}) \wedge ((\mathtt{cR} - \mathtt{cS} \leq 4) \vee (\mathtt{cR} - \mathtt{cS} \geq 8)))$$

This formula results in the automaton structure shown in Fig. 18 by applying the construction explained in Sect. 4.2.3. The CDTNU example presented in Fig. 6 is encoded into the TGA shown in Fig. 19.

5 Conclusion

This paper presented a summary of a variety of temporal network formalisms that have been widely used to represent different kinds of temporal constraints and related information. The paper combined all of the features from those networks into a single, unifying formalism, called a Conditional Disjunctive Temporal Network with Uncertainty (CDTNU). It then presented a way of encoding the dynamic controllability problem for any CDTNU into a reachability game for a linear-size Timed Game Automaton, thereby generating, for the first time, a sound and complete algorithm for determining the dynamic controllability of CSTNUs, DTNUs and CDTNUs.

Strategy extraction. One interesting characteristic of the algorithms for checking TGA reachability [6] is the possibility of obtaining a strategy for winning the game or an unbeatable counter-strategy for the opponent. This feature is useful also in the context of temporal networks, as a counter-strategy for the TGA obtained from a network, corresponds to the strategy for scheduling the controllable time-points required by the dynamic controllability problem definition. In practice, one can use the counter-strategy generated from the TGA encoding to schedule time-points, as each controllable time point has a single transition associated with it: when the counter-strategy prescribes to take that transition, then it is time to schedule the time point.

CDTNU and temporal workflows models. As discussed in Sects. 1.1 and 2, in the literature there are different formal proposals about how to extend workflow models or process-aware-system models in order to allow them to represent and manage significant kinds of temporal aspects [8,14,16,17,20,28,29]. Each of such formal models defines which kinds of temporal aspects/constraints can be represented in each component of the model and characterizes how any workflow/process instance execution has to satisfy the specified temporal constraints in order to be a successful execution of the instance. In other words, each formal model proposes an extension of the temporal consistency/controllability concept at model level. Moreover, some of such proposals presents also algorithms to verify the consistency/controllability of a workflow/process instance. Such algorithms usually transform the input workflow/process instance into an equivalent or quasi-equivalent STN/STNU /CSTNU instance in order to exploit the well-known consistency/controllability checking algorithms for such temporal models.

However, this translation is not always straightforward, because the temporal behavior of some workflow patterns can be represented only by using disjunctive or conditional constrains that cannot be represented as STN/STNU /CSTNU sub-networks. For example, temporal aspects regarding the workflow *temporized parallel join connector* or the workflow *multiple temporized receive pattern* [21] have to be expressed as disjunctive constraints [13]. In related work [13,28], the authors proposed to override the CSTNU limitation concerning disjunctive constraints by considering and checking multiple CSTNU sub-networks (two for each translated join connector/multiple temporized receive pattern) for determining some upper bounds that have to be used in the final CSTNU translation. Another approach [18] uses *Hyper Temporal Networks (HyTNs),* an extension of STNs, in order to directly represent these kind of disjunctive constraints while maintaining an efficient consistency check. Currently, HyTNs cannot represent conditional or contingent constraints.

For such workflow/process-aware models, the adoption of the CDTNU model as the internal temporal model would allow a significant simplification of the translation phase because the above workflow patterns can be directly represented without any preliminary analysis or restrictions. Moreover, CDTNU adoption allows the representation of temporal characterizations in other more sophisticated workflow patterns. Such patterns have not yet been considered [30] due to the difficulty of representing their temporal features only using STNs, STNUs or CSTNUs.

Future work. There are many avenues for future work. First, there needs to be an extensive empirical evaluation of our approach, especially since there are many different options for translating CSTNUs and DTNUs into TGAs. Which options will yield the most efficient DC-checking algorithm is an open question. Moreover, given the peculiar nature of the encoding, it may be possible to specialize TGA-solving algorithms using dedicated heuristics or pruning techniques that exploit the particular features of the CDTNU-to-TGA encoding.

Appendix 1: The semantics of dynamic controllability for STNUs

Although the intuitive description of the execution semantics for STNUs given in Sect. 3.1 makes reference to both the agent and the environment, formal treatments of the execution semantics have so far only defined execution strategies for the agent; strategies available to the environment have only been implicitly determined by the sets of possible outcomes of the agent's decisions [22,35]. Thus, the semantics of dynamic controllability for STNUs has effectively described a one-player game where the outcomes of the agent's decisions are non-deterministic. This appendix introduces a novel formulation of the execution semantics for STNUs as a two-player game between Agnes (the agent) and Vera (the environment), where Agnes controls the execution of free time-points and Vera controls the contingent durations. Agnes seeks an execution strategy that will ensure the satisfaction of all constraints in $\mathcal{C}$ no matter what durations Vera chooses; Vera seeks a strategy that will ensure that at least one constraint in $\mathcal{C}$ is *unsatisfied* no matter what Agnes does. As will be seen, this formulation highlights an important asymmetry in the execution semantics: Agnes is *not* able to react instantaneously to observations of contingent time-points executing, but Vera *is* able to react instantaneously to executions of free time-points.

Previous semantics for the dynamic controllability of STNUs

The literature contains two equivalent versions of the semantics of dynamic controllability of STNUs [22,35]. This section summarizes the version presented by Hunsberger [22], which is expressed in terms of *real-time execution decisions* (RTEDs). For convenience, the following description presumes an agent named Agnes.

To begin, a *partial schedule* represents the current state of affairs from the agent's perspective—namely, the time-points that have been executed so far.

Definition 9 (*Partial Schedule* [22]) A partial schedule for an STNU, $(\mathcal{T}, \mathcal{C}, \mathcal{L})$, is a set, ψ, of assignments to time-points in $\mathcal{T}$.

- $TPs(\psi) \subset \mathcal{T}$ denotes the set of time-points appearing in ψ;
- $Vals(\psi) \subset \mathbb{R}$ denotes the set of values appearing in ψ;
- for any $X \in TPs(\psi)$, $\psi(X)$ denotes the value assigned to X; and
- $\texttt{now}_\psi = \max\{v \mid v \in Vals(\psi)\}$ is the time of the latest execution event in ψ. (If $\psi = \emptyset$, let $\texttt{now}_\psi = -\infty$.)

Time-points in $TPs(\psi)$ are said to be *executed.* A partial schedule is called *respectful* if its assignments do not violate the bounds on any contingent link.

Intuitively, a partial schedule ψ assigns a real value to a subset of the time-points in the network, and represents an execution history: the (free or uncontrollable) time-points in ψ are the ones that have already been executed, and the assigned values are the times at which they were executed. The time-points that are not in ψ are the ones that have not yet been executed.

Given a partial schedule ψ, Agnes must decide what to do next. She has two options: (1) wait for something to happen (i.e., wait for some contingent time-point to execute); or (2) conditionally commit to executing a set of free time-points at some time, $T_f > \texttt{now}_\psi$. For example, given $\psi = \{(A_2, 0), (X, 1)\}$, for which $\texttt{now}_\psi = 1$, Agnes could decide to wait until the contingent time-point C_2 eventually executes. Alternatively, she could decide that "if nothing happens before time 7, I shall execute A_1 at time 7." The decisions available to Agnes are called *real-time execution decisions* (RTEDs).

Definition 10 (*RTED, for Agnes* [22]) Let ψ be a respectful partial schedule. An RTED for Agnes has one of two forms: $\texttt{wait}$ or (T_f, χ_f). A $\texttt{wait}$ decision is applicable if at least one contingent time-point, C, is *active* in ψ (i.e., C's activation time-point has already been executed, but C has not). A (T_f, χ_f) decision (i.e., "If nothing happens *before* time T_f, execute the time-points in χ_f at time T_f") is applicable if $T_f > \texttt{now}_\psi$ and χ_f is a non-empty subset of unexecuted free time-points (i.e., $\chi_f \neq \emptyset$ and $\chi_f \cap TPs(\psi) = \emptyset$).

Given a partial schedule ψ and some RTED Δ, the outcome of the decision Δ typically depends on the range of possible durations for one or more contingent links, as follows.

Definition 11 (*Situations* [35]) Let $\mathcal{L} = \{(A_1, \ell_1, u_1, C_1), \ldots, (A_k, \ell_k, u_k, C_k)\}$ be the set of contingent links in a given STNU $\mathcal{S}$. Then the *space of situations* for $\mathcal{S}$ is the set, $\Omega = [\ell_1, u_1] \times [\ell_2, u_2] \times \ldots \times [\ell_k, u_k]$; and any $\omega = (\omega_1, \omega_2, \ldots, \omega_k) \in \Omega$ is called a *situation.* A situation ω is *respected* by a partial schedule ψ if the durations specified in ω are consistent with not only the execution times in ψ, but also the constraint that all time-points that are unexecuted in ψ must occur after $\texttt{now}_\psi$ [22].

Note that if ψ is a partial schedule that respects a situation ω, and $A_i \in TPs(\psi)$, but its corresponding contingent time-point $C_i \notin TPs(\psi)$ (i.e., C_i is active in ψ), then it follows that $\texttt{now}_\psi < \psi(A_i) + \omega_i$, since C_i must be executed after $\texttt{now}_\psi$.

Definition 12 (*Outcome of a* $\texttt{wait}$ *decision* [22]) Let ψ be a partial schedule for which at least one contingent time-point is active, and let ω be a situation that is respected by ψ. The outcome of the $\texttt{wait}$ decision in that context depends on: (1) $tnc(\psi, \omega)$, the time of the *next* contingent execution according to ψ and ω; and (2) $\chi^*(\psi, \omega)$, the set of contingent time-points that will execute *next* (i.e., at the time $tnc(\psi, \omega)$). In particular:

$$tnc(\psi, \omega) = \min\{\psi(A_i) + \omega_i \mid A_i \in TPs(\psi), C_i \notin TPs(\psi)\}; \text{ and}$$

$$\chi^*(\psi, \omega) = \{C_i \mid A_i \in TPs(\psi), C_i \notin TPs(\psi), \psi(A_i) + \omega_i = tnc(\psi, \omega)\}.$$

The outcome of the `wait` decision is notated $\mathcal{O}(\psi, \omega, \texttt{wait})$ and is given by:

$$\mathcal{O}(\psi, \omega, \texttt{wait}) \;=\; \psi \;\cup\; \{(C_i, tnc(\psi, \omega)) \mid C_i \in \chi^*(\psi, \omega)\}$$

Definition 13 (*Outcome of a* (T_f, χ_f) *Decision* [22]) Let ψ be a partial schedule for which at least one free time-point is unexecuted, and let ω be a situation that is respected by ψ. For convenience, let $t = tnc(\psi, \omega)$ (or $t = \infty$ if no contingent time-points are active in ψ), and let $\chi^* = \chi^*(\psi, \omega)$. The outcome of a (T_f, χ_f) decision in that context, notated $\mathcal{O}(\psi, \omega, (T_f, \chi_f))$, depends on the relationship between t and T_f. In particular:

$$\mathcal{O}(\psi, \omega, (T_f, \chi_f)) \;=\; \psi \;\cup\; \begin{cases} \{(C_i, t) \mid C_i \in \chi^*\}, & \text{if } t < T_f \\ \{(X, t) \mid X \in \chi_f\}, & \text{if } T_f < t \\ \{(Y, t) \mid Y \in \chi_f \cup \chi^*\}, & \text{if } T_f = t \end{cases}$$

In the first case, some contingent time-points happened to execute before the time T_f arrived; in the second case, only the time-points in χ_f were executed; in the third case, rarely expected in practice, some contingent time-points happened to execute precisely at the time T_f and, thus, both contingent and free time-points were executed simultaneously.

Definition 14 (*RTED-based Strategy* [22]) An *RTED-based strategy* for an STNU $\mathcal{S}$ is a mapping R from respectful partial schedules to real-time execution decisions. Thus, if ψ is a respectful partial schedule, then $R(\psi)$ is an RTED.

Lemma 1 *If R is an RTED-based strategy for an STNU $\mathcal{S}$, and ω is any situation, then R and ω together determine a unique (complete) schedule, notated $\psi(R, \omega)$, that results from following the strategy R in the situation ω [22].*

Definition 15 (*Dynamic Controllabilty for an STNU*) An STNU $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ is dynamically controllable if there exists an RTED-based strategy R for $\mathcal{S}$ such that for each situation ω, the complete schedule $\psi(R, \omega)$ that results from following the strategy R satisfies all of the constraints in $\mathcal{C}$.

Dynamic controllability for STNUs as a two-player game

This section provides an alternative characterization of the semantics of dynamic controllability for STNUs by explicitly representing the decisions available to the environment. For convenience, the environment is represented by an agent Vera. In any given context, a pair of decisions—one by Agnes and one by Vera—together determine a unique outcome.

The kinds of decisions available to Vera are different from those available to Agnes in two important respects. First, Vera's version of an RTED—called an RTED*—allows a decision of the form, "if nothing happens before *or at* time T_u, then I shall execute the contingent time-points in the set $\chi_u \subseteq \mathcal{T}_u$ at time T_u." Note that when time T_u arrives, should Vera observe Agnes executing any time-points *at time* T_u, Vera has the option of *instantaneously changing her mind.* Second, in such cases, Vera may *instantaneously react* by executing some other contingent time-points *at time* T_u. Such decisions are called *instantaneous reactions.* For example, suppose Vera had decided that "if nothing happens before *or at* time 7, then I shall execute C_2 at time 7", but when time 7 arrived, she observed Agnes executing some time-point(s). Vera could withdraw her decision to execute C_2 and instantaneously react by deciding to execute some other contingent time-point(s) *at time* 7.

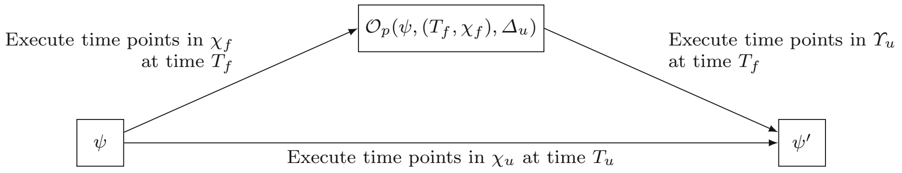

$\mathcal{O}_p(\psi, (T_f, \chi_f), \Delta_u)$
Execute time points in χ_f at time T_f
Execute time points in Υ_u at time T_f
ψ
ψ'
Execute time points in χ_u at time T_u

Fig. 20 Deriving the outcome ψ' of decisions by Agnes and Vera from the partial schedule ψ

Definition 16 (RTED*, *for Vera*) Let ψ be a respectful partial schedule. A *before-or-at RTED* (RTED*) has one of two forms: `wait` or (T_u, χ_u). A `wait` decision is only applicable if no contingent time-points are currently active in ψ. A (T_u, χ_u) decision (i.e., "If nothing happens *before-or-at* time T_u, I shall execute the time-points in χ_u at time T_u") is applicable only if $T_u > \texttt{now}_\psi$, and χ_u is a non-empty subset of currently-activated contingent time-points each of whose execution window includes T_u; and all other contingent time-points that are unexecuted in ψ are either unactivated in ψ or have execution windows that extend beyond T_u.

Definition 17 (*Instantaneous reaction, for Vera*) Let ψ be a respectful partial schedule. Let χ° be the set of contingent time-points that are currently active in ψ whose execution windows happen to *terminate precisely* at $\texttt{now}_\psi$; and let $\chi^\star$ be any (possibly empty) subset of the contingent time-points that are currenlty active in ψ whose execution windows include $\texttt{now}_\psi$, but also extend beyond $\texttt{now}_\psi$. An *instantaneous reaction* is a decision (by Vera) to execute the contingent time-points in the set $\chi^\circ \cup \chi^\star$ at the time $\texttt{now}_\psi$.

To accommodate Vera's ability to react instantaneously, the outcome for a pair of decisions—one by Agnes, one by Vera—is defined in two stages: partial and full.

Definition 18 (*Partial Outcome*) Let ψ be a respectful partial schedule; let Δ_f be an RTED for Agnes; and let Δ_u be an RTED* for Vera. The *partial outcome,* $\mathcal{O}_p(\psi, \Delta_f, \Delta_u)$, is defined as follows.[9]

(1a) $\mathcal{O}_p(\psi, \texttt{wait}, (T_u, \chi_u)) = \psi \cup \{(C, T_u) \mid C \in \chi_u\}$.
(1b) $\mathcal{O}_p(\psi, (T_f, \chi_f), (T_u, \chi_u)) = \psi \cup \{(C, T_u) \mid C \in \chi_u\}$, if $T_u < T_f$.
(2a) $\mathcal{O}_p(\psi, (T_f, \chi_f), \texttt{wait}) = \psi \cup \{(X, T_f) \mid X \in \chi_f\}$.
(2b) $\mathcal{O}_p(\psi, (T_f, \chi_f), (T_u, \chi_u)) = \psi \cup \{(X, T_f) \mid X \in \chi_f\}$, if $T_f \leq T_u$.

Note that in cases (1a) and (1b), the partial outcome includes only the execution of the contingent time-points in χ_u at time T_u. Cases (2a) and (2b) are analogous, in that the partial outcome includes only the execution of the free time-points in χ_f at time T_f, except that Vera is also able to instantaneously react by executing one or more contingent time-points, *also at time* T_f, as described below.

Definition 19 (*Full Outcome*) Let $\psi_p = \mathcal{O}_p(\psi, \Delta_f, \Delta_u)$ be a partial outcome, as described above; and let Υ_u be a set of contingent time-points that constitute an instantaneous reaction to ψ_p. The *full outcome*, $\mathcal{O}(\psi, \Delta_f, \Delta_u, \Upsilon_u)$, is the same as ψ_p, except that in cases (2a) and (2b), the schedule is augmented to include the execution of the time-points in Υ_u at time T_f.

Figure 20 illustrates the possible pathways from a partial schedule ψ to the full outcome $\psi' = \mathcal{O}(\psi, \Delta_f, \Delta_u, \Upsilon_u)$. Note that $\texttt{now}_{\psi'}$ is either T_f or T_u, depending on which pathway

[9] Note that a `wait` decision cannot be simultaneously applicable for both Agnes and Vera.

Table 1 The outcomes ψ' for sample decisions by Agnes and Vera for the STNU from Fig. 3

$\psi = \{(A_2, 0), (X, 1)\}$; $\Delta_f = (7, \{A_1\})$; $\Delta_u = (6, \{C_2\})$
$\psi' = \{(A_2, 0), (X, 1), (C_2, 6)\}$; Υ_u irrelevant
$\psi = \{(A_2, 0), (X, 1)\}$; $\Delta_f = (7, \{A_1\})$; $\Delta_u = (8, \{C_2\})$
$\psi' = \{(A_2, 0), (X, 1), (A_1, 7), (C_2, 7)\}$, where $\Upsilon_u = \{C_2\}$
$\psi = \{(A_2, 0), (X, 1)\}$; $\Delta_f = (7, \{A_1\})$; $\Delta_u = (8, \{C_2\})$
$\psi' = \{(A_2, 0), (X, 1), (A_1, 7)\}$, where $\Upsilon_u = \emptyset$

is taken. Note, too, that the full outcome, ψ', is typically a partial schedule, except at the very end when all of the time-points have been executed. Table 1 shows the outcomes that result from sample decisions by Agnes and Vera in the case of the STNU from Fig. 3. In each case, $\psi' = \mathcal{O}(\psi, \Delta_f, \Delta_u, \Upsilon_u)$.

Definition 20 (RTED*-*based Strategy for Vera*) An *RTED*-based strategy* (for Vera) is a pair of mappings, (f_1, f_2), where f_1 is a mapping from respectful partial schedules to RTED*s; and f_2 is a mapping from respectful partial schedules to instantaneous reactions.

Definition 21 (*Outcomes of Strategy Pairs*) Let ψ be a respectful partial schedule; R an RTED-based strategy; and $R^\star = (f_1, f_2)$ an RTED*-based strategy. The one-step outcome, $\mathcal{O}^1(\psi, R, R^\star)$, is defined by:

$$\mathcal{O}^1(\psi, R, R^\star) = \mathcal{O}(\psi, R(\psi), f_1(\psi), f_2(\mathcal{O}_p(\psi, R(\psi), f_1(\psi)))).$$

The *terminal outcome, $\mathcal{O}^*(R, R^\star)$*, is the *complete* schedule that results from the following recursive definition: $\psi_0 = \emptyset$ and $\psi_{i+1} = \mathcal{O}^1(\psi_i, R, R^\star)$.

The constraints on the decisions generated by $R^\star$—namely, that Vera must observe the bounds on the contingent durations—ensure that each ψ_i in the sequence will be respectful, given that $\psi_0 = \emptyset$ is trivially respectful.

Given the above execution semantics for STNUs, the corresponding definition of dynamic controllability is straightforward.

Definition 22 (*Dynamic Controllability*) An STNU, $(\mathcal{T}, \mathcal{C}, \mathcal{L})$, is *dynamically controllable* if there exists an RTED-based strategy R, such that for *all* RTED*-based strategies $R^\star$, the variable assignments in the complete schedule, $\mathcal{O}^*(R, R^\star)$, satisfy all of the constraints in $\mathcal{C}$.

Theorem 1 *Definition 22 is equivalent to the prior definition of dynamic controllability (Definition 15).*

Proof Let $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ be any STNU. First, suppose that $\mathcal{S}$ is dynamically controllable according to the RTED-based semantics. Then there exists an RTED-based execution strategy R such that for any situation ω, the full schedule that results from following R in ω satisfies all of the constraints in $\mathcal{C}$. Let that strategy R be the one chosen by Agnes in the two-player game semantics. Let $R^* = (f_1, f_2)$ be any strategy for Vera. It will be shown that the terminal outcome $\mathcal{O}^*(R, R^*)$ that results from Agnes and Vera playing these two strategies against each other necessarily satisfies the constraints in $\mathcal{C}$. In particular, it will be shown by induction that each (partial or full) schedule obtained at any point during the execution phase by following R and R^* according to the two-player game semantics can also be obtained by following R in some situation in the RTED-based semantics.

Base Case. Let ψ_0 be the empty partial schedule. This is the starting partial schedule in either semantics.
Recursive Case. Let ψ be any partial schedule obtained by following R and R^* in the two-player game semantics. There are three sub-cases to consider.

- $R(\psi) = \texttt{wait}$; $R^*(\psi) = (T_u, \chi_u)$. In this case, the partial outcome involves the execution of the contingent time-points in χ_u at the time T_u. Since the applicability conditions for Vera's RTED* decision requires the execution times for contingent time-points to respect the lower and upper bounds on the corresponding contingent links, the resulting partial outcome is a partial schedule obtainable from any situation ω that is respected by ψ and includes the durations specified by the contingent time-points in χ_u.
- $R(\psi) = (T_f, \chi_f)$; $R^*(\psi) = (T_u, \chi_u)$, where $T_u < T_f$. This case is essentially the same as the first case, since $T_u < T_f$.
- $R(\psi) = (T_f, \chi_f)$; $R^*(\psi) = \texttt{wait}$. In this case, the partial outcome involves the execution of the executable time-points in χ_f. Since Vera can only use the `wait` decision when the partial schedule ψ does not contain any currently active contingent links, this must be the outcome in the RTED-based semantics, too. There can be no instantaneous reaction by Vera in this case.
- $R(\psi) = (T_f, \chi_f)$; $R^*(\psi) = (T_u, \chi_u)$, where $T_f \leq T_u$. This case is the same as the preceding case except that Vera may choose to react instantaneously (i.e., $f_2(\psi)$ may not be empty). The applicability conditions of instantaneous reactions require the contingent time-points in $f_2(\psi)$ to be currently active in ψ, and such that their execution windows include the time $\texttt{now}_\psi$. In addition, any contingent time-points that happen to have their execution window terminate precisely at $\texttt{now}_\psi$ must be included in $f_2(\psi)$. Thus, the full outcome is the same as in the preceding case except that some contingent time-points may also execute at the time T_f. Again, this corresponds to any situation ω that is respected by ψ, while also respecting the contingent durations determined by the executuion of the contingent time-points in $f_2(\psi)$.

For the other direction, suppose that $\mathcal{S}$ is *not* dynamically controllable according to the RTED-based semantics. In other words, for any RTED-based strategy R, there is a situation ω_R such that the outcome $\mathcal{O}^*(R, \omega_R)$ that results from following the strategy R in the situation ω_R does *not* satisfy the constraints in $\mathcal{C}$. (Any situation with this property will be said to *thwart* the strategy R.) It will be shown that there must be a strategy $R^* = (f_1, f_2)$ for Vera that will ensure that Agnes loses the two-player game. The proof is by induction. The proposition to prove is the following:

> Let ψ be any partial schedule that can be reached by following any RTED-based strategy R in any thwarting situation ω_R, according to the RTED-based semantics. Then there is an RTED* decision Δ_u (that depends only on ψ, not on R) and an instantaneous reaction Υ_u for Vera such that the full outcome obtained from $R(\psi)$, Δ_u and Υ_u according to the two-player game semantics is a schedule that is identical to one obtained by following R in some thwarting situation ω_R.

Let ψ be a partial schedule that can be reached by following some RTED-based execution strategy R in some thwarting situation ω_R, according to the RTED-based semantics. Now, if no contingent time-points are currently active in ψ, then Agnes must choose a (T_f, χ_f) decision, and Vera must choose the `wait` decision. But in that case, the outcome is fully determined: the time-points in χ_f will be executed at time T_f. Furthermore, the outcome is the same whether using the RTED-based semantics or the two-player game semantics.

On the other hand, suppose that at least one contingent time-point is currently active in ψ. Let Θ_ψ be the set of RTED-based execution strategies for Agnes that can generate the partial schedule ψ at some point during the execution of the network, if followed in some thwarting situation. For each $t > \mathtt{now}_\psi$, let $\Theta(t)$ be the subset of Θ_ψ that contains all strategies θ whose decisions, $\theta(\psi)$, specify execution times *greater than* t. Now, for any strategy $\theta \in \Theta(t)$, there must be a situation ω_θ that thwarts θ; however, that situation may involve the execution of contingent time-point(s) at some time *before* t (i.e., at some time ρ, where $\mathtt{now} < \rho < t$). Of particular interest are the values of $t > \mathtt{now}_\psi$ for which *all* of the strategies in $\Theta(t)$ can be thwarted by situations that do not involve executing any contingent time-points *before* time t. In particular, let Γ be the set of real numbers $t > \mathtt{now}_\psi$ for which *every* strategy $\theta \in \Theta(t)$ can be thwarted by a situation that is consistent with no *new* contingent executions occurring *before* time t (i.e., at any time ρ such that $\mathtt{now}_\psi < \rho < t$).

Now, suppose that $\Gamma = \emptyset$. Let Agnes adopt the following strategy: wait until some contingent time-point happens to execute. Let $t > \mathtt{now}_\psi$ be the time of that next contingent execution. Since $t \notin \Gamma$, there must be some strategy, $\theta \in \Theta(t)$, that could only be thwarted by situations that involve the execution of contingent time-points before time t. Since no contingent time-points executed before time t, that strategy is not thwarted by the current situation and, thus, is a winning strategy for Agnes, which is a contradiction. Thus, $\Gamma \neq \emptyset$.

Next, let $T_u = \inf\{t \mid t > \mathtt{now}_\psi \text{ and } t \notin \Gamma\}$. Now, T_u is well defined since Γ is non-empty and bounded below by $\mathtt{now}_\psi$. Consider the possibility that $T_u = \mathtt{now}_\psi$. This implies that for any time $t > \mathtt{now}_\psi$, there is a time $t' \in (\mathtt{now}_\psi, t)$ such that $t' \notin \Gamma$. But then a similar argument as that used to show that Γ is not empty can be used to show that T_u cannot equal $\mathtt{now}_\psi$. In this case, given the time t of the next contingent execution, there must be a time $t' \in (\mathtt{now}_\psi, t)$ such that $t' \notin \Gamma$ and, hence, some strategy $\theta \in \Theta(t')$ that could only be thwarted by contingent executions before time $t' < t$. Since no such executions occurred, that strategy could be followed by Agnes as a winning strategy, a contradiction. Thus, $T_u > \mathtt{now}_\psi$. It remains to be seen whether $T_u \in \Gamma$.

Next, let Γ^* be the subset of $(\mathtt{now}_\psi, T_u]$ such that for each $t \in \Gamma^*$, there exists a (possibly empty) set $\chi(t)$ of contingent time-points such that every strategy $\theta \in \Theta(t)$ can be thwarted by a situation that is consistent with (1) no new contingent executions *before* time t; and (2) the execution of *all* of the contingent time-points in $\chi(t)$ *at* time t. Now, suppose Γ^* were empty. Then let $t \in (\mathtt{now}_\psi, T_u) \subseteq \Gamma$ be arbitrary; and consider the following strategy for Agnes: wait until the time t, or the execution of the next contingent time-point, whichever happens first. If no contingent time-points happen to execute before time t, then let $t' = t$; otherwise, let t' be the time at which the first contingent time-point executed. In either case, since $t' \in \Gamma$, but $t' \notin \Gamma^*$, there could not be a single set $\chi(t')$ as described earlier. Therefore, there would have to be at least two strategies, θ_1 and θ_2, in $\Theta(t')$ whose thwarting would require two *different* sets of contingent time-points executing at time t'. Agnes could then choose to follow whichever strategy, θ_1 or θ_2, was not thwarted by the execution events that occurred at time t'. Since that chosen strategy could only have been thwarted by execution events which did not occur, it must be a winning strategy, which is a contradiction. Therefore $\Gamma^* \neq \emptyset$.

Next, let $T_u^* = \inf\{t \mid t > \mathtt{now}_\psi \text{ and } t \notin \Gamma^*\}$. Consider the possibility that $T_u^* = \mathtt{now}_\psi$. Then for any $t > \mathtt{now}_\psi$, there exists a t' such that $\mathtt{now}_\psi < t' < t$ and $t' \notin \Gamma^*$. Let Agnes wait until the time of the next contingent execution, say at time $t > \mathtt{now}_\psi$. Then there exists a time t' strictly between $\mathtt{now}_\psi$ and t such that $t' \notin \Gamma^*$. In that case, there exist strategies θ_1 and θ_2 in $\Theta(t')$ whose thwarting situations required different sets of contingent executions at time $t' < t$. Since no such contingent executions occurred, Agnes can simply choose whichever strategy has thereby become a winning strategy, yielding a contradiction. Therefore, $T_u^* > \texttt{now}_\psi$.

There are now three cases to consider:

> *Case 1: $T_u^* = T_u$, but $T_u \notin \Gamma$.* Suppose that for all $t \in (\texttt{now}_\psi, T_u)$, $\chi(t) = \emptyset$. In other words, for each $t \in (\texttt{now}_\psi, T_u)$, every $\theta \in \Theta(t)$ can be thwarted by situations in which no contingent time-points execute *at or before* time t. But that implies that every $\theta \in \Theta(T_u)$ can be thwarted by situations in which no contingent time-points execute *before* time T_u and, hence, that $T_u \in \Gamma$, a contradiction. Therefore, it must be that for some $t^* \in (\texttt{now}_\psi, T_u)$, $\chi(t^*) \neq \emptyset$. Let Vera's RTED* decision be $(t^*, \chi(t^*))$.
> *Case 2: $T_u^* = T_u \in \Gamma$.* Suppose that $T_u^* \notin \Gamma^*$. Then there must be two strategies, θ_1 and θ_2, in $\Theta(T_u^*)$ that can only be thwarted by situations involving two *different* sets of contingent time-points at time T_u^*. But then Agnes could simply wait until time T_u^* to see which of the two strategies was not thwarted, to yield a winning strategy. But that is a contradiction. Therefore, $T_u^* \in \Gamma$.
> Now, suppose that $\chi(T_u^*) = \emptyset$. That is, every strategy in $\Theta(T_u^*)$ can be thwarted by situations that do not involve any new contingent executions at or before T_u^*. Let Agnes employ the following strategy: wait until the next contingent execution. Suppose it happens at some time $t > T_u^*$. By the definition of T_u and the fact that $T_u \in \Gamma$, it follows that there must be some t' strictly between T_u and t such that $t' \notin \Gamma$. But then there must be a strategy $\theta \in \Theta(t')$ whose thwarting requires the execution of a contingent time-point before time $t' < t$. Since no such execution occurred, Agnes can employ θ as a winning strategy, which is a contradiction. Thus, $\chi(T_u^*) \neq \emptyset$. Vera's RTED* decision can then be $(T_u^*, \chi(T_u^*))$.
> *Case 3: $T_u^* < T_u$.* As in Case 2, it follows here that $T_u^* \in \Gamma^*$. Now, let t be any time such that $T_u^* < t < T_u$. Let Agnes wait until the next contingent execution or the time t, whichever comes first. Let $t^\dagger$ be that time. By the definition of T_u^* as an infemum, and the fact that $T_u^* \in \Gamma^*$, it follows that there is some t' strictly between T_u^* and $t^\dagger$ such that $t' \notin \Gamma^*$, but $t' \in \Gamma$ (since $t' < T_u$). But then there exist strategies θ_1 and θ_2 in $\Theta(t')$ whose thwarting situations require different sets of contingent time-points to execute at time $t' < t^\dagger \leq t$. Since no such contingent executions occurred, Alice can simply choose whichever strategy has thereby become a winning strategy, yielding a contradiction. Therefore, it cannot be that $T_u^* < T_u$.

Only Cases 1 and 2 avoid a contradiction; and in each of those cases generates a decision for Vera of the form $\Delta_u = (t, \chi)$, where χ is a set of contingent time-points that are to be executed at time t if Agnes does not execute any time-points at or before t. It remains to show that all possible outcomes of the decisions of Agnes and Vera result in a schedule that can be obtained by following a strategy R in some thwarting situation ω_R.

First, suppose Agnes uses a `wait` decision. In that case, the contingent time-points in χ will be executed at time t. By the construction of the χ set (cf. the definition of Γ^*), it follows that all strategies in $\Theta(t)$, of which `wait` is one, can be thwarted by situations that are consistent with this outcome. Similar remarks apply to Agnes using a (T_f, χ_f) decision where $T_f > t$.

Second, suppose Agnes uses a (T_f, χ_f) decision where $T_f \leq t$. Then the partial outcome will involve the execution of the executable time-points in χ_f at time $T_f \leq t$, but not the contingent time-points in χ. Now, since $T_f \leq t$, it follows that $T_f \leq T_u^*$. Thus, for each time $t' < T_f$, all strategies in $\Theta(t')$—of which, Agnes' (T_f, χ_f) is one—must be thwartable by situations involving no new contingent time-points before time t'. But then, for any $t^\dagger < T_f$, there is some t' such that $t^{\dagger} < t' < T_f$, from which it follows that no contingent time-points need be executed at or before $t^{\dagger}$. Thus, no contingent time-points need be executed before time T_f. However, thwarting the strategies that involve the execution of the time-points in χ_f at time T_f may require the execution of some contingent time-points at time T_f. A single set of such time-points must be sufficient; otherwise, it would contradiction the thwartability of those strategies. That set of time-points constitutes an instantaneous reaction by Vera.

Thus, in all cases, Vera has a decision (t, χ) available—that only depends on ψ, not on R—that, together with a possible instantaneous reaction, generates an outcome according to the two-player game semantics that is identical to an outcome that is obtained by following an RTED-based strategy in a thwarting situation. □

Appendix 2: The semantics of dynamic controllability for CDTNUs

In this section, the dynamic execution semantics for STNUs is extended to accommodate the features of CSTNUs and DTNUs, resulting in a dynamic execution semantics for CDTNUs. For a CDTNU, $(\mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P, \mathcal{L})$, the agent seeks a strategy for executing the *free* time-points in $\mathcal{T}_f \subseteq \mathcal{T}$ whose labels are in accordance with the current scenario, such that all constraints in $\mathcal{C}$ will necessarily be satisfied *no matter what durations the environment "chooses" for the contingent links in* $\mathcal{L}$, *and no matter which truth values the environment "chooses" for the propositions in* P. The decisions that constitute such a strategy can depend only on execution events that occurred in the past; however, the strategy can be dynamic in that it may react—*after a positive delay*—to observations of contingent time-points executing or propositional letters being assigned truth values.

First, the partial schedules from Defnition 9 are extended to accommodate observation time-points. In this context, an extended partial schedule is not only a possibly partial assignment of values to time-points, but also a possibly partial assignment of truth values to propositional letters.

Definition 23 (*Extended Partial Schedule*) An *extended partial schedule* for a CDTNU, $(\mathcal{T}, \mathcal{C}, L, \mathcal{OT}, \mathcal{O}, P, \mathcal{L})$), is (ψ, σ), where ψ is a partial schedule (i.e., a partial assignment to time-points in $\mathcal{T}$, as in Definition 9), and σ is a set of tuples of the form (X, b) where $X \in \mathcal{OT} \cap TPs(\psi)$ is an already-executed observation time-point, and b is either $\top$ or $\bot$ (i.e., *true* or *false*). The *label* for the extended partial schedule, (ψ, σ), is the conjunction of literals determined by the truth values in σ. For example, if p is true in σ, and q is false, and those are the only Boolean variables that have been observed so far according to ψ and σ, then the label for (ψ, σ) is $p\neg q$.

Intuitively, ψ records the execution times for those time-points that have already executed; and σ records the truth values of the propositional letters corresponding to observation time-points that have already executed. In short, the extended partial schedule represents all of the information on which execution decisions may depend.

The RTEDs available to Agnes in the case of a CDTNU are essentially the same as in the case of STNUs with one minor condition: the time-points in the set χ_f must have labels that are subsumed by the label associated with the current extended partial schedule. In other words, the labels on the time-points must be true given the label for the extended partial schedule. For example, if the label of (ψ, σ) is $p\neg q$, then the labels of any time-points in the set χ_f must be one of: the empty label, p, $\neg q$, or $p\neg q$.

Definition 24 (*RTED, for Agnes, in a CDTNU*) Let (ψ, σ) be an extended partial schedule, where ψ is respectful. An RTED for Agnes has one of two forms: `wait` or (T_f, χ_f). A `wait` decision is applicable if at least one contingent time-point, C, is *active* in ψ (i.e., C's activation time-point has already been executed, but C has not). A (T_f, χ_f) decision (i.e., "If nothing happens *before* time T_f, execute the time-points in χ_f at time T_f") is applicable if $T_f > \mathtt{now}_\psi$, χ_f is a non-empty subset of unexecuted free time-points (i.e., $\chi_f \neq \emptyset$ and $\chi_f \cap TPs(\psi) = \emptyset$) and for each $X \in \chi_f$, the label $L(X)$ is subsumed by the label of (ψ, σ).

For Vera, there are two significant changes:

(1) For an RTED*: the execution times for contingent time-points must accommodate the case of contingent durations that may fall anywhere within a union of disjoint intervals.
(2) For an instantaneous reaction: in cases where the partial outcome of the decisions by Agnes and Vera includes the execution of observation time-points, then for each such observation time-point, Vera must instantaneously specify a truth value for the corresponding Boolean propositional letter.

In the case of an RTED*, it is convenient to call the union of distinct intervals for a given contingent duration an *extended execution window.*

Definition 25 (RTED*, *for Vera, in a CDTNU*) Let (ψ, σ) be an extended partial schedule, where ψ respects at least one situation. A *before-or-at RTED* (RTED*) has one of two forms: `wait` or (T_u, χ_u). A `wait` decision is only applicable if no contingent time-points are currently active in ψ. A (T_u, χ_u) decision (i.e., "If nothing happens *before-or-at* time T_u, I shall execute the time-points in χ_u at time T_u") is applicable only if $T_u > \mathtt{now}_\psi$; χ_u is a non-empty subset of currently-activated contingent time-points each of whose *extended execution window* includes T_u; and the extended execution window for each currently-activated contingent time-point that is *not* in χ_u extends beyond the time T_u.

Definition 26 (*Instantaneous reaction, for Vera, in a CDTNU*) Let (ψ, σ) be an extended partial schedule in which at least one contingent time-point C is activated and whose *extended execution window* includes $\mathtt{now}_\psi$ (i.e., one of the possible durations for C would result in C executing at $\mathtt{now}_\psi$). An *instantaneous reaction* is a decision (by Vera) to: (1) execute a set of such time-points at time $\mathtt{now}_\psi$; and (2) assign truth values for each of the observation time-points in ψ that are not yet assigned in σ. If $\mathtt{now}_\psi$ happens to be the last possible time at which a currently-activated contingent time-point C can execute, then the instantaneous reaction must include C.

The partial and full outcomes for these augmented decisions for Agnes and Vera are analogous to those in the case of an STNU. The principal difference is that if a partial outcome involves the execution of an observation time-point, then Vera's instantaneous reaction must assign a truth value to the corresponding Boolean propositional letter.

Note that the disjunctive constraints that may appear within the set $\mathcal{C}$ of constraints in the CDTNU do not affect the execution semantics at all. In other words, they do not affect the execution decisions that are available to either Agnes or Vera. Instead, they represent constraints that Agnes wants to satisfy.

With these changes, the definition of dynamic controllability for CDTNUs is analogous to that for STNUs.

Appendix 3: Proof of correctness for the STNU-to-TGA encoding

This section presents the theoretical results that confirm the correctness of the STNU-to-TGA encoding given in Sect. 4.2.1. It also explicates the correspondence between strategies for STNUs and their TGA counterparts.

Theorem 2 *Let $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{L})$ be any STNU; and let Θ be the encoding of $\mathcal{S}$ as a TGA, as described in Sect. 4.2.1. Then Θ correctly captures the execution semantics for $\mathcal{S}$ in the sense that any sequence of partial schedules that can be generated for $\mathcal{S}$ according to the execution semantics for STNUs corresponds to a run for Θ that can be generated by following its transitions according to the TGA semantics.*

Proof The following invariant is proved by induction. Each respectful partial schedule ψ that can be generated for $\mathcal{S}$ corresponds to a state of Θ in which the location is $\mathtt{vera}$, $\mathtt{c}_\delta = 0$, $\mathtt{now}_\psi = \hat{c}$, for each executed time-point X, $\psi(X) = \hat{c} - cX$, and for each unexecuted time-point Y, $\psi(Y) = \hat{c}$. For the base case, the initial partial schedule, $\psi_0 = \emptyset$, corresponds to the initial state of Θ in which the location is $\mathtt{vera}$, all clocks are at zero, and all time-points are unexecuted. Note that ψ_0 is trivially respectful.

Now, suppose that ψ is a respectful partial schedule that can be generated according to the execution semantics for STNUs, and that satisfies the hypothesized invariant. Let θ be the corresponding state of the TGA. Since $\mathtt{c}_\delta = 0$, the only transitions that are immediately enabled are the loops whereby contingent time-points are executed. These transitions, if taken, correspond to the instantaneous reaction decisions for Vera, in which a set Υ_u of one or more contingent time-points can be executed simultaneously. However, suppose that Vera does not make any such transitions at $\mathtt{c}_\delta = 0$. Once $\mathtt{c}_\delta > 0$, both Agnes and Vera have transitions that they could make at any time. For example, Vera might decide to execute one or more contingent time-points when $\mathtt{c}_\delta = 3$. That would correspond to an RTED*-based decision, (T_u, χ_u), where $T_u = \mathtt{now}_\psi + 3$ and χ_u contains the time-points to be executed. Since each transition by Vera resets $\mathtt{c}_\delta$ to 0, Agnes is unable to interrupt Vera's simultaneous execution of contingent time-points. The resulting outcomes are equivalent to the partial schedules that arise in Cases (1a) and (1b) of Definition 18. The guards on Vera's transitions, which enforce the duration bounds for the contingent links, ensure that the resulting partial schedule is respectful. Also, when Vera's sequence of "simultaneous" transitions complete, $\hat{c}$ equals the time of the most recent execution (i.e., $\mathtt{now}_\psi + 3$). In addition, for each newly executed time-point, C, the clock cC is set to 0, ensuring that $\hat{c} - \mathtt{cC}$ equals the execution time of C. Since both clocks will never again be reset, this difference remains fixed forever.

On the other hand, suppose that Agnes decided to execute the time-points in χ_f at an earlier time, say, $\mathtt{now}_\psi + 2$. This would correspond to her making the transition to the $\mathtt{agnes}$ location and instantaneously executing the time-points in χ_f at that time and, then, immediately returning to the $\mathtt{vera}$ location. Since $\mathtt{agnes}$ is an *urgent* state, the global clock equals $\mathtt{now}_\psi + 2$ when the return transition is made. This sequence of transitions corresponds to the *partial* outcomes in Cases (2a) and (2b) in Definition 18, where Agnes' decision is (T_f, χ_f), where $T_f = \mathtt{now}_\psi + 2$. Furthermore, if Vera chooses to instantaneously execute some contingent time-points at that same time, $\mathtt{now}_\psi + 2$, that will correspond to an instantaneous reaction, as specified in Definition 17.

Finally, if at time $\mathtt{now}_\psi$, Agnes and Vera both decided to execute some time-points at time $\mathtt{now}_\psi + 1$, then the STNU semantics ensures that Agnes' time-points will be executed, and that Vera will be able to instantaneously react, if she chooses. This corresponds to Agnes' transition having *priority* over Vera's transition. Agnes transitions to the $\mathtt{agnes}$

state, executes her time-points, and returns to the vera state, with the global clock ending up at $\texttt{now}_\psi + 1$.

Since, in all cases, the resulting state of the TGA satisfies the desired invariant property, the result is proven. □

Theorem 3 *Let $\mathcal{S}$ be any STNU; let Θ be the encoding of $\mathcal{S}$; and let σ be a winning TGA counter-strategy for Agnes. Then there is an equivalent RTED-based strategy for Agnes that will ensure the satisfaction of all constraints in $\mathcal{S}$ no matter how the contingent durations turn out.*

Proof Let $\mathcal{S}$, Θ and σ be as described in the statement above. Therefore, $\sigma : L \times \mathbb{R}^{\mathcal{X}}_{>=0} \to Act_u \cup \{\lambda\}$, where Act_u is the set of uncontrollable actions (for Agnes).

Suppose the TGA has just entered the state, $(\texttt{vera}, v)$, where v represents the vector of clock values. As has already been noted, for any time-point X and associated clock $\texttt{cX}$: (1) before X executes, $\texttt{cX} = \hat{c}$; and (2) after X executes, $\texttt{cX} < \hat{\texttt{t}}$ and the fixed difference, $\hat{\texttt{t}} - \texttt{cX}$, equals the time at which X executed. Thus, the vector of clock values specifies a partial schedule, ψ. Now, suppose that $\texttt{now}_\psi < \hat{c}$ (i.e., that some positive time has elapsed since the last execution event in ψ). The only way that could have happened is if the state $(\texttt{vera}, v)$ had been preceded by one or more useless loops (i.e., loops using only the gain and pass transitions to go back and forth between vera and agnes without executing any time-points). Let $(\texttt{vera}, v')$ be the state immediately preceding the first such useless loop. Then for some positive ϵ, $v = v' + \epsilon$ (i.e., the clock values in v are ϵ units larger than their corresponding values in v'). And by construction, $\texttt{now}_\psi = v'(\hat{c})$.

Next, let D be the minimum time that can elapse from v before the strategy σ recommends a non-trivial transition to the agnes location. That is: $D = \min\{d \mid \sigma(\texttt{vera}, v' + d) \neq \lambda, \sigma(\texttt{agnes}, v'+d) \neq \texttt{pass}\}$. Let $v_0 = v'+D$. The unique sequence of *execution* transitions at the agnes location is: $\tau_1 = \sigma(\texttt{agnes}, v_0)$, $\tau_2 = \sigma(\texttt{agnes}, v_1)$, $\tau_3 = \sigma(\texttt{agnes}, v_2), \ldots$, where each v_{i+1} is the same as v_i, except that the clock for the just-executed time-point is 0 in v_{i+1}. This sequence must terminate, since there are only finitely many time-points, and each can be executed only once. If τ_m is the last execution transition, it follows that $\texttt{pass} = \sigma(\texttt{agnes}, v_m)$. That transition leads back to the state, $(\texttt{vera}, v_m)$, where v_m is the same as v', except that the clocks for the time-points executed by the transitions, $\tau_1, \ldots, \tau_m$, are all zero in v_m.

Next, let $T_f = v_0(\hat{c})$ be the global time at which σ recommends its first non-trivial transition to agnes; and let χ_f be the set of time-points that correspond to the execution transitions, $\tau_1, \ldots, \tau_m$. Then (T_f, χ_f) is an RTED for ψ that corresponds to what the strategy σ recommends at $(\texttt{vera}, v')$. Note that Vera may decide to instantaneously react by executing some contingent time-points also at time T_f, an outcome that is sanctioned by the execution semantics for STNUs. Finally, it may happen that Vera decides to intervene *before* time T_f arrives, by executing one or more contingent time-points and effectively generating a new partial schedule, ψ^*. In that case, the same procedure could be applied to ψ^* to generate an appropriate RTED. Since the guard on the transition from vera to agnes requires a positive time delay, that RTED is properly prohibited from any kind of instantaneous reaction (by Agnes).

This procedure provides a mapping from any $(\texttt{vera}, v)$ state that is reachable following the winning strategy σ. In addition, the sequences of partial schedules generated by following the RTEDs correspond to runs that can be produced by σ. Thus, the complete schedules generated by the RTEDs are guaranteed to satisfy all STNU constraints assuming Vera observes the bounds on all contingent links. □

References

1. Abdeddaim, Y., Asarin, E., Sighireanu, M.: Simple algorithm for simple timed games. In: TIME, pp. 99–106 (2009)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**(11), 832–843 (1983)
3. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)
4. Augusto, J.C.: Temporal reasoning for decision support in medicine. Artif. Intell. Med. **33**(1), 1–24 (2005)
5. Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K., Lime, D.: Uppaal-Tiga: time for playing games!. In: Damm, W., Hermanns, H. (eds.) Proceedings of the 19th Conference on Computer Aided Verification (CAV-2007). Lecture Notes in Computer Science, vol. 4590, pp. 121–125. Springer, Berlin (2007)
6. Cassez, F., David, A., Fleury, E., Larsen, K.G., Lime, D.: Efficient on-the-fly algorithms for the analysis of timed games. In: CONCUR, pp. 66–80 (2005)
7. Cesta, A., Fratini, S., Orlandini, A., Finzi, A.: Flexible plan verification: feasibility results. Fundam. Inform. **107**(2–3), 111–137 (2011)
8. Cheikhrouhou, S., Kallel, S., Guermouche, N., Jmaiel, M.: Toward a time-centric modeling of business processes in BPMN 2.0. In: International Conference on Information Integration and Web-based Applications and Services, pp. 154–163. ACM (2013)
9. Cimatti, A., Hunsberger, L., Micheli, A., Posenato, R., Roveri, M.: Sound and complete algorithms for checking the dynamic controllability of temporal networks with uncertainty, disjunction and observation. In: Cesta, A., Combi, C., Laroussinie, F. (eds.) 21st International Symposium on Temporal Representation and Reasoning, TIME 2014, Verona, Italy, September 8–10, 2014, pp. 27–36. IEEE Computer Society (2014). doi:10.1109/TIME.2014.21
10. Cimatti, A., Hunsberger, L., Micheli, A., Roveri, M.: Using timed game automata to synthesize execution strategies for simple temporal networks with uncertainty. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27–31, 2014, Québec City, Québec, Canada, pp. 2242–2249 (2014)
11. Cimatti, A., Micheli, A., Roveri, M.: Solving temporal problems using SMT: weak controllability. In: AAAI, pp. 448–454 (2012)
12. Cimatti, A., Micheli, A., Roveri, M.: Solving strong controllability of temporal problems with uncertainty using SMT. Constraints **20**(1), 1–29 (2015)
13. Combi, C., Gambini, M., Migliorini, S., Posenato, R.: Representing business processes through a temporal data-centric workflow modeling language: an application to the management of clinical pathways. IEEE Trans. Syst. Man Cybern. Syst. **44**(9), 1182–1203 (2014). doi:10.1109/TSMC.2014.2300055
14. Combi, C., Gozzi, M., Posenato, R., Pozzi, G.: Conceptual modeling of flexible temporal workflows. ACM Trans. Autono. Adapt. Syst. (TAAS) **7**(2), 19 (2012). doi:10.1145/2240166.2240169
15. Combi, C., Hunsberger, L., Posenato, R.: An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In: Filipe, J., Fred, A.L.N. (eds.) ICAART 2013—Proceedings of the 5th International Conference on Agents and Artificial Intelligence, vol. 2, Barcelona, Spain, 15–18 February, 2013, pp. 144–156. SciTePress (2013)
16. Combi, C., Posenato, R.: Controllability in temporal conceptual workflow schemata. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8–10, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5701, pp. 64–79. Springer (2009). doi:10.1007/978-3-642-03848-8_6
17. Combi, C., Pozzi, G.: Architectures for a temporal workflow management system. In: Proceedings of the 2004 ACM Symposium on Applied Computing (SAC-2004), pp. 659–666. ACM, New York (2004)
18. Comin, C., Posenato, R., Rizzi, R.: A tractable generalization of simple temporal networks and its relation to mean payoff games. In: Cesta, A., Combi, C., Laroussinie, F. (eds.) 21st International Symposium on Temporal Representation and Reasoning, TIME 2014, Verona, Italy, September 8–10, 2014, pp. 7–16. IEEE Computer Society (2014). doi:10.1109/TIME.2014.19
19. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artif. Intell. **49**, 61–95 (1991)
20. Eder, J., Panagos, E., Rabinovich, M.: Time constraints in workflow systems. In: Jarke, M., Oberweis, A. (eds.) Advanced Information Systems Engineering, LNCS, vol. 1626, pp. 286–300. Springer, Berlin (1999)
21. Hollingsworth, D.: The workflow reference model. http://www.wfmc.org/standards/model.htm (1995)
22. Hunsberger, L.: Fixing the semantics for dynamic controllability and providing a more practical characterization of dynamic execution strategies. In: Lutz, C., Raskin, J. (eds.) TIME 2009, 16th International Symposium on Temporal Representation and Reasoning, Bressanone-Brixen, Italy, 23–25 July 2009, Proceedings, pp. 155–162. IEEE Computer Society (2009). doi:10.1109/TIME.2009.25

23. Hunsberger, L.: A fast incremental algorithm for managing the execution of dynamically controllable temporal networks. In: Markey, N., Wijsen, J. (eds.) TIME 2010–17th International Symposium on Temporal Representation and Reasoning, Paris, France, 6–8 September 2010, pp. 121–128. IEEE Computer Society (2010). doi:10.1109/TIME.2010.16
24. Hunsberger, L.: A faster execution algorithm for dynamically controllable stnus. In: Sánchez, C., Venable, K.B., Zimányi, E. (eds.) 2013 20th International Symposium on Temporal Representation and Reasoning, Pensacola, FL, USA, September 26–28, 2013, pp. 26–33. IEEE Computer Society (2013). doi:10.1109/TIME.2013.13
25. Hunsberger, L.: A faster algorithm for checking the dynamic controllability of simple temporal networks with uncertainty. In: Duval, B., van den Herik, H.J., Loiseau, S., Filipe, J. (eds.) ICAART 2014 - Proceedings of the 6th International Conference on Agents and Artificial Intelligence, vol. 1, ESEO, Angers, Loire Valley, France, 6–8 March, 2014, pp. 63–73. SciTePress (2014). doi:10.5220/0004758100630073
26. Hunsberger, L., Posenato, R., Combi, C.: The dynamic controllability of conditional STNs with uncertainty. In: Proceedings of the Workshop on Planning and Plan Execution for Real-World Systems: Principles and Practices (PlanEx) at ICAPS-2012, pp. 1–8 (2012). arXiv:1212.2005
27. Kleene, S.: Mathematical Logic. Wiley, Hoboken (1967)
28. Lanz, A., Posenato, R., Combi, C., Reichert, M.: Controllability of time-aware processes at run time. In: Meersman, R., Panetto, H., Dillon, T.S., Eder, J., Bellahsene, Z., Ritter, N., Leenheer, P.D., Dou, D. (eds.) On the Move to Meaningful Internet Systems: OTM 2013 Conferences—Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9–13, 2013. Proceedings, Lecture Notes in Computer Science, vol. 8185, pp. 39–56. Springer (2013). doi:10.1007/978-3-642-41030-7_4
29. Lanz, A., Posenato, R., Combi, C., Reichert, M.: Simple temporal networks with partially shrinkable uncertainty. In: Loiseau, S., Filipe, J., Duval, B., van den Herik, H.J. (eds.) ICAART 2015—Proceedings of the International Conference on Agents and Artificial Intelligence, vol. 2, Lisbon, Portugal, 10–12 January, 2015, pp. 370–381. SciTePress (2015)
30. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Mylopoulos, J., Sadeh, N.M., Shaw, M.J., Szyperski, C., Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling 11th International Workshop, BPMDS 2010, and 15th International Conference, EMMSAD 2010, pp. 94–107. Springer, Berlin (2010)
31. Lewis, H.R., Papadimitriou, C.H.: Elements of the Theory of Computation, 2nd edn. Prentice-Hall Inc, Upper Saddle River (1998)
32. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems. In: STACS, pp. 229–242 (1995)
33. Morris, P.: A structural characterization of temporal dynamic controllability. In: Principles and Practice of Constraint Programming (CP-2006), Lecture Notes in Computer Science, vol. 4204, pp. 375–389. Springer (2006)
34. Morris, P.: Dynamic controllability and dispatchability relationships. In: Simonis, H. (ed.) Integration of AI and OR Techniques in Constraint Programming—11th International Conference (CPAIOR-2014), Lecture Notes in Computer Science, vol. 8451, pp. 464–479. Springer (2014)
35. Morris, P., Muscettola, N., Vidal, T.: Dynamic control of plans with temporal uncertainty. In: Nebel, B. (ed.) Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-2001), pp. 494–499. Morgan Kaufmann (2001)
36. Morris, P.H., Muscettola, N.: Temporal dynamic controllability revisited. In: AAAI, pp. 1193–1198 (2005)
37. Orlandini, A., Finzi, A., Cesta, A., Fratini, S.: TGA-based controllers for flexible plan execution. In: KI, no. 7006 in LNAI, pp. 233–245. Springer (2011)
38. Peintner, B., Venable, K.B., Yorke-Smith, N.: Strong controllability of disjunctive temporal problems with uncertainty. In: Principles and Practice of Constraint Programming (CP-2007), pp. 856–863 (2007)
39. Rossi, F., Venable, K.B., Yorke-Smith, N.: Uncertainty in soft temporal constraint problems: a general framework and controllability algorithms for the fuzzy case. J. Artif. Intell. Res. **27**, 617–674 (2006)
40. Tsamardinos, I., Pollack, M.E.: Efficient solution techniques for disjunctive temporal reasoning problems. Artif. Intell. **151**, 43–89 (2003)
41. Tsamardinos, I., Vidal, T., Pollack, M.: CTP: a new constraint-based formalism for conditional, temporal planning. Constraints **8**(4), 365–388 (2003)
42. Venable, K.B., Volpato, M., Peintner, B., Yorke-Smith, N.: Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In: Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS-2010) in ICAPS-2010, pp. 50–59 (2010)
43. Venable, K.B., Yorke-Smith, N.: Disjunctive temporal planning with uncertainty. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2005), pp. 1721–1722 (2005)

44. Vidal, T.: Controllability characterization and checking in contingent temporal constraint networks. In: KR, pp. 559–570 (2000)
45. Vidal, T., Fargier, H.: Contingent durations in temporal CSPS: from consistency to controllabilities. In: Proceedings of the 4th International Symposium on Temporal Representation and Reasoning (TIME-1997) (1997)
46. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: from consistency to controllabilities. J. Exp. Theor. Artif. Intell. **11**(1), 23–45 (1999)
47. Vidal, T., Ghallab, M.: Temporal constraints in planning: free or not free? In: Proceedings of the International Workshop on Constraint-Based Reasoning (CONSTRAINT-1995) in FLAIRS-1995 (1995)
48. Vidal, T., Ghallab, M.: Dealing with uncertain durations in temporal constraint networks dedicated to planning. In: Wahlster, W. (ed.) Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-1996), pp. 48–54. Wiley, Chichester (1996)