UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

STATIC ENFORCEMENT OF SERVICE DEADLINES

Massimo Bartoletti and Roberto Zunino

February 2010

Technical Report #DISI-10-018

Static Enforcement of Service Deadlines

Massimo Bartoletti[1] and Roberto Zunino[2]

[1] Dipartimento di Matematica e Informatica, Università degli Studi di Cagliari
[2] Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento

Abstract. We consider the problem of statically deciding when a service always provides its functionality within a given amount of time. In a timed π-calculus, we propose a two-phases static analysis guaranteeing that processes enjoy both the maximal progress and the well-timedness properties. Exploiting this analysis, we devise a decision procedure for checking service deadlines.

1 Introduction

Time plays a crucial role in Service-oriented Computing. Given a formal specification of a set of interacting services, a frequent question is whether invoking a service will yield the desired functionality, and *how much time* will be required to complete the whole task. Also, in scenarios where service level agreement and negotiation are regulated by contracts [3,4,7,8], it is common that such contracts involve time constraints on the fulfillment of operations. Temporal guarantees are then in order to devise orchestrations satisfying all the required contracts.

A useful feature of SOC infrastructures would then be the ability to check whether, from a formal specification, one can infer a static time bound for each of the provided functionalities. Detecting such static temporal guarantees is the goal of this paper. We choose, as a minimalistic specification language, a synchronous π-calculus extended with a few primitives to deal with time. We adopt a discrete time model, where time is measured in atomic units, named *ticks*.

A first feature of our calculus is the ability of specifying services which can always complete an arbitrarily complex *internal* computation within a given number of time units. In other words, our calculus will enjoy the *maximal progress* property [18], that is, the passing of time cannot block internal computations.

To obtain maximal progress for our calculus, we prioritize the computation steps of services w.r.t. time ticks. In this way, time cannot pass unless the specification itself explicitly asks for it. This approach provides a lot of power to the specification, which can precisely handle the flowing of time. This enhanced expressive power leads, as a drawback, to the possibility of abuses. For instance, time might be blocked by stuck processes, as well as by processes entering an infinite loop of internal computations. Pragmatically, these kinds of specifications are meaningless: obviously, no real-world service can stop the flowing of time.

A main contribution of this paper is a static analysis allowing to single out such kind of "ill-timed" specifications. Our analysis consists of two phases. In the first phase, we develop a type system for checking *pulsingness*, a property enjoyed

by those processes which will eventually perform (at least) one tick. After a tick, the behaviour is unpredicted by this analysis. A further type system exploits then the results of the first phase, to detect those processes that always remain pulsing after each tick. Such processes enjoy deadlock-freedom and *well-timedness* [18], therefore they guarantee that time will never freeze.

This property is pivotal for our global contribution, an algorithm for deciding if a service actually guarantees some desired property within a given deadline. While this property is undecidable in the general case of ill-timed services, it turns out to be decidable for those services typeable in our two-phase system.

An example. Consider a simple scenario with three services. A service broker B is invoked by a client requiring some functionality f. Two external services are delegated to provide such functionality: a cheap one C, and an expensive one E. The broker first attempts to query the cheap service; if no response arrives within a given timeout, the broker queries the expensive one. Unlike the expensive service, the cheap one will provide the result only if it can be produced within a given time bound. We specify the three services as follows:

$$E = rec\, X.\ e(x,k)_\infty.\chi.(X|\bar{k}\langle f(x)\rangle_\infty.\Omega_\infty)$$
$$C = rec\, X.\ c(x,k)_\infty.\chi.(X|\lceil\chi.\bar{k}\langle f(x)\rangle_\infty.\Omega_\infty + \chi.\chi.\chi.\bar{k}\langle f(x)\rangle_\infty.\Omega_\infty\rceil^2(\Omega_\infty))$$
$$B = rec\, X.\ b(x,k)_\infty.\chi.\nu k'\Big(\bar{c}\langle x,k'\rangle_\infty.\Omega_\infty \big| \lfloor k'(y)_\infty.\bar{k}y_\infty.X\rfloor^3(\bar{e}\langle x,k\rangle_\infty.X)\Big)$$

The semantics of the above services is formalised in Sect. 2; here, we just describe intuitively the behavior of the services, and the usage of our static machinery.

Service E receives as input some datum x and a return channel k. The prefix $e(x,k)_\infty$ just means that the process may tick until the input is available. After that, E computes $f(x)$ and returns the result through k. Again, the output prefix $\bar{k}\langle f(x)\rangle_\infty$ may tick *ad libitum*. After the result has been returned, the process does nothing but perpetually ticking, denoted by Ω_∞. Note that E performs its duty within a single tick, denoted as χ.

In service C, an initial tick is required to handle the request; after that, either one or three ticks (at service discretion) are needed to complete the task. To abort computations taking too much time, C uses a *watchdog* [18], written as $\lceil-\rceil^2(-)$. After two ticks, the computation is abruptly halted, and no reply is provided. Globally, C may then require from 2 to 3 ticks to complete its task.

The broker B first forwards its request to C through channel c. The answer, if any, is received on channel k', and then forwarded to the client on channel k. Whenever C reply takes too long, the broker resorts to the expensive service E. A *timeout* [18], written as $\lfloor-\rfloor^3(-)$, is used by B to stop waiting for the cheap service after three ticks. We expect that the broker B, once invoked, will always reply within a fixed time bound. In the worst case, it takes $1+3+1=5$ ticks.

We can put our machinery at work to check this formally. First, the static analyses of Sects. 3, 4 guarantee that the above specification is *well-timed*. Then, the decision procedure of Th. 5 ensures that five ticks indeed suffice for a reply. The same procedure can also tell us that four ticks do not suffice, so five ticks is the tightest static bound.

Outline of the paper. We present our timed calculus in Sect. 2. The maximal progress property is stated in Th. 1. In Sect. 3 we introduce the first phase of our static analysis, i.e. a type system that recognizes pulsing processes; its soundness is stated in Th. 2. The second step of our analysis is in Sect. 4, featuring a type system for well-timedness (the soundness of which is stated by Th. 4) and our decision procedure for service deadlines (Th. 5). In Sect. 5 we discuss some related work. The appendixes contain the proofs of our results.

2 A timed π-calculus

We now present a timed π-calculus. Its syntax is rather standard: it is a synchronous π-calculus, also allowing for non-guarded choice, timeouts and watchdogs [18]. The prefix χ (*tick*) models the passing of one time unit.

Syntax. Let $\mathcal{N}$ be a countably infinite set of *names*, written as lowercase latin letters $a, b, x, \ldots$, possibly with a bar: $\bar{a}, \bar{b}, \bar{x}$. Prefixes π are defined as follows:

$$\pi ::= \bar{a}x \mid a(x) \mid \tau \mid \chi$$

The prefix $\bar{a}x$ is to send a name x through a; the prefix $a(x)$ is to receive a name through a and bind it to x; the prefix τ is for an internal computation; the prefix χ is for making a single time unit pass.

Processes $P, Q, \ldots$ comprise prefixed processes $\pi.P$, parallel composition $P|Q$, non-deterministic choice $P + Q$, recursion $rec\, X.P$, restriction $(\nu a)P$, timeouts and watchdogs. A *timeout* $\lfloor P \rfloor^k(Q)$ behaves as P if an initial action of P is fired within k ticks; otherwise, it behaves as Q. A *watchdog* $\lceil P \rceil^k(Q)$ behaves as P until the k-th tick; since then, it behaves as Q.

Definition 1 (Processes). *The set $\mathcal{P}$ of* processes $P, Q, \ldots$ *is defined as:*

$$P, Q ::= \pi.P \mid P + Q \mid P|Q \mid rec\, X.P \mid X \mid (\nu a)P \mid \lfloor P \rfloor^k(Q) \mid \lceil P \rceil^k(Q)$$

To keep our syntax minimal, we do not explicitly feature the stuck process 0. Stuckness can be represented as well through the process $rec\, X.X$ (see Ex. 1). As usual, we consider processes up-to alpha-conversion, and up-to the structural congruence relation (the definition is standard, see App. A).

Actions comprise the standard ones of the π-calculus: input, free and bound outputs, and the silent action τ. Additionally, we have the action $\chi(A)$, where the set A is used for making the tick χ a low-priority (non-urgent) action [18]. To do that, A contains the names associated with potential higher-priority actions.

Definition 2 (Actions). *The set $\mathcal{A}$ of* actions *is defined as follows:*

$$\alpha ::= a(x) \mid \bar{a}x \mid \bar{a}(x) \mid \tau \mid \chi(A)$$

where $A \subseteq \mathcal{N} \cup \{\tau\}$ is a finite set. We write $\alpha = \chi(-)$ when $\alpha = \chi(A)$ for some A.

The set $act(P)$ contains the *active names* of a process P, i.e. the names of the channels involved in immediately available actions, possibly including τ. The definition is mostly standard, so it is postponed to App. A.

Semantics. We define the semantics of our calculus through the labelled transition system in Fig. 1.

The rules [Pref, Sum, Par, Comm, Rec, Open, Close, Res] are rather standard, defining the behaviour of inputs, (free and bound) outputs, and silent actions. Note that the side condition $\alpha \neq \chi(-)$ prevents these rules from making the time pass. The rules [Chi, SumChi, ParChi, ResChi] model the passing of time, by allowing $\chi(-)$ actions to be generated. We briefly comment on these.

Rule [Chi] fires a χ prefix, generating a $\chi(\emptyset)$ action. Here, the empty set represents the fact that the process can perform no other action but χ.

Rule [SumChi] allows a time tick to cause the selection of a branch P in a non-deterministic choice $P+Q$. Intuitively, this models the ability of one branch to signal a time-out, making the other branch no longer available. To prioritize computation over the passing of time, this is only permitted when the other branch Q is *stuck*. To do that, we first check in the side condition that no internal move is possible for that branch ($\tau \notin act(Q)$). Then, we augment the set A in the action $\chi(A)$ with the names $act(Q)$, i.e. with the subjects of inputs and outputs which can be fired by Q. Each of these inputs and outputs may be stuck or not, depending on the context. Of course, to maintain our operational semantics compositional, we cannot put any requirement on the context. Rather, we just record all the relevant information inside the label, and defer the stuckness check.

Rule [ParChi] allows a composition $P|Q$ to perform a tick. This is possible if both P and Q perform a tick, unlike rule [Par] requiring just one of them to move. This causes time to pass for *all* parallel components, in a synchronous fashion. The side condition $stuck(A, B)$ performs the check deferred in rule [SumChi]: communication between P and Q is thus prioritized over the passing of time.

Rule [ResChi] allows time to pass for a restricted process $(\nu a)P$. All the deferred checks about $a, \bar{a}$ have already been performed at this point, since we have reached the boundary of the scope for a. We are then allowed to discharge the related proof obligations.

The rules [TO1,TO2,TO3] and [WD1,WD2,WD3] deal with timeouts and watchdogs, respectively. The two sets of rules are mostly similar, the characterising ones being [TO1] and [WD1]. Rule [TO1] drives a timeout $\lfloor P \rfloor^k(Q)$ to a residual of P, when $k > 0$ and P performs a non-tick action. Rule [WD1] allows a derivation of P *within* a watchdog $\lceil P \rceil^k(Q)$, when $k > 0$. The rules [TO2] and [WD2] perform a tick, decreasing the counter of a timeout/watchdog. The rules [TO3] and [WD3] allow to leave a timeout/watchdog when the counter reaches 0.

We now introduce some remarkable processes.

Example 1. Consider the processes:

$$0 = rec\, X.X \qquad \Omega_\infty = rec\, X.\chi.X \qquad \pi_\infty.P = recX.(\chi.X + \pi.P)$$

The process 0 is always stuck. Note that 0 does not enjoy *transition liveness* [18], i.e. it can neither perform an internal action, nor make the time pass. The process Ω_∞ is perpetually ticking, yet it cannot perform any other action. For each prefix π and process P, the process $\pi_\infty.P = recX.(\chi.X + \pi.P)$ may indefinitely defer the action π, making the time pass until it eventually fires π.

$$\frac{}{\pi.P \xrightarrow{\pi} P}\ \pi \neq \chi\ \ [\textsc{Pref}] \qquad \chi.P \xrightarrow{\chi(\emptyset)} P\ \ [\textsc{Chi}]$$

$$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}\ \alpha \neq \chi(-)\ \ [\textsc{Sum}] \qquad \frac{P \xrightarrow{\chi(A)} P'}{P+Q \xrightarrow{\chi(A \cup act(Q))} P'}\ \tau \notin act(Q)\ \ [\textsc{SumChi}]$$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\ \begin{array}{l}\alpha \neq \chi(-)\\ bn(\alpha) \cap fn(Q) = \emptyset\end{array}\ \ [\textsc{Par}] \qquad \frac{P \xrightarrow{\chi(A)} P' \quad Q \xrightarrow{\chi(B)} Q'}{P|Q \xrightarrow{\chi(A \cup B)} P'|Q'}\ stuck(A,B)\ \ [\textsc{ParChi}]$$

$$\frac{P \xrightarrow{\bar{a}x} P' \quad Q \xrightarrow{a(x)} Q'}{P|Q \xrightarrow{\tau} P'|Q'}\ \ [\textsc{Comm}] \qquad \frac{P\{recX.P\,/\,X\} \xrightarrow{\alpha} P'}{recX.P \xrightarrow{\alpha} P'}\ \ [\textsc{Rec}]$$

$$\frac{P \xrightarrow{\bar{a}x} P'}{(\nu x)P \xrightarrow{\bar{a}(x)} P'}\ x \neq a, \bar{a}\ \ [\textsc{Open}] \qquad \frac{P \xrightarrow{\bar{a}(x)} P' \quad Q \xrightarrow{a(x)} Q'}{P|Q \xrightarrow{\tau} (\nu x)(P'|Q')}\ \ [\textsc{Close}]$$

$$\frac{P \xrightarrow{\alpha} P'}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}\ \begin{array}{l}\alpha \neq \chi(-)\\ a \notin n(\alpha)\end{array}\ \ [\textsc{Res}] \qquad \frac{P \xrightarrow{\chi(A)} P'}{(\nu a)P \xrightarrow{\chi(A \setminus \{a,\bar{a}\})} (\nu a)P'}\ \ [\textsc{ResChi}]$$

$$\frac{P \xrightarrow{\alpha} P'}{\lfloor P \rfloor^k(Q) \xrightarrow{\alpha} P'}\ \begin{array}{l}\alpha \neq \chi(-)\\ k > 0\end{array}\ \ [\text{TO1}] \qquad \frac{P \xrightarrow{\alpha} P'}{\lceil P \rceil^k(Q) \xrightarrow{\alpha} \lceil P' \rceil^k(Q)}\ \begin{array}{l}\alpha \neq \chi(-)\\ k > 0\end{array}\ \ [\text{WD1}]$$

$$\frac{P \xrightarrow{\chi(A)} P'}{\lfloor P \rfloor^k(Q) \xrightarrow{\chi(A)} \lfloor P' \rfloor^{k-1}(Q)}\ k > 0\ \ [\text{TO2}] \qquad \frac{P \xrightarrow{\chi(A)} P'}{\lceil P \rceil^k(Q) \xrightarrow{\chi(A)} \lceil P' \rceil^{k-1}(Q)}\ k > 0\ \ [\text{WD2}]$$

$$\frac{Q \xrightarrow{\alpha} Q'}{\lfloor P \rfloor^0(Q) \xrightarrow{\alpha} Q'}\ \ [\text{TO3}] \qquad \frac{Q \xrightarrow{\alpha} Q'}{\lceil P \rceil^0(Q) \xrightarrow{\alpha} Q'}\ \ [\text{WD3}]$$

$$stuck(A,B) \iff \nexists a.\, (a \in A \wedge \bar{a} \in B) \vee (a \in B \wedge \bar{a} \in A)$$

Fig. 1. The transition rules

The following lemma, given a transition $P \xrightarrow{\chi(A)}$, relates the names contained in A with the actions occurring in the immediate transitions from P.

Lemma 1. *For all processes P:*

(1a) $\quad P \xrightarrow{\chi(A)} \ \wedge\ P \xrightarrow{a(x)} \implies a \in A$

(1b) $\quad P \xrightarrow{\chi(A)} \ \wedge\ (P \xrightarrow{\bar{a}x} \ \vee\ P \xrightarrow{\bar{a}(x)}) \implies \bar{a} \in A$

(1c) $\quad P \xrightarrow{\chi(A)} \ \wedge\ P \xrightarrow{\tau} \implies \tau \in A$

A key result about our semantics is the maximal progress property (Th. 1). That is, the time can pass only when no other internal actions are possible.

Theorem 1 (Maximal progress). *If $P \xrightarrow{\tau}$ then $P \not\xrightarrow{\chi(-)}$.*

Proof. By contradiction, assume that $P \xrightarrow{\tau}$ and $P \xrightarrow{\chi(A)}$ for some A. By (1c), it follows that $\tau \in A$. However, by induction on the derivation $P \xrightarrow{\chi(A)}$, it follows that $\tau \notin A$ (the only rule which could potentially add τ is [SumChi], yet its side condition prevents from such behaviour) – contradiction.

We now briefly comment (strong, late) bisimilarity $\sim$ in our calculus. Its definition can be given through standard means [22], so we omit it.

In the π-calculus, (strong, late) bisimilarity $\sim_\pi$ satisfies $P + 0 \sim_\pi P$ and $P|0 \sim_\pi P$, so making 0 a neutral element of $+$ and $|$. This holds even when these laws are not comprised in the structural equivalence relation.

In our calculus, 0 is just a shorthand for $rec\, X.X$. It is easy to see that our 0 is a neutral element for $+$, but not for $|$. For instance, we have $\chi.Q|0 \not\sim \chi.Q$, because the right hand side can perform $\chi(\emptyset)$, while the left hand side is stuck: applying [ParChi] would require 0 to move as well. Therefore, the correct law in this case would be $\chi.Q|0 \sim 0$. Note that parallel composition still admits Ω_∞ as a neutral element, since $\Omega_\infty|P \sim P$ holds. Further interesting laws are $\chi.Q \mid \bar{a}x.P \sim \bar{a}x.(\chi.Q \mid P)$ and $\chi.Q + \tau.P \sim \tau.P$. In the former, the action χ in $\chi.Q$ cannot be fired until P performs χ as well. This allows $\chi.Q$ to be pushed inside the continuation of the output. Similar laws hold if we replace the output prefix with τ or input, as long as no captures occur. In the latter, the side condition of [SumChi] prevents the χ in $\chi.Q + \tau.P$ from being fired, thus, $\chi.Q + \tau.P$ is actually bisimilar to $\tau.P$.

3 Pulsing processes

The maximal progress property allows us to specify processes guaranteed to finish their job in a given amount of ticks. This is because no ticks (i.e. χ actions) can be performed while a process is performing internal computation (i.e. while τ actions are enabled). In a sense, this gives too much power to processes, which are able to avoid the passing of time as long as desired. Indeed, a misbehaving process could perform an *infinite sequence* of τ actions, so "freezing" the passing of time. Stuck processes can stop time as well. Clearly, these processes should not be regarded as faithful specifications of real-world systems, since no real system can hinder the passing of time.

In the rest of the paper, we will study how to statically prevent from these kinds of misbehaviour. More precisely, we will introduce a type system which ensures that typable processes will perform an unbounded number of ticks. Our analysis consists of two steps.

In the first step (developed in this section), we single out the processes that are guaranteed to eventually perform at least one χ action. We say such processes to be *pulsing*. Note that a pulsing process is guaranteed to not get stuck only until the first tick is performed. After that, the behaviour of a pulsing process is no longer constrained.

In the next section, we will present a type system to ensure that the pulsingness of a given process is preserved in each possible run. In that case, after the

first χ the process will still be pulsing. This guarantees that an infinite number of χ will be eventually produced.

We start here by formalising when a process is pulsing.

Definition 3. *The maximal traces of a process P are defined as follows:*

$$Tr(P) = \{\, \alpha_0\alpha_1\cdots \mid P \xrightarrow{\alpha_0}\xrightarrow{\alpha_1}\cdots \,\} \cup \{\, \alpha_0\cdots\alpha_n \mid P \xrightarrow{\alpha_0}\cdots\xrightarrow{\alpha_n}\not\rightarrow \,\}$$

For each trace η and $k \geq 0$, we write $\eta|_k$ for the truncation of η to the first k actions (if $k = 0$, then $\eta|_k = \varepsilon$). Note that, if P is stuck, then $Tr(P) = \{\varepsilon\}$.

Definition 4 (Pulsingness). *We say that a process P is* pulsing *whenever:*

$$\forall \eta \in Tr(P) \;:\; \exists k \;:\; \chi(-) \in \eta_{|k}$$

We now introduce our types. A type R can either be $\top$, denoting the absence of information about a process, or χ, for a process which is ready to perform a tick, or $\Pi\mathsf{S}$, for a process which will behave according to the type S after having accomplished with the *type prefix* Π. The type prefix Π can be of three kinds: τ is for a process which is ready to perform a τ action (but not a tick); ! is for a process ready to perform a τ action or a tick; finally, ? is for a process which is either stuck or will perform some action (of any kind).

Definition 5. *The set of type prefixes and the set $\mathcal{T}$ of types (ranged over by $\mathsf{R}, \mathsf{S}, \ldots$) are defined by the following grammars:*

$$\begin{aligned} \Pi &::= \; \tau \;\mid\; ! \;\mid\; ? \\ \mathsf{R} &::= \; \top \;\mid\; \chi \;\mid\; \Pi\,\mathsf{R} \end{aligned}$$

We now provide a subtyping relation $\sqsubseteq$ for our types and type prefixes.

Definition 6. *We define the total order $\sqsubseteq$ between type prefixes as follows:*

$$\tau \;\sqsubseteq\; ! \;\sqsubseteq\; ?$$

The relation $\sqsubseteq$ over types is then defined as the smallest preorder such that:

$$\chi \sqsubseteq !\mathsf{R} \qquad \mathsf{R} \sqsubseteq \top \qquad \Pi\,\mathsf{R} \sqsubseteq \Pi'\,\mathsf{R}' \;\; \textit{whenever } \Pi \sqsubseteq \Pi' \textit{ and } \mathsf{R} \sqsubseteq \mathsf{R}'$$

Our subtyping relation forms a join semilattice. The lub $\sqcup$ on type prefixes is just the maximum, since type prefixes are totally ordered. The lub on types has the following algebraic characterization (symmetric laws are omitted).

Lemma 2. *The pair $(\mathcal{T}, \sqsubseteq)$ is a join semilattice. Furthermore, we have:* Proof in App. B

$$\top \sqcup \mathsf{R} = \top \qquad \chi \sqcup \mathsf{R} = \begin{cases} !\mathsf{S} & \textit{if } \mathsf{R} = \tau\mathsf{S} \\ \mathsf{R} & \textit{otherwise} \end{cases} \qquad \Pi\,\mathsf{R} \sqcup \Pi'\,\mathsf{R}' = (\Pi \sqcup \Pi')\,(\mathsf{R} \sqcup \mathsf{R}')$$

To assign types to processes, we proceed in a compositional way. We abstract the semantics of parallel composition and choice, by defining corresponding abstract operators on types and type prefixes. Most cases follow the intuition behind our types, so we now comment the most peculiar cases. If a process can perform a τ action, then adding a parallel component will still preserve the ability to fire τ, so $\tau|\Pi = \tau$. The same holds for $\tau+\Pi = \tau$. A choice having a non-stuck process in a branch will be non-stuck: $! + \Pi =!$. For parallel composition, we first note that $!|! =!$ agrees with our intuition: either one side can perform a τ, or both can perform a χ. In both cases, the process at hand is non-stuck, hence can be typed with !. Also, we let $!|? =?$ since the left component could be able to perform a tick, only, while the right component could be stuck: in this case the whole process would be stuck, so we abstract that using ?. Both equations $!|! =!$ and $!|? =?$ are generalized by $!|\Pi = \Pi$. This concludes the equations on type prefixes. Among the type equations, we have $\chi|\mathsf{R} = \mathsf{R}$ since an idle process does not affect its parallel components. Since we are concerned with pulsingness, we can safely define $\chi+\mathsf{R}$ to be R, thus choosing the worst-case branch. We actually refine that equation so that $\chi+?\mathsf{R} =!\mathsf{R}$, reflecting that in this case we know that the abstracted process is not stuck: if the right branch is stuck, we can perform χ. As usual, below we omit the symmetric laws.

Definition 7. *We define the operators $|$ and $+$ between type prefixes as:*

$$\begin{array}{lll} \tau \mid \Pi = \tau & ! \mid \Pi = \Pi & ? \mid ? = ? \\ \tau + \Pi = \tau & ! + \Pi = ! \quad (\textit{if } \Pi \neq \tau) & ?+? = ? \end{array}$$

We define the operators $|$ and $+$ between types as follows:

$$\top \mid \mathsf{R} = \top \quad \chi \mid \mathsf{R} = \mathsf{R} \qquad \Pi\,\mathsf{R} \mid \Pi'\,\mathsf{R}' = (\Pi|\Pi')\,(\mathsf{R}|\Pi'\mathsf{R}' \sqcup \Pi\mathsf{R}|\mathsf{R}' \sqcup \mathsf{R}|\mathsf{R}')$$

$$\top+\mathsf{R} = \top \quad \chi+\mathsf{R} = \begin{cases} !\mathsf{S} & \textit{if } \mathsf{R} =?\mathsf{S} \\ \mathsf{R} & \textit{o.w.} \end{cases} \qquad \Pi\,\mathsf{R}+\Pi'\,\mathsf{R}' = (\Pi+\Pi')\,(\mathsf{R} \sqcup \mathsf{R}')$$

We are now ready to assign a type $\Delta(P)$ to each process P. This is done in an algebraic fashion. A $\chi.P$ process is abstracted with the type χ, neglecting the continuation P, since we only care about pulsingness at this stage. Internal τ actions are abstracted with τ, and potentially blocking prefixes are abstracted with ?. We use our abstract operators to handle choice and parallel composition. Restriction and recursion do not affect the result of Δ. Variables can stand for anything, so we abstract them with $\top$. Timeouts and watchdogs behave as either their first or second components, so we type them accordingly.

Definition 8. *We define the function $\Delta : \mathcal{P} \to \mathcal{T}$ as follows:*

$$\Delta(\chi.P) = \chi \quad \Delta(\tau.P) = \tau\Delta(P) \quad \Delta(a(x).P) = \Delta(\bar{a}x.P) = ?\Delta(P) \quad \Delta(X) = \top$$

$$\Delta(P + Q) = \Delta(P) + \Delta(Q) \qquad \Delta(P|Q) = \Delta(P)|\Delta(Q) \qquad \Delta((\nu a)P) = \Delta(P)$$

$$\Delta(rec\,X.P) = \Delta(P) \qquad \Delta(\lfloor P \rfloor^k(Q)) = \Delta(\lceil P \rceil^k(Q)) = \begin{cases} \Delta(P) & \textit{if } k > 0 \\ \Delta(Q) & \textit{if } k = 0 \end{cases}$$

Example 2. Recall the processes of Ex. 1. We have that:

$$\Delta(0) = \Delta(rec\, X.X) = \Delta(X) = \top$$
$$\Delta(\Omega_\infty) = rec\, X.\chi.X = \Delta(\chi.X) = \chi$$
$$\Delta(\pi_\infty.P) = \Delta(rec\, X.(\chi.X + \pi.P)) = \chi + ?\Delta(P) = !\Delta(P)$$

The first item deals with the stuck process 0: of course, no ticking is guaranteed. The second item guarantees that the first step of the perpetually ticking process Ω_∞ will be a tick. In the last item, $\pi_\infty.P$ is guaranteed to perform an action. This can be either a tick, if the left branch of the sum is chosen, or the prefix π on the right branch. In the second case, the residual pulsingness is that of P.

A subject reduction result holds for our type system. First, processes typed by $\chi, \tau\mathsf{R}, !\mathsf{R}$ actually enjoy the corresponding semantic properties (items 3a, 3b and 3c). Second, typing is preserved by process transitions, *except for ticks* (item 3d). The exception for ticks is a crucial one, yet it still allows our type system to correctly approximate the pulsingness of processes. Indeed, pulsingness does not require to consider what happens after the first tick. In the next section we shall see a further type system, which exploits the current pulsingness analysis to approximate well-timedness of processes.

Lemma 3 (Subject reduction). *For all processes P and for all types R:* Proof in App. B

(3a) $\Delta(P) \sqsubseteq \chi \implies (\forall \alpha, P' : P \xrightarrow{\alpha} P' \implies \alpha = \chi(\emptyset)) \wedge (\exists P' : P \xrightarrow{\chi(\emptyset)} P')$

(3b) $\Delta(P) \sqsubseteq \tau\mathsf{R} \implies \exists P' : P \xrightarrow{\tau} P'$

(3c) $\Delta(P) \sqsubseteq !\mathsf{R} \implies \exists P' : P \xrightarrow{\tau} P' \vee P \xrightarrow{\chi(-)} P'$

(3d) $\Delta(P) \sqsubseteq ?\mathsf{R} \implies \forall \alpha, P' : P \xrightarrow{\alpha} P' \implies \alpha = \chi(-) \vee \Delta(P') \sqsubseteq \mathsf{R}$

We now introduce a static notion of pulsingness. Theorem 2 below states that this static notion correctly approximates the dynamic one (Def. 4).

Definition 9. *We inductively define the predicate $pulsing_k(\mathsf{R})$ as follows:*
- *$pulsing_0(\mathsf{R})$ iff $\mathsf{R} = \chi$.*
- *$pulsing_{k+1}(\mathsf{R})$ iff there exists S such that $\mathsf{R} \sqsubseteq !\mathsf{S}$ and $pulsing_k(\mathsf{S})$.*

We say R is pulsing *if there exists k such that $pulsing_k(\mathsf{R})$.*
We say R is weakly pulsing *if either R is pulsing, or $\mathsf{R} = ?\mathsf{S}$ for some pulsing S.*

The following lemma is crucial in establishing our abstraction correct. Let P be a statically pulsing process, i.e. $pulsing_k(\Delta(P))$ for some k. Item 4a says that P is not stuck. Item 4b says that P will remain pulsing until it performs a tick. Finally, item 4c says that a tick will eventually be performed.

Lemma 4. *For all processes P, if $pulsing_k(\Delta(P))$, then:*

(4a) $\exists \alpha, P' : P \xrightarrow{\alpha} P' \wedge \alpha \in \{\tau, \chi(-)\}$

(4b) $P \xrightarrow{\alpha} P' \wedge \alpha \neq \chi(-) \implies k > 0 \wedge pulsing_{k-1}(\Delta(P'))$

(4c) $\forall \eta \in Tr(P) \,:\, \chi(-) \in \eta_{|k+1}$

Proof. The item (4a) follows by (3c), since $\mathit{pulsing}_k(\Delta(P))$ implies $\Delta(P) \sqsubseteq !\mathsf{R}$ for some R. The item (4b) follows by (3d), since $\mathit{pulsing}_k(\Delta(P))$ implies $\Delta(P) \sqsubseteq ?\mathsf{R}$ for some R. We prove (4c) by induction on k. If $k = 0$, (3a) suffices. If $k > 0$, note that $\eta = \alpha\eta'$ from some α, since P is not stuck by (4a) and η is maximal. If $\alpha = \chi(-)$, we directly obtain the thesis. Otherwise, by (4b) and the ind. hyp. we obtain $\chi(-) \in \eta'_{|k}$, hence $\chi(-) \in \eta_{|k+1}$. □

The main result of this section is that if (statically) the type of a process P is pulsing, then (dynamically) P is pulsing. This follows directly from 4c.

Theorem 2. *If $\mathit{pulsing}(\Delta(P))$, then P is pulsing.*

4 Well-timedness and Service Deadlines

In this section we introduce a type system that guarantees each typable process to produce an infinite number of ticks. Since only a finite number of actions can be performed between two consecutive ticks, this implies the so-called *well-timedness* (or *finite variability*) property [18]. Intuitively, this models the fact that, in any finite time interval, only a finite amount of work can be performed.

Before presenting our type system, we briefly discuss three different formalisations of well-timedness. Intuitively, any definition of this property requires that in all the traces $Tr(P)$ of a process P, the label $\chi(-)$ occurs infinitely often. That is to say, in each process P' reachable from P, $\chi(-)$ occurs in all the traces of P'. This closely corresponds to the formulation of well-timedness in [18].

Definition 10 (Well-timedness). *A process P is* well-timed *iff, whenever $P \to^* P'$ and $\eta \in Tr(P')$, we have $\chi(-) \in \eta$.*

A bit stronger definition would require that, for each residual P' of P, an upper bound k is known within which the action $\chi(-)$ will eventually occur. We call this property *strong* well-timedness.

Definition 11 (Strong well-timedness). *A process P is* strongly well-timed *iff, whenever $P \to^* P'$, there exists k such that, for all $\eta \in Tr(P')$, $\chi(-) \in \eta|_k$.*

An even stronger requirement asks that an upper bound k exists on the number of actions that can be performed between two ticks. That is to say, whatever derivation of P we have at hand, a tick will occur within k steps. This property, named here *bounded* well-timedness, closely corresponds to the *bounded variability* property mentioned in [18].

Definition 12 (Bounded well-timedness). *A process P is* bounded well-timed *iff there exists k such that, for all $P \to^* P'$ and $\eta \in Tr(P')$, $\chi(-) \in \eta|_k$.*

Note that the above three definitions of well-timedness only differ for the placement of an existential quantifier. More precisely, they can be rephrased as:

$$\begin{aligned}
\textit{well-timedness} &: \forall P', \eta : P \to^* P' \wedge \eta \in Tr(P') \implies \exists k : \chi(-) \in \eta|_k \\
\textit{strong well-timedness} &: \forall P' : P \to^* P' \implies \exists k : \forall \eta \in Tr(P') : \chi(-) \in \eta|_k \\
\textit{bounded well timedness} &: \exists k : \forall P', \eta : P \to^* P' \wedge \eta \in Tr(P') \implies \chi(-) \in \eta|_k
\end{aligned}$$

This immediately proves inclusion between the three above well-timedness properties, as formalized by the following lemma.

Lemma 5. *For all processes P: (i) if P is bounded well-timed, then P is strongly well-timed; (ii) if P is strongly well-timed, then P is well-timed.*

Separation results hold for the three notions of well-timedness (Lemma 7), i.e. there exists processes which are "classically" (yet not strongly) well-timed, and processes which are strongly (yet not boundedly) well-timed. However, classical and strong well-timedness coincide for finitely-branching LTSs (Lemma 6).

Lemma 6. *Let P be a process with a finitely-branching LTS. If P is well-timed, then it is also strongly well-timed.* Proof in App. C

Lemma 7. *There exists a well-timed process P which is not strongly well-timed. Also, there exists a strongly well-timed process Q which is not bounded well-timed.*

Proof. Take as P and Q the following process (more details in App. C):

$$P = (\nu a)\,(rec\,X.(X + \bar{a}a_\infty.\Omega_\infty | a(x).\bar{a}a_\infty.\Omega_\infty))$$
$$Q = (\nu a)\,(rec\,X.(\tau.\chi.X + \bar{a}a_\infty.\Omega_\infty |\ R)) \qquad \text{where } R = a(x)_\infty.\bar{a}a_\infty.\Omega_\infty \quad \square$$

Type system. The main goal of our type system (Table 1) is to preserve the pulsingness of a process after each χ step. This will guarantee that an infinite number of χ will be eventually produced. Type judgements have the form $\Gamma \vdash P$, where Γ is a typing environment mapping variables X to (closed) processes. The typing environment is used to keep track of recursive processes. The typing rules are mostly straightforward, so we comment on the most peculiar ones, only.

To preserve pulsingness, rule [T-Chi] checks that the continuation P of a χ-prefix is still pulsing. In the general case, the continuation can be an open process, e.g. $rec\,X.\,\chi.X$. So, before checking P for pulsingness, we subtitute the variable for its own definition by applying the environment Γ. This is done by augmenting Γ with the recursive definition in rule [T-Rec], so that we can then check $P\Gamma$ in rule [T-Chi]. Roughly, this amounts to unfold (once) the recursion.

The rules [T-TO0]/[T-WD0] deal with the case $k = 0$ of timeouts/watchdogs: in such case, we know that only Q matters. In a timeout with $k > 0$, we consider two cases. If $\Delta(P)$ predicts a τ move within k steps, we know that Q will never be executed. Otherwise, we require Q to be typeable and pulsing, as shown in rule [T-TO1]. For a watchdog, we simply check both components in rule [T-WD1].

Note that our typing rules proceed by structural induction on the syntax of processes. Indeed, the right hand side of each judgment in a premise is a strict subprocess of the right hand side of the judgment in the conclusion. Also, when constructing a typing derivation in a bottom-up fashion, at most one rule can be applied at each step. This immediately proves the decidability of type inference.

We can now state subject reduction: pulsingness and typeability, taken together, are preserved through transitions. Technically, the actual inductive statement is stronger, just requiring weak pulsingness in the hypotheses.

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P + Q}\ \text{[T-Sum]} \qquad \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P|Q}\ \text{[T-Par]}$$

$$\frac{\Gamma \vdash P \quad \mathit{pulsing}(\Delta(P\Gamma))}{\Gamma \vdash \chi.P}\ \text{[T-Chi]} \qquad \frac{\Gamma \vdash P \quad \pi \neq \chi}{\Gamma \vdash \pi.P}\ \text{[T-Pref]} \qquad \frac{\Gamma \vdash P}{\Gamma \vdash (\nu a)P}\ \text{[T-Res]}$$

$$\frac{\Gamma, X : (\mathit{rec}\, X.P)\Gamma \vdash P \quad \mathit{pulsing}(\Delta(\mathit{rec}\, X.P\Gamma))}{\Gamma \vdash \mathit{rec}\, X.P}\ \text{[T-Rec]} \qquad \Gamma, X : Q \vdash X\ \text{[T-Var]}$$

$$\frac{\Gamma \vdash Q}{\Gamma \vdash \lfloor P \rfloor^0(Q)}\ \text{[T-TO0]} \qquad \frac{\Gamma \vdash P \quad \big(\tau \in \Delta(P)|_{k+1} \ \vee\ (\Gamma \vdash Q \wedge \mathit{pulsing}(\Delta(Q\Gamma)))\big)}{\Gamma \vdash \lfloor P \rfloor^{k+1}(Q)}\ \text{[T-TO1]}$$

$$\frac{\Gamma \vdash Q}{\Gamma \vdash \lceil P \rceil^0(Q)}\ \text{[T-WD0]} \qquad \frac{\Gamma \vdash P \quad \Gamma \vdash Q \quad \mathit{pulsing}(\Delta(Q\Gamma))}{\Gamma \vdash \lceil P \rceil^{k+1}(Q)}\ \text{[T-WD1]}$$

Table 1. The type system

Theorem 3 (Subject reduction). *For all closed processes P:* Proof in App. C

$$\mathit{weakly\ pulsing}(\Delta(P)) \ \wedge\ \vdash P \ \wedge\ P \xrightarrow{\alpha} P' \implies \mathit{pulsing}(\Delta(P')) \ \wedge \vdash P'$$

We now establish the soundness of our type system. Typeable pulsing processes are well-timed (actually, strong well-timedness is guaranteed as well).

Theorem 4 (Type Soundness). *Let P be a closed process. If $\vdash P$ and $\Delta(P)$ is pulsing, then P is strongly well-timed.*

Proof. Assume $P \to^* P'$. By repeated application of Th. 3, we have $\vdash P'$ and $\mathit{pulsing}(\Delta(P'))$. By Def. 9, there exists k such that $\mathit{pulsing}_k(\Delta(P'))$. Let $\eta \in \mathit{Tr}(P')$. Then, by Lemma 4c, we get $\chi(-) \in \eta_{|k+1}$. □

Example 3. Recall the processes of Ex. 1. We have the following judgements:

$$\nvdash 0 \qquad \vdash \Omega_\infty \qquad \vdash \pi_\infty.P \ \text{ iff } \ \vdash P \ \wedge\ \mathit{pulsing}(\Delta(P))$$

Enforcing Deadlines. We now establish our main decidability result. First, we define when a process satisfies a predicate within a bounded number of ticks.

Definition 13. *Let $\phi : \mathcal{P} \to \{\mathit{false}, \mathit{true}\}$ be a decidable predicate on processes, and let $k \in \mathbb{N}$. We say that a process P satisfies "ϕ within k ticks" if there exist no runs $P = P_0 \xrightarrow{\alpha_0} P_1 \cdots \xrightarrow{\alpha_n} P_n$ such that $\chi(-)$ occurs k times in the trace $\alpha_0 \cdots \alpha_n$ but none of the processes $P_0, \ldots, P_n$ satisfies ϕ.*

In general, it is undecidable to state whether a process P satisfies ϕ within k ticks, as the halting problem can be reduced to this. Trivially, a process P can simulate a Turing machine without producing any tick, and then on termination exposes a barb so to satisfy ϕ. However, for well-timed finite-branching processes, this property turns out to be decidable.

Theorem 5. *Let P be a well-timed process with a finitely branching LTS. Let ϕ be a decidable predicate on processes, and let $k \in \mathbb{N}$. Then, it is decidable whether P satisfies ϕ in k ticks.*

Proof (Sketch). Intuitively, we can decide "P satisfies ϕ in k ticks" by exploring the LTS of P until k ticks are found in every branch. This can be done, e.g., through a depth-first search. When k ticks are found without ϕ being true, we can output *false*; otherwise, when the visit is completed, we output *true*.

This procedure terminates. To prove that, we first label P with the ordinal number $\omega \cdot k$, and then we label each other process in the LTS of P such that, in each transition, the residual has a strictly lower label than the antecedent. For a successor ordinal $\omega \cdot n + m + 1$ it is sufficient to pick $\omega \cdot n + m$ for each residual. When a limit ordinal $\omega \cdot n$ is found at a process Q, we can choose $\omega \cdot (n-1) + m$ where m is the threshold provided for Q by strong well-timedness of P, implied by the hypothesis and Lemma 6. This ensures that we will find at least k ticks before the labels reach zero. We can then establish termination by contradiction as follows. If our procedure does not terminate, it will visit an infinite number of nodes. By König's lemma, an infinite branch exists, which implies the existence of an infinite descending chain of ordinals – contradiction. □

5 Discussion and Related Work

Timed extensions of process calculi have been studied since the late eighties [1,2,5,10,11,16,19,21,24]. A number of these calculi have been compared in [18], by considering their behaviour w.r.t. some relevant time-related properties. We now briefly examine our calculus w.r.t. such properties. To do that, we rephrase them in our idiom, since the formulation in [18] considers delays instead of ticks.

Our calculus enjoys the *maximal progress* property (Th. 1), i.e. internal computation is prioritized over time ticks. This is important in SOC, since we want to model services guaranteed to terminate within a given amount of time. This would not be possible if time could pass with no control from the service.

Another relevant property is *persistency*, stating that the passing of time never conflicts with other transitions.

$$P \xrightarrow{\alpha} \ \wedge \ \alpha \neq \chi(-) \ \wedge \ P \xrightarrow{\chi(-)} P' \implies P' \xrightarrow{\alpha} \qquad \text{(persistency)}$$

This property does not hold in our calculus, e.g. in $\chi.\tau.P + a(x).Q$ the input is no longer available after a tick. A similar effect can be achieved through a timeout. Actually, persistency prevents us from modelling timeouts, because we cannot disable a pending communication after a given amount of time. Since timeouts are crucial in SOC, we admit to violate persistency in our calculus.

Time determinism states that time affects processes in a deterministic way:

$$P \xrightarrow{\chi(-)} P' \ \wedge \ P \xrightarrow{\chi(-)} P'' \implies P' = P'' \qquad \text{(time determinism)}$$

This is not the case in our calculus, as shown by $P = \chi.Q + \chi.R$. Actually, we do not believe that time determinism is an essential property for our purposes: in particular, it is not required for our decision procedure of Th. 5. However,

one can recover time determinism if needed, by using a semantics similar to that in [12], where it is true that, e.g., $P \xrightarrow{\chi(\emptyset)} Q + R$.

The *transition liveness* property states that processes are never stuck:

$$P \xrightarrow{\tau} \ \vee \ P \xrightarrow{\chi(-)} \qquad \text{(transition liveness)}$$

This property can be violated in our calculus, e.g. by 0. This is because we want to completely specify the passing of time within processes, thus making each χ-move explicit. We believe that this choice helps identifying problems in the specifications. For instance, when a potentially blocking action is used, e.g. $a(x).P$, the designer is then invited to decide whether a timeout is needed, and change that to $a(x).P + \chi.Q$, or no timeout is needed, and change that to $a(x)_{\infty}.P$. Again, this property can be recovered, if needed: it suffices to modify our semantics so that an otherwise stuck process will continuosly perform a tick.

A huge number of timed calculi has been proposed over the years. TCSP [10] is a timed extension of CSP, which enjoys several time-related properties, among which maximal progress and (a weak version of) well-timedness. Compared to [10], our calculus also enjoys the *isomorphism* property[18], i.e. the (timed) semantics of the untimed fragment of the calculus coincides with the (untimed) semantics of the untimed calculus. In [23] another Timed CSP is presented, with a synctactic condition on processes which ensures non-stuck processes to be well-timed. Compared to [23], our type system considers stuckness as well.

In [12] the timed process language TPL is presented, with a correct and complete equational theory with respect to the must preorder. There, well-timedness is not addressed. We believe the approach presented here can be adapted to TPL by redefining the types for choice and prefix.

Our procedure for enforcing deadlines consists of two phases: first, we analyse the specification to check well-timedness; then, we apply a (sound and complete) decision procedure for reachability. Actually, attaching reachability directly would be unfeasible, since such problem is undecidable for the π-calculus and for CCS. An alternative approach would be the following. First, one constructs an abstraction of the original specification, using a simpler process algebra. Then, the abstracted specification is model-checked, e.g. as in [6,9,15]. Following this approach would require, in our case, extending the types of Sect. 3, to make them track the behaviour after the first tick.

Recently, the challenges of SOC have fostered the attention to temporal aspects of programming languages for Web services. In [14] a temporal extension to the COWS language is presented. Compared to our calculus, timed COWS does not feature timeouts and watchdogs; a sort of timeout can however be defined through non-deterministic choice. On the other side, our calculus does not feature primitives to handle sessions, while [14] inherits correlation sets from COWS. However, we believe that our techniques can be suitably exploited to guarantee well-timedness of services modelled in timed COWS.

Further approaches to time are stochastic Markov chains and stochastic calculi, which have been widely used in the context of Web Services, e.g. in [20,17]. Also in these approaches, model-checking techniques are applicable [13].

Acknowledgments This research has been partially supported by EU-FETPI Global Computing Project IST-2005-16004 SENSORIA.

References

1. Jos C. M. Baeten and Jan A. Bergstra. Real time process algebra. *Formal Asp. Comput.*, 3(2):142–188, 1991.
2. Jos C. M. Baeten and Jan A. Bergstra. Discrete time process algebra. In *Proc. CONCUR*, pages 401–420, 1992.
3. M. Bartoletti, P. Degano, and G.L. Ferrari. Planning and verifying service composition. *Journal of Computer Security*, 17(5), 2009.
4. M. Bartoletti, P. Degano, G.L. Ferrari, and R. Zunino. Secure service orchestration. In *Proc. FOSAD*, 2007.
5. G. Berry and L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. In *Proc. CMU Seminar on Concurrency*, 1984.
6. Patricia Bouyer. Model-checking timed temporal logics. *ENTCS*, 231, 2009.
7. M. Bravetti and G. Zavattaro. Contract-based discovery and composition of web services. In *Proc. SFM*, 2009.
8. M. Buscemi and U. Montanari. CC-Pi: A constraint-based language for specifying service level agreements. In *Proc. ESOP*, volume 4421 of *LNCS*, 2007.
9. Sagar Chaki, Sriram K. Rajamani, and Jakob Rehof. Types as models: model checking message-passing programs. In *Proc. POPL*, 2002.
10. J. Davies and S. Schneider. An introduction to Timed CSP. Technical Report PRG-75, Oxford University Computing Laboratory, 1989.
11. Matthew Hennessy and Tim Regan. A temporal process algebra. In *Proc. FORTE*, 1990.
12. Matthew Hennessy and Tim Regan. A process algebra for timed systems. *Inf. Comput.*, 117(2):221–239, 1995.
13. Jane Hillston. Process algebras for quantitative analysis. In *Proc. LICS*, 2005.
14. Alessandro Lapadula, Rosario Pugliese, and Francesco Tiezzi. COWS: A timed service-oriented calculus. In *Proc. ICTAC*, 2007.
15. Richard Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proc. FSTTCS*, 1996.
16. Faron Moller and Chris M. N. Tofts. A temporal calculus of communicating systems. In *Proc. CONCUR*, 1990.
17. Rocco De Nicola, Diego Latella, Michele Loreti, and Mieke Massink. MarCaSPiS: a markovian extension of a calculus for services. *ENTCS*, 229(4):11–26, 2009.
18. Xavier Nicollin and Joseph Sifakis. An overview and synthesis on timed process algebras. In *Proc. CAV*, pages 376–398, 1991.
19. Xavier Nicollin and Joseph Sifakis. The algebra of timed processes, ATP: Theory and application. *Inf. Comput.*, 114(1), 1994.
20. Davide Prandi and Paola Quaglia. Stochastic COWS. In *Proc. ICSOC*, 2007.
21. George M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. *Theor. Comput. Sci.*, 58:249–261, 1988.
22. Davide Sangiorgi and David Walker. *The π-calculus: a theory of mobile processes.* Cambridge University Press, 2001.
23. Steve Schneider. An operational semantics for Timed CSP. *Inf. Comput.*, 116(2):193–213, 1995.
24. Wang Yi. CCS + time = an interleaving model for real time systems. In *Proc. ICALP*, 1991.

A Proofs for Section 2

Definition 14. *The structural congruence relation $\equiv$ is the smallest congruence satisfying the following axioms:*

$$\begin{array}{ll} P + Q \equiv Q + P & P + (Q + R) \equiv (P + Q) + R \\ P \mid Q \equiv Q \mid P & P \mid (Q \mid R) \equiv (P \mid Q) \mid R \\ (\nu x)(\nu y)P \equiv (\nu y)(\nu x)P & (\nu x)(P \mid Q) \equiv P \mid (\nu x)Q \quad \textit{if } x \notin \mathit{fn}(P) \end{array}$$

The definition of the set $act(P)$ of the *active names* of a process P follows in Def. 15. The only worth-mentioning cases of the definition are those dealing with recursion and parallel composition. For recursion, note that $act(P)$ is always bounded by the set of the free names of P, possibly including τ. Therefore, computing the fixed point of a recursive process $rec\,X.P$ only requires a finite number of steps. The operator *fix* denotes the least fixed point on the cpo $2^{\mathcal{A}}$. For a parallel composition $P|Q$, the active names also include τ when a name a is active in P and its co-name $\bar{a}$ is active in Q.

Definition 15. *We define the functions $act_\rho, coact_\rho : \mathcal{P} \to 2^{\mathcal{A}}$ from processes to actions as follows, where ρ is an environment that maps variables to $2^{\mathcal{A}}$.*

$$act_\rho(\bar{a}x.P) = \{\bar{a}\} \qquad act_\rho(a(x).P) = \{a\} \qquad act_\rho(\tau.P) = \{\tau\} \qquad act_\rho(\chi.P) = \emptyset$$

$$act_\rho(P + Q) = act_\rho(P) \cup act_\rho(Q) \qquad act_\rho((\nu a)P) = act_\rho(P) \setminus \{a, \bar{a}\}$$

$$act_\rho(P \mid Q) = act_\rho(P) \cup act_\rho(Q) \cup \begin{cases} \emptyset & \textit{if } (act_\rho(P) \cap coact_\rho(Q)) \setminus \{\tau\} = \emptyset \\ \{\tau\} & \textit{otherwise} \end{cases}$$

$$act_\rho(rec\,X.\,P) = \mathit{fix}(\lambda A.\, act_{\rho\{X \mapsto A\}}(P)) \qquad act_\rho(X) = \rho(X)$$

$$act_\rho(\lfloor P \rfloor^k(Q)) = act_\rho(\lceil P \rceil^k(Q)) = \begin{cases} act_\rho(P) & \textit{if } k > 0 \\ act_\rho(Q) & \textit{if } k = 0 \end{cases}$$

$$coact_\rho(P) = \{\, \bar{a} \mid a \in act_\rho(P) \,\} \cup \{\, a \mid \bar{a} \in act_\rho(P) \,\} \cup (\{\tau\} \cap act_\rho(P))$$

The operator *act* behaves coherently with structural congruence (Lemma 9) and with substitutions (Lemma 10).

Example 4. Let 0 be the stuck process $rec\,X.X$. We have that:

$$act(0) = \mathit{fix}\,(\lambda A.\, act_{\{X \mapsto A\}}(X)) = \mathit{fix}\,(\lambda A.\, A) = \emptyset$$

Let now $P = recX.(X \mid (a(x).0 + \bar{a}b))$. We have:

$$act(P) = \mathit{fix}\,(\lambda A.\, act_{\{X \mapsto A\}}(X \mid (a(x).0 + \bar{a}b))) = \{a, \bar{a}, \tau\}$$

The following lemma establishes the corretness of *act* with respect to the operational semantics of processes. That is, the active names of a process P are exactly those names involved in the transitions from P. The proof is straightforward by induction on the structure of P.

Lemma 8. (Correctness of act) *For all processes P:*

$$a \in act(P) \iff \exists x, P' : P \xrightarrow{a(x)} P' \tag{8a}$$

$$\bar{a} \in act(P) \iff \exists x, P' : P \xrightarrow{\bar{a}x} P' \vee P \xrightarrow{\bar{a}(x)} P' \tag{8b}$$

$$\tau \in act(P) \iff \exists P' : P \xrightarrow{\tau} P' \tag{8c}$$

Proof. All the items are straightforward by induction on the structure of P.

Lemma 9. *If $P \equiv Q$ then $act(P) = act(Q)$ and $coact(P) = coact(Q)$.*

Proof. Mostly straightforward. The only interesting case is parallel composition, for which it suffices to show that the condition $(act_\rho(P) \cap coact_\rho(Q)) \setminus \{\tau\} = \emptyset$ is equivalent to the symmetric $(coact_\rho(P) \cap act_\rho(Q)) \setminus \{\tau\} = \emptyset)$.

Lemma 10. $act_\rho(P\{Q/X\}) = act_{\rho\{X \mapsto act_\rho(Q)\}}(P)$.

Proof. Easy induction on the structure of P.

Lemma 11. *If $P \xrightarrow{\chi(A)} P'$, then $A = act(P)$.*

Proof. By induction on the derivation of $P \xrightarrow{\chi(A)} P'$. There are the following cases on the last rule used.

- [Chi]. We have $A = \emptyset = act(P)$.
- [SumChi]. We have $P = Q_0 + Q_1$, $P' = Q_0'$ and $A = A' \cup act(Q_1)$ for some Q_0 and Q_1 such that $Q_0 \xrightarrow{\chi(A')} Q_0'$. By the induction hypothesis, $A' = act(Q_0)$. Then, $A = A' \cup act(Q_1) = act(Q_0) \cup act(Q_1) = act(P)$.
- [ParChi]. We have $P = Q_0|Q_1$, $P' = Q_0'|Q_1'$ and $A = A' \cup B'$ for some Q_0, Q_1, A', B' such that $Q_0 \xrightarrow{\chi(A')} Q_0'$, $Q_1 \xrightarrow{\chi(B')} Q_1'$ and $stuck(A', B')$. By applying twice the induction hypothesis, we have $A' = act(Q_0)$ and $B' = act(Q_1)$. Since $stuck(A', B')$, then either $act(Q_0) \cap coact(Q_1) = \emptyset$ or $act(Q_0) \cap coact(Q_1) = \{\tau\}$. Then, $A = A' \cup B' = act(Q_0) \cup act(Q_1) = act(P)$.
- [Rec]. We have $P = rec\, X.Q$ and $Q\{P/X\} \xrightarrow{\chi(A)} P'$. By the induction hypothesis, $A = act(Q\{P/X\})$. Lemma 10 then concludes.
- [ResChi]. We have $P = (\nu a)Q$, for some Q and A' such that $Q \xrightarrow{\chi(A')} P'$ and $A = A' \setminus \{a, \bar{a}\}$. By the induction hypothesis, $A' = act(Q)$. Then, by Def. 15, $A = A' \setminus \{a, \bar{a}\} = act(Q) \setminus \{a, \bar{a}\} = act(P)$.
- [TO*,WD*]. Straightforward.

Lemma 12. *If $P \xrightarrow{\chi(A)}$ and $P \xrightarrow{\chi(B)}$ then $A = B$.*

Proof. Straightforward from Lemma 11.

B Proofs for Section 3

Lemma 13 (Right Inversion). *For all type prefixes Π, and $\mathsf{R}, \mathsf{S} \in \mathcal{T}$:*

1. *if $\mathsf{R} \sqsubseteq \chi$, then $\mathsf{R} = \chi$*
2. *if $\mathsf{R} \sqsubseteq \Pi\,\mathsf{S}$, then either one of the following cases hold:*
   - *$\Pi = \tau$ and $\mathsf{R} = \tau\mathsf{U}$ for some $\mathsf{U} \sqsubseteq \mathsf{S}$.*
   - *$\Pi \neq \tau$ and either $\mathsf{R} = \chi$ or $\mathsf{R} = \Pi'\,\mathsf{U}$, for some $\Pi' \sqsubseteq \Pi$ and $\mathsf{U} \sqsubseteq \mathsf{S}$.*

Proof. By induction on the derivation of $\mathsf{R} \sqsubseteq \mathsf{S}$.

Lemma 14 (Left Inversion). *For all type prefixes Π, and $\mathsf{R}, \mathsf{S} \in \mathcal{T}$:*

1. *if $\top \sqsubseteq \mathsf{S}$, then $\mathsf{S} = \top$*
2. *if $\chi \sqsubseteq \mathsf{S}$, then $\mathsf{S} \in \{\top, ?\mathsf{U}, !\mathsf{U}, \chi\}$ for some U.*
3. *if $\Pi\,\mathsf{R} \sqsubseteq \mathsf{S}$, then either $\mathsf{S} = \top$, or $\mathsf{S} = \Pi'\,\mathsf{U}$ for some $\Pi' \sqsupseteq \Pi$ and $\mathsf{U} \sqsupseteq \mathsf{R}$.*

Proof. By induction on the derivation of $\mathsf{R} \sqsubseteq \mathsf{S}$.

Proof of Lemma 2. First, we prove that $(\mathcal{T}, \sqsubseteq)$ is a join semilattice. By Lemma 13 and structural induction we can see that $\sqsubseteq$ is antisymmetric, so $(\mathcal{T}, \sqsubseteq)$ is a partial order. Second, we prove the following equations:

$$\top \sqcup \mathsf{R} = \top \tag{1}$$

$$\chi \sqcup \mathsf{R} = \begin{cases} !\mathsf{S} & \text{if } \mathsf{R} = \tau\mathsf{S} \\ \mathsf{R} & \text{otherwise} \end{cases} \tag{2}$$

$$\Pi\,\mathsf{R} \sqcup \Pi'\,\mathsf{S} = (\Pi \sqcup \Pi')\,(\mathsf{R} \sqcup \mathsf{S}) \tag{3}$$

Each of these equations has the form $\mathsf{R} \sqcup \mathsf{S} = \mathsf{U}$, for some R, S and U. By Def. 6, it follows directly by structural induction that U is an upper bound of R and S. It then suffices to show that, for all types V:

$$\mathsf{R} \sqsubseteq \mathsf{V} \;\wedge\; \mathsf{S} \sqsubseteq \mathsf{V} \implies \mathsf{U} \sqsubseteq \mathsf{V}$$

This is done by Lemma 14 and structural induction, as follows.

- Equation (1) ($\top \sqcup \mathsf{R} = \top$) is trivial.
- For (2), we have to prove:

  $$\chi \sqcup \mathsf{R} = \begin{cases} !\mathsf{S} & \text{if } \mathsf{R} = \tau\mathsf{S} \\ \mathsf{R} & \text{otherwise} \end{cases}$$

  We have $\chi \sqsubseteq \mathsf{V}$ and $\mathsf{R} \sqsubseteq \mathsf{V}$. Since $\chi \sqsubseteq \mathsf{V}$, lemma 14 implies that $\mathsf{V} \in \{\top, ?\mathsf{W}, !\mathsf{W}, \chi\}$ for some W. There are two cases.
  If $\mathsf{R} = \tau\mathsf{R}'$ for some R', then $\mathsf{U} = !\mathsf{R}'$. By Lemma 14, either $\mathsf{V} = \top$, or there exist $\Pi' \sqsupseteq \tau$ and $\mathsf{W} \sqsupseteq \mathsf{R}'$ such that $\mathsf{V} = \Pi'\,\mathsf{W}$. Summing up, we have one of the following three subcases.

- $\mathsf{V} = \top$. It is trivially true that $\mathsf{V} = \top \sqsupseteq !\mathsf{R}'$.
- $\mathsf{V} = ?\mathsf{W}$ and $\mathsf{W} \sqsupseteq \mathsf{R}'$. By Def. 6 we have $\mathsf{V} = ?\mathsf{W} \sqsupseteq !\mathsf{W} \sqsupseteq !\mathsf{R}'$.
- $\mathsf{V} = !\mathsf{W}$ and $\mathsf{W} \sqsupseteq \mathsf{R}'$. By Def. 6 we have $\mathsf{V} = !\mathsf{W} \sqsupseteq !\mathsf{R}'$

Otherwise, if R has not the form $\tau\mathsf{R}'$, then $\mathsf{U} = \mathsf{R}$. The thesis follows from the hypothesis $\mathsf{R} \sqsubseteq \mathsf{V}$.

– For (3), we have to prove:

$$\Pi\,\mathsf{R} \sqcup \Pi'\,\mathsf{S} = (\Pi \sqcup \Pi')\,(\mathsf{R} \sqcup \mathsf{S})$$

By Lemma 14, $\Pi\,\mathsf{R} \sqsubseteq \mathsf{V}$ and $\Pi'\,\mathsf{S} \sqsubseteq \mathsf{V}$ imply, respectively:

- $\mathsf{V} = \top$ or $\mathsf{V} = \Pi''\,\mathsf{W}$, for some $\Pi'' \sqsupseteq \Pi$ and $\mathsf{W} \sqsupseteq \mathsf{R}$
- $\mathsf{V} = \top$ or $\mathsf{V} = \Pi''\,\mathsf{W}$, for some $\Pi'' \sqsupseteq \Pi'$ and $\mathsf{W} \sqsupseteq \mathsf{S}$

Summing up, either $\mathsf{V} = \top$, or $\mathsf{V} = \Pi''\mathsf{W}$ for some $\Pi'' \sqsupseteq \Pi \sqcup \Pi'$ and $\mathsf{W} \sqsupseteq \mathsf{R} \sqcup \mathsf{S}$. In both cases, Def. 6 trivially implies that $\mathsf{V} \sqsupseteq (\Pi \sqcup \Pi')\,(\mathsf{R} \sqcup \mathsf{S})$.

Of course, types are stable under structural equivalence.

Lemma 15. *If $P \equiv Q$, then $\Delta(P) = \Delta(Q)$.*

Proof. Easy structural induction.

Substituting an actual process for a free variable can cause a refinement of the type, only.

Lemma 16. $\Delta(P\{Q/X\}) \sqsubseteq \Delta(P)$

Proof. By structural induction. The base case $\Delta(X\{Q/X\}) \sqsubseteq \Delta(X) = \bot$ is trivial. The inductive cases follow from our type operators being monotonic.

Lemma 17. *For all P, P', Q, Q', and for all X, if $\sigma = \{Q/X\}$:*

(17a) $\quad P \xrightarrow{\tau} P' \implies P\sigma \xrightarrow{\tau} P'\sigma$

(17b) $\quad P \xrightarrow{\chi(\emptyset)} P' \wedge \forall \alpha : (Q \xrightarrow{\alpha} Q' \implies \alpha = \chi(\emptyset)) \implies P\sigma \xrightarrow{\chi(\emptyset)} P'\sigma$

Proof of Lemma 3. We start with proving (3a), that is:

$$\Delta(P) \sqsubseteq \chi \implies (\forall \alpha, P' : P \xrightarrow{\alpha} P' \implies \alpha = \chi(\emptyset)) \wedge (\exists P' : P \xrightarrow{\chi(\emptyset)} P')$$

We will first prove that, for all α, if $P \xrightarrow{\alpha} P'$ then $\alpha = \chi(\emptyset)$. This is done by induction on the derivation of $P \xrightarrow{\alpha} P'$. We have the following cases on the last rule used in the derivation $P \xrightarrow{\alpha} P'$. Note that the hypothesis $\Delta(P) \sqsubseteq \chi$ requires that P has one of the following forms: $\chi.Q$, $Q_0 + Q_1$, $Q_0|Q_1$, $(\nu a)Q$, $rec\,X.Q$. Therefore, only the following cases apply:

– case [Chi]. Straightforward.

- case [Sum]. We have $P = Q_0 + Q_1$ with $\alpha \neq \chi$, and either $Q_0 \xrightarrow{\alpha} Q_0'$ and $P' = Q_0'$, or $Q_1 \xrightarrow{\alpha} Q_1'$ and $P' = Q_1'$. Since the two cases are symmetrical, it suffices to consider e.g. $P' = Q_0'$. We shall prove by contradiction that this rule does not apply. By Def. 7, $\Delta(P) = \Delta(Q_0) + \Delta(Q_1) \sqsubseteq \chi$ requires $\Delta(Q_0) = \Delta(Q_1) = \chi$. By the induction hypothesis, it follows that $\alpha = \chi(\emptyset)$. This is a contradiction, since [Sum] requires $\alpha \neq \chi$.
- case [SumChi]. Straightforward.
- case [Par]. Similarly to the case [Sum], $\Delta(P) = \Delta(Q_0)|\Delta(Q_1) \sqsubseteq \chi$ requires $\Delta(Q_0) = \Delta(Q_1) = \chi$, and the induction hypothesis leads to a contradiction since $\alpha \neq \chi$.
- case [ParChi] Straightforward.
- case [Comm]. We have $P = Q_0|Q_1$ with $\alpha = \tau$. Similarly to the case [Sum], $\Delta(P) = \Delta(Q_0)|\Delta(Q_1) \sqsubseteq \chi$ requires $\Delta(Q_0) = \Delta(Q_1) = \chi$, and the induction hypothesis leads to a contradiction since $\alpha = \tau$.
- case [Rec]. We have $P = rec\,X.Q$ for some Q such that $Q\{P/X\} \xrightarrow{\alpha} P'$. By Def. 8 and by hypothesis we have $\Delta(P) = \Delta(Q) \sqsubseteq \chi$. By Lemma 16, it follows that $\Delta(Q\{P/X\}) \sqsubseteq \chi$. Then, by the induction hypothesis we have the thesis $\alpha = \chi(\emptyset)$.
- case [Open]. We have $P = (\nu x)Q$ for some Q such that $Q \xrightarrow{\bar{a}x} P'$, with $\alpha = \bar{a}(x)$. By Def. 8, $\Delta(P) = \Delta(Q) \sqsubseteq \chi$, which by the induction hypothesis implies $\bar{a}x = \chi$ – contradiction.
- case [Close]. Similarly to the case [Sum], $\Delta(P) = \Delta(Q_0)|\Delta(Q_1) \sqsubseteq \chi$ requires $\Delta(Q_0) = \Delta(Q_1) = \chi$, and the induction hypothesis leads to a contradiction since $\bar{a}(x) \neq \chi$.
- case [Res]. Similarly to the case [Sum], we have a contradiction with the side condition $\alpha \neq \chi$.
- case [ResChi],[TO*],[WD*]. Straightforward.

We now prove that, if $\Delta(P) \sqsubseteq \chi$, then $P \xrightarrow{\chi(\emptyset)} P'$ for some P'. By induction on the structure of P, we have the following exhaustive cases:

- $P = \chi.Q$. Straightforward.
- $P = Q_0 + Q_1$. By Def. 7, it must be $\Delta(Q_0) = \Delta(Q_1) = \chi$. By the induction hypothesis on the left component Q_0 (the other case is symmetrical), it follows that $Q_0 \xrightarrow{\chi(\emptyset)} Q_0'$ for some Q_0'. By the proof of the first part of (3a), we have that $Q_1 \xrightarrow{\alpha} Q_1'$ implies $\alpha = \chi(\emptyset)$. Then, by Lemma 8 it follows that $act(Q_1) = \emptyset$. The rule [SumChi] then concludes.
- $P = Q_0|Q_1$. By Def. 7, it must be $\Delta(Q_0) = \Delta(Q_1) = \chi$. By applying twice the induction hypothesis, it follows that $Q_0 \xrightarrow{\chi(\emptyset)} Q_0'$ for some Q_0', and $Q_1 \xrightarrow{\chi(\emptyset)} Q_1'$ for some Q_1'. The rule [ParChi] then concludes.
- $P = (\nu a)Q$. We have $\Delta(Q) = \Delta(P) \sqsubseteq \chi$. By the induction hypothesis, there exists Q' such that $Q \xrightarrow{\chi(\emptyset)} Q'$. The rule [ResChi] then concludes.
- $P = rec\,X.Q$. We have $\Delta(Q) = \Delta(P) \sqsubseteq \chi$. Let $\sigma = \{P/X\}$. The first part of (3a) implies that, whenever $Q \xrightarrow{\alpha} Q'$, then $\alpha = \chi(\emptyset)$. Then, by (17b) it follows that $Q\sigma \xrightarrow{\chi(\emptyset)} Q'\sigma$. The rule [Rec] then concludes.

– $P = \lfloor Q_0 \rfloor^k(Q_1)$, $P = \lceil Q_0 \rceil^k(Q_1)$. Direct by the induction hypothesis.

For (3b), we have to prove that:

$$\Delta(P) \sqsubseteq \tau\mathsf{R} \implies \exists P' : P \xrightarrow{\tau} P'$$

We proceed by induction on the structure of P. According to Def. 6 and Def. 8, $\Delta(P) \sqsubseteq \tau\mathsf{R}$ requires that P has one of the following forms:

– $P = \tau.Q$, and $\Delta(P) = \tau\Delta(Q)$. The rule [Pref] then yields $P \xrightarrow{\tau} Q$.
– $P = Q_0 + Q_1$ and $\Delta(P) = \Delta(Q_0) + \Delta(Q_1)$. By Def. 7, $\Delta(Q_0) + \Delta(Q_1) \sqsubseteq \tau\mathsf{R}$ implies that $\Delta(Q_0)$ and $\Delta(Q_1)$ have one of the following forms (since the operator $+$ is commutative, the symmetrical cases are omitted)

$$\begin{array}{lll} \Delta(Q_0) = \tau\mathsf{S_0} & \Delta(Q_1) = \chi & \textit{with}\ \ \mathsf{S_0} \sqsubseteq \mathsf{R} \\ \Delta(Q_0) = \tau\mathsf{S_0} & \Delta(Q_1) = \Pi\,\mathsf{S_1} & \textit{with}\ \ \mathsf{S_0} \sqcup \mathsf{S_1} \sqsubseteq \mathsf{R} \end{array}$$

  In both cases, we have $\Delta(Q_0) \sqsubseteq \tau\mathsf{S_0}$, hence by the induction hypothesis there exists Q_0' such that $Q_0 \xrightarrow{\tau} Q_0'$. Then, by the rule [Sum] we conclude that $P \xrightarrow{\tau} Q_0'$.
– $P = Q_0|Q_1$ and $\Delta(P) = \Delta(Q_0)|\Delta(Q_1)$. By Def. 7, $\Delta(Q_0)|\Delta(Q_1) \sqsubseteq \tau\mathsf{R}$ implies that $\Delta(Q_0)$ and $\Delta(Q_1)$ have one of the following forms (since the operator $|$ is commutative, the symmetrical cases are omitted)

$$\begin{array}{llll} \Delta(Q_0) = \tau\mathsf{S_0} & \Delta(Q_1) = \chi & \textit{with}\ \ \mathsf{S_0} \sqsubseteq \mathsf{R} \\ \Delta(Q_0) = \tau\mathsf{S_0} & \Delta(Q_1) = \Pi\,\mathsf{S_1} & \textit{with}\ \ \mathsf{S_0}|\Pi\,\mathsf{S_1} \sqcup \tau\mathsf{S_0}|\mathsf{S_1} \sqcup \mathsf{S_0}|\mathsf{S_1} \sqsubseteq \mathsf{R} \end{array}$$

  In both cases, we have $\Delta(Q_0) \sqsubseteq \tau\mathsf{S_0}$, hence by the induction hypothesis there exists Q_0' such that $Q_0 \xrightarrow{\tau} Q_0'$. Then, by the rule [Par] we conclude that $P \xrightarrow{\tau} Q_0'|Q_1$.
– $P = (\nu a)Q$, and $\Delta(P) = \Delta(Q) \sqsubseteq \tau\mathsf{R}$. By the induction hypothesis, there exists Q' such that $Q \xrightarrow{\tau} Q'$. The rule [Res] then concludes.
– $P = rec\,X.Q$, and $\Delta(P) = \Delta(Q) \sqsubseteq \tau\mathsf{R}$. By Lemma 16, it follows that $\Delta(Q\{P/X\}) \sqsubseteq \Delta(Q) \sqsubseteq \mathsf{R}$. By the induction hypothesis, there exists Q' such that $Q\{P/X\} \xrightarrow{\tau} Q'$. The rule [Rec] then concludes.
– $P = \lfloor Q_0 \rfloor^k(Q_1)$, $P = \lceil Q_0 \rceil^k(Q_1)$. Direct by the induction hypothesis.

For (3c), we have to prove that:

$$\Delta(P) \sqsubseteq !\mathsf{R} \implies \exists P' : P \xrightarrow{\tau} P' \vee P \xrightarrow{\chi(-)} P'$$

We proceed by induction on the structure of P. According to Def. 6 and Def. 8, $\Delta(P) \sqsubseteq !\mathsf{R}$ requires that P has one of the following forms:

– $P = \chi.Q$. By the rule [Chi], $P \xrightarrow{\chi(\emptyset)} Q$.
– $P = \tau.Q$. By the rule [Pref], $P \xrightarrow{\tau} Q$.

– $P = Q_0 + Q_1$ and $\Delta(P) = \Delta(Q_0) + \Delta(Q_1)$. By Def. 7, $\Delta(Q_0) + \Delta(Q_1) \sqsubseteq !\mathsf{R}$ implies that $\Delta(Q_0)$ and $\Delta(Q_1)$ have one of the following forms (since the operator $+$ is commutative, the symmetrical cases are omitted)

$\Delta(Q_0) = \chi$	$\Delta(Q_1) = \chi$	
$\Delta(Q_0) = \tau\mathsf{S}_0$	$\Delta(Q_1) = \chi$	*with* $\mathsf{S}_0 \sqsubseteq \mathsf{R}$
$\Delta(Q_0) = \tau\mathsf{S}_0$	$\Delta(Q_1) = \Pi\,\mathsf{S}_1$	*with* $\mathsf{S}_0 \sqcup \mathsf{S}_1 \sqsubseteq \mathsf{R}$
$\Delta(Q_0) = \chi$	$\Delta(Q_1) = \Pi\,\mathsf{S}_1$	*with* $\mathsf{S}_1 \sqsubseteq \mathsf{R}$
$\Delta(Q_0) = !\mathsf{S}_0$	$\Delta(Q_1) = \Pi\,\mathsf{S}_1$	*with* $\mathsf{S}_0 \sqcup \mathsf{S}_1 \sqsubseteq \mathsf{R}$ *and* $\Pi \neq \tau$

For the first case, we have shown in (3a) that $Q_0 \xrightarrow{\chi(\emptyset)} Q_0'$ for some Q_0', and that, whenever $Q_1 \xrightarrow{\alpha} Q_1'$ for some Q_1', then $\alpha = \chi$. By Lemma 8, it follows that $\tau \notin act(Q_1)$. The rule [SumChi] then concludes, with $P' = Q_0'$.
For the second and third cases, we have shown in (3b) that $Q_0 \xrightarrow{\tau} P'$. Therefore, the rule [Sum] yields the thesis $P \xrightarrow{\tau} P'$.

For the fourth case, we have shown in (3a) that $Q_0 \xrightarrow{\chi(\emptyset)} Q_0'$ for some Q_0'. There are two further subcases.

- if $\tau \notin act(Q_1)$, then the thesis follows directly from the rule [SumChi], with $P' = Q_0'$.
- if $\tau \in act(Q_1)$, then Lemma 8 implies that $Q_1 \xrightarrow{\tau} Q_1'$ for some Q_1'. Therefore, the thesis follows from the rule [Sum], with $P' = Q_1'$.

For the fifth case, we have $\Delta(Q_0) \sqsubseteq !\mathsf{S}_0$, hence by the induction hypothesis there exists P' such that either $Q_0 \xrightarrow{\tau} P'$ or $Q_0 \xrightarrow{\chi(\emptyset)} P'$. There are three further subcases.

- If $Q_0 \xrightarrow{\tau} Q_0'$, then the thesis follows directly from the rule [Sum], with $P' = Q_0'$.
- if $Q_0 \xrightarrow{\chi(A)} Q_0'$ and $\tau \notin act(Q_1)$, then the thesis follows directly from the rule [SumChi], with $P' = Q_0'$.
- if $Q_0 \xrightarrow{\chi(A)} Q_0'$ and $\tau \in act(Q_1)$, then by Lemma 8 we have that $Q_1 \xrightarrow{\tau} Q_1'$ for some Q_1'. The thesis follows from the rule [Sum], with $P' = Q_1'$.

– $P = Q_0|Q_1$ and $\Delta(P) = \Delta(Q_0)|\Delta(Q_1)$. By Def. 7, $\Delta(Q_0)|\Delta(Q_1) \sqsubseteq !\mathsf{R}$ implies that $\Delta(Q_0)$ and $\Delta(Q_1)$ have one of the following forms (since the operator $|$ is commutative, the symmetrical cases are omitted)

$\Delta(Q_0) = \chi$	$\Delta(Q_1) = \chi$	
$\Delta(Q_0) = \chi$	$\Delta(Q_1) = \Pi\,\mathsf{S}_1$	*with* $\Pi \sqsubseteq !$ *and* $\mathsf{S}_1 \sqsubseteq \mathsf{R}$
$\Delta(Q_0) = \tau\mathsf{S}_0$	$\Delta(Q_1) = \Pi\,\mathsf{S}_1$	*with* $\mathsf{S}_0 \vert \Pi\,\mathsf{S}_1 \sqcup \tau\mathsf{S}_0 \vert \mathsf{S}_1 \sqcup \mathsf{S}_0 \vert \mathsf{S}_1 \sqsubseteq \mathsf{R}$
$\Delta(Q_0) = !\mathsf{S}_0$	$\Delta(Q_1) = \Pi\,\mathsf{S}_1$	*with* $\Pi \sqsubseteq !$ *and* $\mathsf{S}_0 \vert \Pi\,\mathsf{S}_1 \sqcup !\mathsf{S}_0 \vert \mathsf{S}_1 \sqcup \mathsf{S}_0 \vert \mathsf{S}_1 \sqsubseteq \mathsf{R}$

For the first case, we have shown in (3a) that $Q_0 \xrightarrow{\chi(\emptyset)} Q_0'$ for some Q_0', and $Q_1 \xrightarrow{\chi(\emptyset)} Q_1'$ for some Q_1'. The rule [ParChi] then concludes, with $P' = Q_0'|Q_1'$ (note that $stuck(\emptyset, \emptyset)$ is true).

For the second case, we have shown in (3a) that $Q_0 \xrightarrow{\chi(\emptyset)} Q_0'$ for some Q_0'. Since $\Delta(Q_1) \sqsubseteq !\mathsf{R}$, the induction hypothesis implies that either $Q_1 \xrightarrow{\tau} Q_1'$ or $Q_1 \xrightarrow{\chi(-)} Q_1'$, for some Q_1'. There are then two subcases.

- if $Q_1 \xrightarrow{\tau} Q_1'$, then the thesis follows directly from the rule [Par], with $P' = Q_0|Q_1'$.
- if $Q_1 \xrightarrow{\chi(-)} Q_1'$, then the thesis follows directly from the rule [ParChi] with $P' = Q_0'|Q_1'$ (note that $stuck(\emptyset, A)$ is true for all A).

For the third case, we have shown in (3b) that $Q_0 \xrightarrow{\tau} Q_0'$ for some Q_0'. The thesis follows directly from the rule [Par], with $P' = Q_0'|Q_1$.
For the fourth case, since $\Delta(Q_0) \sqsubseteq !\mathsf{S_0}$ then the induction hypothesis implies one of the following two subcases:

- $Q_0 \xrightarrow{\tau} Q_0'$. In this case, the rule [Par] concludes, with $P' = Q_0'|Q_1$.
- $Q_0 \xrightarrow{\chi(A)} Q_0'$, for some Q_0'. In this case, since $\Delta(Q_1) \sqsubseteq !\mathsf{R}$ we can apply again the induction hypothesis and obtain that either $Q_1 \xrightarrow{\tau} Q_1'$ or $Q_1 \xrightarrow{\chi(B)} Q_1'$, for some Q_1'. In the former case, the rule [Par] concludes, with $P' = Q_0|Q_1'$. In the latter case, we have two further subcases, according to whether $stuck(A, B)$ is true or false. If $stuck(A, B)$ is true, then the rule [ParChi] yields the thesis with $P' = Q_0'|Q_1'$. If $stuck(A, B)$ is false, then by definition there must exist a name a such that either $a \in A, \bar{a} \in B$ or $a \in B, \bar{a} \in A$. Since the two cases are symmetrical, we consider e.g. the first one. By Lemma 11, we have that $A = act(Q_0)$ and $B = act(Q_1)$. Then, Lemma 8 implies that $Q_0 \xrightarrow{a(x)} Q_0''$ for some Q_0'', and one of the following cases holds for Q_1: either $Q_1 \xrightarrow{\bar{a}x} Q_1''$, or $Q_1 \xrightarrow{\bar{a}(x)} Q_1''$ for some Q_1''. The thesis is obtained through the [Comm] or the [Close] rules, respectively.

– $P = (\nu a)Q$. We have $\Delta(P) = \Delta(Q) \sqsubseteq !\mathsf{R}$, hence the induction hypothesis yields either $Q \xrightarrow{\tau} Q'$ or $Q \xrightarrow{\chi(-)} Q'$, for some Q'. We then obtain thesis through the rule [Res] or the rule [ResChi], respectively.
– $P = rec\, X.Q$, We have $\Delta(P) = \Delta(Q) \sqsubseteq !\mathsf{R}$. By Lemma 16, $\Delta(Q\{P/X\}) \sqsubseteq \Delta(Q) \sqsubseteq !\mathsf{R}$. The induction hypothesis then gives either $\Delta(Q\{P/X\}) \xrightarrow{\tau} P'$ or $\Delta(Q\{P/X\}) \xrightarrow{\chi(-)} P'$, from which the rule [Rec] concludes.
– $P = \lfloor Q_0 \rfloor^k(Q_1)$, $P = \lceil Q_0 \rceil^k(Q_1)$. Direct by the induction hypothesis.

For (3d), we have to prove that:

$$\Delta(P) \sqsubseteq ?\mathsf{R} \implies \forall \alpha, P' : P \xrightarrow{\alpha} P' \implies \alpha = \chi(-) \vee \Delta(P') \sqsubseteq \mathsf{R}$$

We proceed by induction on the derivation of $P \xrightarrow{\alpha} P'$. We have the following cases, according to the last rule used in the derivation:

– case [Pref]. We have $P = \pi.P'$ with $\pi \neq \chi$. If $\pi \neq \tau$, the thesis follows from $\Delta(P) = ?\Delta(P')$, since $?\Delta(P') \sqsubseteq ?\mathsf{R}$ implies $\Delta(P') \sqsubseteq \mathsf{R}$ by Def. 6. Otherwise, if $\pi = \tau$, the thesis follows from $\Delta(P) = \tau\Delta(P')$.

– case [Chi]. We have $P = \chi.P'$ and $\alpha = \chi(\emptyset)$, which implies the thesis.
– case [Sum]. We have $P = Q_0 + Q_1$, and either $Q_0 \xrightarrow{\alpha} Q_0'$ and $P' = Q_0'$, or $Q_1 \xrightarrow{\alpha} Q_1'$ and $P' = Q_1'$. Since the two cases are symmetrical, it suffices to consider e.g. $P' = Q_0'$. By Def. 8 and by hypothesis, we have $\Delta(P) = \Delta(Q_0)+\Delta(Q_1) \sqsubseteq ?\mathsf{R}$. By Lemma 13, we have that either $\Delta(Q_0)+\Delta(Q_1) = \chi$, or $\Delta(Q_0) + \Delta(Q_1) = \Pi\,\mathsf{S}$ for some Π and $\mathsf{S} \sqsubseteq \mathsf{R}$.
  By Def. 7, the case $\Delta(Q_0) + \Delta(Q_1) = \chi$ is only possible when $\Delta(Q_0) = \Delta(Q_1) = \chi$. By (3a), from $\Delta(Q_0) = \chi$ it follows that $\alpha = \chi(A)$ for some A, which implies the thesis.
  If $\Delta(Q_0) + \Delta(Q_1) = \Pi\,\mathsf{S}$ then by Def. 7 it must be the case that $\Delta(Q_0) = \Pi_0\,\mathsf{S}_0$, $\Delta(Q_1) = \Pi_1\,\mathsf{S}_1$, $\Pi = \Pi_0 + \Pi_1$, and $\mathsf{S} = \mathsf{S}_0 \sqcup \mathsf{S}_1$. Since $\mathsf{S} \sqsubseteq \mathsf{R}$, then $\mathsf{S}_0 \sqsubseteq \mathsf{R}$. By Def. 6, $\Delta(Q_0) = \Pi_0\,\mathsf{S}_0$ implies $\Delta(Q_0) \sqsubseteq ?\mathsf{S}_0 \sqsubseteq ?\mathsf{R}$. By the induction hypothesis, either $\alpha = \chi(A)$ for some A, or $\Delta(Q_0') \sqsubseteq \mathsf{R}$. The first case implies the thesis. In the second case, the thesis follows from $\Delta(P') = \Delta(Q_0') \sqsubseteq \mathsf{R}$.
– case [SumChi]. We have $\alpha = \chi(A)$, which implies the thesis.
– case [Par]. We have $P = Q_0|Q_1$, and either $Q_0 \xrightarrow{\alpha} Q_0'$ and $P' = Q_0'|Q_1$, or $Q_1 \xrightarrow{\alpha} Q_1'$ and $P' = Q_0|Q_1'$. Since the two cases are symmetrical, it suffices to consider e.g. $P' = Q_0'|Q_1$. By Def. 8 and by hypothesis, we have $\Delta(P) = \Delta(Q_0)|\Delta(Q_1) \sqsubseteq ?\mathsf{R}$. By Lemma 13, we have that either $\Delta(Q_0)|\Delta(Q_1) = \chi$, or $\Delta(Q_0)|\Delta(Q_1) = \Pi\,\mathsf{S}$ for some Π and $\mathsf{S} \sqsubseteq \mathsf{R}$.
  By Def. 7, the case $\Delta(Q_0)|\Delta(Q_1) = \chi$ is only possible when $\Delta(Q_0) = \Delta(Q_1) = \chi$. By (3a), from $\Delta(Q_0) = \chi$ it follows that $\alpha = \chi(A)$ for some A, which implies the thesis.
  If $\Delta(Q_0)|\Delta(Q_1) = \Pi\,\mathsf{S}$ then by Def. 7 it must be the case that $\Delta(Q_0) = \Pi_0\,\mathsf{S}_0$, $\Delta(Q_1) = \Pi_1\,\mathsf{S}_1$, $\Pi = \Pi_0|\Pi_1$, and $\mathsf{S} = \Pi_0\,\mathsf{S}_0|\mathsf{S}_1 \sqcup \mathsf{S}_0|\Pi_1\,\mathsf{S}_1 \sqcup \mathsf{S}_0|\mathsf{S}_1$. By Def. 6, $\Pi\,\mathsf{S} \sqsubseteq ?\mathsf{R}$ implies $\mathsf{S} \sqsubseteq \mathsf{R}$. Hence, $\mathsf{S}_0|\Pi_1\,\mathsf{S}_1 \sqsubseteq \mathsf{R}$. Also, $\Delta(Q_0) = \Pi_0\,\mathsf{S}_0$ implies $\Delta(Q_0) \sqsubseteq ?\mathsf{S}_0$. By the induction hypothesis, we have that either $\alpha = \chi(A)$ for some A, or $\Delta(Q_0') \sqsubseteq \mathsf{S}_0$. In the first case, the thesis holds trivially; in the second case, the thesis follows from:
$$\Delta(P') \;=\; \Delta(Q_0') \mid \Delta(Q_1) \;\sqsubseteq\; \mathsf{S}_0 \mid \Pi_1\,\mathsf{S}_1 \;\sqsubseteq\; \mathsf{R}$$
– case [ParChi]. We have $\alpha = \chi(A)$, which implies the thesis.
– case [Comm]. We have $P = Q_0 \mid Q_1$, $\alpha = \tau$, and $P' = Q_0' \mid Q_1'$, for some Q_0 and Q_1 such that $Q_0 \xrightarrow{\bar{a}(x)} Q_0'$ and $Q_1 \xrightarrow{a(x)} Q_1'$.
  By Def. 8 and by hypothesis, we have $\Delta(P) = \Delta(Q_0)|\Delta(Q_1) \sqsubseteq ?\mathsf{R}$. By Lemma 13, we have that either $\Delta(Q_0)|\Delta(Q_1) = \chi$, or $\Delta(Q_0)|\Delta(Q_1) = \Pi\,\mathsf{S}$ for some Π and $\mathsf{S} \sqsubseteq \mathsf{R}$.
  The first case is dealt with similarly to the corresponding case for rule [Par]. For the second case, by Def. 7 we have that $\Delta(Q_0) = \Pi_0\,\mathsf{S}_0$, $\Delta(Q_1) = \Pi_1\,\mathsf{S}_1$, $\Pi = \Pi_0|\Pi_1$, and $\mathsf{S} = \Pi_0\,\mathsf{S}_0|\mathsf{S}_1 \sqcup \mathsf{S}_0|\Pi_1\,\mathsf{S}_1 \sqcup \mathsf{S}_0|\mathsf{S}_1$. By Def. 6, $\Pi\,\mathsf{S} \sqsubseteq ?\mathsf{R}$ implies $\mathsf{S} \sqsubseteq \mathsf{R}$. Hence, $\mathsf{S}_0|\mathsf{S}_1 \sqsubseteq \mathsf{R}$. Also, $\Delta(Q_0) = \Pi_0\,\mathsf{S}_0$ and $\Delta(Q_1) = \Pi_1\,\mathsf{S}_1$ imply $\Delta(Q_0) \sqsubseteq ?\mathsf{S}_0$ and $\Delta(Q_1) \sqsubseteq ?\mathsf{S}_1$, respectively. By applying twice the induction hypothesis, we have that $\Delta(Q_0') \sqsubseteq \mathsf{S}_0$ and $\Delta(Q_1') \sqsubseteq \mathsf{S}_1$. The thesis follows from:
$$\Delta(P') \;=\; \Delta(Q_0') \mid \Delta(Q_1') \;\sqsubseteq\; \mathsf{S}_0 \mid \mathsf{S}_1 \;\sqsubseteq\; \mathsf{R}$$

– case [Rec]. We have $P = rec\, X.Q$, for some Q such that $Q\{P/X\} \xrightarrow{\alpha} P'$. By Def. 8 and by hypothesis, $\Delta(P) = \Delta(Q) \sqsubseteq ?\mathsf{R}$. By Lemma 16, $\Delta(Q\{P/X\}) \sqsubseteq \Delta(Q)$. The thesis follows by the induction hypothesis.
– case [Open]. We have $P = (\nu x)Q$ and $P' = Q'$, for some Q and Q' such that $Q \xrightarrow{\bar{a}x} Q'$. By Def. 8 and by hypothesis, $\Delta(P) = \Delta(Q) \sqsubseteq ?\mathsf{R}$. Then, by the induction hypothesis it follows that $\Delta(Q') \sqsubseteq \mathsf{R}$ (note that $\bar{a}x \neq \chi(A)$ for any A. The thesis follows then from $\Delta(P') = \Delta(Q')$.
– case [Close]. Similar to the case [Comm].
– case [Res]. We have $P = (\nu a)Q$ and $P' = (\nu a)Q'$, for some Q such that $Q \xrightarrow{\alpha} Q'$ with $\alpha \neq \chi$. By Def. 8 and by hypothesis, $\Delta(P) = \Delta(Q) \sqsubseteq ?\mathsf{R}$. Then, by the induction hypothesis, we have $\Delta(Q') \sqsubseteq \mathsf{R}$ (the case $Q \xrightarrow{\chi(A)} Q'$ is excluded by $\alpha \neq \chi$). Then, $\Delta(P') = \Delta(Q') \sqsubseteq \mathsf{R}$.
– case [ResChi]. We have $P = (\nu a)Q$ and $P' = (\nu a)Q'$, for some Q' such that $Q \xrightarrow{\chi(B)} Q'$ and $\alpha = \chi(B \setminus \{a, \bar{a}\})$, which implies the thesis.
– case [TO*],[WD*]. Straightforward.

Lemma 18. *Let* $pulsing(\mathsf{R})$ *and* $pulsing(\mathsf{S})$*. Then,*

(18a) $pulsing(\mathsf{R} \sqcup \mathsf{S})$
(18b) $pulsing(\mathsf{R}|\mathsf{S})$
(18c) $pulsing(\mathsf{R} + \mathsf{S})$

Proof. Let r, s be the pulsingness degrees of R, S, respectively.

– For (18a), we proceed by induction on the pair (r, s), ordered pointwise.
  • If $r = s = 0$, hence $\mathsf{R} = \mathsf{S} = \chi$, the thesis follows from $\mathsf{R} \sqcup \mathsf{S} = \chi$.
  • Otherwise, if $r = 0$, $s > 0$, then we have $\mathsf{S} \sqsubseteq !\mathsf{S}'$ where $pulsing_{s-1}(\mathsf{S}')$. Then, $\mathsf{R} \sqcup \mathsf{S} \sqsubseteq \chi \sqcup !\mathsf{S}' = !\mathsf{S}' = \mathsf{S}$ which is pulsing. The case $r > 0, s = 0$ is analogous.
  • Otherwise, if $r > 0, s > 0$, we have $\mathsf{R} \sqsubseteq !\mathsf{R}'$ and $\mathsf{S} \sqsubseteq !\mathsf{S}'$ where $pulsing_{r-1}(\mathsf{R}')$ and $pulsing_{s-1}(\mathsf{S}')$. Then, $\mathsf{R} \sqcup \mathsf{S} \sqsubseteq !\mathsf{R}' \sqcup !\mathsf{S}' = !(\mathsf{R}' \sqcup \mathsf{S}')$. We conclude by the induction hypothesis, stating that $\mathsf{R}' \sqcup \mathsf{S}'$ is pulsing.
– For (18b), we proceed by induction on the pair (r, s), ordered pointwise.
  • If $r = 0$, hence $\mathsf{R} = \chi$, the thesis follows from $\mathsf{R}|\mathsf{S} = \mathsf{S}$. The case $s = 0$ is analogous.
  • Otherwise, we have $\mathsf{R} \sqsubseteq !\mathsf{R}'$ and $\mathsf{S} \sqsubseteq !\mathsf{S}'$ where $pulsing_{r-1}(\mathsf{R}')$ and $pulsing_{s-1}(\mathsf{S}')$. Therefore,
$$\mathsf{R}|\mathsf{S} \sqsubseteq !(\mathsf{R}'|!\mathsf{S}' \sqcup !\mathsf{R}'|\mathsf{S}' \sqcup \mathsf{R}'|\mathsf{S}')$$
  since $|$ is monotonic. By the induction hypothesis, $\mathsf{R}'|!\mathsf{S}'$, $!\mathsf{R}'|\mathsf{S}'$, and $\mathsf{R}'|\mathsf{S}'$ are pulsing. By (18a), we have that $\mathsf{R}|\mathsf{S} \sqsubseteq !\mathsf{W}$ for some pulsing W, implying the thesis.
– For (18c), we proceed by cases on the pair (r, s).
  • If $r = s = 0$, hence $\mathsf{R} = \mathsf{S} = \chi$, the thesis follows from $\mathsf{R} + \mathsf{S} = \chi$.

- Otherwise, if $r = 0$, $s > 0$, then we have $\mathsf{S} \sqsubseteq !\mathsf{S}'$ where $\mathit{pulsing}_{s-1}(\mathsf{S}')$. Then, $\mathsf{R} + \mathsf{S} \sqsubseteq \chi + !\mathsf{S}' = !\mathsf{S}' = \mathsf{S}$ which is pulsing. The case $r > 0, s = 0$ is analogous.
- Otherwise, if $r > 0, s > 0$, we have $\mathsf{R} \sqsubseteq !\mathsf{R}'$ and $\mathsf{S} \sqsubseteq !\mathsf{S}'$ where $\mathit{pulsing}_{r-1}(\mathsf{R}')$ and $\mathit{pulsing}_{s-1}(\mathsf{S}')$. Then, $\mathsf{R} + \mathsf{S} \sqsubseteq !\mathsf{R}' + !\mathsf{S}' = !(\mathsf{R}' \sqcup \mathsf{S}')$. We conclude by (18a).

Lemma 19.

(19a) $\mathsf{R} \sqsubseteq \mathsf{S} \;\wedge\; \mathit{pulsing}(\mathsf{S}) \implies \mathit{pulsing}(\mathsf{R})$

(19b) $\neg \mathit{pulsing}(?\mathsf{R}|\mathsf{S})$

(19c) $\mathit{pulsing}(\mathsf{R}|\mathsf{S}) \implies \mathit{pulsing}(\mathsf{R})$

(19d) $\mathit{weakly\ pulsing}(\mathsf{R}|\mathsf{S}) \implies \mathit{weakly\ pulsing}(\mathsf{R})$

(19e) $\mathit{weakly\ pulsing}(\mathsf{R} + \mathsf{S}) \implies \mathit{weakly\ pulsing}(\mathsf{R})$

Proof.

- The item (19a) follows directly from Def. 9.
- For (19b), we proceed by induction on the structure of S.
  If $\mathsf{S} = \chi$, $?\mathsf{R}|\chi = ?\mathsf{R}$ is not pulsing. Otherwise, $\mathsf{S} = \Pi\,\mathsf{S}'$. If $\Pi \in \{!, ?\}$, we have $\Pi|? = ?$, making $?\mathsf{R}|\Pi\,\mathsf{S}'$ not pulsing. So, we must have $\Pi = \tau$. We obtain:
  $$?\mathsf{R} \mid \tau\mathsf{S}' \;=\; \tau(\mathsf{R}|\Pi\,\mathsf{S}' \sqcup ?\mathsf{R}|\mathsf{S}' \sqcup \mathsf{R}|\mathsf{S}')$$
  By contradiction, assume the above is pulsing. Then, also $\mathsf{R}|\Pi\,\mathsf{S}' \sqcup ?\mathsf{R}|\mathsf{S}' \sqcup \mathsf{R}|\mathsf{S}'$ is such, which in turn implies $\mathit{pulsing}(?\mathsf{R}|\mathsf{S}')$. This contradicts the inductive hypothesis.
- For (19c), we proceed by induction on the pulsingness degree k of $\mathsf{R}|\mathsf{S}$.
  If $k = 0$, then $\mathsf{R}|\mathsf{S} = \chi$, and by Def. 7 this is possible only if $\mathsf{R} = \mathsf{S} = \chi$.
  If $k > 0$, then $\mathsf{R}|\mathsf{S} \sqsubseteq !\mathsf{W}$ where $\mathit{pulsing}_{k-1}(\mathsf{W})$. If $\mathsf{R}|\mathsf{S} = \chi$, we proceed as for $k = 0$. Otherwise, by Def. 7, we have $\mathsf{R} = \Pi\,\mathsf{R}'$ and $\mathsf{S} = \Pi'\,\mathsf{S}'$ satisfying $(\Pi|\Pi') \sqsubseteq !$ and $\Pi\,\mathsf{R}'|\mathsf{S} \sqcup \mathsf{R}'|\Pi'\,\mathsf{S}' \sqcup \mathsf{R}'|\mathsf{S}' \sqsubseteq \mathsf{W}$. In particular, we have $\mathsf{R}'|\mathsf{S}' \sqsubseteq \mathsf{W}$ which, by induction hypothesis, implies $\mathit{pulsing}(\mathsf{R}')$. By (19b), we have $\Pi \neq ?$, so $\Pi \sqsubseteq !$. This implies $\mathsf{R} \sqsubseteq !\mathsf{R}'$, thus proving $\mathit{pulsing}(\mathsf{R})$.
- For (19d), we proceed by case analysis.
  - If $\mathsf{R} = \chi$, the thesis follows trivially.
  - If $\mathsf{S} = \chi$, the thesis follow from $\mathsf{R}|\chi = \mathsf{R}$.
  - Otherwise, we have $\mathsf{R} = \Pi\,\mathsf{R}'$ and $\mathsf{S} = \Pi'\,\mathsf{S}'$. So, we have $\mathsf{R}|\mathsf{S} = (\Pi|\Pi')\,(\mathsf{R}'|\Pi'\,\mathsf{S}' \sqcup \Pi\,\mathsf{R}'|\mathsf{S}' \sqcup \mathsf{R}'|\mathsf{S}')$ being weakly pulsing. This implies that $\mathsf{R}'|\Pi'\,\mathsf{S}' \sqcup \Pi\,\mathsf{R}'|\mathsf{S}' \sqcup \mathsf{R}'|\mathsf{S}'$ is pulsing, hence $\mathsf{R}'|\mathsf{S}'$ is such. By (19c), R' is pulsing, therefore R is weakly pulsing.
- For (19e), we proceed by case analysis.
  - If $\mathsf{R} = \chi$, the thesis follows trivially.
  - Otherwise, if $\mathsf{R} = ?\mathsf{R}'$ and $\mathsf{S} = \chi$, we have $\mathsf{R} + \mathsf{S} = !\mathsf{R}'$ weakly pulsing, therefore R' is pulsing, implying the R is weakly pulsing.
  - Otherwise, if $\mathsf{R} = \Pi\,\mathsf{R}'$ with $\Pi \neq ?$ and $\mathsf{S} = \chi$, we have that $\mathsf{R} + \mathsf{S} = \mathsf{R}$ is weakly pulsing by hypothesis.

- Otherwise, if $\mathsf{R} = \mathit{\Pi}\,\mathsf{R}'$ and $\mathsf{S} = \mathit{\Pi}'\,\mathsf{S}'$, we have that $\mathsf{R} + \mathsf{S} = (\mathit{\Pi} + \mathit{\Pi}')\,(\mathsf{R}' \sqcup \mathsf{S}')$ is weakly pulsing. Hence, $\mathsf{R}' \sqcup \mathsf{S}'$ is pulsing, which implies that R' is pulsing, and so R is weakly pulsing.

Lemma 20. *If $\Delta(P)$ is weakly pulsing and $P \xrightarrow{\alpha} P'$ for some $\alpha \neq \chi(-)$, then $\Delta(P')$ is pulsing.*

Proof. Straightforward from (3d).

C Proofs for Section 4

Proof of Lemma 6. Consider the truncation of $LTS(P)$ obtained by removing each subtree reached through a χ step. By hypothesis, this truncated LTS is a finitely branching tree. By contradiction, assume that P is not strongly well-timed. This implies that the truncated LTS has arbitrarily long paths, hence it is an infinite tree. By König's Lemma, an infinite path also exists, so $LTS(P)$ has an infinite χ-free trace as well. This violates well-timedness.

Proof of Lemma 7. For the first part, take:

$$P = (\nu a)(rec\, X.(X + \bar{a}a_\infty.\Omega_\infty | a(x).\bar{a}a_\infty.\Omega_\infty))$$

In one τ step, this becomes a process of the form:

$$Q_i = (\nu a)(\bar{a}a_\infty.\Omega_\infty | a(x).\bar{a}a_\infty.\Omega_\infty | \cdots | a(x).\bar{a}a_\infty.\Omega_\infty)$$

where $a(x).\bar{a}a_\infty.\Omega_\infty$ is replicated i times, for some i. Also, the converse holds: any such Q_i can be reached by P in one τ step. Note in passing that this makes $LTS(P)$ infinitely branching. Each Q_i has a single trace $\tau^i\chi^\omega$. Hence, we have P well-timed, since we can pick the limit k to be $i + 2$. Moreover, we do not have P strongly well-timed, since for any limit k, we can pick the trace of Q_{k+1} which does not perform χ in the first k steps.

For the second part, let:

$$\begin{aligned} Q &= (\nu a)Q' \\ Q' &= rec\, X.(\tau.\chi.X + \bar{a}a_\infty.\Omega_\infty \mid R) \\ R &= a(x)_\infty.\bar{a}a_\infty.\Omega_\infty \end{aligned}$$

Each run of Q can be split in two phases. In the first phase, the left branch of the choice is taken and the process builds up a number of parallel R components:

$$Q \xrightarrow{\tau}\xrightarrow{\chi} (\nu a)(Q'|R) \xrightarrow{\tau}\xrightarrow{\chi} (\nu a)(Q'|R|R) \xrightarrow{\tau}\xrightarrow{\chi} (\nu a)(Q'|R|R|R) \to \cdots$$

In the second phase, the right branch of the choice is taken, stopping recursion. This generates a sequence of communications, each one "consuming" an R:

$$\begin{aligned} \cdots \to (\nu a)(Q'|R|R|R) &\xrightarrow{\tau} (\nu a)(\bar{a}a_\infty.\Omega_\infty|R|R) \xrightarrow{\tau} (\nu a)(\Omega_\infty|\bar{a}a_\infty.\Omega_\infty|R) \\ &\xrightarrow{\tau} (\nu a)(\Omega_\infty|\Omega_\infty|\bar{a}a_\infty.\Omega_\infty) \end{aligned}$$

Note that the traces of Q are of the form $(\tau\chi)^i\tau^i\chi^\omega$. The process Q is strongly well-timed, since we can increment the limit k during the first phase, counting the number of R components. If Q keeps on building R's, it performs a χ every other step, so any $k > 2$ is fine. Otherwise, if Q starts the second phase we know that that will generate a χ after all the R's are consumed.

However, Q is *not* bounded well-timed. Given the limit k, Q can just build $k+1$ R-components, and perform $k+1$ consecutive τ steps in the second phase.

The following technical lemma states that variables occurring in Γ, but not in P, can be neglected from a judgment $\Gamma \vdash P$.

Lemma 21. *For all processes P, Q and for all variables X:*

$$\Gamma, X : Q \vdash P \ \wedge \ X \notin fv(P) \implies \Gamma \vdash P$$

Substitution of a pulsing typeable process for a variable preserves typing.

Lemma 22 (Substitution). *For all processes P and closed Q, and for all X:*

$$\mathit{pulsing}(\Delta(Q)) \ \wedge \ \Gamma \vdash Q \ \wedge \ \Gamma, X : Q \vdash P \implies \Gamma \vdash P\{Q/X\}$$

Proof. By induction on the structure of P, there are the following cases:

- case [T-Chi]. By the induction hypothesis: $\Gamma, X : Q \vdash P$ implies $\Gamma \vdash P\{Q/X\}$.
  $\mathit{pulsing}(\Delta(P\{Q/X\}\Gamma))$ and Q closed implies $\mathit{pulsing}(\Delta(P\Gamma\{Q/X\}))$
- the cases [T-Sum], [T-Par], [T-Pref], [T-Res] are straightforward by the induction hypothesis.
- case [T-Rec]. It is easy to verify that:

$$\frac{\Gamma, X : Q, Y : (\mathit{rec}\, Y.P)\Gamma\{Q/X\} \vdash P \quad \mathit{pulsing}(\Delta((\mathit{rec}\, Y.P\Gamma)\{Q/X\}))}{\Gamma, X : Q \vdash \mathit{rec}\, Y.P}$$

  implies the following:

$$\frac{\Gamma, Y : (\mathit{rec}\, Y.P\{Q/X\})\Gamma \vdash P\{Q/X\} \quad \mathit{pulsing}(\Delta((\mathit{rec}\, Y.P\{Q/X\})\Gamma))}{\Gamma \vdash \mathit{rec}\, Y.P\{Q/X\}}$$

- the remaining cases are trivial.

Theorem 3. For all closed P:

$$\mathit{weakly\ pulsing}(\Delta(P)) \ \wedge \ \vdash P \ \wedge \ P \xrightarrow{\alpha} P' \implies \mathit{pulsing}(\Delta(P')) \ \wedge \vdash P'$$

Proof. By induction on the derivation of $P \xrightarrow{\alpha} P'$. There are the following exhaustive cases.

– case [Pref]. We have $P = \pi.P'$, $\pi \neq \chi$, and $\alpha = \pi$. Each derivation of $\vdash P$ has the following form:

$$\frac{\vdash P' \quad \pi \neq \chi}{\vdash \pi.P'}\ [\text{T-Pref}]$$

which implies $\vdash P'$. Also, $\mathit{pulsing}(\Delta(P'))$ follows from Lemma 20.

– case [Chi]. We have $P = \chi.P'$ and $\alpha = \chi(\emptyset)$. Each derivation of $\vdash P$ has the following form:

$$\frac{\vdash P' \quad \mathit{pulsing}(P')}{\vdash \chi.P'}\ [\text{T-Chi}]$$

The thesis follows directly from the hypotheses of the typing rule.

– case [Sum]. We have $P = Q_0 + Q_1$ for some Q_0 such that $Q_0 \xrightarrow{\alpha} P'$ (the other case is symmetrical). Each derivation of $\vdash P$ has the following form:

$$\frac{\vdash Q_0 \quad \vdash Q_1}{\vdash Q_0 + Q_1}\ [\text{T-Sum}]$$

By Lemma 19e, *weakly pulsing*$(\Delta(Q_0+Q_1))$ implies *weakly pulsing*$(\Delta(Q_0))$. Then, by the induction hypothesis it follows that $\vdash P'$ and $\mathit{pulsing}(\Delta(P'))$.

– case [SumChi]. We have $P = Q_0 + Q_1$ and $\alpha = \chi(A \cup \mathit{act}(Q_1))$ for some Q_0 such that $Q_0 \xrightarrow{\chi(A)} P'$ (the other case is symmetrical). Similarly to the case [Sum], *weakly pulsing*$(\Delta(Q_0))$ is obtained through Lemma 19e, so the the thesis follows from the induction hypothesis.

– case [Par]. We have $P = Q_0|Q_1$ and $P' = Q_0'|Q_1$ for some Q_0 such that $Q_0 \xrightarrow{\alpha} Q_0'$ and $\alpha \neq \chi(-)$ (the other case is symmetrical). Each derivation of $\vdash P$ has the following form:

$$\frac{\vdash Q_0 \quad \vdash Q_1}{\vdash Q_0|Q_1}\ [\text{T-Par}]$$

By Lemma 19e, *weakly pulsing*$(\Delta(Q_0|Q_1))$ implies *weakly pulsing*$(\Delta(Q_0))$. Then, by the induction hypothesis it follows that $\vdash Q_0'$ and $\mathit{pulsing}(\Delta(Q_0'))$. We have the following typing derivation:

$$\frac{\vdash Q_0' \quad \vdash Q_1}{\vdash Q_0'|Q_1}\ [\text{T-Par}]$$

Also, $\mathit{pulsing}(\Delta(P'))$ follows from Lemma 20.

– case [ParChi]. We have $P = Q_0|Q_1$ and $P' = Q_0'|Q_1'$ for some Q_0, Q_1 such that $Q_0 \xrightarrow{\chi(A')} Q_0'$, $Q_1 \xrightarrow{\chi(B')} Q_1'$, and $\alpha = \chi(A' \cup B')$. Each derivation of $\vdash P$ has the following form:

$$\frac{\vdash Q_0 \quad \vdash Q_1}{\vdash Q_0|Q_1}\ [\text{T-Par}]$$

By Lemma 19e, *weakly pulsing*$(\Delta(Q_0|Q_1))$ implies *weakly pulsing*$(\Delta(Q_0))$ and *weakly pulsing*$(\Delta(Q_1))$. Then, by applying twice the induction hypothesis it follows that $\vdash Q'_0$, $\vdash Q'_1$, *pulsing*$(\Delta(Q'_0))$ and *pulsing*$(\Delta(Q'_1))$. We have the following typing derivation:

$$\frac{\vdash Q'_0 \quad \vdash Q'_1}{\vdash Q'_0|Q'_1}\ [\text{T-Par}]$$

By Lemma 18b, it follows that *pulsing*$(\Delta(Q'_0))$ and *pulsing*$(\Delta(Q'_1))$ imply *pulsing*$(\Delta(Q'_0) \mid \Delta(Q'_1))$, i.e. *pulsing*$(\Delta(P'))$.

- case [Comm]. We have $P = Q_0|Q_1$, $P' = Q'_0|Q'_1$ and $\alpha = \tau$, for some Q_0, Q_1 such that $Q_0 \xrightarrow{\bar{a}x} Q'_0$ and $Q_1 \xrightarrow{a(x)} Q'_1$. The proof proceeds similarly to the case [ParChi].
- case [Rec]. We have $P = rec\,X.Q$ for some Q such that $Q\{P/X\} \xrightarrow{\alpha} P'$. Each typing derivation of $\vdash P$ has the following form:

$$\frac{X : P \vdash Q \quad \mathit{pulsing}(\Delta(P))}{\vdash rec\,X.Q}\ [\text{T-Rec}]$$

By Lemma 16 and Def. 8, $\Delta(Q\{P/X\}) \sqsubseteq \Delta(Q) = \Delta(P)$. Therefore, since *pulsing*$(\Delta(P))$, then Lemma 19a implies that *pulsing*$(\Delta(Q\{P/X\}))$. Lemma 22 then implies $\vdash Q\{P/X\}$. The thesis follows then directly from the induction hypothesis.

- case [Open]. We have $P = (\nu x)Q$ and $\alpha = \bar{a}(x)$, for some Q such that $Q \xrightarrow{\bar{a}x} P'$. Each typing derivation for $\vdash P$ has the following form:

$$\frac{\vdash Q}{\vdash (\nu x)Q}\ [\text{T-Res}]$$

By Def. 8, *weakly pulsing*$(\Delta(P))$ implies *weakly pulsing*$(\Delta(Q))$. Then, the thesis follows directly by the induction hypothesis.

- case [Close]. We have $P = Q_0|Q_1$, $P' = (\nu x)(Q'_0|Q'_1)$ and $\alpha = \tau$, for some Q_0, Q_1 such that $Q_0 \xrightarrow{\bar{a}(x)} Q'_0$ and $Q_1 \xrightarrow{a(x)} Q'_1$. The proof is mostly the same as the case [Comm].
- case [Res]. The proof is mostly the same as the case [Open].
- case [ResChi]. The proof is mostly the same as the case [Open].
- cases [TO*],[WD*]. Simple use of the induction hypothesis.