Foundations of web transactions

Cosimo Laneve Gianluigi Zavattaro

Department of Computer Science, University of Bologna, Italy

Abstract. A timed extension of π-calculus with a transaction construct – the calculus $\texttt{web}\pi$ – is studied. The underlying model of $\texttt{web}\pi$ relies on networks of processes; time proceeds asynchronously at the network level, while it is constrained by the *local urgency* at the process level. Namely process reductions cannot be delayed to favour idle steps. The extensional model – the *timed bisimilarity* – copes with time and asynchrony in a different way with respect to previous proposals. In particular, the discriminating power of timed bisimilarity is weaker when local urgency is dropped. A labelled characterization of timed bisimilarity is also discussed.

1 Introduction

Web Services technologies intend to provide standard mechanisms for describing the interface and the services available on the web, as well as protocols for locating such services and invoking them (see e.g. WSDL [9] and UDDI [16]). To describe interfaces, services, and protocols new *web programming languages*, the so-called *orchestration* and *choreography* languages, are currently investigated. Examples of these languages are Microsoft XLANG [17] and its visual environment BizTalk, IBM WSFL [13], BPEL [2], WS-CDL [12], and WSCI [12].

Most of the web programming languages also include the notion of *web transaction*, as a unit of work involving activities that may last long periods of time. These transactions, being orthogonal to administrative domains, have the typical atomicity and isolation properties relaxed, and instead of assuming a perfect roll-back in case of failure, support the explicit programming of the compensation activity.

Despite of the great interest for web transactions, the Web Services community has not reached a common agreement on a unique notion of this form of transaction. The paper [14] gives a valuable critical comparison among three transaction protocols: BTP, WS-C/T, and WS-CAF. Other few papers (we are aware of), that discuss the formal semantics of compensable activities in this context, rely on specific proposals: the work [8] is mainly inspired by XLANG, the calculus of Butler and Ferreira [7] is inspired by BPBeans, the πt-calculus [5] considers BizTalk, the work [6] deals with short-lived transactions in BizTalk.

In this paper we follow a rather different and radical approach: we define a calculus of web transactions – the calculus $\texttt{web}\pi$ – that is independent of the different proposals discussed above and that allows to grab (we hope) the key concepts. Three major aspects are considered in $\texttt{web}\pi$: interruptible processes, failure handlers that are activated when the main process is interrupted, and time. Time has been considered because it is fundamental for dealing with the typical latency of web activities or with message losses. For instance, in ticketing services of airplane companies, the services should cancel reservations that are not confirmed within a certain period of time. Since $\mathtt{web}\pi$ is an extension of π-calculus, and the latter is emerging as one of the referring models for Web Services orchestration and choreography (it has inspired the design of languages such as XLANG and WS-CDL), we trust that the mathematical underpinnings of $\mathtt{web}\pi$ are digestible by the web service community.

The underlying model of $\mathtt{web}\pi$ includes machines and processes. The formers define networks; the latters define the computational content of locations of the networks. A location is a uniprocessor machine, written $[\, P \,]_{\widetilde{x}}$, with its own clock that is not synchronized with the clock of other locations (time progresses asynchronously between different locations). Namely, if M and N are locations, then progress of the compound machine is defined by the rule

$$\frac{\mathsf{M} \to \mathsf{M}'}{\mathsf{M} \,|\, \mathsf{N} \to \mathsf{M}' \,|\, \mathsf{N}} \tag{1}$$

Names $\widetilde{x}$ in $[\, P \,]_{\widetilde{x}}$ indicate that the location is responsible for accepting messages on such names (a name always indexes a unique location). We assume that, within a location, operations cannot be delayed in favour of idle operations – this property is called *local urgency*. For example, consider two processes running on the same location: a printer process of a warning message with a timeout and an idle process waiting for an external event. Local urgency means that, if the external event doesn't occur, then the printer process cannot be delayed. Said otherwise, the time may elapse in a location either because the process inside progresses or because no progress is possible. These two alternatives are respectively defined by the rules

$$\frac{P \to Q}{[\, P \,]_{\widetilde{x}} \to [\, Q \,]_{\widetilde{x}}} \qquad \frac{P \not\to}{[\, P \,]_{\widetilde{x}} \to [\, \phi(P) \,]_{\widetilde{x}}}$$

where ϕ is a function making the time elapse of one unit. In particular, the rightmost rule permits the elapsing of one time unit only in the case when no computational step is possible inside a machine.

Processes extend the asynchronous π-calculus with *transactions* $\langle\!| P \;;\; Q |\!\rangle_x^n$, where P and Q are the *body* and the *compensation*, respectively, n indicates the deadline, and x is the name of the transaction. The body of a transaction executes either until termination or until the transaction fails. On failure, the compensation is activated. A transaction may fail in two different ways, either explicitly (when the abort message $\overline{x}$ is consumed, where x is the name of the transaction to be aborted) or implicitly (when the deadline is reached). The deadline may be reached either because of computational steps of the body or because of computational steps of processes in parallel. Assuming that every step costs one time slot, these two alternatives are defined by the rules

$$\frac{P \to P'}{\langle\!| P \,;\, Q |\!\rangle_x^{n+1} \to \langle\!| P' \,;\, Q |\!\rangle_x^{n}} \qquad \frac{P \to P'}{P \,|\, Q \to P' \,|\, \phi(Q)}$$

Comparing the last rule and rule (1), we obtain a model for $\mathtt{web}\pi$ that is locally synchronous and globally asynchronous.

Regarding time, we have been influenced by the work of Berger and Honda about π-calculus with timers [4, 3]. A timer process $\mathsf{timer}^n(P, Q)$ behaves like P, but triggers Q if P does not move within n time units. Transactions have a rather different behaviour: in $\langle\!| P \,;\, Q |\!\rangle_x^{n}$ the process Q may be activated provided the execution of P is not terminated. Transactions have two interruption mechanisms: one associated to timeouts (as for the timers); the other is explicit – the abort message. Additionally, the model of time in [4, 3] is different from the one considered here. Berger and Honda have a rule

$$P \to \phi(P)$$

that allows the time elapse even if P may progress. In $\mathtt{web}\pi$ this rule is restricted to locations $[\, P \,]_{\tilde{x}}$, where it is reasonable to verify $P \not\to$ since P collect all the entities competing for the location processor.

The calculus $\mathtt{web}\pi$ is initially equipped with a reduction semantics, consisting of a reduction relation and a barbed bisimulation. The reduction relation defines reductions that take one unit of time. The barbed bisimulation, called *timed bisimilarity*, is sensible to the number of internal moves (it is a strong equivalence). Timed bisimilarity is also sensible to local urgency: its discriminating power of timed bisimilarity is weaker when local urgency is dropped.

In order to support direct proofs of equality, $\mathtt{web}\pi$ is also equipped with a labelled semantics. In particular, we define a labelled transition system and consider the standard notion of asynchronous bisimulation [1] that admits inputs to be also mimicked by internal moves. It turns out that asynchronous bisimulation is not a congruence because it is not substitution and time closed (this is the same as in [3]) and it is not closed by a property checking whether a process manifests an input that is not underneath a transaction. When asynchronous bisimulation is appropriately closed, the resulting equivalence, called *labelled time bisimilarity*, is equal to time bisimilarity when the discriminating power of contexts is augmented with the match operator.

The paper is structured as follows. For the sake of presentation, we separate processes and machines. The syntax and the reduction relation of $\mathtt{web}\pi$ processes and machines are respectively defined in Sections 2 and 5. Section 3 introduces timed bisimilarity and demonstrates that the discriminating power of timed bisimilarity is weaker when local urgency is dropped. Section 4 defines the labelled semantics, the corresponding congruence relation, and its relationship with timed bisimilarity. Section 6 draws some conclusive remarks.

2 The calculus $\mathtt{web}\pi$

The syntax relies on countable sets of *names*, ranged over by $x, y, z, u, \cdots$. Tuples of names are written $\widetilde{u}$. Natural numbers $\{0, 1, 2, 3, \cdots\}$ or ∞ are ranged over by $n, m, \cdots$. The syntax of $\mathtt{web}\pi$ defines *processes* P.

$$P \ ::= \quad \mathbf{0} \quad | \quad \overline{x}\,\widetilde{u} \quad | \quad x(\widetilde{u}).P \quad | \quad (x)P \quad | \quad P\,|\,P \quad | \quad !x(\widetilde{u}).P \quad | \quad \langle\!| P \ ; \ P |\!\rangle_x^n$$

A process can be the inert process $\mathbf{0}$, a message $\overline{x}\,\widetilde{u}$ sent on a name x that carries a tuple of names $\widetilde{u}$, an input $x(\widetilde{u}).P$ that consumes a message $\overline{x}\,\widetilde{w}$ and behaves like $P\{^{\widetilde{w}}/_{\widetilde{u}}\}$, a restriction $(x)P$ that behaves as P except that inputs and messages on x are prohibited, a parallel composition of processes, a replicated input $!x(\widetilde{u}).P$ that consumes a message $\overline{x}\,\widetilde{w}$ and behaves like $P\{^{\widetilde{w}}/_{\widetilde{u}}\} \,|\, !x(\widetilde{u}).P$, or a (web) transaction $\langle\!| P \ ; \ Q |\!\rangle_x^n$ that behaves as the *body* P except that, if the body does not terminate, the *compensation* Q is triggered after n steps or because of a transaction abort message $\overline{x}$. The label n, called the *time stamp* of the transaction, is a natural number or ∞. The *timeless transaction* $\langle\!| P \ ; \ Q |\!\rangle_x$ is an abbreviation for $\langle\!| P \ ; \ Q |\!\rangle_x^\infty$, and we assume that $\infty + 1 = \infty$. It is possible to write out-of-time transactions $\langle\!| P \ ; \ Q |\!\rangle_x^0$: the semantics (in particular, the structural congruence) will simplify these processes on-the-fly. It is worth to notice that the syntax of $\mathtt{web}\pi$ processes extends the asynchronous π-calculus with the transaction process.

The input $x(\widetilde{u}).P$, restriction $(x)P$, and replicated input $!x(\widetilde{u}).P$ are binders of names $\widetilde{u}$, x, and $\widetilde{u}$, respectively. The scope of these binders are the processes P. We use the standard notions of α-equivalence, *free* and *bound names* of processes, noted $\mathtt{fn}(P)$, $\mathtt{bn}(P)$, respectively. In particular,

- $\mathtt{fn}(\langle\!| P \ ; \ Q |\!\rangle_x^n) = \mathtt{fn}(P) \cup \mathtt{fn}(Q) \cup \{x\}$ and α-equivalence equates $(x)(\langle\!| P \ ; \ Q |\!\rangle_x^n)$ with $(z)(\langle\!| P\{^z/_x\} \ ; \ Q\{^z/_x\} |\!\rangle_z^n)$ provided $z \notin \mathtt{fn}(\langle\!| P \ ; \ Q |\!\rangle_x^n)$;

In the following we let $\prod_{i \in I} P_i$ be the parallel composition of the processes P_i. We also let $\tau.P$ be the process $(z)(\overline{z} \,|\, z().P)$ where $z \notin \mathtt{fn}(P)$.

Remark 1.
1. The process $\langle\!| P \ ; \ Q |\!\rangle_x^n$ is intended to define a "web" transaction (the keyword "web" is always omitted in the following). It has not to be confused with "database" transactions, which usually grant atomicity and isolation properties. These two properties are usually not retained by transactional activities over the web.
2. An high-level programming language using $\mathtt{web}\pi$ transactions should neglect names marking transactions, such as x in $\langle\!| P \ ; \ Q |\!\rangle_x^n$. Our insight is that these names are process identifiers of transactions, therefore they are dynamically generated by the run-time support of the language. This design choice may be easily implemented by using a distinguished name called this. Then programmers may write $\langle\!| P \ ; \ Q |\!\rangle^n$, which means $(\mathsf{this})(\langle\!| P \ ; \ Q |\!\rangle_{\mathsf{this}}^n)$. A further consequence of this insight is that two different transactions always bear different names marking them. Even if we conform with this intuition in every example, we purposely do not enforce in $\mathtt{web}\pi$ a discipline for the use of names marking transactions.

2.1 The reduction relation

Following the tradition of π-calculus [15], the reduction relation of $\mathtt{web}\pi$ is defined by using a structural congruence that equates all agents one never wants to distinguish.

Definition 1. *The* structural congruence $\equiv$ *is the least congruence closed with respect to α-renaming, satisfying the abelian monoid laws for parallel (associativity, commutativity, and $\mathbf{0}$ as identity), and the following axioms:*

1. *the scope laws:*

$$\begin{array}{rl} (u)\mathbf{0} \equiv \mathbf{0}, \qquad (u)(v)P \equiv (v)(u)P, & \\ P \,|\, (u)Q \equiv (u)(P \,|\, Q)\ , & \textit{if } u \notin \mathtt{fn}(P) \\ \langle\!| (z)P \ ;\ Q |\!\rangle_x^n \equiv (z)\langle\!| P \ ;\ Q |\!\rangle_x^n\ , & \textit{if } z \notin \{x\} \cup \mathtt{fn}(Q) \\ \langle\!| P \ ;\ (z)Q |\!\rangle_x^0 \equiv (z)\langle\!| P \ ;\ Q |\!\rangle_x^0\ , & \textit{if } z \notin \{x\} \cup \mathtt{fn}(P) \end{array}$$

2. *the repetition law:*

$$!x(\widetilde{u}).P \equiv x(\widetilde{u}).P \,|\, !x(\widetilde{u}).P$$

3. *the transaction laws:*

$$\begin{array}{c} \langle\!| \mathbf{0} \ ;\ Q |\!\rangle_x^n \equiv \mathbf{0} \\ \langle\!| \langle\!| P \ ;\ Q |\!\rangle_y^n \,|\, R \ ;\ R' |\!\rangle_x^m \equiv \langle\!| P \ ;\ Q |\!\rangle_y^n \,|\, \langle\!| R \ ;\ R' |\!\rangle_x^m \end{array}$$

4. *the floating laws:*

$$\begin{array}{c} \langle\!| \overline{z}\,\widetilde{u} \,|\, P \ ;\ Q |\!\rangle_x^n \equiv \overline{z}\,\widetilde{u} \,|\, \langle\!| P \ ;\ Q |\!\rangle_x^n \\ \langle\!| y(\widetilde{v}).P \,|\, P' \ ;\ \overline{z}\,\widetilde{u} \,|\, Q |\!\rangle_x^0 \equiv \overline{z}\,\widetilde{u} \,|\, \langle\!| y(\widetilde{v}).P \,|\, P' \ ;\ Q |\!\rangle_x^0 \end{array}$$

The scope laws and the repetition law are standard; let us discuss the transaction and floating laws that are unusual. The law $\langle\!| \mathbf{0} \ ;\ Q |\!\rangle_x^n \equiv \mathbf{0}$ defines committed transactions, namely transactions with $\mathbf{0}$ as body. These transactions, being committed, are equivalent to $\mathbf{0}$ and, therefore, cannot fail anymore. The law $\langle\!| \langle\!| P \ ;\ Q |\!\rangle_y^n \,|\, R \ ;\ R' |\!\rangle_x^m \equiv \langle\!| P \ ;\ Q |\!\rangle_y^n \,|\, \langle\!| R \ ;\ R' |\!\rangle_x^m$ moves transactions outside parent transactions, thus flattening the nesting of transactions. Notwithstanding this flattening, parent transactions may still affect children transactions by means of transaction names. The law $\langle\!| \overline{z}\,\widetilde{u} \,|\, P \ ;\ R |\!\rangle_x^n \equiv \overline{z}\,\widetilde{u} \,|\, \langle\!| P \ ;\ R |\!\rangle_x^n$ floats messages outside transactions, thus modelling the fact that messages are particles that independently move towards their inputs. The intended semantics is the following. If a process emits a message, this message traverses the surrounding transaction boundaries, until it reaches the corresponding input. The law $\langle\!| y(\widetilde{v}).P \,|\, P' \ ;\ \overline{z}\,\widetilde{u} \,|\, Q |\!\rangle_x^0 \equiv \overline{z}\,\widetilde{u} \,|\, \langle\!| y(\widetilde{v}).P \,|\, P' \ ;\ Q |\!\rangle_x^0$ models floatings of messages from compensations of out-of-time transactions whose bodies contain an input guarded process (failed transactions, see below).

The dynamic behaviour of processes is defined by the reduction relation. The main technical difficulty of this notion is time elapsing. In web models, time of different machines does not progress synchronously. Therefore we assume that each machine of the network has its own clock that is not synchronized with other clocks. On the contrary, all the processes running in the same location, compete for the same processor time. This competition is modelled in webπ by assuming that every reduction costs one time slot. Henceforth, when a subprocess performs a reduction, the flow of time is communicated to all the competing processes. This "flow of time" communication is a formal expedient for describing the elapse of one time slot without defining any machine clock. Should we have used a machine clock, as it happens in practice for running processes, then the time stamps of transactions could have been replaced with an absolute clock time that is compared with the machine clock when the transaction thread is executed.

The operation of decreasing by 1 the time stamps of active transactions on the same machine is modelled by the *time stepper function* below, that adapts the corresponding function in [4] to webπ. The definitions of this function and another auxiliary function are in order:

input predicate $\mathtt{inp}(P)$**:** this predicate verifies whether a process contains an input that is not underneath a transaction. It is the least relations such that:

$$
\begin{array}{ll}
\mathtt{inp}(x(\widetilde{u}).P) & \\
\mathtt{inp}((x)P) & \text{if } \mathtt{inp}(P) \\
\mathtt{inp}(P \,|\, Q) & \text{if } \mathtt{inp}(P) \text{ or } \mathtt{inp}(Q) \\
\mathtt{inp}(!x(\widetilde{u}).P) &
\end{array}
$$

time stepper function $\phi(P)$**:** this function decreases the time stamps by 1. For the missing cases, $\phi(P) = P$.

$$
\begin{array}{rcl}
\phi((x)P) & = & (x)\phi(P) \\
\phi(P \,|\, Q) & = & \phi(P) \,|\, \phi(Q) \\
\phi(\langle\!| P \; ; \; R |\!\rangle_x^0) & = & \begin{cases} \langle\!| \phi(P) \; ; \; \phi(R) |\!\rangle_x^0 & \text{if } \mathtt{inp}(P) \\ \langle\!| \phi(P) \; ; \; R |\!\rangle_x^0 & \text{otherwise} \end{cases} \\
\phi(\langle\!| P \; ; \; R |\!\rangle_x^{n+1}) & = & \langle\!| \phi(P) \; ; \; R |\!\rangle_x^n
\end{array}
$$

The stepper function is defined by induction on the syntax. The critical processes are the out-of-time transactions $\langle\!| P \; ; \; R |\!\rangle_x^0$. In this case, the input predicate is used to verify whether (a) the body P contains input-guarded processes or (b) not. In (a) the compensation is active, and the time must elapse for the transactions therein (and for transactions inside the body). In (b), since $\mathtt{inp}(P)$ is false, the time only elapses for the transactions inside the body. In fact, this definition is sound provided the time stepper function does not modify the input predicate and preserves structural congruence. For example, if $\mathtt{inp}(P)$ is false then $\langle\!| P \; ; \; R |\!\rangle_x^0 \equiv P$, and since $\phi(\langle\!| P \; ; \; R |\!\rangle_x^0) = \langle\!| \phi(P) \; ; \; R |\!\rangle_x^0$, we must verify that $\langle\!| \phi(P) \; ; \; R |\!\rangle_x^0$ is structurally congruent to $\phi(P)$. This is actually the case, as a consequence of the following proposition.

Proposition 1. *1.* $\mathtt{inp}(P)$ *if and only if* $\mathtt{inp}(\phi(P))$.
2. $P \equiv Q$ *implies* $\mathtt{inp}(P) = \mathtt{inp}(Q)$ *and* $\phi(P) \equiv \phi(Q)$.

The input predicate permits the formal definitions of failed and commited transactions.

Definition 2. *A transaction* $\langle\!| P \;;\; Q |\!\rangle_x^0$ *is* failed *if* $\mathtt{inp}(P)$ *is true; it is* committed *if* $\mathtt{inp}(P)$ *is false.*

We observe that a failed transaction $\langle\!| P \;;\; Q |\!\rangle_x^0$ may be always rewritten into a structurally congruent process $(\widetilde{z})\langle\!| y(\widetilde{u}).P' \,|\, P'' \;;\; Q |\!\rangle_x^0$, for some $\widetilde{z}$, y, $\widetilde{u}$, P', and P''. This "canonical form" has been used in the second floating law and is used in the definition of the following reduction relation.

Definition 3. *The* reduction relation $\rightarrow$ *is the least relation satisfying the reductions:*

$$\text{(COM)}\qquad \overline{x}\,\widetilde{v} \,|\, x(\widetilde{u}).P \;\;\rightarrow\;\; P\{\widetilde{v}/\widetilde{u}\}$$

$$\text{(FAIL)}\qquad \overline{x} \,|\, \langle\!| z(\widetilde{u}).P \,|\, Q \;;\; R |\!\rangle_x^{n+1} \;\;\rightarrow\;\; \langle\!| z(\widetilde{u}).P \,|\, \phi(Q) \;;\; R |\!\rangle_x^0$$

and closed under $\equiv$, (x)-, *and the rules:*

$$\frac{P \rightarrow Q}{P \,|\, R \rightarrow Q \,|\, \phi(R)} \qquad \frac{P \rightarrow Q}{\langle\!| P \;;\; R |\!\rangle_x^{n+1} \rightarrow \langle\!| Q \;;\; R |\!\rangle_x^{n}} \qquad \frac{P \rightarrow Q}{\begin{array}{l}\langle\!| y(\widetilde{v}).R \,|\, R' \;;\; P |\!\rangle_x^0 \\ \quad \rightarrow \langle\!| y(\widetilde{v}).R \,|\, \phi(R') \;;\; Q |\!\rangle_x^0\end{array}}$$

Rule (COM) is standard in process calculi and models the input-output interaction. Rule (FAIL) models transaction failures: when a transaction abort (a message on a transaction name) is emitted, the corresponding transaction is terminated by turning the time stamp to 0, thus activating the compensation (see the last inference rule). On the contrary, aborts are not possible if the transaction is already terminated, namely every input-guarded process in the body has completed its own work (this is never the case if the body contains replicated inputs). The inference rules lift reductions to parallel and transaction contexts, updating them because a time slot is elapsed.

In order to clarify the semantics, the reductions of few sample processes are reported. The process

$$\overline{z} \,|\, \overline{x} \,|\, \langle\!| x().\mathbf{0} \;;\; \overline{y} \,|\!\rangle_z^n$$

has the following two computations ($n > 0$):

$$\begin{array}{lll}
\overline{z} \,|\, \overline{x} \,|\, \langle\!| x().\mathbf{0} \;;\; \overline{y} \,|\!\rangle_z^n & \equiv \; \overline{x} \,|\, \overline{z} \,|\, \langle\!| x().\mathbf{0} \;;\; \overline{y} \,|\!\rangle_z^n & \\
& \rightarrow \; \overline{x} \,|\, \langle\!| x().\mathbf{0} \;;\; \overline{y} \,|\!\rangle_z^0 & \text{by (FAIL) and parallel closure} \\
& \equiv \; \overline{x} \,|\, \overline{y} \,|\, \langle\!| x().\mathbf{0} \;;\; \mathbf{0} |\!\rangle_z^0 & \\[1ex]
\overline{z} \,|\, \overline{x} \,|\, \langle\!| x().\mathbf{0} \;;\; \overline{y} \,|\!\rangle_z^n & \rightarrow \; \overline{z} \,|\, \langle\!| \mathbf{0} \;;\; \overline{y} \,|\!\rangle_z^{n-1} & \text{by (COM) and parallel closure} \\
& \equiv \; \overline{z} &
\end{array}$$

In the first computation, the message $\overline{x}$ is not consumed because the body of the transaction is cancelled on transaction failure. In the second one, the message $\overline{y}$ cannot be produced because the compensation process is garbage collected on transaction commit.

Consider now the process $P = (z,z')(\overline{x} \mid \langle\!| x().\mathbf{0} \ ; \ \overline{y} \, |\!\rangle_z^1 \mid \langle\!| x().\mathbf{0} \ ; \ \overline{y} \, |\!\rangle_{z'}^1)$. It evolves as follows

$$\begin{aligned} P \ &\equiv (z,z') \ (\langle\!| \overline{x} \mid x().\mathbf{0} \ ; \ \overline{y} \, |\!\rangle_z^1 \mid \langle\!| x().\mathbf{0} \ ; \ \overline{y} \, |\!\rangle_{z'}^1) \\ &\to (z,z')\langle\!| x().\mathbf{0} \ ; \ \overline{y} \, |\!\rangle_{z'}^0 \qquad \text{by (COMM), restriction and transaction closure} \end{aligned}$$

and in a similar way, but with z instead of z'. We remark that the process $Q = \overline{x} \mid x().\overline{y}$ has a similar behaviour. However the processes $\phi(P)$ and $\phi(Q)$ have different behaviours. In particular $\phi(P) \equiv \overline{x} \mid \overline{y} \mid \overline{y}$, while $\phi(Q) = Q$.

In webπ it is easy to delay a process P of n steps. To this aim, let $x \notin \mathtt{fn}(P)$ then

$$(x)\langle\!| x().\mathbf{0} \ ; \ P |\!\rangle_x^n$$

behaves like $\mathbf{0}$ for n time units, and evolves to P afterwards.

It is worth to notice that the reduction relation of processes does not define the dynamics of temporarily blocked transactions as the one above. Indeed, by definition $(x)\langle\!| x().\mathbf{0} \ ; \ P |\!\rangle_x^n \not\to$ if $n > 0$. This sloppiness is due to the fact that the process reduction is defined in a compositional way and therefore cannot express the absence of a reduction, which is a global property of the processor running the process. One solution to this problem is to introduce a rule like $P \to \phi(P)$ in [4]. However this solution is at odd with local urgency: it states that a machine (processor) may idle, even if there are some actions that can be performed. We prefer to keep the present reduction (intensional) semantics and to stick to an extensional semantics that is a congruence, thus defining the meaning of a process when it is plugged in any possible context.

3 Timed bisimilarity

The extensional semantics of webπ – the *timed bisimilarity* – relies on the notions of barb and contexts. A process P has a *barb* x, and write $P \downarrow x$, if P manifests an output on the free name x. Formally:

$$\begin{array}{rl} \overline{x}\,\widetilde{u} \downarrow x & \\ (z)P \downarrow x & \text{if } P \downarrow x \text{ and } x \neq z \\ (P \mid Q) \downarrow x & \text{if } P \downarrow x \text{ or } Q \downarrow x \\ \langle\!| P \ ; \ R |\!\rangle_z^0 \downarrow x & \text{if } P \downarrow x \text{ or } (\mathtt{inp}(P) \text{ and } R \downarrow x) \\ \langle\!| P \ ; \ R |\!\rangle_z^{n+1} \downarrow x & \text{if } P \downarrow x \end{array}$$

Therefore inputs (both simple and replicated) have no barb. This is standard in asynchronous calculi: an observer has no direct way of knowing if the message he has sent has been received.

Context processes, noted $\mathsf{C}[\cdot]$, are defined by the following grammar:

$$\begin{array}{rl} \mathsf{C}[\cdot] \ ::= & [\cdot] \quad | \quad x(\widetilde{u}).\mathsf{C}[\cdot] \quad | \quad (x)\mathsf{C}[\cdot] \quad | \quad \mathsf{C}[\cdot]|P \quad | \quad !x(\widetilde{u}).\mathsf{C}[\cdot] \\ | & \langle\!| \mathsf{C}[\cdot] \ ; \ P |\!\rangle_x^n \quad | \quad \langle\!| P \ ; \ \mathsf{C}[\cdot] |\!\rangle_x^n \end{array}$$

Definition 4. *A* timed barbed bisimulation $\mathcal{S}$ *is a symmetric relation between processes such that* $P \, \mathcal{S} \, Q$ *implies*

1. *if $P \downarrow x$ then $Q \downarrow x$;*
2. *if $P \rightarrow P'$ then $Q \rightarrow Q'$ and $P' \,\mathcal{S}\, Q'$;*

Timed bisimilarity*, denoted with $\sim_t$, is the largest timed barbed bisimulation that is also a congruence.*

As an illustration of timed bisimilar processes we discuss few examples. The following identity adapts an equation of asynchronous bisimilarity [1] to $\mathtt{web}\pi$, thus suggesting that timed bisimilarity is asynchronous:

$$\langle\!| x(u).\overline{x}\, u \mid \tau.\mathbf{0} \;;\; P |\!\rangle_z^1 \sim_t \langle\!| \tau.(v)v().\mathbf{0} \;;\; P |\!\rangle_z^1$$

It is worth to notice that $\mathbf{0} \not\sim_t x(u).\overline{x}\, u$. For instance the context $\mathsf{C}[\cdot] = (z)([\cdot] \mid \overline{x}\, w \mid \langle\!| x(u).\mathbf{0} \;;\; \overline{v}\, |\!\rangle_z^1)$ separates the two processes. Due to local urgency, the transaction z cannot fail in $\mathsf{C}[\mathbf{0}]$ (thus the message $\overline{v}$ cannot be produced), while it can fail in $\mathsf{C}[x(\widetilde{u}).\overline{x}\,\widetilde{u}]$ (thus activating the compensation $\overline{v}$).

Timed bisimilarity may be inferred by considering only a subset of contexts and applying substitutions.

Lemma 1. *(Context Lemma) Let* timed-prime bisimilarity*, in notation $\sim_t'$, be the largest timed barbed bisimulation such that if $P \sim_t' Q$ then, for every R, x, n, S, $\widetilde{w}$, $\widetilde{z}$: $\langle\!| P\{\widetilde{w}/\widetilde{z}\} \;;\; R |\!\rangle_x^n \mid S \;\sim_t'\; \langle\!| Q\{\widetilde{w}/\widetilde{z}\} \;;\; R |\!\rangle_x^n \mid S$. Then $\sim_t = \sim_t'$.*

It is worth to notice that the corresponding lemma about π-calculus reduces contexts to those whose shape is $[\cdot]\{\widetilde{u}/\widetilde{v}\} \mid R$.

We conclude this section by demonstrating that the discriminating power of $\sim_t$ is weaker when local urgency is dropped. To this aim, we consider a new reduction relation of processes denoted with $\rightarrow^\phi$ defined by augmenting Definition 3 (where $\rightarrow^\phi$ is substituted for $\rightarrow$) with the idle rule:

$$\begin{array}{l}\text{(IDLE)}\\ P \rightarrow^\phi \phi(P)\end{array}$$

The (IDLE) rule allows time to pass asynchronously even when other reductions are possible. Let $\sim_t^{\text{idle}}$ be defined as $\sim_t$ considering the reduction relation $\rightarrow^\phi$ instead of $\rightarrow$. Then $(x)x().\mathbf{0} \sim_t^{\text{idle}} (x)x().\mathbf{0} \mid z().\overline{z}$ while $(x)x().\mathbf{0} \not\sim_t (x)x().\mathbf{0} \mid z().\overline{z}$. However a model of time similar to (IDLE) can be simulated with the local urgency assumption. It sufficies to put in the context a process always able to perform internal synchronizations; thus letting the time to pass.

Proposition 2. *$P \sim_t Q$ implies $P \sim_t^{\text{idle}} Q$.*

Proof. (Sketch) Let τ^* be the process $(x)(\overline{x} \mid !x().\overline{x})$. An easy check gives that, for every P, $P \rightarrow^\phi Q$ if and only if $\tau^* \mid P \rightarrow \tau^* \mid Q$. The proposition follows directly by this property. $\square$

4 The labelled semantics

Even if the context lemma restricts the shape of contexts for inferring timed bisimilarity, direct proofs remain particularly difficult. A standard device to avoid such quantification consists of introducing a labelled operational model and equipping it with an (asynchronous) bisimulation.

Let μ range over input labels $\overset{\diamond}{x}(\widetilde{u})$ and $\overset{\circ}{x}(\widetilde{u})$, bound output labels $(\widetilde{z})\overline{x}\,\widetilde{u}$ where $\widetilde{z} \subseteq \widetilde{u}$, and $\overset{\diamond}{\tau}$ and $\overset{\circ}{\tau}$. Let $\star$ range over $\{\diamond, \circ\}$; we define $\overset{\circ}{\overset{\star}{x}}(\widetilde{u}) = \overset{\circ}{x}(\widetilde{u})$, $\overset{\circ}{(\widetilde{z})\overline{x}\,\widetilde{u}} = (\widetilde{z})\overline{x}\,\widetilde{u}$, and $\overset{\circ}{\overset{\star}{\tau}} = \overset{\circ}{\tau}$. Let also $\mathtt{fn}(\overset{\star}{\tau}) = \emptyset$, $\mathtt{fn}(\overset{\star}{x}(\widetilde{u})) = \{x\}$, $\mathtt{fn}(\overline{x}\,\widetilde{u}) = \{x\} \cup \widetilde{u}$, and $\mathtt{fn}((\widetilde{z})\overline{x}\,\widetilde{u}) = \{x\} \cup \widetilde{u} \setminus \widetilde{z}$. Finally, let $\mathtt{bn}(\mu)$ be $\widetilde{z}$ if $\mu = (\widetilde{z})\overline{x}\,\widetilde{u}$, be $\widetilde{u}$ if $\mu = \overset{\star}{x}(\widetilde{u})$, and be $\emptyset$, otherwise. We implicitly identify terms up to α-renaming $\equiv_\alpha$: that is, if $P \equiv_\alpha Q$ and $Q \xrightarrow{\mu} P'$ then $P \xrightarrow{\mu} P'$.

Definition 5. *The* transition relation *of* $\mathtt{web}\pi$ *processes, noted* $\xrightarrow{\mu}$, *is the least relation satisfying the rules:*

(IN)
$$x(\widetilde{u}).P \xrightarrow{\overset{\diamond}{x}(\widetilde{u})} P$$

(OUT)
$$\overline{x}\,\widetilde{u} \xrightarrow{\overline{x}\,\widetilde{u}} \mathbf{0}$$

(RES)
$$\frac{P \xrightarrow{\mu} Q \quad x \notin \mathtt{fn}(\mu)}{(x)P \xrightarrow{\mu} (x)Q}$$

(OPEN)
$$\frac{P \xrightarrow{(\widetilde{v})\overline{x}\,\widetilde{u}} Q \quad w \neq x \quad w \in \widetilde{u} \setminus \widetilde{v}}{(w)P \xrightarrow{(w\widetilde{v})\overline{x}\,\widetilde{u}} Q}$$

(PAR)
$$\frac{P \xrightarrow{\mu} Q \quad \mathtt{bn}(\mu) \cap \mathtt{fn}(R) = \emptyset}{P \mid R \xrightarrow{\mu} Q \mid \phi(R)}$$

(COM)
$$\frac{P \xrightarrow{(\widetilde{w})\overline{x}\,\widetilde{v}} P' \quad Q \xrightarrow{\overset{\star}{x}(\widetilde{u})} Q' \quad \widetilde{w} \cap \mathtt{fn}(Q) = \emptyset}{P \mid Q \xrightarrow{\overset{\star}{\tau}} (\widetilde{w})(P' \mid Q'\{\widetilde{v}/\widetilde{u}\})}$$

(REPIN)
$$!x(\widetilde{u}).P \xrightarrow{\overset{\diamond}{x}(\widetilde{u})} P \mid !x(\widetilde{u}).P$$

(ABORT)
$$\llparenthesis P \,;\, R \rrparenthesis_x^{n+1} \xrightarrow{\overset{\circ}{x}()} \llparenthesis P \,;\, R \rrparenthesis_x^{0}$$

(SELF)
$$\frac{P \xrightarrow{\overline{x}} Q}{\llparenthesis P \,;\, R \rrparenthesis_x^{n+1} \xrightarrow{\overset{\circ}{\tau}} \llparenthesis Q \,;\, R \rrparenthesis_x^{0}}$$

(TRANS)
$$\frac{P \xrightarrow{\mu} Q \qquad \mathtt{bn}(\mu) \cap (\mathtt{fn}(R) \cup \{x\}) = \emptyset}{\llparenthesis P \,;\, R \rrparenthesis_x^{n+1} \xrightarrow{\overset{\circ}{\mu}} \llparenthesis Q \,;\, R \rrparenthesis_x^{n}}$$

(TRANS-B)
$$\frac{P \xrightarrow{\overset{\circ}{\mu}} Q \qquad \mathtt{bn}(\overset{\circ}{\mu}) \cap (\mathtt{fn}(R) \cup \{x\}) = \emptyset \quad \mathtt{inp}(P)}{\llparenthesis P \,;\, R \rrparenthesis_x^{0} \xrightarrow{\overset{\circ}{\mu}} \llparenthesis Q \,;\, \phi(R) \rrparenthesis_x^{0}}$$

(TRANS-C)
$$\frac{P \xrightarrow{\overset{\circ}{\mu}} Q \qquad \mathtt{bn}(\overset{\circ}{\mu}) \cap (\mathtt{fn}(R) \cup \{x\}) = \emptyset \quad \neg\mathtt{inp}(P)}{\llparenthesis P \,;\, R \rrparenthesis_x^{0} \xrightarrow{\overset{\circ}{\mu}} \llparenthesis Q \,;\, R \rrparenthesis_x^{0}}$$

(TRANS-F)
$$\frac{R \xrightarrow{\mu} R' \quad \mathtt{bn}(\mu) \cap (\mathtt{fn}(P) \cup \{x\}) = \emptyset \quad \mathtt{inp}(P)}{\llparenthesis P \,;\, R \rrparenthesis_x^{0} \xrightarrow{\overset{\circ}{\mu}} \llparenthesis \phi(P) \,;\, R' \rrparenthesis_x^{0}}$$

The transitions of $P \mid Q$ have mirror cases that have been omitted.

The first seven rules are almost standard in π-calculus. Exceptions are (replicated) inputs whose transitions are labelled with $\overset{\diamond}{x}(\widetilde{u})$, and rule (PAR) that uses the time stepper function. The symbol $\diamond$ is used to mark input transitions that are not underneath a transaction. These transitions must be blocked if they are due to bodies of failed transactions. Transitions that are underneath transactions are marked with a $\circ$ symbol: see rule (TRANS). These transitions are never blocked: see rules (TRANS-B) and (TRANS-C). We discuss the other rules. Rule (ABORT) models transaction termination due to an abort message. It amounts to turning the time stamp to 0. We remark that abort is not possible if the time stamp is already 0. The label is marked with $\circ$ because the transition is assumed to be underneath a transaction. Rule (SELF) is similar to (ABORT), taking into account the case when the abort message is raised by the body of the transaction. Rule (TRANS) lifts transitions to transaction contexts and decreases the transaction time stamp because a transition of the body is going to occur. This rule applies also to outputs transitions, thus looking at odd with the reduction relation, where messages are moved outside transaction bodies by means of a structural rule. Actually this is only apparent: in the reduction relation, the decreasing of the time stamp is performed by the contextual rules for parallel composition (by ϕ) or for transactions. Rules (TRANS-B) and (TRANS-C) lift transitions of bodies of transactions to out-of-time transaction contexts. According to this rule output transitions are always enabled because $\overset{\circ}{(\widetilde{z})\overline{x}\,\widetilde{u}}= (\widetilde{z})\overline{x}\,\widetilde{u}$. On the contrary, input and τ transitions are enabled provided they are underneath not failed transaction contexts. The two rules separate the cases whether the compensation is active or not. Rule (TRANS-F) lifts transitions of compensations to failed transaction contexts. We observe that the transition in the conclusion is labelled with a $\circ$. This means that the transition cannot be blocked by an external failed transaction boundary.

The following statement guarantees that transitions in the bodies of failed transactions preserve the input predicate. If this was not the case, a committed transaction could become failed, thus enabling transitions of the compensation.

Proposition 3. *If* $P \xrightarrow{\overset{\circ}{\mu}} Q$ *and* $\mathtt{inp}(P)$ *then* $\mathtt{inp}(Q)$.

We are now in place for formalizing a correspondence result between the labelled and the reduction semantics.

Proposition 4. *Let* P *be a* $\mathtt{web}\pi$ *process. Then*

1. $P \downarrow v$ *if and only if* $P \xrightarrow{(\widetilde{z})\overline{v}\,\widetilde{u}}$, *for some* $\widetilde{z}$ *and* $\widetilde{u}$;
2. $P \xrightarrow{\overset{\star}{\tau}} Q$ *implies* $P \to Q$;
3. $P \to Q$ *implies there is* R *such that* $R \equiv Q$ *and* $P \xrightarrow{\overset{\star}{\tau}} R$.

The labelled bisimulation that we consider recalls the asynchronous bisimulation [1] for processes. In the following definition $\star, \bullet$ range over $\{\diamond, \circ\}$

Definition 6. *An* asynchronous bisimulation *is a symmetric binary relation $\mathcal{S}$ between processes such that $P\mathcal{S}Q$ implies*

1. *if $P \xrightarrow{\overset{\star}{\tau}} P'$ then $Q \xrightarrow{\overset{\bullet}{\tau}} Q'$ and $P'\mathcal{S}Q'$,*
2. *if $P \xrightarrow{(\widetilde{v})\overline{x}\,\widetilde{u}} P'$ and $\widetilde{v} \cap \mathtt{fn}(Q) = \emptyset$, then $Q \xrightarrow{(\widetilde{v})\overline{x}\,\widetilde{u}} Q'$ and $P'\mathcal{S}Q'$;*
3. *if $P \xrightarrow{\overset{\star}{x}(\widetilde{u})} P'$ and $\widetilde{u} \cap \mathtt{fn}(Q) = \emptyset$, then*
   (a) either $Q \xrightarrow{\overset{\bullet}{x}(\widetilde{u})} Q'$ and $P'\mathcal{S}Q'$,
   (b) or $Q \xrightarrow{\overset{\bullet}{\tau}} Q'$ and $P'\mathcal{S}(Q' \,|\, \overline{x}\,\widetilde{u})$.

Asynchronous bisimilarity*, in notation $\sim_a$, is the largest asynchronous bisimulation.*

The item 3 of the definition of asynchronous bisimulation allows to match an input transition with a τ transition. This item permits to equate the following processes, that have been already discussed in the previous Section:

$$\langle\!| x(u).\overline{x}\,u \,|\, \tau.\mathbf{0} \;;\; P |\!\rangle^{1}_{z} \sim_a \langle\!| \tau.(v)v().\mathbf{0} \;;\; P |\!\rangle^{1}_{z}$$

Remark 2. Our approach is different from [3]. Berger uses a standard bisimulation definition on a transition system extended with the Honda-Tokoro rule $\mathbf{0} \xrightarrow{x(\widetilde{u})} \overline{x}\,\widetilde{u}$ [11]. On the contrary, we stick to the approach in [1], where a slightly modified bisimulation (with the item *3.(b)*) is applied to a standard transition system.

Asynchronous bisimulation equates structurally congruent processes:

Proposition 5. *$P \equiv Q$ implies $P \sim_a Q$.*

In contrast with asynchronous π-calculus, $\sim_a$ is not a congruence for $\mathtt{web}\pi$ because it is not closed with respect to input, parallel composition, and transaction contexts. This may be remedied by appropriately closing the equivalence. With respect to [3], where closures regarded substitutions and time, we also need to close by the input predicate.

Definition 7. *A binary relation $\mathcal{R}$ over processes is*

- substitution-closed *if $P\,\mathcal{R}\,Q$ implies, for every substitution σ, $P\sigma\,\mathcal{R}\,Q\sigma$;*
- time-closed *if $P\,\mathcal{R}\,Q$ implies $\phi(P)\,\mathcal{R}\,\phi(Q)$;*
- input-predicate-closed *if $P\,\mathcal{R}\,Q$ implies $\mathtt{inp}(P) = \mathtt{inp}(Q)$.*

These are counterexamples showing that the asynchronous bisimulation $\sim_a$ is neither substitution-closed, nor time-closed, nor input-predicate-closed.

1. As regards substitution closure, we adapt a counterexample in [3]. Let

$$\begin{aligned} P &\stackrel{def}{=} (a)(\overline{x}\,a \,|\, !y(u).\overline{u}\,) \\ Q &\stackrel{def}{=} (a)(\overline{x}\,a \,|\, !y(u).\overline{u} \,|\, (z)\langle\!| y(u).(\overline{u} \,|\, a().\overline{b}\,) \;;\; \mathbf{0} |\!\rangle^{2}_{z}) \end{aligned}$$

We have that $P \sim_a Q$ but $P\{^y/_x\} \not\sim_a Q\{^y/_x\}$ because $Q\{^y/_x\}$ may produce the message $\overline{b}$ while this is not the case for $P\{^y/_x\}$. The main difference between this counterexample and the one reported in [3] is that we do not exploit nesting of transactions. The equivalence result between P and Q relies on the fact that, in general, $!y(u).\overline{u} \mid (z)\langle\!| y(u).(\overline{u} \mid x(\widetilde{v}).P) \ ; \ \mathbf{0} |\!\rangle^1_z) \sim_a !y(u).\overline{u}$ and $(a)(\overline{x}\, a) \sim_a (a)(\overline{x}\, a \mid (z)\langle\!| a().\overline{b} \ ; \ \mathbf{0} |\!\rangle^1_z)$.

2. As regards time closure, we adapt another counterexample in [3]. Let

$$P \stackrel{def}{=} (z)\langle\!| \tau.\overline{x} \ ; \ \mathbf{0} |\!\rangle^1_z$$
$$Q \stackrel{def}{=} (z)\langle\!| \tau.\tau.\mathbf{0} \ ; \ \overline{x} \,|\!\rangle^1_z$$

then $P \sim_a Q$ but $\phi(P) \not\sim_a \phi(Q)$ because $\phi(Q) \xrightarrow{\overline{x}}$ and $\phi(P)$ cannot.

3. As regards input-predicate closure, let

$$P \stackrel{def}{=} \mathbf{0}$$
$$Q \stackrel{def}{=} (z)z()$$

then $P \sim_a Q$ and $\mathtt{inp}(P) \neq \mathtt{inp}(Q)$. Since $\mathtt{inp}(P)$ is different from $\mathtt{inp}(Q)$, it is possible to separate P and Q by using contexts such as $\langle\!| [\cdot] \ ; \ \overline{y} \,|\!\rangle^0_x$.

Definition 8. Labelled timed bisimilarity*, in notation $\simeq_a$, is the greatest asynchronous bisimulation contained into $\sim_a$ that is also substitution-closed, time-closed, and input-predicate closed.*

Lemma 2. *$\simeq_a$ is a congruence.*

We are now in place to report the correspondence result between the labelled timed bisimilarity and the timed bisimulation congruence.

Proposition 6. *$P \simeq_a Q$ implies $P \sim_t Q$.*

Proof. By Proposition 4, $\simeq_a$ is a timed barbed bisimulation, and by Lemma 2 it is also a congruence. The statement follows because $\sim_t$ is the largest one. □

The converse implication of Proposition 6 also holds in the asynchronous π-calculus (with strong semantics) [10]. The technique shows that if $P \sim_t Q$ then the bisimulation game between P and Q of $\simeq_a$ holds (the closures of the definition of $\simeq_a$ hold easily). This is obtained by means of small contexts checking that bound outputs of P and Q are the same up-to alpha-equivalence. These contexts disappear after few steps (namely, if $P \xrightarrow{\mu} P'$ then $\mathsf{C}[P] \xrightarrow{\tau} \cdots \xrightarrow{\tau} P'$). Unfortunately, this technique applies badly to $\mathtt{web}\pi$ because such "checking steps" make the time elapse in P and Q. Namely, if $P \xrightarrow{\mu} P'$ then $\mathsf{C}[P] \xrightarrow{\tau} \cdots \xrightarrow{\tau} \phi^n(P')$, for some n (rather than $n = 0$). Since we are missing a direct proof (even if we conjecture the equality $\simeq_a = \sim_t$), we use an alternative, weaker technique that has been proposed for the weak asynchronous bisimulation [1].

Let us extend the $\mathtt{web}\pi$ syntax with the rule:

$$P \ ::= \quad \cdots \quad | \quad [x = y]P$$

A match process $[x = y]P$ executes P provided x is equal to y. Let $[x_i = y_i]^{i \in I} P$ be the sequence of name matches $[x_i = y_i]$ followed by the process P. The semantics of name match is defined by the structural congruence rule

$$[x = x]P \equiv P \ .$$

Let also $\mathtt{inp}([x = x]P) = \mathtt{inp}(P)$. Finally, let $\sim_{t,M}$ be the largest timed barbed bisimulation that is a congruence with respect to contexts in $\mathtt{web}\pi$ extended with the name match (namely $\mathsf{C}[\cdot] ::= \cdots \quad | \quad [x = y]\mathsf{C}[\cdot]$). It is easy to demonstrate that $\simeq_a \subseteq \sim_{t,M} \subseteq \sim_t$ (the first containment is proved with arguments similar to Proposition 6).

Lemma 3. *If $P \sim_{t,M} Q$ then $P \simeq_a Q$.*

Proof. It is easy to verify that $\sim_{t,M}$ is substitution-closed, timed-closed, and input-predicate-closed. We demonstrate that, for any move $P \xrightarrow{\mu} P'$, there exist contexts $\mathsf{C}[\cdot]$ such that $\mathsf{C}[P] \sim_{t,M} \mathsf{C}[Q]$ implies $Q \xrightarrow{\mu'} Q'$ and one of the items 1 – 3 of Definition 6 is satisfied. We report only the two most significant cases. Let $\star, \bullet \in \{\circ, \diamond\}$.

$P \xrightarrow{\overset{\star}{x}(\widetilde{u})} P'$. We consider the context $\mathsf{C}[\cdot] = \overline{x}\,\widetilde{u} \,|\, [\cdot]$. Then $\mathsf{C}[P] \rightarrow P'$. As $P \sim_{t,M} Q$ then $\mathsf{C}[P] \sim_{t,M} \mathsf{C}[Q]$, and there is Q' such that $\mathsf{C}[Q] \rightarrow Q'$ and $P' \sim_{t,M} Q'$. There are two cases, either $Q \xrightarrow{\overset{\bullet}{x}(\widetilde{u})} Q'$ (thus the item 3.(a) of the Definition 6 is satisfied) or $Q \xrightarrow{\overset{\bullet}{\tau}} Q''$ and $P' \sim_{t,M} Q'' \,|\, \overline{x}\,\widetilde{u}$ (thus the item 3.(b) of the Definition 6 is satisfied).

$P \xrightarrow{(\widetilde{v})\overline{x}\,\widetilde{u}} P'$. Let $\widetilde{u} = u_1 \ldots u_n$. Let also $F = \{i \mid u_i \notin \widetilde{v}\}$, $B = \{i \mid u_i \in \widetilde{v}\}$, $E = \{(i,j) \mid i < j \text{ and } u_i = u_j \text{ and } u_i, u_j \in B\}$, $D = \{(i,j) \mid i < j \text{ and } u_i \neq u_j \text{ and } u_i, u_j \in B\}$. Consider the context

$$\begin{aligned} \mathsf{C}_\mu[\cdot] = x(z_1 \ldots z_n).(\ & \textstyle\prod_{i \in F}[z_i = u_i]a_i \mid \prod_{i \in B, u \in \mathtt{fn}(P) \cup \mathtt{fn}(Q)}[z_i = u]b_{i,u} \\ & \mid \textstyle\prod_{(i,j) \in E}[z_i = z_j]c_i \mid \prod_{(i,j) \in D}[z_i = z_j]d_i \ ) \mid [\cdot] \end{aligned}$$

where all the names $a_i, b_{i,u}, c_i, d_i$ are fresh and pairwise different. Then $\mathsf{C}_\mu[P] \rightarrow P'' \sim_{t,M} \prod_{i \in H} \overline{e_i} \,|\, P'$, where e_i are a subset of labels of $a_i, b_{i,u}, c_i, d_i$. As $P \sim_{t,M} Q$ are timed bisimilar, then also $\mathsf{C}_\mu[Q] \rightarrow Q''$ where $P'' \sim_{t,M} Q''$. By definition of $\mathsf{C}[\cdot]$, it must be the case that $Q'' \sim_{t,M} \prod_{i \in fi} \overline{e_i} \,|\, Q'$, for some Q' and $Q \xrightarrow{(\widetilde{v})\overline{x}\,\widetilde{u}} Q'$. An easy reasoning permits to state that, if $d \notin \mathtt{fn}(R) \cup \mathtt{fn}(R')$, then $\overline{d} \,|\, R \sim_{t,M} \overline{d} \,|\, R'$ if and only if $R \sim_{t,M} R'$. Applying this result we conclude that $P' \sim_{t,M} Q'$ (thus the item 2. of the Definition 6 is satisfied). □

5 Machines

In this section we study the syntax and the reduction relation of $\mathtt{web}\pi$ machines. The extensional semantics is omitted in this contribution: a thorough analysis of the extensional semantics for machines (and the induced equality on processes) will be addressed in the full paper.

The syntax of *machines* M is defined by the following rules.

$$\mathsf{M} ::= \quad \mathbf{0} \quad | \quad [\,P\,]_{\widetilde{x}} \quad | \quad (x)\mathsf{M} \quad | \quad \mathsf{M}\,|\,\mathsf{M}$$

A machine may be empty; a location $[\,P\,]_{\widetilde{x}}$ running the process P and accepting all messages on names in the set $\widetilde{x}$; a machine $(x)\mathsf{M}$ with local name x; or a network of locations. The symbols $\mathbf{0}$ and $|$ are overloaded because they also denote the empty and parallel processes, respectively; the actual meaning is made clear from the context. The index $\widetilde{x}$ in the location $[\,P\,]_{\widetilde{x}}$ indicates a set $\widetilde{x}$, even if it is denoted with the same notation of tuples.

We assume that a name may index at most one machine. Formally, let $\mathtt{ln}(\mathsf{M})$ be defined as $\mathtt{ln}(\mathbf{0}) = \emptyset$, $\mathtt{ln}([\,P\,]_{\widetilde{x}}) = \widetilde{x}$, $\mathtt{ln}((x)\mathsf{M}) = \mathtt{ln}(\mathsf{M}) \setminus \{x\}$, and $\mathtt{ln}(\mathsf{M}\,|\,\mathsf{N}) = \mathtt{ln}(\mathsf{M}) \cup \mathtt{ln}(\mathsf{N})$. Networks $\mathsf{M}\,|\,\mathsf{N}$ are constrained to satisfy the property $\mathtt{ln}(\mathsf{M}) \cap \mathtt{ln}(\mathsf{N}) = \emptyset$.

The *structural congruence* $\equiv$ is the least congruence closed with respect to α-renaming, satisfying the abelian monoid laws for parallel (associativity, commutativity and $\mathbf{0}$ as identity), and the following axioms:

1. the scope laws:

$$\begin{array}{c} (u)\mathbf{0} \equiv \mathbf{0}, \qquad (x)(z)\mathsf{M} \equiv (z)(x)\mathsf{M}, \\ \mathsf{M}\,|\,(x)\mathsf{N} \equiv (x)(\mathsf{M}\,|\,\mathsf{N})\,, \quad \textit{if } x \notin \mathtt{fn}(\mathsf{M}) \\ [\,(x)P\,]_{\widetilde{z}} \equiv (x)\,[\,P\,]_{\widetilde{z}x}\,, \quad \textit{if } x \notin \widetilde{z} \end{array}$$

2. the lifting law:

$$[\,P\,]_{\widetilde{x}} \equiv [\,Q\,]_{\widetilde{x}}\,, \qquad \textit{if } P \equiv Q$$

The first three scope laws are standard. The last one is used to extrude a name outside a machine; the effect is that the extruded name is added to the set of the names on which the machine is the receptor. The lifting law lifts to machines the structural congruence defined on processes.

The *reduction relation* for machines is the least relation closed under $\equiv$, (x)-, and parallel composition, and satisfying the reductions:

$$\text{(INTRA)}\ \frac{P \to Q}{[\,P\,]_{\widetilde{x}} \to [\,Q\,]_{\widetilde{x}}} \qquad \text{(TIME)}\ \frac{P \not\to}{[\,P\,]_{\widetilde{x}} \to [\,\phi(P)\,]_{\widetilde{x}}} \qquad \begin{array}{l} \text{(DELIV)} \\ [\,\overline{x}\,\widetilde{v}\,|\,P\,]_{\widetilde{z}}\,|\,[\,Q\,]_{\widetilde{y}x} \\ \qquad \to [\,P\,]_{\widetilde{z}}\,|\,[\,\overline{x}\,\widetilde{v}\,|\,Q\,]_{\widetilde{y}x} \end{array}$$

As a consequence of the closure under parallel composition, time progress asynchronously between machines. Namely, if $\mathsf{M} \to \mathsf{M}'$ then also $\mathsf{M}\,|\,\mathsf{N} \to \mathsf{M}'\,|\,\mathsf{N}$. In particular, the time of N does not elapse. Rule (INTRA) lifts the local reductions to the machine. Rule (TIME) reflects our approach for modeling the time. In particular, as local computations are urgent, this rule permits the elapsing of one time unit – the application of ϕ – only in the case when no internal computation is possible inside a machine. Rule (DELIV) delivers a message to the unique machine having x in the index. This rule does not consume time both in the sender and in the receiver machines. This does not mean that communication takes no time. Delays of deliveries follow from asynchrony between machines and nondeterminism of reductions due to (DELIV). Alternatively, one could extend the syntax of machines by adding messages in parallel with machines and replacing (DELIV) with two rules: one putting a message outside the sender machine, the other actually delivering the message to the receiver machine. The present solution has been preferred for simplicity.

It is worth to notice that, in the present model, a message may be either consumed in the same machine in which it has been produced (see rule (COM) in the reduction relation of processes) or delivered to another machine in the network (the unique responsible for accepting that message). This appears a bit counterintuitive: a machine that is not responsible to accept messages on a given name may actually consume messages that have been produced locally. In fact, in practice this scenario never occurs. If a machine defines a name x and exports it to other machines, then the machines receiving x may use it with output capability only. Since $\mathtt{web}\pi$ processes are unrestricted, the present reduction relation of machines results a conservative extension of the practical scenario.

6 Conclusions

We have studied $\mathtt{web}\pi$, a process calculus extending the asynchronous π-calculus with a timed transaction construct. The main theoretical contribution of this paper is the investigation of the extensional semantics of $\mathtt{web}\pi$, the timed bisimilarity, and of its labelled counterpart.

A number of issues have been overlooked. We retain that the following twos are particularly significant to judge the benefits of $\mathtt{web}\pi$. First of all, $\mathtt{web}\pi$ has been motivated by the need of assessing the proposals of web programming languages. It will be foundational if it is possible to translate these proposals in $\mathtt{web}\pi$, in particular the transactional protocols that are defined therein. The techniques developed in this paper will be necessary for comparing the translations. The next step is therefore the translation in $\mathtt{web}\pi$ of some emerging technology, such as BPEL.

The second issue has a theoretical flavour. The identity of $\sim_t$ and $\simeq_a$ has been only conjectured because we were not able to provide a direct proof. To measure the discriminating power of $\simeq_a$, we have introduced an operator that is able to perform several tests and emit a message in one step. This expedient appears useless when machines are used because it is possible to delegate a different location to perform the tests and emit the message (the time spent by a location for a computation has no effect on the time of other locations). While this remark does not help in solving our conjecture, it prompts the investigation of the extensional semantics of machines.

References

1. R. M. Amadio, I. Castellani, and S. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.
2. T. Andrews and et.al. Business Process Execution Language for Web Services. Version 1.1. Specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems, 2003.
3. M. Berger. Basic theory of reduction congruence for two timed asynchronous π-calculi. In *CONCUR '04: Proceedings of the 15th International Conference on Concurrency Theory*, volume 3170 of *LNCS*, pages 115–130. Springer-Verlag, 2004.
4. M. Berger and K. Honda. The two-phase commitment protocol in an extended pi-calculus. In *EXPRESS '00: Proceedings of the 7th International Workshop on Expressiveness in Concurrency*, volume 39.1 of *ENTCS*. Elsevier Science Publishers, 2000.
5. L. Bocchi, C. Laneve, and G. Zavattaro. A calculus for long running transactions. In *FMOODS'03, Proceedings of the 6th IFIP International Conference on Formal Methods for Open Object-based Distributed Systems*, volume 2884 of *LNCS*, pages 124–138. Springer-Verlag, 2003.
6. R. Bruni, C. Laneve, and U. Montanari. Orchestrating transactions in join calculus. In *CONCUR 2002: Proceedings of the 13th International Conference on Concurrency Theory*, volume 2421 of *LNCS*, pages 321–337. Springer Verlag, 2002.
7. M. Butler and C. Ferreira. An operational semantics for StAC, a language for modelling long-running business transactions. In *COORDINATION'04, Proceedings of the 6th International Conference on Coordination Models and Languages*, volume 2949 of *LNCS*, pages 87–104. Springer-Verlag, 2004.
8. M. Butler, T. Hoare, and C. Ferreira. A trace semantics for long-running transactions. In Proceedings of 25 Years of CSP, London, 2004.
9. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL 1.1). W3C Note, 2001.
10. C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *ICALP '98, Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, volume 1443 of *LNCS*, pages 844–855. Springer-Verlag, 1998.
11. K. Honda and M. Tokoro. On asynchronous communication semantics. In *Proceedings of Object-Based Concurrent Computing (ECOOP '91)*, volume 612 of *LNCS*, pages 21–52. Springer Verlag, 1992.
12. N. Kavantzas, G. Olsson, J. Mischkinsky, and M. Chapman. Web Services Choreography Description Languages. Oracle Corporation, 2003.
13. F. Leymann. Web Services Flow Language (wsfl 1.0). Technical report, IBM Software Group, 2001.
14. M. Little. Web services transactions: Past, present and future. Proceedings of the XML Conference and Exposition, Philadelphia, USA, 2003.
15. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100(1):1–77, 1992.
16. OASIS. Introduction to UDDI: Important features and functional concepts. Organization for the Advancement of Structured Information Standards, 2004.
17. S. Thatte. XLANG: Web services for business process design. Microsoft Corporation, 2001.