Action Abstraction in Timed Process Algebra
The Case for an Untimed Silent Step

Michel A. Reniers and Muck van Weerdenburg

Technical University Eindhoven (TU/e)
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
M.A.Reniers@tue.nl, M.J.van.Weerdenburg@tue.nl

Abstract. This paper discusses action abstraction in timed process algebras. It is observed that the leading approaches to action abstraction in timed process algebra all maintain the timing of actions, even if these actions are abstracted from.

This paper presents a novel approach to action abstraction in timed process algebras. Characteristic for this approach is that in abstracting from an action, also its timing is abstracted from. We define an abstraction operator and a timed variant of rooted branching bisimilarity and establish that this notion is an equivalence relation and a congruence.

1 Introduction

One of the main tools in analysing processes in a process-algebraic setting is abstraction. Abstraction allows for the removal of information that is regarded as unobservable (or irrelevant) for the verification purpose at hand. Abstraction is introduced in the form of an action abstraction operator, called hiding, or in the form of data abstraction through abstract interpretations. In action hiding, certain action names are made anonymous and/or unobservable by replacing them by a predefined *silent step* (also called internal action) denoted by τ.

In the field of untimed process algebra, there is reasonable consensus about the properties of the silent step. In ACP-style process algebras [1] the notion of (rooted) branching bisimilarity, as put forward by Van Glabbeek and Weijland in [2,3], is mostly adopted. The few timed versions of rooted branching bisimilarity found in the literature (see [4,5,6]) and of weak bisimilarity (see [7,8,9,10]) all agree on maintaining the timing of actions, even if these actions are abstracted from. In all of these approaches the passing of time by itself (i.e., without subsequent action execution or termination) can be observed. As a consequence, not as many identifications between processes can be made as is desirable for verification purposes. This hinders the verification of correctness of real-time systems and therefore this situation needs to be improved.

In this paper, we study an action abstraction mechanism that not only abstracts from an action, but also from its timing. We introduce an *untimed silent step* into a timed process algebra. We define a timed version of rooted branching bisimilarity based on this untimed silent step, show that it is an equivalence and

F. Arbab and M. Sirjani (Eds.): FSEN 2007, LNCS 4767, pp. 287–301, 2007.

a congruence, and present a remarkably straightforward axiomatisation for this notion of equivalence. We give a short account of the identifications between processes that can be obtained using this equivalence. This is done by showing simplifications of the PAR protocol using the notions of equivalence from the literature and the notion introduced in this paper.

It should be mentioned that when studying timed process algebras (or timed automata for that matter), one encounters a number of different interpretations of the interaction between actions and time. There are the so-called two-phase models, where the progress of time is modeled separately from action execution, and there is the time-stamped setting, where time progress and action execution are modeled together. Two-phase models are used in [11], and time-stamped models are found in timed μCRL [12], for example. In this paper, we study timed rooted branching bisimilarity in the context of an absolute time, time-stamped model.

Structure. First, we introduce a simple timed process algebra with absolute timing and a time-stamped model (Sect. 2). This process algebra serves as a vehicle for our discussions on abstraction and equality of processes. It contains primitives that are fundamental to virtually every timed process algebra. In Sect. 3, we discuss the notions of timed rooted branching bisimilarity as they are encountered in the literature. In Sect. 4, we adapt the timed process algebra to incorporate our ideas for abstraction and equality for timed processes interpreted in a time-stamped model. In Sect. 5, we illustrate the consequences of our definitions on the PAR protocol. In Sect. 6, we present axioms for timed strong bisimilarity and timed rooted branching bisimilarity. In Sect. 7, we discuss the possibilities and impossibilities of adapting our notions to other settings in timed process algebra from the literature. Section 8 wraps up the paper.

2 The Universe of Discourse

In this section, we introduce a simple time-stamped process algebra without abstraction. This process algebra serves well for a more formal exposition of our discomfort with the existing ways of dealing with abstraction in timed process algebra and for a discussion of the possible solutions. Also, this process algebra will be used for the treatment of the chosen solution.

The timed process algebra presented in this section, $\mathrm{BSP}^{@}_{\mathrm{abs}}$ (for *Basic Sequential Processes* with *absolute time* and *time-stamping*), is an extension of the process theory BSP from [13] with absolute-timing and time-stamping (both syntactically and semantically) inspired by the process algebra *timed* μCRL [12][1].

We first present the starting point of our deliberations. We assume a set Time that is totally ordered by $\leq$ with smallest element 0 that represents the time domain[2]. We also assume a set Act of actions, *not* containing τ.

[1] Note that in the original semantics of timed μCRL [14], a two-phase model is used with states consisting of a closed process term and a moment in time, and separate action transitions $\xrightarrow{a}$ and a time transition $\xrightarrow{\iota}$.

[2] It does not matter for the treatment whether this time domain is discrete or dense.

The signature of the process algebra $\mathrm{BSP}^{@}_{\mathrm{abs}}$ consists of the following constants and operators:

- for each $t \in \mathrm{Time}$, a timed deadlock constant $0^{@t}$. The process $0^{@t}$ idles up to time t and then deadlocks.
- for each $t \in \mathrm{Time}$, a timed termination constant $1^{@t}$. The process $1^{@t}$ idles up to time t and then terminates successfully.
- for each $a \in \mathrm{Act}$ and $t \in \mathrm{Time}$, an action prefix operator $a^{@t}.\_$. The process $a^{@t}.p$ represents the process that idles up to time t, executes action a at that time and after that behaves as process p insofar time allows.
- the alternative-composition operator $\_ + \_$. The process $p + q$ represents the nondeterministic choice between the processes p and q. The choice is resolved by the execution of an action or an occurrence of a termination.
- for each $t \in \mathrm{Time}$, a time-initialisation operator $t \gg \_$. The process $t \gg p$ is p limited to those alternatives that execute their first action not before time t.

Terms can be constructed using variables and the elements from the signature. Closed terms are terms in which no variables occur. We decide to allow the execution of more than one action at the same moment of time (in some order). There are no fundamental reasons for this choice: we could equally well have adopted the choice to disallow such *urgent* actions.

Next, we provide a structured operational semantics for the closed terms from this process algebra. We define the following transition relations and predicates:

- a time-stamped action-transition relation $\_ \xrightarrow{a}_t \_$ (with $a \in \mathrm{Act}$ and $t \in \mathrm{Time}$), representing the execution of an action a at time t.
- a time-stamped termination predicate $\_ \downarrow_t$ (with $t \in \mathrm{Time}$), representing successful termination at time t.
- a time-parameterised delay predicate $\_ \rightsquigarrow_t$ (with $t \in \mathrm{Time}$), representing that a process can idle until time t (at least).

The reason for including the delay predicate is to discriminate between differently timed deadlocks: $0^{@3} \rightsquigarrow_3$, whereas $0^{@2} \not\rightsquigarrow_3$. These transition relations and predicate are defined by means of a so-called term deduction system [15]. The deduction rules are presented in Table 1. In this table and others in the rest of this paper, x, x', y and y' are variables representing arbitrary process terms, $a \in \mathrm{Act}$ is an action name, $I \subseteq \mathrm{Act}$ and $t, u \in \mathrm{Time}$.

Note that the time-initialisation operator is used in the structured operational semantics to impose upon a process the absolute time point that has been reached by previous activity.

Timed strong bisimilarity (as defined in [12], for example) is a congruence for all operators from this process algebra. One can quite easily obtain a sound and complete axiomatisation of timed strong bisimilarity. The details are omitted as they are of no importance to the goal of this paper.

Table 1. Structured Operational Semantics of $\mathrm{BSP}^{@}_{\mathrm{abs}}$

$$\frac{}{0^{@t} \rightsquigarrow_u} [u \leq t] \qquad \frac{}{1^{@t} \downarrow_t} \qquad \frac{}{1^{@t} \rightsquigarrow_u} [u \leq t] \qquad \frac{}{a^{@t}.x \xrightarrow{a}_t t \gg x}$$

$$\frac{}{a^{@t}.x \rightsquigarrow_u} [u \leq t] \qquad \frac{x \xrightarrow{a}_t x'}{\begin{array}{l} x + y \xrightarrow{a}_t x' \\ y + x \xrightarrow{a}_t x' \end{array}} \qquad \frac{x \downarrow_t}{\begin{array}{l} x + y \downarrow_t \\ y + x \downarrow_t \end{array}} \qquad \frac{x \rightsquigarrow_t}{\begin{array}{l} x + y \rightsquigarrow_t \\ y + x \rightsquigarrow_t \end{array}}$$

$$\frac{x \xrightarrow{a}_u x'}{t \gg x \xrightarrow{a}_u x'} [t \leq u] \qquad \frac{x \downarrow_u}{t \gg x \downarrow_u} [t \leq u] \qquad \frac{}{t \gg x \rightsquigarrow_u} [u \leq t] \qquad \frac{x \rightsquigarrow_u}{t \gg x \rightsquigarrow_u}$$

3 Abstraction and the Timed Silent Step

In order to facilitate abstraction of actions, usually a special atomic action $\tau \notin \mathrm{Act}$ is assumed that represents an *internal action* or *silent step*. Also, an abstraction operator τ_I (for $I \subseteq \mathrm{Act}$) is used for specifying which actions need to be considered internal. This leads to the following extensions to the signature of the process algebra:

- for each $t \in \mathrm{Time}$, a silent step prefix operator $\tau^{@t}.\_$. The process $\tau^{@t}.p$ represents the process that idles up to time t, executes silent step τ at that time and after that behaves as process p insofar time allows.
- for each $I \subseteq \mathrm{Act}$, an abstraction operator τ_I. The process $\tau_I(p)$ represents process p in which all actions from the set I are made invisible (i.e., replaced by silent step τ).

To express execution of a silent step at a certain time t the predicate $\_ \xrightarrow{\tau}_t \_$ is used. The silent step prefix operator has precisely the same deduction rules as the action prefix operator (with a replaced by τ). The deduction rules for the abstraction operator are given below.

$$\frac{x \xrightarrow{a}_t x'}{\tau_I(x) \xrightarrow{a}_t \tau_I(x')} [a \notin I] \qquad \frac{x \xrightarrow{a}_t x'}{\tau_I(x) \xrightarrow{\tau}_t \tau_I(x')} [a \in I]$$

$$\frac{x \xrightarrow{\tau}_t x'}{\tau_I(x) \xrightarrow{\tau}_t \tau_I(x')} \qquad \frac{x \downarrow_t}{\tau_I(x) \downarrow_t} \qquad \frac{x \rightsquigarrow_t}{\tau_I(x) \rightsquigarrow_t}$$

Again, congruence of timed strong bisimilarity is obvious and obtaining a sound and complete axiomatisation of timed strong bisimilarity is not difficult either.

Timed Rooted Branching Bisimilarity. In the rest of this section, we discuss several timed versions of the well-known notion of rooted branching bisimilarity [2,3]. We refer to the relevant literature for definitions of these notions. We

only present some characteristic equalities and inequalities between processes to illustrate the notions.

In [4, Chapter 6], Klusener defines notions of timed rooted branching bisimilarity for a timed process algebra in a setting that does not allow for consecutive actions at the same moment in time, i.e., non-urgent actions. Two semantics and equivalences are defined, both in a setting with time-stamped action transitions. The first semantics, the so-called *idle* semantics employs idle transitions to model time passing. The second, called the *term* semantics, uses an ultimate delay predicate instead. Characteristic for the equivalences is that an action transition a at time t in one process may be mimicked in another process by a well-timed sequence (i.e., a sequence in which the timing of the subsequent actions does not decrease) of silent steps that is ultimately followed by an a-transition at time t. The intermediate states need to be related with the original state (at the right moment in time). Klusener shows that in his setting these two semantics and equivalences coincide. In almost the same setting[3], using the term semantics, Fokkink proves a completeness result for the algebra of regular processes [16,17]. By means of the following examples we will discuss the equivalences of Klusener. For these examples it is possible to eliminate the abstraction operator from the process terms. We have not done so in order to be able to use these examples again in their current form in the next section (where we have a slightly different syntax).

Example 1 (No-Choice Silent Step). The three processes $\tau_{\{b\}}(a^{@1}.b^{@2}.c^{@4}.0^{@5})$, $\tau_{\{b\}}(a^{@1}.b^{@3}.c^{@4}.0^{@5})$ and $a^{@1}.c^{@4}.0^{@5}$ are obviously considered equal. Thus, the timing of the action that is hidden is of no importance insofar it does not disallow other actions from occurring (due to ill-timedness).

Example 2 (Time-Observed Silent Step). The processes $\tau_{\{b\}}(a^{@1}.(b^{@2}.(c^{@3}.0^{@4}+d^{@3}.0^{@4})+d^{@3}.0^{@4}))$ and $a^{@1}.(c^{@3}.0^{@4}+d^{@3}.0^{@4})$ are distinguished by the notion of timed rooted branching bisimilarity from [4, Chapter 6]. The reason is that in the first process at time 2 it may be determined that the d will be executed at time 3, while in the latter process term the choice between the c and the d at 3 can not be done earlier than at time 3.

Example 3 (Swapping). The processes $\tau_{\{b\}}(a^{@1}.(b^{@2}.c^{@3}.0^{@4} + d^{@3}.0^{@4}))$ and $\tau_{\{b\}}(a^{@1}.(c^{@3}.0^{@4}+b^{@2}.d^{@3}.0^{@4}))$ are considered equal with respect to Klusener's notion of equality, since in both processes it is decided at time 2 whether the c or the d will be executed at time 3.

It is interesting to note that, if one considers Klusener's definition of timed rooted idle branching bisimilarity in a setting in which urgent actions are allowed, the swapping of silent steps as portrayed in this example does not hold anymore. With timed rooted branching bisimilarity as defined for the term semantics though, it remains valid. This is due to the fact that the latter notion explicitly limits the behaviour of processes.

[3] Fokkink uses a relative-time syntax and semantics and defines the ultimate delay predicate slightly different.

Example 4 (Time-Choice). According to [4] the processes $\tau_{\{b\}}(a^{@1}.\ (b^{@3}.0^{@4} + c^{@2}.0^{@4}))$ and $a^{@1}.(0^{@4} + c^{@2}.0^{@4})$ are equal, since the passage of time already decides at time point 2 whether or not the alternative $c^{@2}.0^{@4}$ occurs or not.

Baeten and Bergstra introduce the silent step to relative time, absolute time and parametric time (i.e., a mixture of both relative and absolute time) versions of ACP with discrete time in [5]. A difference with the work of Klusener is that time steps are represented explicitly in the syntax in [5]. In [18], a complete axiomatisation for timed rooted branching bisimilarity is provided, for a variant of this theory. With respect to the four examples presented before, the only difference between Klusener's notion and Baeten and Bergstra's notion is that the latter does *not* consider the processes from Example 3 (Swapping) equal.

In [6], Van der Zwaag defines a notion of timed branching bisimilarity for a process algebra that has almost the same syntax and semantics as ours. In the setting studied by Van der Zwaag there is no successful termination. In [19], Fokkink et al. show that the notion of timed branching bisimilarity as put forward by Van der Zwaag is not an equivalence for dense time domains and therefore they present a stronger notion of timed branching bisimilarity that is an equivalence indeed. Also, the definitions are extended to include successful termination. These notions of timed rooted branching bisimilarity are similar to that of Baeten and Bergstra for the examples presented before.

The way in which abstraction of actions leads to very precisely timed silent steps can be considered problematic (from a practical point of view). This was also recognised by Baeten, Middelburg and Reniers in [20] in the context of a relative-time discrete-time process algebra with two-phase time specifications. The equivalences as described above are not coarse enough in practical cases such as the PAR protocol. An attempt is made to establish a coarser equivalence (called abstract branching bisimilarity) that "treats an internal action always as redundant if it is followed by a process that is only capable of idling till the next time slice." This leads to an axiom (named DRTB4) of the form $\tau_{\{a\}}(a^{@t}.x) = \tau_{\{a\}}(t \gg x)$ (in a different syntax).

Although we support the observation of the authors from [20] that a coarser notion of equivalence is needed, we have a major problem with the treatment of this issue in [20]. The authors have sincere difficulties in defining the equivalence on the structured operational semantics. This difficulty is ultimately solved by using the (standard) definition of rooted branching (tail) bisimilarity from [18] in combination with a structured operational semantics that is a silent-step-saturated version of the original semantics.

4 Untimed Silent Step

In this section, we present a novel abstraction mechanism in timed process algebra that is inspired by the opinion that *the timing of a silent step as such is not observable*. Therefore, one might consider defining an abstraction operator that abstracts from an action *and* from its timing. One should be careful though,

that abstraction from the timing of action a may not result in an abstraction of the consequences of this timing of a for the rest of the process!

In the next section, we formally present our novel approach to action abstraction in timed process algebras. First we give the consequences of our intuition about the equality (called timed rooted branching bisimilarity, denoted by $\underline{\leftrightarrow}_{\mathrm{rb}}$, see Sect. 4.2 for a definition) of the example processes from the previous section.

The timing of the action that is hidden is of no importance insofar it does not disallow other actions from occurring (due to ill-timedness). Therefore, the processes from Example 1 (No-Choice) should be considered equal:

$$\tau_{\{b\}}(a^{@1}.b^{@2}.c^{@4}.0^{@5}) \quad \underline{\leftrightarrow}_{\mathrm{rb}} \quad \tau_{\{b\}}(a^{@1}.b^{@3}.c^{@4}.0^{@5}) \quad \underline{\leftrightarrow}_{\mathrm{rb}} \quad a^{@1}.c^{@4}.0^{@5}$$

The processes from Example 2 (Time-Observed Silent Step) are equal in our setting since we do not wish to consider the timing of the internal step relevant:

$$\tau_{\{b\}}(a^{@1}.(b^{@2}.(c^{@3}.0^{@4}+d^{@3}.0^{@4})+d^{@3}.0^{@4})) \quad \underline{\leftrightarrow}_{\mathrm{rb}} \quad a^{@1}.(c^{@3}.0^{@4}+d^{@3}.0^{@4})$$

The processes from Example 3 (Swapping) are different processes, since by executing the silent step, an option that was there before has disappeared:

$$\tau_{\{b\}}(a^{@1}.(b^{@2}.c^{@3}.0^{@4}+d^{@3}.0^{@4})) \quad \underline{\leftrightarrow}_{\mathrm{rb}} \quad \tau_{\{b\}}(a^{@1}.(c^{@3}.0^{@4}+b^{@2}.d^{@3}.0^{@4}))$$

Since we do not allow to take the timing of the abstracted action into account, we cannot have the equality of the processes from Example 4 (Time-Choice):

$$\tau_{\{b\}}(a^{@1}.(b^{@3}.0^{@4}+c^{@2}.0^{@4})) \quad \underline{\leftrightarrow}_{\mathrm{rb}} \quad a^{@1}.(0^{@4}+c^{@2}.0^{@4})$$

In contrast with the other equivalences discussed in this paper, the process $\tau_{\{b\}}(a^{@1}.(b^{@3}.0^{@4}+c^{@2}.0^{@4}))$ can only be 'simplified' to $a^{@1}.(\tau.0^{@4}+c^{@2}.0^{@4})$. Thus the silent step remains.

In our opinion, in [20] too many silent steps can be omitted. Consider for example the process $\tau_{\{a\}}(a^{@1}.0^{@2}+b^{@3}.0^{@4})$. In [20], it is considered to be equal to $b^{@3}.0^{@4}$. In our opinion, the execution of the internal step disables the execution of action b altogether.

4.1 Abstraction Using the Untimed Silent Step

We propose to extend the process algebra from Sect. 2 with the following primitives instead of the timed silent action prefix operators and abstraction operator from Sect. 3:

- the silent step prefix operator $\tau.\_$. The process $\tau.p$ performs an internal action (not at any specific time) and thereafter behaves as p. Note that the occurrence of such an internal action cannot result in disabling an action from p.
- for each $I \subseteq \mathrm{Act}$, the abstraction operator τ_I. The process $\tau_I(p)$ represents process p where all actions from the set I are made invisible (replaced by the untimed silent step τ). It should be noted that the consequences of the timing of the abstracted action are not abstracted from.

In the structured operational semantics, we add a relation $\_ \xrightarrow{\tau} \_$ that represents the execution of an untimed silent step. For alternative composition and time initialisation we add deduction rules for this new transition relation (the first two deduction rules in Table 2). In the second deduction rule for the abstraction operator one can see that a timed action is replaced by an untimed silent step

Table 2. Structured Operational Semantics of untimed silent step and abstraction operator

$$\frac{x \xrightarrow{\tau} x'}{x + y \xrightarrow{\tau} x' \quad y + x \xrightarrow{\tau} x'} \qquad \frac{x \xrightarrow{\tau} x'}{t \gg x \xrightarrow{\tau} t \gg x'} \qquad \frac{}{\tau.x \xrightarrow{\tau} x} \qquad \frac{x \rightsquigarrow_t}{\tau.x \rightsquigarrow_t}$$

$$\frac{x \xrightarrow{a}_t x'}{\tau_I(x) \xrightarrow{a}_t \tau_I(x')}\ [a \notin I] \qquad \frac{x \xrightarrow{a}_t x'}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}\ [a \in I] \qquad \frac{x \xrightarrow{\tau} x'}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}$$

$$\frac{x \downarrow_t}{\tau_I(x) \downarrow_t} \qquad \frac{x \rightsquigarrow_t}{\tau_I(x) \rightsquigarrow_t}$$

in case the action is to be abstracted from. Also note that the consequences of the timing of the action are imposed on the rest of the process by means of the time-initialisation operator in the deduction rule for action-transitions of the action prefix operator (in Table 1). This means that the process x' incorporates the fact that time t has been reached.

Example 5. Somewhat surprisingly, the process $p = a^{@2}.\tau_{\{b\}}(b^{@1}.0^{@4})$ is not ill-timed. This is a consequence of our decision that the timing of abstracted actions is not observable. Thus the process p is equal to $a^{@2}.0^{@4}$ and of course also to $a^{@2}.\tau_{\{b\}}(b^{@3}.0^{@4})$ (which can hardly be considered ill-timed).

4.2 Timed Rooted Branching Bisimilarity

In the following definition we use the notation $p \Rightarrow q$ to denote that q can be reached from p by executing an arbitrary number (possibly zero) of τ-transitions. The notation $p \xrightarrow{(\tau)} q$ means $p \xrightarrow{\tau} q$ or $p = q$.

Definition 1 (Timed Rooted Branching Bisimilarity). *Two closed terms p and q are* timed branching bisimilar*, notation $p \leftrightarrows_{\mathrm{b}} q$, if there exists a symmetric binary relation R on closed terms, called a* timed branching bisimulation *relation, relating p and q such that for all closed terms r and s with $(r, s) \in R$:*

1. *if $r \xrightarrow{a}_t r'$ for some $a \in$ Act, $t \in$ Time and closed term r', then there exist closed terms s^* and s' such that $s \Rightarrow s^* \xrightarrow{a}_t s'$, $(r, s^*) \in R$ and $(r', s') \in R$;*

2. *if* $r \xrightarrow{\tau} r'$ *for some closed term* r'*, then there exist closed terms* s^* *and* s' *such that* $s \Rightarrow s^* \xrightarrow{(\tau)} s'$*,* $(r, s^*) \in R$ *and* $(r', s') \in R$*;*
3. *if* $r \downarrow_t$ *for some* $t \in$ Time*, then there exists a closed term* s^* *such that* $s \Rightarrow s^* \downarrow_t$ *and* $(r, s^*) \in R$*;*
4. *if* $r \rightsquigarrow_t$ *for some* $t \in$ Time*, then there exists a closed term* s^* *such that* $s \Rightarrow s^* \rightsquigarrow_t$ *and* $(r, s^*) \in R$*.*

If R *is a timed branching bisimulation relation, we say that the pair* (p, q) *satisfies the* root condition *with respect to* R *if*

1. *if* $p \xrightarrow{a}_t p'$ *for some* $a \in$ Act*,* $t \in$ Time *and closed term* p'*, then there exists a closed term* q' *such that* $q \xrightarrow{a}_t q'$ *and* $(p', q') \in R$*;*
2. *if* $p \xrightarrow{\tau} p'$ *for some closed term* p'*, then there exists a closed term* q' *such that* $q \xrightarrow{\tau} q'$ *and* $(p', q') \in R$*;*
3. *if* $p \downarrow_t$ *for some* $t \in$ Time*, then* $q \downarrow_t$*;*
4. *if* $p \rightsquigarrow_t$ *for some* $t \in$ Time*, then* $q \rightsquigarrow_t$*.*

Two closed terms p *and* q *are called* timed rooted branching bisimilar*, notation* $p \underline{\leftrightarrow}_{\mathrm{rb}} q$*, if there is a timed branching bisimulation relation* R *relating* p *and* q *such that the pairs* (p, q) *and* (q, p) *satisfy the root condition with respect to* R*.*

Note that we have actually defined a timed version of the notion of semi-branching bisimilarity of [21].

4.3 Properties of Timed Rooted Branching Bisimilarity

In this section, we show that timed rooted branching bisimilarity as defined in the previous section is indeed an equivalence. Moreover we show that it is a congruence for the rather restricted set of operators introduced. Proofs of the theorems given in this section can be found in [22].

Theorem 1. *Timed rooted branching bisimilarity is an equivalence relation.*

Theorem 2. *Timed strong bisimilarity and timed rooted branching bisimilarity are congruences for all operators from the signature of the process algebra* $\mathrm{BSP}^{@}_{\mathrm{abs}}$*.*

Furthermore, obviously timed rooted branching bisimilarity identifies strictly more processes than timed strong bisimilarity does.

Theorem 3. *Timed strongly bisimilar processes are timed rooted branching bisimilar: i.e.,* $\underline{\leftrightarrow} \subset \underline{\leftrightarrow}_{\mathrm{rb}}$*.*

From the examples presented in the previous sections, we can easily conclude that our notion of equality is incomparable with the notions from Klusener [4], Baeten and Bergstra [5] and Van der Zwaag [6]. We claim that the notion of abstract branching bisimilarity from [20] is coarser than ours.

5 Case Study: PAR Protocol

In [20] the Positive Acknowledgement Retransmission protocol is used to illustrate the need for a coarser equivalence. In this paper, we will use the same protocol to illustrate our notion of timed rooted branching bisimilarity. An informal description of the protocol can be found in [20]. For comparison, we present a linearised version of the protocol in which the internal communications are abstracted from and as many silent steps as possible have been removed/omitted using the notion of abstraction and timed rooted branching bisimilarity from [5]. This result is obtained by translating the result from [20] to our setting. Note that we have used notations such as $\sum_{t'} p$ that describe a potentially infinite alternative composition consisting of one alternative of p for each t'. We refrain from giving operational semantics for this operator, called summation [12] or alternative quantification [23].

$$
\begin{aligned}
X_{b,t} &= \sum_{t'} \sum_{d \in D} r_1(d)^{@t+t'} . Y_{d,b,t+t'+t_S} \\
Y_{d,b,t} &= \tau^{@t+t_K} . s_2(d)^{@t+t_K+t_R} . Z_{d,b,t+t_K+t_R+t'_R} + \sum_{k \leq t_K} \tau^{@t+k} . Y_{d,b,t+t'_S} \\
Z_{d,b,t} &= \tau^{@t+t_L} . X_{\overline{b},t+t_L} + \sum_{l \leq t_L} \tau^{@t+l} . U_{d,b,t+t'_S-t_K-t_R-t'_R} \\
U_{d,b,t} &= \tau^{@t+t_K} . V_{d,b,t+t_K+t'_R} + \sum_{k \leq t_K} \tau^{@t+k} . U_{d,b,t+t'_S} \\
V_{d,b,t} &= \tau^{@t+t_L} . X_{\overline{b},t+t_L} + \sum_{l \leq t_L} \tau^{@t+l} . U_{d,b,t+t'_S-t_K-t'_R}
\end{aligned}
$$

Below, we present a linearised version based on the notion of abstraction and timed rooted branching bisimilarity as proposed in this paper:

$$
\begin{aligned}
X_{b,t} &= \sum_{t'} \sum_{d \in D} r_1(d)^{@t+t'} . Y_{d,b,t+t'+t_S} \\
Y_{d,b,t} &= \tau . s_2(d)^{@t+t_k+t_R} . U'_{d,b,t,t_R} + \tau . Y_{d,b,t+t'_S} \\
U'_{d,b,t,u} &= \tau . X_{\overline{b},t+t_K+u+t'_R+t_L} + \tau . U'_{d,b,t+t'_S,0}
\end{aligned}
$$

The silent steps that are left are essential. The silent steps in Y determine whether or not an error occurred in channel K and those in U' determine the same for channel L. As these errors result in an additional delay before the next action occurs, they are not redundant. In [22] a more detailed discussion of this case study can be found.

6 Axioms for Timed Rooted Branching Bisimilarity

In Table 3 we present axioms for timed strong bisimilarity. The axioms (A1)-(A3) express some standard properties of alternative composition. Axiom (WT) (for well-timedness) describes that the time that is reached by executing an action is passed on to the subsequent process. The axioms (A6a)-(A6d) describe the properties of timed deadlocks, especially the circumstances under which they can be removed from the process description. An important equality that can be derived for closed terms p is $p + 0^{@0} = p$.

Axioms (I1)-(I7) describe how the time-initialisation operator can be eliminated from terms. Note that the silent step neglects this operator (axiom (I6)). Axioms (H1)-(H6) describe how the abstraction operator can be eliminated. Note that the timing of an action that is abstracted from is passed on to the rest of the process (axiom (H4)). The time-initialisation operator in the right-hand side of axiom (H3) is needed in order to enforce the timing restriction from the action prefix before applying further abstractions.

Table 3. Axioms for timed strong bisimilarity and timed rooted branching bisimilarity

(A1)	$x + y = y + x$	(A6a)	$0^{@t} + 0^{@u} = 0^{@\max(t,u)}$
(A2)	$(x + y) + z = x + (y + z)$	(A6b)	$u \leq t \Rightarrow 1^{@t} + 0^{@u} = 1^{@t}$
(A3)	$x + x = x$	(A6c)	$u \leq t \Rightarrow a^{@t}.x + 0^{@u} = a^{@t}.x$
(WT)	$a^{@t}.x = a^{@t}.t \gg x$	(A6d)	$u \leq t \Rightarrow \tau.(x + 0^{@t}) + 0^{@u} = \tau.(x + 0^{@t})$
(I1)	$t \gg 0^{@u} = 0^{@\max(t,u)}$	(H1)	$\tau_I(0^{@t}) = 0^{@t}$
(I2)	$u < t \Rightarrow t \gg 1^{@u} = 0^{@t}$	(H2)	$\tau_I(1^{@t}) = 1^{@t}$
(I3)	$u \geq t \Rightarrow t \gg 1^{@u} = 1^{@u}$		
(I4)	$u < t \Rightarrow t \gg a^{@u}.x = 0^{@t}$	(H3)	$a \notin I \Rightarrow \tau_I(a^{@t}.x) = a^{@t}.\tau_I(t \gg x)$
(I5)	$u \geq t \Rightarrow t \gg a^{@u}.x = a^{@u}.x$	(H4)	$a \in I \Rightarrow \tau_I(a^{@t}.x) = \tau.\tau_I(t \gg x)$
(I6)	$t \gg \tau.x = \tau.t \gg x$	(H5)	$\tau_I(\tau.x) = \tau.\tau_I(x)$
(I7)	$t \gg (x + y) = t \gg x + t \gg y$	(H6)	$\tau_I(x + y) = \tau_I(x) + \tau_I(y)$

We claim that the axioms from Table 3 are sound and complete for timed strong bisimilarity on closed terms. These axioms are (of course; see Theorem 3) also valid for timed rooted branching bisimilarity. In Table 4, one additional axiom is presented for timed rooted branching bisimilarity. The reader should notice that this axiom resembles the untimed axiom for rooted branching bisimilarity $a.(\tau.(x + y) + x) = a.(x + y)$ meticulously. Also, it is expected that the axioms from both tables provide a sound and complete axiomatisation of timed rooted branching bisimilarity on closed terms.

Table 4. Axiom for timed rooted branching bisimilarity

(B) $a^{@t}.(\tau.(x+y)+x) = a^{@t}.(x+y)$

7 Other Timed Process Algebra Settings

The process algebra that we have chosen as our universe of discourse can be classified (both syntactically and semantically) as an absolute-time time-stamped process algebra. As mentioned before, in the literature there are some other versions available, with respect to both the syntax used and the semantics adopted. In this section, we discuss, with respect to the semantics, how the abstraction technique presented here for an absolute-time time-stamped process algebra can be carried over to other types of timed process algebras and what problems are expected to arise in doing so.

In a setting where the time-stamping mechanism uses relative time the treatment becomes even simpler. In such a setting $a^{@t}.p$ means that a is to be executed t time after the execution of the previous action (or after the conception of the process). As a consequence of this relative-timing the problem of ill-timedness is avoided. Therefore, the time-initialisation operator can be left out. Instead, one needs to have a mechanism for updating the relative time-stamp of the initial actions of the subsequent process due to abstraction:

$$\frac{x \xrightarrow{a}_t x' \quad a \in I}{\tau_I(x) \xrightarrow{\tau} t \circledast \tau_I(x')}$$

where $t \circledast p$ means that t time has to be added to the time-stamp of the first visible action from p. For example $3 \circledast a^{@5}.p$ behaves as $a^{@8}.p$. An example of such an operator is the time shift operator $(t)\_$ (also with negative t!) that has been used by Fokkink for defining timed branching bisimilarity in [16].

We have chosen to carry out our deliberations in a time-stamped setting because this setting allows for a very natural definition of the abstraction operator since the timing of the action (before abstraction) and the action itself are tightly coupled in the model. To illustrate the difficulties that arise in defining abstraction in a two-phase model, we look at the following processes (in the syntax of [24,25]). Note that $\sigma.\_$ is a time step prefix operator and $\underline{\underline{a}}.\_$ is an immediate action prefix operator.

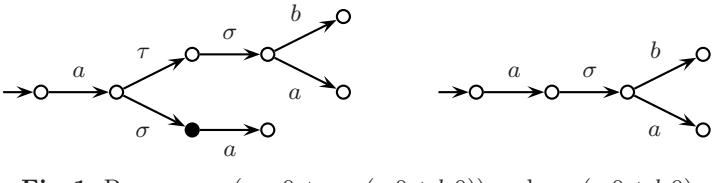

Fig. 1. Processes $\underline{\underline{a}}.(\sigma.\underline{\underline{a}}.\underline{\underline{0}} + \underline{\underline{\tau}}.\sigma.(\underline{\underline{a}}.\underline{\underline{0}} + \underline{\underline{b}}.\underline{\underline{0}}))$ and $\underline{\underline{a}}.\sigma.(\underline{\underline{a}}.\underline{\underline{0}} + \underline{\underline{b}}.\underline{\underline{0}})$

As we have discussed in Sect. 4, we consider these processes equivalent. However, to express this in an equivalence, we need to be able to relate the states of both processes. In the diagram above one can see that the first process can make a time transition that results in a state (the black one) that has no corresponding state in the second process. The essence of this problem is that one tries to relate states that are reached solely by time steps such as the black one. We thus believe the solution is to not necessarily relate such states, even if they exist.

8 Concluding Remarks

In this paper, we have introduced a notion of abstraction that abstracts from the identity of an action as well as its timing, resulting in an untimed silent step. We have developed an accompanying notion of equality of processes, also called timed rooted branching bisimilarity. We have shown that this notion is an equivalence relation and a congruence for all operators considered in this paper and as such may be a meaningful tool in analysing and verifying systems. A first experiment, on the PAR protocol, indicates that our notions allow for a much clearer and smaller representation of the abstract system than the standard notions do. An axiomatisation of timed rooted branching bisimilarity for closed process terms is given with an axiom for the removal of untimed silent steps that resembles the well-known axiom from untimed process algebra.

In this paper, we have made many claims about the timed process algebra with untimed silent steps. Of course, these claims need to be substantiated further. Also, it is worthwhile to study our notion of abstraction in other timed settings, most notably those with relative timing and where timing is described by separate timing primitives (decoupled from actions) as in [11] and most other mainstream timed process algebras.

We have illustrated the differences and similarities between the different definitions of timed rooted branching bisimilarity from literature and the version introduced in this paper by means of examples only. A more thorough comparison is needed. Also, a comparison with timed versions of weak bisimilarity (e.g., [7,8,9,10]) should be performed.

In order to illustrate that our restriction to the limited set of operators is not inspired by fundamental limitations, in [22], we have extended the timed process algebra with sequential composition and parallel composition as these operators are frequently encountered in timed process algebras in the ACP community. It turns out that the deduction rules are standard. Also it is shown that timed rooted branching bisimilarity as defined in this paper in a congruence for those operators.

The success of an abstraction mechanism and notion of equality not depend only on the theoretical properties (though important) of these notions, but much more so on the practical suitability of these notions. Therefore, we need to perform more case studies to observe whether these notions contribute to a better/easier verification of correctness and/or properties of relevant systems. In

this direction, we are also interested in a weaker version of the notion of equivalence presented in this paper that additionally considers the processes from Example 4 (Time-Choice) equal.

We are, in line with our previous work ([24,25]), very interested in obtaining a collection of theories that are nicely related by means of conservativity results and embeddings. Therefore, it is interesting to extend the rather limited timed process algebra from this paper with untimed action prefix operators $a.\_$ in order to formally study, in one framework, the relationship between rooted branching bisimilarity on untimed processes and our timed version.

A complementary way of specifying a timed system is by means of a timed (modal) logic. It is worthwhile to get a deeper understanding of our notion of action abstraction and timed rooted branching bisimilarity by considering the relationship with modal logics for timed systems as has been done for strong bisimilarity [26] and Hennessy-Milner logic [27]. We have good hope that the majority of the logics that are used for the specification of properties of timed systems are preserved by our notion of timed rooted branching bisimilarity.

Acknowledgements. We acknowledge useful comments from Jos Baeten, Pieter Cuijpers, Wan Fokkink, Jan Friso Groote, Bas Luttik, Bas Ploeger, Yaroslav Usenko and Tim Willemse.

References

1. Bergstra, J., Klop, J.: Process algebra for synchronous communication. Information and Control 60(1/3), 109–137 (1984)
2. van Glabbeek, R., Weijland, W.: Branching time and abstraction in bisimulation semantics (extended abstract). In: Ritter, G. (ed.) Information Processing 1989, pp. 613–618. North-Holland, Amsterdam (1989)
3. van Glabbeek, R., Weijland, W.: Branching time and abstraction in bisimulation semantics. Journal of the ACM 43(3), 555–600 (1996)
4. Klusener, A.: Models and Axioms for a Fragment of Real Time Process Algebra. PhD thesis, Eindhoven University of Technology (1993)
5. Baeten, J., Bergstra, J.: Discrete time process algebra with abstraction. In: Reichel, H. (ed.) FCT 1995. LNCS, vol. 965, pp. 1–15. Springer, Heidelberg (1995)
6. van der Zwaag, M.B.: The cones and foci proof technique for timed transition systems. Information Processing Letters 80(1), 33–40 (2001)
7. Moller, F., Tofts, C.: Behavioural abstraction in TCCS. In: Kuich, W. (ed.) Automata, Languages and Programming. LNCS, vol. 623, pp. 559–570. Springer, Heidelberg (1992)
8. Chen, L.: A model for real-time process algebras (extended abstract). In: Borzyszkowski, A.M., Sokolowski, S. (eds.) MFCS 1993. LNCS, vol. 711, pp. 372–381. Springer, Heidelberg (1993)
9. Quemada, J., de Frutos, D., Azcorra, A.: TIC: A TImed calculus. Formal Aspects of Computing 5(3), 224–252 (1993)
10. Ho-Stuart, C., Zedan, H., Fang, M.: Congruent weak bisimulation with dense real-time. Information Processing Letters 46(2), 55–61 (1993)
11. Baeten, J.C.M., Middelburg, C.A.: Process Algebra with Timing. EATCS Monographs. Springer, Heidelberg (2002)

12. Reniers, M., Groote, J., van der Zwaag, M., van Wamel, J.: Completeness of timed μCRL. Fundamenta Informaticae 50(3-4), 361–402 (2002)
13. Baeten, J., Basten, T., Reniers, M.: Process algebra: Equational theories of communicating processes (2007)
14. Groote, J.: The syntax and semantics of timed μCRL. Technical Report SEN-R9709, CWI, Amsterdam (1997)
15. Aceto, L., Fokkink, W., Verhoef, C.: Structural operational semantics. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 197–292. Elsevier, Amsterdam (2001)
16. Fokkink, W.: Clock, Trees and Stars in Process Theory. PhD thesis, University of Amsterdam (1994)
17. Fokkink, W.: An axiomatization for regular processes in timed branching bisimulation. Fundamenta Informaticae 32, 329–340 (1997)
18. Baeten, J., Bergstra, J., Reniers, M.: Discrete time process algebra with silent step. In: Plotkin, G., Stirling, C., Tofte, M. (eds.) Proof, Language, and Interaction: Essays in Honour of Robin Milner. Foundations of Computing Series, pp. 535–569. MIT Press, Cambridge (2000)
19. Fokkink, W., Pang, J., Wijs, A.: Is timed branching bisimilarity an equivalence indeed? In: Pettersson, P., Yi, W. (eds.) FORMATS 2005. LNCS, vol. 3829, pp. 258–272. Springer, Heidelberg (2005)
20. Baeten, J., Middelburg, C., Reniers, M.: A new equivalence for processes with timing: With an application to protocol verification. Technical Report CSR 02-10, Eindhoven University of Technology, Department of Computer Science (2002)
21. Basten, T.: Branching bisimilarity is an equivalence indeed! Information Processing Letters 58(3), 141–147 (1996)
22. Reniers, M., van Weerdenburg, M.: Action abstraction in timed process algebra: The case for an untimed silent step. Technical Report CSR 06-32, Eindhoven University of Technology, Department of Computer Science (2006)
23. Luttik, S.: Choice Quantification in Process Algebra. PhD thesis, University of Amsterdam (April 2002)
24. Baeten, J., Mousavi, M., Reniers, M.: Timing the untimed: Terminating successfully while being conservative. In: Middeldorp, A., van Oostrom, V., van Raamsdonk, F., de Vrijer, R. (eds.) Processes, Terms and Cycles: Steps on the Road to Infinity. LNCS, vol. 3838, pp. 251–279. Springer, Heidelberg (2005)
25. Baeten, J., Reniers, M.: Duplication of constants in process algebra. Journal of Logic and Algebraic Programming 70(2), 151–171 (2007)
26. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) Theoretical Computer Science. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
27. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of the ACM 32, 137–161 (1985)