A Real-Time Process Algebra with Open Intervals and Maximal Progress

Harald Fecher

Universität Mannheim
Fakultät für Mathematik und Informatik,
D7, 27, 68131 Mannheim, Germany
hfecher@pi2.informatik.uni-mannheim.de
published at http://www.ii.uib.no/~nwpt00

Abstract. For real-time process algebras where some actions may occur in open intervals it is hard to meet the maximal progress assumption. A problem is that the maximal progress assumption forces an internal action in the interval $(0, 1)$ to perform at time 0 and on the other hand we have to distinguish, for example, between internal actions in $(0, 1)$ and internal actions in $[0, 1)$, since the latter will always prevent the former from performing if they are combined by a choice operator. To solve this, we introduce a non-observable time step. By this approach it gets necessary to distinguish normal actions from those actions which are getting enabled after a non-observable time step, since the latter are only used to produce internal actions. This is done by extending the set of actions by a marked copy of it. The presented real time process algebra is an extension of Milner's CCS. The algebra can in addition be used to model dynamic priority of actions at the same point in time. We introduce different equivalence relations based on bisimulation.

1 Introduction

Real-time models are used to model time-dependent physical problems. The common approach is to restrict action occurrences to closed, more precisely to left closed intervals [RR88, Sch00, MT90, Wan90, Mur92, FZ94, CL96, LL97]. Contrary to this, *timed automata* [AD94] and some process algebras [BB91, Klu91, Dan91, Che92, Jef92] allow actions to occur in open intervals in order to increase their expressive power.

Another important property of timed process algebras is the *maximal progress assumption*, i.e. an action must occur at the instant at which all participants are ready. This assumption thus guarantees that time cannot proceed if an internal action is enabled. Typical time process algebras which meet this assumption are [HR95, Wan90, Sch00, CL96, LL97].

But none of the mentioned algebras satisfy both properties. In [Dan91], the author considered a *progress assumption*, which means that an internal action must not be delayed more than to its upper time limit, i.e. it is not allowed to disable an internal action by idling. His assumption is contrary to our assumption where idling is completely forbidden once the internal action is enabled. In real time process algebras with priority it is possible to model right open intervals by enabling an internal action with a higher priority at the right end of the interval. But in those algebras it is still impossible to model left open intervals.

Taking both properties, the maximal progress assumption and action occurrence in open intervals, the internal action in the interval $(0, 1)$ is forced to perform at time 0, since otherwise it would contradict with the maximal progress assumption. But on the other hand, we have to distinguish for example between internal actions in $(0, 1)$ and internal actions in $[0, 1)$, since the latter will always prevent the former from performing if they are combined by a choice operator. Therefore, we introduce a non-observable time step. The only changes by a non-observable time step are that actions which reached their upper time limit get disabled and actions in the interval $(0, d)$ or $(0, d]$ get enabled to produce internal actions.[1] To distinguish normal actions from those which are only allowed to produce internal actions, we extend the set of actions by a marked copy of this set.

To illustrate the mentioned ideas, we present a real-time process algebra, termed *Open Timed CCS*, which is in fact an extension of Millner's CCS [Mil89]. We use relative time and we assume that all atomic actions take no time, i.e. they are instantaneous.

In Open Timed CCS it is also possible to model dynamic priority of actions [CLN99] at the same point in time in contrary to [BCL99], where it is necessary to pass real time to model dynamic priority. We model dynamic priority in Open Timed CCS by using internal actions which get enabled in open intervals.

The paper is organized as follows: Section 2 describes syntax and the operational semantics of the process algebra, Open Timed CCS. Some properties of the operational semantics are also discussed. In Section 3 an example of process controlling is modelled in Open Timed CCS. Section 4 establishes the concepts of bisimulation equivalences. A possibility to model priorities in Open Timed CCS is introduced in Section 5. A conclusion is given in Section 6.

2 Process Algebra

In this section we define Open Timed CCS. We keep close to the syntax and semantics of Chen [Che92].

2.1 Syntax

Let Γ be a set of action names, not containing τ. Elements of Γ are denoted by a, b. Define the set of co-names as $\overline{\Gamma} := \{\bar{a} \mid a \in \Gamma\}$. Then the set $\Lambda := \Gamma \cup \overline{\Gamma}$ is the set of *atomic actions*. Further, define the set $\widehat{\Lambda} := \{\hat{c} \mid c \in \Lambda\}$ of *marked atomic actions*. The marked actions are used in the operational semantics to

[1] In [BB93] infinitesimals are used to model the occurrences of two actions at the same point in time, since their process algebra does not allow this.

indicate that these actions are only used to produce *internal actions* (τ) in contrary to the actions in Λ, which are allowed to perform observably. Finally, define $\mathrm{Act} := \Lambda \cup \widehat{\Lambda} \cup \{\tau\}$, ranged over by α.

Definition 1. *The three endofunctions on* Act*:* $\bar{}$, $\hat{}$ *and* $\check{}$ *are defined as follows*

$$\bar{\alpha} := \begin{cases} \bar{a} & \text{if } a = \alpha \\ a & \text{if } \bar{a} = \alpha \\ \hat{\bar{a}} & \text{if } \hat{a} = \alpha \\ \hat{a} & \text{if } \hat{\bar{a}} = \alpha \\ \tau & \text{if } \tau = \alpha \end{cases} \qquad \hat{\alpha} := \begin{cases} \hat{a} & \text{if } a = \alpha \\ \hat{\bar{a}} & \text{if } \bar{a} = \alpha \\ \hat{a} & \text{if } \hat{a} = \alpha \\ \hat{\bar{a}} & \text{if } \hat{\bar{a}} = \alpha \\ \tau & \text{if } \tau = \alpha \end{cases} \qquad \check{\alpha} := \begin{cases} a & \text{if } a = \alpha \\ \bar{a} & \text{if } \bar{a} = \alpha \\ a & \text{if } \hat{a} = \alpha \\ \bar{a} & \text{if } \hat{\bar{a}} = \alpha \\ \tau & \text{if } \tau = \alpha \end{cases}$$

The function $\bar{}$ is commonly used in the literature for the description of synchronisation. The function $\hat{}$, called *mark function*, marks every action except for the internal action. It is used in the operational semantics to describe the behaviour of actions after a non-observable time step. The function $\check{}$, called *unmark function*, unmarks every action except for the internal action. It is used in the operational semantics to describe the behaviour of actions after a real-time step.

A *Λ-relabelling function* f' is an endofunction on Λ such that $\overline{f'(a)} = f'(\bar{a})$ holds. A *relabelling function* f is an endofunction on *Act* such that it is an extension of an Λ-relabelling function, i.e. there is an Λ-relabelling function f' such that

$$f(\alpha) = \begin{cases} f'(\alpha) & \text{if } \alpha \in \Lambda \\ \widehat{f'(a)} & \text{if } \hat{a} = \alpha \text{ and } a \in \Lambda \\ \tau & \text{if } \tau = \alpha \end{cases}$$

Assume a fixed set of process variables $\mathcal{V}_{\mathcal{E}}$ ranged over by X. The process expressions of *Open Timed CCS*, ranged over by E, F, are defined by the following BNF-grammar.

$$E ::= \mathrm{NIL} \mid \alpha^{\prec' e'}_{\succ e}.E \mid E + E \mid E|E \mid E\backslash a \mid E[f] \mid X \mid \mu X.E$$

where $\succ \in \{>, \geq\}$, $\prec' \in \{<, \leq\}$, $e \in \mathbb{R}^+_0$, and $e' \in \mathbb{R}^+_0 \cup \{\infty\}$ the positive reals with zero and infinity. Let $\mathcal{E}$ be the set of all process expressions.

The process NIL can do nothing but idling. Prefix $\alpha^{\prec' e'}_{\succ e}.E$ represents the process which performs the action α between e and e', where it depends on the elements $\succ$ ($\prec'$) if time e (respectively e') is included or excluded. After α has happened the process behaves like E. A choice between process E and E' is represented by the summation $E + E'$. The choice is made by an action execution of E or E', or by time if only one process may delay to that time and no internal action gets enabled before that time (the process which cannot delay is dropped from the future computation). The choice by time proceeding is deterministic. Process $E|E'$ represents the parallel composition of the two processes. Each process may perform actions independently or they synchronise on complementary actions (a and $\bar{a}$), where no distinction between marked and unmarked actions is made. Time has to proceed synchronously, i.e. $E|E'$ is able to delay time d only if E and E' are able to do so. The restriction process $E\backslash a$

behaves like the process E except that it does not allow the actions $a, \bar{a}, \hat{a}, \hat{\bar{a}}$ to perform. The relabelling process $E[f]$ performs the action $f(\alpha)$ if the process E performs the action α. The recursion process $\mu X.E$ allows the specification of infinite behaviours. Here the $\mu.X$ operator in $\mu X.E$ binds all free occurrences of the process variable X in E. Therefore, we get the usual definition of free and bound occurrences of process variables.

We say a process expression is *closed* if there is no free occurrences of process variables. We say a process E is *weakly guarded* if every process variable X of E is in some subterm of form $\alpha_{\succ e}^{\prec' e'}.E'$ of E. Formally, we have

Definition 2. *Define $\Psi \subseteq \mathcal{E} \times 2^{\mathcal{V}_\mathcal{E}}$ by structural induction on $\mathcal{E}$:*

$$
\begin{array}{ll}
\Psi(\mathrm{NIL}, M) & \Psi(E+E', M) \iff \Psi(E, M) \wedge \Psi(E', M) \\
\Psi(\alpha_{\succ e}^{\prec' e'}.E, M) \iff (E, \emptyset) \in \Psi & \Psi(E|E', M) \iff \Psi(E, M) \wedge \Psi(E', M) \\
\Psi(E\backslash a, M) \iff \Psi(E, M) & \Psi(X, M) \iff X \notin M \\
\Psi(E[f], M) \iff \Psi(E, M) & \Psi(\mu X.E, M) \iff \Psi(E, M \cup \{X\})
\end{array}
$$

A process expression is weakly guarded *if and only if $\Psi(E, \mathcal{V}_\mathcal{E})$. Let $\mathcal{G}$ be the set of all weakly guarded expressions.*

We say a process expression to be *regular* if it is weakly guarded, closed and every action which occurs in it is not from $\widehat{\Lambda}$. Sometimes we write $\alpha_{\succ e}^{\prec' e'}$ instead of $\alpha_{\succ e}^{\prec' e'}.\mathrm{NIL}$.

2.2 Operational Semantics

As the time domain $\mathcal{T}$, we consider the positive real numbers ($\mathbb{R}^+$) extended by the least element ι, which denotes the non-observable time step. The non-observable time step is used to model dynamic behaviour change at the same real time. Elements from $\mathcal{T}$ are denoted by t and elements from $\mathbb{R}^+$ are denoted by d. Furthermore, let $\mathcal{T}_0$ be $\mathcal{T}$ with the additional least element 0.

The semantics of Open Timed CCS is defined as the labelled transition system $(\mathcal{E}, \rightarrow)$ where $\rightarrow \subset \mathcal{G} \times (\mathrm{Act} \,\dot\cup\, \mathcal{T}) \times \mathcal{E}$. Here $\dot\cup$ denotes the disjoint union. Note, that for simplicity we only consider weakly guarded expressions on the left hand side of $\rightarrow$. For further details see Remark 7. We separate in the transition system the performing of actions and the passing of time.

To define the transition rules we use Moller & Tofts' maximal delay times of processes before any action [MT90]. Since we consider also right open intervals we have to change the range of the function. Therefore, we define $\mathcal{T}'$ to be $\mathcal{T} \times \{<, \leq\}$ with the lexicographical order, where $<$ is smaller than $\leq$. Furthermore, $\mathcal{T}_c$ be $\mathcal{T}'$ with the additional least element 0 and the additional greatest element ∞.

Definition 3. *Let $|\ |_{\mathcal{T}_c} : \mathcal{G} \rightarrow \mathcal{T}_c$ be defined as follows*

$$
\begin{array}{ll}
|\mathrm{NIL}|_{\mathcal{T}_c} = \infty & |E|E'|_{\mathcal{T}_c} = \min(|E|_{\mathcal{T}_c}, |E'|_{\mathcal{T}_c}) \\
\left|\alpha_{\succ e}^{\prec' e'}.E\right|_{\mathcal{T}_c} = \begin{cases} 0 & \text{if } e' = 0 \\ \infty & \text{if } e' = \infty \\ (e', \prec') & \text{otherwise} \end{cases} & \begin{array}{l} |E\backslash a|_{\mathcal{T}_c} = |E|_{\mathcal{T}_c} \\ |E[f]|_{\mathcal{T}_c} = |E|_{\mathcal{T}_c} \\ |\mu X.E|_{\mathcal{T}_c} = |E|_{\mathcal{T}_c} \end{array} \\
|E+E'|_{\mathcal{T}_c} = \max(|E|_{\mathcal{T}_c}, |E'|_{\mathcal{T}_c}) &
\end{array}
$$

Here $|E|_{\mathcal{T}_c} = (d, <)$ means that the process E is not allowed to idle till time d in contrary to $|E|_{\mathcal{T}_c} = (d, \leq)$ where E is not allowed to idle more than d. Since we do not consider maximal progress in this definition, function $|\ |_{\mathcal{T}_c}$ states to which time a process is allowed to idle if no internal action gets earlier enabled.

Remark 4. Since we only consider weakly guarded expressions we may omit the definition for X and use the definition for $\mu X.E$ without substitution. It can be showed that $|\ |_{\mathcal{T}_c}$ is well defined.

To handle the maximal progress assumption in the operational semantics we use a function *Sort* which gives the available actions of a weakly guarded expression with respect to time. In the definition of *Sort* no distinction between marked and atomic action is made. Furthermore, the maximal progress assumption is not considered in the definition, so not every action is really available. But if an action is in Sort at time t then it is available at time t or an internal action has been enabled before.

Definition 5. *Let* $\mathrm{Sort} : \mathcal{G} \to (\mathcal{T}_0 \to 2^{\Lambda \cup \{\tau\}})$ *be defined as*

$$\begin{aligned}
\mathrm{Sort}(\mathrm{NIL})_{(t)} &= \emptyset \\
\mathrm{Sort}(\alpha^{\prec' e'}_{\succ e}.E)_{(t)} &= \begin{cases} \{\check{\alpha}\} & \textit{if } t \succ e \textit{ and } t \prec' e' \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathrm{Sort}(E + E')_{(t)} &= \mathrm{Sort}(E)_{(t)} \cup \mathrm{Sort}(E')_{(t)} \\
\mathrm{Sort}(E|E')_{(t)} &= \begin{cases} \mathrm{Sort}(E)_{(t)} \cup \mathrm{Sort}(E')_{(t)} \cup I & \textit{if } (t, \leq) \leq |E|E'|_{\mathcal{T}_c} \\ \emptyset & \textit{otherwise} \end{cases} \\
& \qquad \textit{where } I = \begin{cases} \{\tau\} & \textit{if } \mathrm{Sort}(E)_{(t)} \cap \overline{\mathrm{Sort}(E')_{(t)}} \neq \emptyset \\ \emptyset & \textit{otherwise} \end{cases} \\
\mathrm{Sort}(E \backslash a)_{(t)} &= \mathrm{Sort}(E)_{(t)} \backslash \{a, \bar{a}\} \\
\mathrm{Sort}(E[f])_{(t)} &= f(\mathrm{Sort}(E)_{(t)}) \\
\mathrm{Sort}(\mu X.E)_{(t)} &= \mathrm{Sort}(E)_{(t)}
\end{aligned}$$

Note that we only consider weakly guarded expressions and therefore, this definition is sufficient (compare with Remark 4).

Since we do not have *persistency* like in [Wan90] our *Sort*-function can not be used to conclude from the current time to earlier times, more precisely if τ is not in *Sort* at time d, it is still possible that τ is in *Sort* at an earlier time. Therefore, we define a formula Φ, depending on an element of $\mathcal{T}$ and on a weakly guarded expression, which holds if only if there is an internal action enabled before the considered time.

$$\Phi(t, E) = \forall t' \in \mathcal{T}_0 \, : \, t' < t \Rightarrow \tau \notin \mathrm{Sort}(E)_{(t')}$$

The formula Φ is decidable since we are restricted on weakly guarded expressions and therefore, the *Sort*-function gets a step function[2] with finitely many steps.

The transition rules are presented in Table 1. The action of the action prefix

[2] A step function is constant between two successive steps.

$$\frac{}{\mathrm{NIL} \xrightarrow{t} \mathrm{NIL}}$$

$$\frac{0 \succ e \qquad 0 \prec' e'}{\alpha^{\prec' e'}_{\succ e}.E \xrightarrow{\alpha} E}$$

$$\frac{0 < e'}{\alpha^{\prec' e'}_{> 0}.E \xrightarrow{\iota} \hat{\alpha}^{\prec' e'}_{\geq 0}.E}$$

$$\frac{0 < e' \qquad (\alpha = \tau \Rightarrow 0 < e) \qquad (e = 0 \Rightarrow \succ = \geq)}{\alpha^{\prec' e'}_{\succ e}.E \xrightarrow{\iota} \alpha^{\prec' e'}_{\succ e}.E}$$

$$\frac{d \prec' e' \qquad d \leq e}{\alpha^{\prec' e'}_{\succ e}.E \xrightarrow{d} \check{\alpha}^{\prec' e'-d}_{\succ e-d}.E}$$

$$\frac{d \prec' e' \qquad \alpha \neq \tau \qquad d > e}{\alpha^{\prec' e'}_{\succ e}.E \xrightarrow{d} \check{\alpha}^{\prec' e'-d}_{\geq 0}.E}$$

$$\frac{E \xrightarrow{\alpha} E'}{E + F \xrightarrow{\alpha} E'}$$

$$\frac{E \xrightarrow{t} E'}{E + F \xrightarrow{t} E'} \; (|F|_{\mathcal{T}_c} < (t, \leq)) \wedge \Phi(t, F)$$

$$\frac{E \xrightarrow{\alpha} E'}{F + E \xrightarrow{\alpha} E'}$$

$$\frac{E \xrightarrow{t} E'}{F + E \xrightarrow{t} E'} \; (|F|_{\mathcal{T}_c} < (t, \leq)) \wedge \Phi(t, F)$$

$$\frac{E \xrightarrow{t} E' \qquad F \xrightarrow{t} F'}{E + F \xrightarrow{t} E' + F'}$$

$$\frac{E \xrightarrow{\alpha} E'}{E|F \xrightarrow{\alpha} E'|F}$$

$$\frac{E \xrightarrow{t} E' \qquad F \xrightarrow{t} F'}{E|F \xrightarrow{t} E'|F'} \; \Phi(t, E|F)$$

$$\frac{E \xrightarrow{\alpha} E'}{F|E \xrightarrow{\alpha} F|E'}$$

$$\frac{E \xrightarrow{\alpha_1} E' \qquad F \xrightarrow{\alpha_2} F' \qquad \check{\bar{\alpha}}_1 = \check{\alpha}_2 \qquad \alpha_i \neq \tau}{E|F \xrightarrow{\tau} E'|F'}$$

$$\frac{E \xrightarrow{\alpha} E' \qquad \check{\alpha} \notin \{a, \bar{a}\}}{E \backslash a \xrightarrow{\alpha} E' \backslash a}$$

$$\frac{E \xrightarrow{t} E'}{E \backslash a \xrightarrow{t} E' \backslash a}$$

$$\frac{E \xrightarrow{\alpha} E'}{E[f] \xrightarrow{f(\alpha)} E'[f]}$$

$$\frac{E \xrightarrow{t} E'}{E[f] \xrightarrow{t} E'[f]}$$

$$\frac{E[\mu X.E/X] \xrightarrow{\alpha} E'}{\mu X.E \xrightarrow{\alpha} E'}$$

$$\frac{E[\mu X.E/X] \xrightarrow{t} E'}{\mu X.E \xrightarrow{t} E'}$$

Table 1. Operational Semantics

process is allowed to perform if and only if the current time (time 0) is an element of the action's interval.

The first rule of time passing by action prefix considers the case of a non-observable time step when the action's interval is left open at zero. In this case the action gets enabled to produce an internal action. Therefore, we change its interval by including the current time, i.e. take the left closed interval at zero, and mark the action. The non-observable time step is only allowed when the upper time limit of the action's interval is not reached, i.e. $0 < e'$.

The second rule of time passing by action prefix considers the case of a non-observable time step when the action's interval is different from the first rule. In this case a non-observable time step is only allowed if the upper time limit of the action's interval is not reached and no internal action is enabled. The non-observable time step keeps the process unchanged. If we would mark the action also in this case, the process $\left(\tau^{\leq 1}_{>0}.b^{\leq 0}_{\geq 0}\right)\Big| a^{\leq 1}_{\geq 0}$ would not be able to perform the a-action after the b-action at the same time ($\left(\tau^{\leq 1}_{>0}.b^{\leq 0}_{\geq 0}\right)\Big| a^{\leq 1}_{\geq 0} \xrightarrow{\iota}\xrightarrow{\tau} b^{\leq 0}_{\geq 0}|\hat{a}^{\leq 1}_{\geq 0} \xrightarrow{b}\not\xrightarrow{a}$), which is in our sense counterintuitive.

The assumption, that a non-observable time step is not allowed if a process reached its upper time limit, is justified, since we expect that the process $b^{\leq 0}_{\geq 0}|\tau^{\leq 1}_{>0}$ is not allowed to perform the internal action.

Real time is allowed to pass by an action prefix process if no internal action gets earlier enabled and the upper time limit is not exceeded. The interval is moved by the amount of the passing time and the action gets unmarked, since if it was marked it is now in 'the middle of its interval' and therefore behaves normally. For example, we expect that the process $a^{\leq 3}_{>0}$ is able to perform the a-action after a non-observable time step followed by the passing of 1 real time ($a^{\leq 3}_{>0} \xrightarrow{\iota}\xrightarrow{1}\xrightarrow{a}$). We do not allow, for example, the process $a^{\leq 1}_{\geq 0}$ to delay 1 time unit, since otherwise the process $a^{\leq 1}_{\geq 0}|b^{\leq 1}_{\geq 1}$ gets enabled to perform the b-action. The two rules of real time passing by action prefix only differs in the handling of the lower bound of the interval.

If only one expression is allowed to idle to time t, we skip the other expression in the summation if it does not enable an internal action before time t. To determine the time till one expression is allowed to idle we use the function $|\ |_{\mathcal{T}_c}$. We consider the difference whether the time is included or excluded, since otherwise the process $a^{\leq 1}_{\geq 0} + b^{\leq 1}_{\geq 1}$ cannot delay 1 real time, because $a^{\leq 1}_{\geq 0}$ cannot delay 1 real time.

To perform an internal action by the parallel operator, we do not distinguish between marked and unmarked actions (the action $\bar{a}$ (and also $\hat{\bar{a}}$) can communicate with $\hat{a}$ and, of course, with a as well), since the process $a^{\leq 0}_{>0}|\bar{a}^{\leq 1}_{\geq 0}$ should perform an internal action after a non-observable time step ($\hat{a}^{\leq 1}_{\geq 0}|\bar{a}^{\leq 3}_{\geq 0} \xrightarrow{\tau}$).

The rest of the rules is standard. They are illustrated, for example, in [Wan90, Che92].

Remark 6. Since we only consider weakly guarded expressions it can be proved that the rules for recursion can be replaced by the following rules:

$$\frac{E \stackrel{\alpha}{\longrightarrow} E'}{\mu X.E \stackrel{\alpha}{\longrightarrow} E'[\mu X.E/X]} \qquad \frac{E \stackrel{t}{\longrightarrow} E'}{\mu X.E \stackrel{t}{\longrightarrow} E'[\mu X.E/X]}$$

These rules are often more friendly for structural induction.

Remark 7. We only consider weakly guarded expressions on the left hand side of $\rightarrow$, since an expression like $F := \mu X.\left(a^{\leq 3}_{\geq 2} + X \middle| \left(\bar{a}^{\leq 3}_{\geq 2}\right)\right)$ makes difficulties. We expect that after a delay of 2 time units F will perform an internal action. But if we want to delay F by the delay rules we get a circular reasoning. [Che92].

Remark 8. The marked actions are only used for the operational semantics. We are usually only interested in regular expressions. There is also no intuitive interpretation of processes which contain marked actions like $a^{\leq 2}_{\geq 0}.\hat{b}^{\leq 1}_{\geq 0}$. But the marked actions can be used to make an action at a specific time point available only for communication. For example, consider the process $E| \left(\hat{a}^{\leq 0}_{\geq 0} + \mathrm{NIL}\right)$. Here all $\bar{a}$-action which can be performed by E at time 0 can be translated to internal action without allowing the a-action explicitly.

2.3 Properties

In this subsection we consider some model properties [NS92] with respect to $(\mathcal{E}, \rightarrow)$. The proofs are omitted.

Time Determinism. The *time determinism* is satisfied if and only if the resulting process by time passing is uniquely determined.

Proposition 9. $(\mathcal{E}, \rightarrow)$ *is* time deterministic, *i.e.*

$$\forall E, E', E'' \in \mathcal{E}, t \in \mathcal{T} \,:\, \left(E \stackrel{t}{\longrightarrow} E' \wedge E \stackrel{t}{\longrightarrow} E''\right) \Rightarrow E' = E''$$

Time Additivity. The *time additivity* is satisfied if and only if: a process E can pass $t_1 + t_2$ time to the process E' iff E can pass t_1 time to a process F and F can pass t_2 time to the process E'.

Since we do not have an addition operator on the whole time domain we weaken the time additivity property to the following results:

Proposition 10. *Let* $E, E', E'' \in \mathcal{E}$ *and* $d, d' \in \mathbb{R}^+$ *then*

1. $\left(\exists F \in \mathcal{E} \,:\, E \stackrel{d}{\longrightarrow} F \wedge F \stackrel{d'}{\longrightarrow} E'\right) \iff E \stackrel{d+d'}{\longrightarrow} E'$
2. $\left(\exists F \in \mathcal{E} \,:\, E \stackrel{\iota}{\longrightarrow} F \wedge F \stackrel{d}{\longrightarrow} E'\right) \iff E \stackrel{d}{\longrightarrow} E'$
3. $E \stackrel{\iota}{\longrightarrow} E'' \wedge E'' \stackrel{\iota}{\longrightarrow} E' \Longrightarrow E \stackrel{\iota}{\longrightarrow} E'$

The implication from right to left in the 3. case of Proposition 10 does not hold because of the maximal progress assumption, for example, $\tau^{\leq 3}_{>0}$ only allows one non-observable time step.

Proposition 10 shows that no further state explosion is caused by introducing non-observable time steps.

Deadlock. A deadlock is a process which neither is allowed to pass time nor to perform any action. Deadlocks are expressible in $(\mathcal{E}, \rightarrow)$, for example by the process, $a^{\leq 0}_{\geq 1}$.

Action Urgency. Processes which *must* perform an action without letting time to pass, i.e. $\exists E, E' \in \mathcal{E}, c \in \Lambda : E \xrightarrow{c} E' \wedge \forall t \in \mathcal{T}, E'' \in \mathcal{E} : E \overset{t}{\not\longrightarrow} E''$, exists in $(\mathcal{E}, \rightarrow)$. A typical example is the process $a^{\leq 0}_{\geq 0}$.

Maximal Progress. It is the property that an action must occur at the instant at which all participants are ready, hence time must not proceed if an internal action is enabled.

Proposition 11. $(\mathcal{E}, \rightarrow)$ *satisfies the* maximal progress, *i.e.*

$$\forall E \in \mathcal{E}, t \in \mathcal{T} : E \xrightarrow{t} \;\Rightarrow E \overset{\tau}{\not\longrightarrow} \quad \wedge \; \forall\, t' < t : E \xrightarrow{t'} \overset{\tau}{\not\longrightarrow}$$

Persistency. *Persistency* is the property that the progress of time cannot suppress the ability to perform an action. Formally, $\forall E, E', E'' \in \mathcal{E}, \alpha \in \mathrm{Act}, t \in \mathcal{T} : E \xrightarrow{\alpha} E' \wedge E \xrightarrow{t} E'' \Rightarrow \exists F \in \mathcal{E} : E'' \xrightarrow{\alpha} F$. Since summation makes a choice by time, $(\mathcal{E}, \rightarrow)$ does not satisfy the persistency property. The process $a^{\leq 0}_{\geq 0} + \mathrm{NIL}$ denotes a counterexample.

Finite Variability. If a process can perform only finitely many actions in a finite time interval, it has the *finite variability* property. $(\mathcal{E}, \rightarrow)$ fails to have this property as the process $\mu X. a^{\leq 0}_{\geq 0}.X$ illustrates.

Bounded Control. Let us denote the actions which the process E can immediately perform by the initial actions of E, formally, $init(E) = \{\alpha \in \mathrm{Act} \mid \exists E' : E \xrightarrow{\alpha} E'\}$. The *bounded control* property holds if and only if for any delay t, there is a bounded number of changes of the initial actions and it is possible to find a global clock period that allows to handle these changes. Formally $\exists \tilde{d} \in \mathbb{R}^+ : \forall E, E' \in \mathcal{E}, t \in \mathcal{T} : \left(E \xrightarrow{t} E' \wedge init(E) \neq init(E)\right) \Rightarrow t \geq \tilde{d}$. Note that we only allow $\tilde{d}$ to be a real number. The reason of this restriction is that the real numbers cannot be divided by non-observable time steps.

The bounded control property is not satisfied by $(\mathcal{E}, \rightarrow)$, since we consider real time with instantaneous actions, so $a^{\leq \tilde{d}}_{\geq \tilde{d}/2}$ denotes a counterexample. Furthermore, the non-observable time step also yields counterexamples, like $a^{\leq 2}_{>0}$.

3 Example

Let us consider a plane which regularly sends an *ok* signal to the tower confirming that everything is all right. If the last received *ok* signal is more than 5 minutes ago, the safety officer of the tower is allowed to send off a rescue team. If an *ok* signal arrives before the rescue team is sent off, the safety officer is no longer allowed to send them off until another last received *ok* signal is more than 5 minutes ago. The safety officer can be modeled by $E_{s-officer}$ through the following example, where the plane is represented by E_p, the activity of the rescue team by the F process and the sending of the rescue team by the *send*-action.

Example 12. Let

$$E_{s-officer} = \left(\left(\mu X. \left(ok^{<\infty}_{\geq 0}.X + send^{<\infty}_{>5}.F\right)\right) |E_p\right) \setminus \{ok\}$$

where E_p be a process which sometimes send an *ok*-signal (i.e. perform an $\overline{ok}$-action). The process $E_{s-officer}$ can be considered as a controller of E_p, who may react with a *send*-action whenever he does not get an *ok*-action within the last 5 time units.

To model such a kind of problem we need both properties, the maximal progress assumption and the action occurrences in open intervals. The maximal progress assumption is needed here, because otherwise the *send*-action could stay enabled even if an $\overline{ok}$-action was enabled by E_p. The open intervals are used to ensure at time 5 that no *send*-action gets enabled if an $\overline{ok}$-action is enabled.

Process algebras with time and priority can also model this example, except that the *send*-action gets also enabled at time 5, which is not problematic in practice. The advantage of Open Timed CCS is that it needs no priority constructions to model this problem, hence it is more user friendly, since in reality a lot of problems are described by 'after' or similar words. Another advantages of Open Timed CCS is that such kind of problems are modelled more intuitively, because by the priority approach you get interpretation problems, if $\overline{ok}$-actions with different priorities will be used.

In Open Timed CCS the maximal progress assumption at open intervals comes into play, if the controller of the process $E_{s-officier}$ from Example 12 is replaced by a machine, in the sense that it always will perform *send* if it is enabled. This is done in the following process.

$$E_{auto-s-officier} = \left(\left(\overline{send}^{<\infty}_{\geq 0}.\mathrm{NIL}\right) |E_{s-officier}\right) \setminus \{send\}$$

4 Equivalences

In this section we introduce different kinds of equivalences which are based on bisimulation:

1. *strong $\mathcal{T}$-equivalence* which considers the marked actions as adequate actions,
2. *priority $\mathcal{T}$-equivalence* which neglects marked actions,
3. *halfway $\mathcal{T}$-equivalence* which neglects marked actions and abstracts away of non-observable time steps and
4. *weak $\mathcal{T}$-equivalence* which neglects marked actions and abstracts away of non-observable time steps and internal actions.

There exists of course much more equivalences like abstracting away of time [LW97], but since they can defined straightforward, we omit them.

To get a more uniformed definition of these equivalences we define what a *bisimulation* over a transition system is and what its corresponding equivalence is.

Definition 13. *Consider the transition system $(\mathcal{M}, \mapsto)$, where $\mapsto \subset \mathcal{M} \times \mathcal{L} \times \mathcal{M}$. A* bisimulation *over the transition system $(\mathcal{M}, \mapsto)$ is a binary relation $\mathcal{R}$ over $\mathcal{M}$ such that if $(P, Q) \in \mathcal{R}$ implies that for any $\ell \in \mathcal{L}$*

1. *if $P \overset{\ell}{\mapsto} P'$, then there is a $Q' \in \mathcal{M}$ such that $Q \overset{\ell}{\mapsto} Q'$ and $(P', Q') \in \mathcal{R}$;*
2. *if $Q \overset{\ell}{\mapsto} Q'$, then there is a $P' \in \mathcal{M}$ such that $P \overset{\ell}{\mapsto} P'$ and $(P', Q') \in \mathcal{R}$.*

We say that P, Q are bisumulation-equivalent *with respect to $(\mathcal{M}, \mapsto)$, if and only if there is a bisimulation $\mathcal{R}$ over* $(\mathrm{M}, \mapsto)$ *such that $(P, Q) \in \mathcal{R}$.*

It can be proved, that the bisumulation-equivalent relation is indeed an equivalent relation [Mil89].

4.1 Strong $\mathcal{T}$-Equivalence

Definition 14. *Suppose E and F are closed and weakly guarded. We say that E and F are* strong $\mathcal{T}$-equivalent*, denoted $E \sim F$, if they are bisumulation-equivalent with respect to $(\mathcal{E}, \rightarrow)$.*

Strong $\mathcal{T}$-equivalence is a congruence and it is used if the observer is allowed to combine the processes with others (context observation).

4.2 Priority $\mathcal{T}$-Equivalence

Definition 15. *Let the transition system $(\mathcal{E}, \rightharpoonup)$ be defined by $\rightharpoonup := \rightarrow \backslash (\mathcal{G} \times \widehat{\Lambda} \times \mathcal{E})$.*

Suppose E and F are closed and weakly guarded. We say that E and F are priority $\mathcal{T}$-equivalent*, denoted $E \sim_p F$, if they are bisumulation-equivalent with respect to $(\mathcal{E}, \rightharpoonup)$.*

Priority $\mathcal{T}$-equivalence neglects marked action, but it still reflects the priority information (compare with Section 5), since it does not abstract away of non-observable time steps.

The priority $\mathcal{T}$-equivalence fails to be a congruence, for example $\tau_{>0}^{\leq 1} + a_{>0}^{\leq 1}.b_{\geq 0}^{\leq 0}$ and $\tau_{>0}^{\leq 1}$ are priority $\mathcal{T}$-equivalent. But if we use them in parallel with $\bar{a}_{>0}^{\leq 1}$, they will behave differently, since the former may perform a b-action.

4.3 Halfway $\mathcal{T}$-Equivalence

Definition 16. *Let the transition system* $(\mathcal{E}, \rightharpoondown)$ *be defined by*

$$
\begin{aligned}
\stackrel{\iota}{\rightharpoondown} &:= (\stackrel{\iota}{\rightarrow})^* \\
\stackrel{\alpha}{\rightsquigarrow} &:= \stackrel{\iota}{\rightharpoondown} \circ \stackrel{\alpha}{\rightharpoonup} \circ \stackrel{\iota}{\rightharpoondown} \\
\stackrel{d}{\rightsquigarrow} &:= \bigcup_{d=\sum_{i=1}^{n} d_i} \stackrel{\iota}{\rightharpoondown} \circ \xrightarrow{d_1} \circ \stackrel{\iota}{\rightharpoondown} \cdots \stackrel{\iota}{\rightharpoondown} \circ \xrightarrow{d_n} \circ \stackrel{\iota}{\rightharpoondown}
\end{aligned}
$$

where $\alpha \in \Lambda \cup \{\tau\}$ *and* $d, d_i \in \mathbb{R}^+$.

Suppose E and F are closed and weakly guarded. We say that E and F are halfway $\mathcal{T}$-equivalent, *denoted* $E \asymp F$, *if they are bisumulation-equivalent with respect to* $(\mathcal{E}, \rightharpoondown)$.

By this equivalence only real time passing and action occurrences (unmarked and internal) are observable. This is relevant if one is not interested in priority property, but can detect that the computer is working (perform an internal action). It fails to be a congruence as the priority $\mathcal{T}$-equivalence.

4.4 Weak $\mathcal{T}$-Equivalence

Definition 17. *Let the transition system* $(\mathcal{E}, \rightsquigarrow)$ *be defined by*

$$
\begin{aligned}
\stackrel{\tau}{\rightsquigarrow} &:= (\stackrel{\tau}{\rightarrow} \cup \stackrel{\iota}{\rightarrow})^* \\
\stackrel{\alpha}{\rightsquigarrow} &:= \stackrel{\tau}{\rightsquigarrow} \circ \stackrel{\alpha}{\rightharpoonup} \circ \stackrel{\tau}{\rightsquigarrow} \\
\stackrel{d}{\rightsquigarrow} &:= \bigcup_{d=\sum_{i=1}^{n} d_i} \stackrel{\tau}{\rightsquigarrow} \circ \xrightarrow{d_1} \circ \stackrel{\tau}{\rightsquigarrow} \cdots \stackrel{\tau}{\rightsquigarrow} \circ \xrightarrow{d_n} \circ \stackrel{\tau}{\rightsquigarrow}
\end{aligned}
$$

where $\alpha \in \Lambda$ *and* $d, d_i \in \mathbb{R}^+$.

Suppose E and F are closed and weakly guarded. We say that E and F are weak $\mathcal{T}$-equivalent, *denoted* $E \approx F$, *if they are bisumulation-equivalent with respect to* $(\mathcal{E}, \rightsquigarrow)$.

Weak $\mathcal{T}$-equivalence is of course no congruence, since the weak bisimulation of Milner [Mil89] is none and the example from Subsection 4.2 still holds.

Remark 18. The approach by defining a new transition relation and then using bisimulation-equivalence corresponds to the usual definition of weak bisimulation as it is used for example in [Mil89, LW97].

Remark 19. From the universal algebra theory, we know that for every equivalence relation there exists a maximal congruence relation contained in it. Hence, there exists a maximal congruence relation for the priority $\mathcal{T}$-equivalence, for the halfway $\mathcal{T}$-equivalence and for the weak $\mathcal{T}$-equivalence.

5 Priority in Open Timed CCS

One possibility to model priority in process algebra is to extend the syntax by attaching an additional natural number to each action [CH90, CLN99]. The natural numbers determine the priority of actions. In those process algebras actions may only perform if no internal action with a higher priority is enabled. This approach is called *static priority*, since the priorities keep unchanged during the system execution.

Another possibility to introduce priority to process algebras is to make use of the maximal progress assumption [BCL99]. Here an action has higher priority if it is enabled earlier and the maximal progress assumption ensures that an action only performs if no internal action with a higher priority is enabled, i.e. no internal action has been earlier enabled. This approach is a *dynamic priority* approach, since the priorities get changed during the running time. But by this approach it is not possible to model priority at the same point in time if $\mathbb{R}^+$ is the considered time domain.

Open Timed CCS allows to model dynamic priority at the same point in time. The idea is to put a $\tau_{>d}^{\leq d+1}$ expression in front of an action with lower priority, where d is determined by the left hand side of the action's interval. It is also necessary to adjust the interval of the action respectively. The more such internal actions are put in front of an action, the less is the priority of this action. For example, the process $a_{\geq 2}^{\leq 7}.\mathrm{NIL}$, where a shall have priority -2 will be translated to the process $\tau_{>2}^{\leq 3}.\tau_{>0}^{\leq 1}.a_{\geq 0}^{\leq 5}.\mathrm{NIL}$. Since in Open Timed CCS internal actions stipulate a choice in the summation operator, we have to copy each process of higher priority behind each introduced internal action. The following example demonstrates the translation.

Example 20. Consider the the process $a_{\geq 2}^{\leq 7}.E + b_{\geq 2}^{\leq 6}.F$, where a has priority 0, which is the highest priority, and b has priority -2. Then the process

$$a_{\geq 2}^{\leq 7}.E + \tau_{>2}^{\leq 3}.\left(a_{\geq 0}^{\leq 5}.E + \tau_{>0}^{\leq 1}.\left(a_{\geq 0}^{\leq 5}.E + b_{\geq 0}^{\leq 4}.F\right)\right)$$

has the desired behaviour.

In [Jef91], the author translates a timed process algebra into a prioritized process algebra, but he only consider static priority[3] since in his framework a process may either *immediately* engage in an action or idle forever. He uses arbitrary summations to get the expressive power of his algebra.

6 Conclusion

We have proposed the process algebra Open Timed CCS, which is a real time extension of CCS. The algebra satisfies the maximal progress assumption and allows actions to perform in open intervals. A possibility to model a priority at the

[3] in contrary to our approach

same point in time has been introduced, and equivalences based on bisimulation techniques have been presented.

A future work will be to establish equational laws of the considered bisimulation equivalences. Another work will be to the extension of other process algebras like [Wan90, Sch00, CL96, LL97] by our ideas.

In [GL92] it has been argued that process algebras like Open Timed CCS fail to satisfy an expansion theorem, therefore it is of interest to extend Open Timed CCS by time variables as in [Wan91]. If we want to use this extension, we first have to decide decide whether a non-observable time step should influence the later behaviour or not. For example, should the process $\tau(x)^{\leq 1}_{>0}.a^{\leq 0-x}_{\geq 0-x}$ be allowed to perform an a-action or not? If no non-observable time step should influence the later behaviour, the extension by time variables seems to be straightforward. Otherwise, a different time domain has to be chosen. The non-standard reals [Rob70] might help, but then reasoning becomes more complicated. The extension of Open Timed CCS by time variables is also of interest to get at least the expressive power of timed automata [AD94].

References

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[BB91] J C M Baeten and J A Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[BB93] J.C.M. Baeten and J.A. Bergstra. Real time process algebra with infinitesimals. Technical Report P9325, University of Amsterdam, Programming Research Group, 1993.

[BCL99] Girish Bhat, Rance Cleaveland, and Gerald Lüttgen. A practical approach to implementing real-time semantics. *Annals of Software Engineering*, 7:127–155, 1999.

[CH90] R. Cleaveland and M. Hennessy. Priorities in process algebras. *Information and Computation*, 87(1/2):58–77, 1990.

[Che92] L. Chen. An interleaving model for real–time systems. *LNCS*, 620:81–92, 1992.

[CL96] R. Cleaveland and G. Luettgen. An algebraic theory of distributed real-time. Technical Report TR-96-04, Department of Computer Science, North Carolina State University, 1996.

[CLN99] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebras. Technical Report TR-99-3, Institute for Computer Applications in Science and Engineering, 1999.

[Dan91] M. Daniels. Modelling real–time behaviour with an interval time calculus. *LNCS*, 571:53–72, 1991.

[FZ94] C. Fidge and J. Zic. A simple, expressive real-time CCS. Technical Report 94-27, Software Verification Reseach Centre, 1994.

[GL92] J. C. Godskesen and K. G. Larsen. Real–time calculi and expansion theorems. *LNCS*, 652:302–315, 1992.

[HR95] M. Hennessy and T. Regan. A process algebra for timed systems. *Information and Computation*, 117(2):221–239, 1995.

[Jef91] A. Jeffrey. Translating timed process algebra into prioritized process algebra. *LNCS*, 571:493–506, 1991.

[Jef92] A. Jeffrey. A linear time process algebra. *LNCS*, 575:432–442, 1992.

[Klu91] A. S. Klusener. Completeness in real time process algebra (extended abstract). *LNCS*, 527:376–392, 1991.

[LL97] L. Léonard and G. Leduc. An introduction to ET-LOTOS for the description of time-sensitive systems. *Computer Networks and ISDN Systems*, 29(3):271–292, 1997.

[LW97] K. G. Larsen and Y. Wang. Time-abstracted bisimulation: Implicit specifications and decidability. *Information and Computation*, 134(2):75–101, 1997.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. *LNCS*, 458:401–415, 1990.

[Mur92] D. Murphy. Intervals and actions in a timed process algebra. Arbeitspapiere 680, Gesellschaft für Mathematik und Datenverarbeitung (GMD), 1992.

[NS92] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. *LNCS*, 600:526–548, 1992.

[Rob70] A. Robinson. *Non-Standard Analysis.* Noth-Holland, 1970.

[RR88] G. M. Reed and A. W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science*, 58:249–261, 1988.

[Sch00] S. Schneider. *Concurrent and Real-time Systems: The CSP Approach.* Wiley, 2000.

[Wan90] Y. Wang. Real-time behaviour of asynchronous agents. *LNCS*, 458:502–520, 1990.

[Wan91] Y. Wang. CCS+time = an interleaving model for real time systems. *LNCS*, 510:217–228, 1991.