

## Completeness of Timed $\mu$ CRL

**M. A. Reniers and J. F. Groote<sup>†\*</sup>**

*TU/e, Eindhoven, The Netherlands*

*M.A.Reniers@tue.nl; J.W.Groote@tue.nl*

**M.B. van der Zwaag and J. van Wamel<sup>†</sup>**

*CWI, Amsterdam, The Netherlands*

*mbz@cwi.nl; jos@cwi.nl*

---

**Abstract.** In [25] a straightforward extension of the process algebra  $\mu$ CRL was proposed to explicitly deal with time. The process algebra  $\mu$ CRL has been especially designed to deal with data in a process algebraic context. Using the features for data, only a minor extension of the language was needed to obtain a very expressive variant of time. But [25] contains syntax, operational semantics and axioms characterising timed  $\mu$ CRL. It did not contain an in depth analysis of theory of timed  $\mu$ CRL. This paper fills this gap, by providing soundness and completeness results. The main tool to establish these is a mapping of timed to untimed  $\mu$ CRL and employing the completeness results obtained for untimed  $\mu$ CRL.

**Keywords:** real-time, process algebra, completeness,  $\mu$ CRL

## 1. Introduction

Process algebras are very nice tools to study fundamental concepts such as actions and interactions, nondeterminism and parallelism, behavioural equivalences and internal or hidden actions. In their plain form (e.g. CCS [59], CSP [46] or ACP [12, 11]), these languages are not very expressive in the sense that only very simple protocols and distributed systems can properly be described. This is the reason why these plain process algebras have been extended with data ([41, 32]). The most expressive [56, 57] and by far the most developed is  $\mu$ CRL (*micro Common Representation Language*) [32, 33], a process

---

\*The authors thank all persons who assisted in developing  $\mu$ CRL throughout the years. For timed  $\mu$ CRL they have benefited a lot from comments from Jos Baeten, Jan Bergstra, Bas Luttik, Radu Mateescu, Alban Ponse, Jan Springintveld, Jan-Joris Vereijken and Tim Willemse.

<sup>†</sup>Address for correspondence: HG 6.76, TU/e, Postbox 513, NL-5600 MB Eindhoven, The Netherlands

algebra in the style of ACP extended with equational abstract datatypes. The main design objectives for  $\mu$ CRL were that

- $\mu$ CRL had to be sufficiently expressive to describe ‘real-life systems’, generally consisting of a set of interacting programs.
- $\mu$ CRL had to be simple and sufficiently clear to form a suitable basis for mathematical analysis.
- The definition of  $\mu$ CRL had to be precise, in order to allow the independent construction of computer tools for assisting in the actual development of systems.

The process algebra  $\mu$ CRL is the basis of several proof methodologies [15, 31, 35, 36, 34, 28, 33, 39] and numerous verifications (see [14, 26, 52, 30, 13, 24, 29, 49, 66]), some of which have been computer checked. It has also been the basis for several fundamental studies about for instance expressiveness and decidability [62, 56, 57], and visualisation of huge state spaces [40]. Finally, a tool set around  $\mu$ CRL (see [19, 51]) has been constructed which is not based on finite state spaces, but on linear process operators [15, 16], allowing automatic treatment of processes with huge or infinite state spaces.

The language  $\mu$ CRL primarily describes the potential ordering of actions that processes can perform, which is called the behaviour of a process. It is easy to express that an action  $a$  must happen before an action  $b$ , and that action  $c$  happens in parallel with these. There is, however, no possibility to explicitly refer to time. E.g.  $\mu$ CRL cannot express that an action  $a$  must be executed at a certain time, an action  $b$  must happen within five seconds after another action  $a$ , or that if nothing happens for 10 seconds, an *alarm* action must be issued.

However, time is explicitly used in almost any computerised system. Computers rely on notions such as time-outs, scheduling and interrupts. Hardware and software clocks are common, and so are repeating time-based tasks. Real-time systems, which are required to perform certain tasks within predetermined time intervals, even add another dimension. For such systems time is not only important for the internal behaviour, but also when it comes to the interaction with the environment. Therefore, we have extended  $\mu$ CRL to handle time. The major design considerations for the extension with time are the following:

- A specification in timed  $\mu$ CRL that makes no reference to time should look exactly the same as an untimed  $\mu$ CRL specification. Moreover, the intuitive meaning of such a specification should be equal in both languages. Formally, the two semantics of this specification differ, as the semantics of timed  $\mu$ CRL explicitly refers to time, whereas the semantics of classical  $\mu$ CRL does not mention time at all. In other words, timed  $\mu$ CRL must be a conservative extension of untimed  $\mu$ CRL. One advantage thereof is that given a tool for manipulating  $\mu$ CRL<sub>*t*</sub> specifications, this tool can also be used for untimed  $\mu$ CRL.
- The extension with time should be natural and concise, fitting in the style of description of  $\mu$ CRL.
- The definition of timed  $\mu$ CRL should be such that it is suitable to adapt many of the existing proof techniques from the untimed setting to the timed setting.

In the past years, many timed process algebras have been developed [3, 42, 60, 61, 63] for different purposes. In extending a process algebra with time choices have to be made with respect to the nature of the time domain, the way time is described syntactically, and the way time is incorporated semantically [9].

**Nature of the time domain** Process algebras that are proposed in the literature can, with respect to the nature of the time domain, be classified as discrete-time process algebras (e.g. [60, 63, 61, 42, 5]), dense-time process algebras (e.g. [64, 3]), and process algebras with an unspecified time domain (e.g. [69, 18, 55]). As in untimed  $\mu$ CRL the user of the language is allowed to specify algebraic data types, no need was felt for restricting the language to a specific time domain. In timed  $\mu$ CRL, the time domain is specified by the user, regardless of whether it is a discrete or a dense time domain. The only restrictions on the time domain (denoted  $T$ ) are that a total ordering (denoted  $\leq$ ) and a least element with respect to this ordering (denoted  $0$ ) must be provided.

**Time-stamping and delay mechanisms** Although many different constructions can be found in the literature for denoting timing aspects, coarsely the following dichotomy is found: either time is denoted through a time-stamping mechanism ([3, 63, 6]), or through a delay mechanism ([60, 61, 42, 5, 7]). Usually a time-stamping mechanism is used in process algebras with delayable atomic actions for the purpose of restricting the uncontrolled timing of such atomic actions, and delay mechanisms are employed in process algebras with nondelayable atomic actions for describing the passage of time (explicitly). Following the first design consideration, the meaning of the untimed process  $p$  in the timed setting should be the execution of the atomic actions of  $p$  at arbitrary times while maintaining their ordering, the only alternative being the execution of all behaviour of an untimed process at one specific time. Hence, timed  $\mu$ CRL has a time-stamping mechanism for describing timing aspects. In contrast to most process algebras with a time-stamping mechanism the time-stamping is not performed directly on the atomic actions, but rather on complete processes. The process  $p \cdot t$ , with  $p$  a process and  $t$  an element of the time domain, denotes the process  $p$  with the restriction that the first action of  $p$  must be executed at time  $t$ . In combination with the other operators of timed  $\mu$ CRL, timing aspects such as time-outs, deadlines and intervals can be expressed.

**Absolute time and relative time** Sometimes, when specifying real-time systems, it is convenient to describe the passage of time with respect to a global clock. At other times, it is convenient to describe passage of timing relative to the previous action. The first mechanism is called absolute timing ([20, 63]), the second relative timing (e.g. [60, 69, 48]). For timed  $\mu$ CRL, the choice was made for an absolute time approach for the following reasons. Firstly, references to absolute time are found in the systems that we want to describe. Secondly, the axioms for the parallel composition operator are easier in a setting with absolute-time than in a setting with relative-time. Thirdly, although relative time is convenient for the description of time with respect to sequential composition, with respect to timing constraints between actions in a parallel context, a relative timing mechanism is much more difficult to use. Finally, descriptions with relative and absolute notions of time can be converted to each other, as has been shown in [5, 6, 7]. In these papers process algebras have been studied that combine absolute timing and relative timing in one theory. These process algebras are called parametric time process algebras.

**Urgent actions and multi-actions** Timed  $\mu$ CRL should be as suitable for analysis and verification purposes as  $\mu$ CRL turned out to be. We did a number of exercises with timed verifications, which has resulted in allowing simultaneous, subsequent actions. For instance, it is allowed to write  $a \cdot 1 \cdot b \cdot 1$ , which means that  $a$  takes place at time 1, followed by  $b$  at time 1. Here we clearly sacrificed naturality of the language in favour of ease of use. Similar decisions can be found in e.g. [50, 2, 4, 5, 45] and in most

discrete-time process algebras. The most important reason to allow simultaneity is the elimination of the parallel operator in terms of the form  $a \cdot 1 \parallel b \cdot 1$ . Using linear process operators [15], it is generally possible to eliminate the parallel operator without exponential blowup in this way. One of the alternatives that have been investigated, is the use of multi-actions, e.g.  $\langle a \cdot b \rangle \cdot 1$  [3], but this gives rise to an exponential number of multi-actions when expanding  $n$  parallel processes.

In a setting with a minimal element  $\mathbf{0}$  of sort  $T$ , this design decision has consequences for the *left merge* operator  $\parallel$ . A process  $p \parallel \delta \cdot \mathbf{0}$  ( $\delta \cdot \mathbf{0}$  denotes deadlock at time zero) is not simply equal to  $\delta \cdot \mathbf{0}$ , but it can perform that part of  $p$  that is enabled at  $\mathbf{0}$ . This phenomenon is called  $\parallel$ -*leaking* (see [37] for more details).

**Expressiveness** The first serious exercises in timed  $\mu\text{CRL}$  appeared in a recent paper [37]. In that paper various basic results were derived, such as theorems for *basic terms*, the expansion of terms with operators for parallelism, elimination of parallelism, and commutativity of the merge and communication merge (the operators  $\parallel$  and  $|$ ). After that, a paper with three case studies of simple hybrid systems in  $\mu\text{CRL}_t$  [38] and a paper containing the analysis of a conveyor belt system [70] appeared.

Numerous case studies in which time is not involved have indicated that untimed  $\mu\text{CRL}$  has the appropriate expressiveness for the specification and verification of untimed systems. In the context of ACP-like process algebras, embeddings and transformations of timed process algebras in which different choices have been made have been studied extensively. These comparisons have not indicated differences in expressiveness of the different process algebras with respect to those choices.

Recently, a timed variant of the *cones and foci* verification technique of [35] has been published [74].

**Structure of this article** The definition of timed  $\mu\text{CRL}$  has been guided by simplicity, elegance and its suitability to adapt many of the existing proof technologies to the timed setting. In [25] the language was explained, syntax and operational semantics were given and an equational characterisation of the language was provided. There were no completeness results in the paper and a claim of soundness turned out to be not completely justified. This paper fills these gaps by providing soundness and completeness theorems, relative to some completeness assumptions on the datatypes. For simplicity and generality, the datatypes are assumed to be given as algebras satisfying a number of elementary properties, contrary to the definition in [25], where equational, inductive datatypes were employed.

The proof of completeness follows the approach of [73], where a timed  $\mu\text{CRL}$  is translated to untimed  $\mu\text{CRL}$ , in such a way that the completeness results of [27, 57] can directly be applied to achieve completeness. Besides providing a completeness result in a relatively easy way, there is an additional advantage of this approach, which may turn out to be very helpful in the analysis of timed systems. As mentioned above, there are many analysis techniques and a very capable tool for untimed  $\mu\text{CRL}$ . Transferring these to the timed setting may be a costly and hard operation. However, if we translate timed  $\mu\text{CRL}$  expressions to untimed  $\mu\text{CRL}$ , everything achieved for untimed  $\mu\text{CRL}$  carries over automatically. We have especially high hopes for the tool set, as the translation can be done automatically in this case.

**Future work** We also hope that this approach may help to solve one of the longer standing open problems in timed process algebra, namely a complete axiomatisation of branching or weak bisimulation in a setting as expressive as timed  $\mu\text{CRL}$ . There are numerous completeness results for weak bisimulations

in timed settings with restricted expressivity (e.g. [23]). The only attempt that we know where the expressiveness of the language comes close to that of  $\mu\text{CRL}$  can be found in [50]. Here, an axiom for branching bisimulation is given, which is claimed to be sound and complete. Although we do not doubt soundness, the completeness claim has not been appropriately justified, and turns out not to be easily reconstructable.

**Related work** Timed process algebras have received a lot of attention in the literature, and many different formalisms have been proposed. Many of these have been mentioned in this introduction (see for a more structured overview [68, 8]). Besides the timed process algebras also other formalisms can be found in the literature that allow for the description of timing aspects. We mention timed automata [2] and hybrid automata [1, 58, 43].

The typical difference with timed  $\mu\text{CRL}$  is that timed automata are a semantical concept, much less algebraical, and as such there is no set of characteristic axioms or even a precisely defined set of operators to construct these. Timed automata are in a sense just finite state machines extended with clocks. These formalism sparked off a lot of research for instance in timed model checking and equivalence checking (using simulation relations). Using a technique known as convex regions, it is possible to capture the infinite flow of time into a finite automaton. This result has been the basis for various tool sets, such as HyTech [44], Uppaal [54] and Kronos [72], of which Uppaal allows simple datatypes as natural numbers and lists. A lot of effort has been put in the study of the relations between such automata and process algebras. Notably are the translation of timed automata to the process algebra ACP with prefix integration from [22], the use of timed automata as a model for a process algebra from [21], and the translations of timed and hybrid automata to a variant of  $\mu\text{CRL}_t$  from [71].

At a different end of the spectrum there are the timed specification languages of which extended LOTOS [17] and LOTOS NT [67] are among the most ambitious, although older languages such as SDL [47] also combine time within an expressive context. The goal of these languages is to allow to express distributed timed systems as elegantly as possible. As such, the languages are rich in syntax and much of the effort around these languages is in building tools around it. The difference with timed  $\mu\text{CRL}$  is that these languages are not apt, and not intended for more fundamental research.

## 2. The axiom system $p\text{CRL}_t$

The axiom system  $p\text{CRL}_t$  for *pico* CRL with time is presented. It serves as the basic framework for our studies. We work in a setting without the silent step  $\tau$ , and without abstraction or general operators for renaming. The addition of operators for parallelism in Section 5 leads to full  $\mu\text{CRL}_t$ . We define a notion of *basic terms* and prove that all terms over the signature  $\Sigma(p\text{CRL}_t)$  without process variables are derivably equal to basic terms.

### 2.1. Abstract data types

The processes described in the language  $\mu\text{CRL}$  generally exchange data. For the specification of data we use (equational) abstract data types with an explicit distinction between constructor functions and ‘normal’ functions. Moreover, all properties of a data type must be explicitly declared, which makes it clear which assumptions can be used for proving properties of data or processes.

In this paper, we do not treat the syntactical details of the data language in depth. We simply assume the existence of a data signature. A data signature consists of a set  $S$  of sort symbols and a set  $F$  of function declarations. We assume disjoint infinite sets of variables  $V_s$  for the sort symbols  $s \in S$ . Let  $V = \bigcup_{s \in S} V_s$ . The set of terms of sort  $s$  is denoted by  $T_s$ . Furthermore, we assume the existence of a sort symbol  $B$  for the booleans with function declarations  $t$  and  $f$ , and the usual connectives  $\neg$ ,  $\wedge$ , and  $\vee$ . For each sort symbol  $s$ , we assume a data algebra with universe  $\mathcal{D}_s$ . The set  $\mathcal{D}_B$  has two elements: the interpretation of  $t$  and the interpretation of  $f$ .

Finally, we assume the existence of a sort symbol  $T$  of time elements and we require that it is totally ordered and that it contains a least element. The total order is denoted  $\leq$  and the least element is denoted  $\mathbf{0}$ . Throughout this paper we abbreviate terms  $\neg(u \leq t)$  to  $t < u$ . A term of the form  $t[e/d]$  denotes the term  $t$  with variable  $d$  replaced by term  $e$ .

Besides the above requirements one is free to specify the kind of time domain one requires. This is done to accommodate those preferring discrete time, and others who prefer a notion of dense time. One can for instance define that  $\mathcal{D}_T$  has only a finite number of elements, or one can define an ordinal-like structure on it.

## 2.2. The syntax of $p\text{CRL}_t$

The signature of the theory  $p\text{CRL}_t$  consists of a data signature and a process signature. The process signature consists of the sort symbol  $P$  and the following function declarations:

1. *action declarations*  $\mathbf{a} : s_1 \times \cdots \times s_n \rightarrow P$  for sort symbols  $s_i$  ( $i = 0, \dots, n$ ) from the data signature.
2. *deadlock*  $\delta : \rightarrow P$ . Timed  $\mu\text{CRL}$  contains a constant  $\delta$ , which can be used to express that from now on, no action can be performed any more. It models inaction, for instance in the case where a number of computers are waiting for each other, and the whole system is blocked.
3. *alternative composition*  $\_ + \_ : P \times P \rightarrow P$ . The process represented by  $p + q$  behaves like  $p$  or  $q$ , depending on which of the two performs the first action.
4. *sequential composition*  $\_ \cdot \_ : P \times P \rightarrow P$ . The process represented by  $p \cdot q$  first performs the actions of  $p$ , until  $p$  terminates, and then continues with the actions in  $q$ .
5. *conditional operator*  $\_ \triangleleft \_ \triangleright \_ : P \times B \times P \rightarrow P$ . The “ $\_ \triangleleft \_ \triangleright \_$ ”-operator is the conditional operator of  $\mu\text{CRL}$ , and it operates precisely as a *then-else*-construct. The process term  $p \triangleleft b \triangleright q$  behaves like  $p$  if  $b$  is equal to  $t$ , and if  $b$  is equal to  $f$  it behaves like  $q$ .
6. *alternative quantification*  $\sum_v \_ : P \rightarrow P$  for each data variable  $v \in V$ . The sum operator  $\sum_v p$  behaves like  $p[d_1/v] + p[d_2/v] + \dots$ , i.e., as the possibly infinite choice between  $p[d_i/v]$  for any data term  $d_i$  of the sort of  $v$ .
7. *at-operator*  $\_ \_ : P \times T \rightarrow P$ . A key feature of timed  $\mu\text{CRL}$  is that it can be expressed at which time certain actions must take place. This is done using the “at”-operator. The process  $p \_ t$ , behaves like the process  $p$ , with the restriction that the first action of  $p$  must take place at time  $t$ .

8. *initialisation operator*  $_{-}\gg_{-} : T \times P \rightarrow P$ . The process  $t \gg p$  behaves as the process  $p$  as if it was started at time  $t$ . Thus the initialisation operator can be used to restrict the behaviour of a process to the part that can idle until the specified moment of time.

In the sequel, we will call these function declarations operators. For action declarations we usually write  $\mathbf{a} : s_1 \times \cdots \times s_n$  or, if no confusion can arise,  $\mathbf{a}$  instead of  $\mathbf{a} : s_1 \times \cdots \times s_n \rightarrow P$ . The set of all action declarations is denoted  $Act$ . *Action terms* are terms of the form  $\mathbf{a}(d_1, \dots, d_n)$ , where  $\mathbf{a} : s_1 \times \cdots \times s_n \in Act$  and  $d_i$  a data term of sort  $s_i$  (for  $i = 0, \dots, n$ ); the set of all action terms is denoted  $AT$ . We write  $AT_\delta$  for  $AT \cup \{\delta\}$ . Furthermore, we assume the existence of an infinite set of process variables  $V_P$  which is disjoint from the set of data variables  $V$ .

Process terms are built from action terms, data terms, variables and process operators. The set of all process terms is denoted  $T_P$ . For all operators in the language, characterising axioms are provided following the process algebraic tradition, in order to define which equivalences hold between processes. In fact, axiomatic reasoning with processes forms the essence of  $\mu\text{CRL}$ . Examples of equational manipulations with processes are given throughout this paper. Whereas the axioms provide a more syntactical perspective, operational semantics are used to interpret process terms in terms of potential behaviours of a system (see Section 3.2).

We list the various operators of  $p\text{CRL}_t$  in decreasing binding strength:

$$\begin{array}{c} \cdot \\ \gg \\ \triangleleft \triangleright \\ \sum_v \\ + \end{array}$$

In timed  $\mu\text{CRL}$  infinite behaviour can be described by using recursive equations of the form  $X(\vec{d}) = p$ , where  $X$  is a recursion variable that is parameterised by the values from  $\vec{d}$  and  $p$  is a process term in which recursion variables can occur. In this article, however, recursion is not treated with respect to the completeness result.

### 2.3. The theory $p\text{CRL}_t$

The axioms of  $p\text{CRL}_t$  are the axioms given by the user for the data types such as the booleans  $B$  and the time domain  $T$ , and the axioms given in Table 2. As proof theory we use generalised equational logic with a congruence rule for binders. For a precise expository on this proof theory we refer to [27, 57]. In Table 1, we present the proof theory for  $p\text{CRL}_t$ . Here,  $E$  represents the set of axioms of  $p\text{CRL}_t$  and  $\Sigma$  is the signature of the theory  $p\text{CRL}_t$ .

*Process-closed terms* are terms without process variables, but possibly with bound and free data variables.

The two elementary operators to construct processes are the sequential composition operator, written as  $\cdot$  and the alternative composition operator, written as  $+$ . The axioms A1–A5 in Table 2 describe the elementary properties of the sequential and alternative composition operators. For instance, the axioms A1, A2 and A3 express that  $+$  is commutative, associative and idempotent. These laws motivate why

$t = t'$	for every $t = t' \in E$
$\frac{t = t'}{t[e/v] = t'[e/v]}$	for every $v \in V_s$ and $e \in T_s$
$\frac{t_1 = t'_1 \cdots t_n = t'_n}{F(t_1, \dots, t_n) = F(t'_1, \dots, t'_n)}$	for every $F : s_1 \times \cdots \times s_n \rightarrow s' \in \Sigma$
$\frac{t = t'}{\sum_v t = \sum_v t'}$	
$t = t$	$\frac{t = t'}{t' = t}$
	$\frac{t_1 = t_2 \quad t_2 = t_3}{t_1 = t_3}$

Table 1. Generalised equational logic for  $p\text{CRL}_t$ .

in some cases parentheses may be omitted. For instance, it is allowed to write  $a + b + c$ , because, due to A2, it does not matter how brackets are put. The process algebra that consists of atomic actions, alternative and sequential composition only and for which A1–A5 are the only axioms is usually called *Basic Process Algebra* [11].

In the calculations in this paper we work modulo associativity and commutativity of  $+$ , and we do not mention the use of simple properties of the functions on data such as  $\neg$ ,  $\vee$ ,  $\wedge$ ,  $\leq$ , and  $eq$ .

In Table 2,  $p\text{CRL}_t$  axioms for the sum operator are listed. The sum operator is a difficult operator, because it acts as a binder. This introduces a range of problems with substitutions. When we substitute a process  $p$  for a variable  $x$  in the scope of a sum operator, no variable in  $p$  may become bound. If this appears to happen, we must first rename all bound occurrences of that variable in  $p$  into a variable that does not occur in  $p$ . This renaming is called  $\alpha$ -conversion. We consider processes modulo  $\alpha$ -conversion, so the terms  $\sum_v p$  and  $\sum_w p[w/v]$  are equal if  $w$  does not occur freely in  $v$ . Consequently, we may only substitute the action  $\mathbf{a}(v)$  for  $x$  in the left hand side of axiom SUM1, after renaming the bound variable  $v$  into  $w$ . So, SUM1 is a concise way of saying that if  $v$  does not appear in  $p$ , then we may omit the sum operator in  $\sum_v p$ . As another example, consider axiom SUM4. It says that we may distribute the sum operator over an alternative composition operator.

The process  $p \triangleleft b \triangleright q$  behaves like the process  $p$  in case the boolean term  $b$  equals  $t$  and it behaves like the process  $q$  if the boolean term  $b$  equals  $f$ . The conditional operator can be used to describe alternatives in behaviour based on the values of data variables. Its properties are given as the axioms C1–C7, SCA, SUM12', ATA5, and ATB5

The process  $p \cdot t$  behaves like the process  $p$ , with the restriction that the first action of  $p$  must take place at time  $t$ . So, if we assume that the natural numbers denote moments in time, the process described by  $a \cdot 1 \cdot b \cdot 2 \cdot c \cdot 3$  specifies that the actions  $a$ ,  $b$  and  $c$  must take place at times 1, 2 and 3, respectively.

A1	$x + y = y + x$	SUM1	$\sum_v x = x$
A2	$x + (y + z) = (x + y) + z$	SUM3	$\sum_v p = \sum_v p + p$
A3	$x + x = x$	SUM4	$\sum_v (p + q) = \sum_v p + \sum_v q$
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$	SUM5	$(\sum_v p) \cdot x = \sum_v (p \cdot x)$
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$	SUM12'	$(\sum_v p) \triangleleft b \triangleright \delta \cdot \mathbf{0} = \sum_v p \triangleleft b \triangleright \delta \cdot \mathbf{0}$
A6 <sup>-</sup>	$a + \delta = a$	PE	$p \triangleleft eq(v, w) \triangleright \delta \cdot \mathbf{0} = p[v/w] \triangleleft eq(v, w) \triangleright \delta \cdot \mathbf{0}$
A6'	$x + \delta \cdot \mathbf{0} = x$		
A7	$\delta \cdot x = \delta$		
C1	$x \triangleleft t \triangleright y = x$		
C2	$x \triangleleft f \triangleright y = y$		
C3	$x \triangleleft b \triangleright y = x \triangleleft b \triangleright \delta \cdot \mathbf{0} + y \triangleleft \neg b \triangleright \delta \cdot \mathbf{0}$		
C4	$(x \triangleleft b_1 \triangleright y) \triangleleft b_2 \triangleright y = x \triangleleft (b_1 \wedge b_2) \triangleright y$		
C5	$x \triangleleft b_1 \triangleright \delta \cdot \mathbf{0} + x \triangleleft b_2 \triangleright \delta \cdot \mathbf{0} = x \triangleleft (b_1 \vee b_2) \triangleright \delta \cdot \mathbf{0}$		
C6	$(x \triangleleft b \triangleright y) \cdot z = x \cdot z \triangleleft b \triangleright y \cdot z$		
C7	$(x + y) \triangleleft b \triangleright z = x \triangleleft b \triangleright z + y \triangleleft b \triangleright z$		
SCA	$(x \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot (y \triangleleft b \triangleright \delta \cdot \mathbf{0}) = x \cdot y \triangleleft b \triangleright \delta \cdot \mathbf{0}$		
AT1	$x = \sum_t x^t$		
AT2	$a^t \cdot x = a^t \cdot (t \gg x)$		
ATA1	$a^t u = (a^t \triangleleft u \leq t \triangleright \delta \cdot t) \triangleleft t \leq u \triangleright \delta \cdot u$		
ATA2	$(x + y)^t = x^t + y^t$		
ATA3	$(x \cdot y)^t = x^t \cdot y$		
ATA4	$(\sum_v p)^t = \sum_v p^t$		
ATA5	$(x \triangleleft b \triangleright y)^t = x^t \triangleleft b \triangleright y^t$		
ATB1	$t \gg a^u = a^u \triangleleft t \leq u \triangleright \delta \cdot t$		
ATB2	$t \gg (x + y) = t \gg x + t \gg y$		
ATB3	$t \gg (x \cdot y) = (t \gg x) \cdot y$		
ATB4	$t \gg \sum_v p = \sum_v t \gg p$		
ATB5	$t \gg (x \triangleleft b \triangleright y) = (t \gg x) \triangleleft b \triangleright (t \gg y)$		

Table 2. Axioms of  $p\text{CRL}_t$ :  $x, y, z \in V_P$ , process-closed  $p, q \in T_P$ ,  $a \in AT_\delta$ ,  $t, u \in V_T$ ,  $b, b_1, b_2 \in V_B$  and  $v, w \in V$ .

The process  $t \gg p$  behaves like the process  $p$  with the restriction that only the alternatives of  $p$  that can execute their first action at or after time  $t$  are considered. The main reason of introducing this operator is that it turns out to be convenient in providing an operational semantics for process terms later in this article. However, it is also convenient in describing that a process, while executing actions, cannot reverse time. This is described by axiom AT2. The initialisation operator is not likely to occur in specifications. The axioms ATB1-ATB5 describe how it can be eliminated from a process-closed term.

The constant  $\delta$  is the ‘classical’ deadlock of process algebra, which can not execute actions and never terminates. An important property of  $\delta$  is  $\delta \cdot p = \delta$ . It says that the process  $p$  after the deadlock  $\delta$  cannot be executed. It is formulated in Table 2 as axiom A7.

If an action happens at time  $t$ , then a subsequent action can take place at time  $t$  or afterwards. This means that in  $a \cdot 2 \cdot b \cdot 3$  and  $a \cdot 2 \cdot b \cdot 2$  the action  $b$  can happen, and in  $a \cdot 2 \cdot b \cdot 1$  the action  $b$  is blocked. Actually, the last example above is equivalent to  $a \cdot 2 \cdot \delta \cdot 2$ , which says that after action  $a$  we have a deadlock at time 2. In order to let  $b$  take place as prescribed, we have to reverse time. As this is clearly in conflict with reality, we choose to stop time at time 2. A process  $\delta \cdot t$  is called a *time deadlock*. Whenever a specification prescribes timing behaviour that cannot be realised, it will exhibit a time deadlock, i.e., time is stopped at a certain point. Specifications with time deadlocks can clearly not be implemented.

The difference between a time deadlock  $\delta \cdot t$  and deadlock  $\delta$  is that the former exists at any time before  $t$  and at  $t$ , but not after  $t$ , whereas the latter exists at any time. As a consequence we can not in general have  $p + \delta = p$  – an identity that is, in untimed  $p$ CRL, valid for every process term  $p$ . Consider for example the process  $a \cdot 2 + \delta$ . This process can perform the action  $a$  at time 2 and then terminates, but it can also let time pass until after time 2, in which case the option to execute  $a$  vanishes. So, contrary to untimed  $p$ CRL,  $\delta$  is not a neutral element for alternative composition in  $p$ CRL $_t$ . This role is taken over by the process  $\delta \cdot \mathbf{0}$ . This can easily be seen, as every  $p$ CRL $_t$  process exists at time  $\mathbf{0}$ . In the axiomatisation, we see that the  $p$ CRL axiom A6 ( $x + \delta = x$ ) is absent, and has been replaced by axiom A6' ( $x + \delta \cdot \mathbf{0} = x$ ). The difference between the process terms  $\delta \cdot \mathbf{0}$  and  $\delta$  can be illustrated by the following:  $a \cdot 2 + \delta \cdot \mathbf{0} = a \cdot 2$  and  $a \cdot 2 + \delta \neq a \cdot 2$ .

We see that if a time deadlock  $\delta \cdot u$  occurs in a process term with  $\delta$  as an alternative, it vanishes:

$$\delta + \delta \cdot t \stackrel{\text{AT1}}{=} \sum_t \delta \cdot t + \delta \cdot t \stackrel{\text{SUM3}}{=} \sum_t \delta \cdot t \stackrel{\text{AT1}}{=} \delta.$$

For processes  $p$  that do not refer to time explicitly, we still like to have the identity  $p + \delta = p$ . Using axiom A6 $^-$  ( $a + \delta = a$ ) this identity can be derived for untimed process-closed terms  $p$ .

## 2.4. Example specifications

We continue by giving some examples to illustrate the expressive power of the language timed  $p$ CRL. For example, a counter that counts the number of  $a$  actions that occur, issuing a  $b$  action after each 10 occurrences of an  $a$  action, can be described by the process  $Counter(0)$ , where  $Counter(n : Nat)$  is given by the following recursive equation:

$$Counter(n : Nat) = a \cdot Counter(n + 1) \triangleleft n < 10 \triangleright b \cdot Counter(0).$$

In the following example we describe a single place buffer, repeatedly reading a value  $d$  using action name  $\mathbf{r}$ , and then delivering that value via action name  $\mathbf{s}$ :

$$Buffer = \sum_d \mathbf{r}(d) \cdot \mathbf{s}(d) \cdot Buffer.$$

In the specification of real-time systems frequently deadlines have to be specified. In the following example we illustrate how to express that an action  $b$  has to occur within  $d$  time after the occurrence of an action  $a$ :

$$\sum_{t:T} a \cdot t \cdot \left( \sum_{t':T} b \cdot t' \triangleleft t' \leq t + d \triangleright \delta \cdot \mathbf{0} \right).$$

In the last example we specify a clock process that produces actions  $tick$  every time unit. However, it does not do so accurately. Each  $tick$  action can occur after the previous  $tick$  action after at least  $1 - \xi$  and at most  $1 + \xi$  time has elapsed:

$$X(t : T) = \sum_{\epsilon:T} tick \cdot (t + 1 + \epsilon) \cdot X(t + 1 + \epsilon) \triangleleft -\xi \leq \epsilon \leq \xi \triangleright \delta \cdot \mathbf{0}.$$

## 2.5. Simple identities

Calculating in (timed)  $\mu$ CRL generally requires numerous basic identities. In this section, we present some identities as lemmas for easy future reference. In derivations in this section and the following sections, we frequently apply the axioms for booleans and for the time domain. We will do so without explicitly mentioning them.

Axiom SUM3 is typically used to extract a specific alternative from an alternative quantification where the bound variable, say  $v$ , is replaced by a data term, say  $d$ , of the appropriate sort. The following lemma ensures the correctness of such an extraction.

**Lemma 2.1.** Let  $s$  be an arbitrary sort. For process-closed  $p \in T_P$ ,  $v \in V_s$ , and  $d \in T_s$ , it holds that

$$\sum_v p = \sum_v p + p[d/v].$$

**Proof:**

Consider the following derivation:

$$\sum_v p = \left( \sum_v p \right) [d/v] \stackrel{\text{SUM3}}{=} \left( \sum_v p + p \right) [d/v] = \left( \sum_v p \right) [d/v] + p[d/v] = \sum_v p + p[d/v].$$

□

### Lemma 2.2. (Laws for Conditional Expressions)

For  $x, y : V_P$  and  $b, c : V_B$ , it holds that  $x \triangleleft b \triangleright x = x$ .

**Proof:**

$$x \triangleleft b \triangleright x \stackrel{\text{C3}}{=} x \triangleleft b \triangleright \delta \cdot \mathbf{0} + x \triangleleft \neg b \triangleright \delta \cdot \mathbf{0} \stackrel{\text{C5}}{=} x \triangleleft t \triangleright \delta \cdot \mathbf{0} \stackrel{\text{C1}}{=} x.$$

□

**Lemma 2.3.** For  $t \in V_T$ , it holds that  $\delta \cdot \mathbf{0} \cdot t = \delta \cdot \mathbf{0}$ .

**Proof:**

$$\delta \cdot \mathbf{0} \cdot t \stackrel{\text{ATA1}}{=} (\delta \cdot \mathbf{0} \triangleleft t \leq \mathbf{0} \triangleright \delta \cdot \mathbf{0}) \triangleleft \mathbf{0} \leq t \triangleright \delta \cdot t \stackrel{2.2}{=} \delta \cdot \mathbf{0} \triangleleft \mathbf{0} \leq t \triangleright \delta \cdot \mathbf{0} \stackrel{\text{C1}}{=} \delta \cdot \mathbf{0}.$$

□

In the following lemma we present some facts about the initialisation operator.

**Lemma 2.4.** For  $t, u \in V_T$  and  $a \in AT_\delta$ , it holds that

1.  $t \gg (t \gg a^c u) = t \gg a^c u$ ;
2.  $t \gg \delta^c \mathbf{0} = \delta^c t$ .

**Proof:**

1.  $t \gg (t \gg a^c u) \stackrel{\text{ATB1}}{=} t \gg (a^c u \triangleleft t \leq u \triangleright \delta^c t) \stackrel{\text{ATB5}}{=} (t \gg a^c u) \triangleleft t \leq u \triangleright (t \gg \delta^c t) \stackrel{\text{ATB1}}{=} (a^c u \triangleleft t \leq u \triangleright \delta^c t) \triangleleft t \leq u \triangleright \delta^c t \stackrel{\text{C4}}{=} a^c u \triangleleft t \leq u \triangleright \delta^c t \stackrel{\text{ATB1}}{=} t \gg a^c u$ .
2.  $t \gg \delta^c \mathbf{0} \stackrel{\text{ATB1}}{=} \delta^c \mathbf{0} \triangleleft t \leq \mathbf{0} \triangleright \delta^c t = \delta^c \mathbf{0} \triangleleft eq(t, \mathbf{0}) \triangleright \delta^c t \stackrel{\text{PE}}{=} \delta^c t \triangleleft eq(t, \mathbf{0}) \triangleright \delta^c t \stackrel{2.2}{=} \delta^c t$ .

□

**Lemma 2.5.** For  $v \in V_T$  and  $t \in V_T$ , it holds that  $\sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} = \delta^c t$ .

**Proof:**

$$\begin{aligned}
& \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} \\
\stackrel{2.1}{=} & \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} + (\delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0})[t/v] \\
= & \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} + \delta^c t \triangleleft t \leq t \triangleright \delta^c \mathbf{0} \\
\stackrel{\text{C1}}{=} & \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} + \delta^c t \\
\stackrel{\text{AT1}}{=} & \sum_v \delta^c t^c v + \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} \\
\stackrel{\text{ATA1,2.2}}{=} & \sum_v \delta^c t \triangleleft t \leq v \triangleright \delta^c v + \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} \\
\stackrel{\text{C3}}{=} & \sum_v (\delta^c t \triangleleft t \leq v \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) + \sum_v \delta^c v \triangleleft v \leq t \triangleright \delta^c \mathbf{0} \\
\stackrel{\text{C5}}{=} & \sum_v (\delta^c t \triangleleft t \leq v \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) \\
+ & \sum_v (\delta^c v \triangleleft eq(t, v) \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) \\
\stackrel{\text{C5,PE}}{=} & \sum_v (\delta^c t \triangleleft t < v \triangleright \delta^c \mathbf{0} + \delta^c t \triangleleft eq(t, v) \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) \\
+ & \sum_v (\delta^c t \triangleleft eq(t, v) \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) \\
\stackrel{\text{SUM4,A3}}{=} & \sum_v (\delta^c t \triangleleft t < v \triangleright \delta^c \mathbf{0} + \delta^c t \triangleleft eq(t, v) \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) \\
\stackrel{\text{C5}}{=} & \sum_v (\delta^c t \triangleleft t \leq v \triangleright \delta^c \mathbf{0} + \delta^c v \triangleleft t > v \triangleright \delta^c \mathbf{0}) \\
\stackrel{\text{C3}}{=} & \sum_v \delta^c t \triangleleft t \leq v \triangleright \delta^c v \\
\stackrel{\text{ATA1,2.2}}{=} & \sum_v \delta^c t^c v \\
\stackrel{\text{AT1}}{=} & \delta^c t.
\end{aligned}$$

□

## 2.6. Basic terms

We provide a basic syntactic format for  $p\text{CRL}_t$ -terms. In this format we use the notation  $\sum_{\vec{v}}$  to represent a finite sequence of alternative quantifications  $\sum_{v_1} \sum_{v_2} \cdots \sum_{v_n}$  ( $n \geq 0$ ).

**Definition 2.1. (Basic terms)**

The set of basic terms is inductively defined as follows:

1.  $\sum_{\bar{v}} a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ , with  $a \in AT_\delta$ ,  $t$  a time term and  $b$  a boolean term, is a basic term;
2. if  $p$  is a basic term, then  $\sum_{\bar{v}} a \cdot t \cdot p \triangleleft b \triangleright \delta \cdot \mathbf{0}$ , with  $a \in AT$ ,  $t$  a time term and  $b$  a boolean term, is a basic term;
3. if  $p$  and  $q$  are basic terms, then  $p + q$  is a basic term.

If a basic term is of the first form, then we say it is of *type 1*. Similarly for forms 2 and 3.

**Theorem 2.1. (Basic Term Theorem)**

If  $q$  is a process-closed term over  $\Sigma(p\text{CRL}_t)$ , then there is a basic term  $p$  such that  $\mu\text{CRL}_t \vdash q = p$ .

**Proof:**

We apply induction on the number of symbols in process-closed term  $q$ . The following cases can be distinguished based on the structure of process-closed term  $q$ .

1.  $q \equiv a$  for some  $a \in AT_\delta$ . Obviously  $a \stackrel{\text{AT1}}{=} \sum_t a \cdot t \stackrel{\text{C1}}{=} \sum_t a \cdot t \triangleleft t \triangleright \delta \cdot \mathbf{0}$ . This is a type 1 basic term.
2.  $q \equiv q_1 + q_2$  for some process-closed terms  $q_1$  and  $q_2$ . By induction we have the existence of basic terms  $p_1$  and  $p_2$  such that  $q_1 = p_1$  and  $q_2 = p_2$ . Hence,  $q = q_1 + q_2 = p_1 + p_2$ , which is a type 3 basic term.
3.  $q \equiv q_1 \cdot q_2$  for some process-closed terms  $q_1$  and  $q_2$ . By induction we have the existence of basic terms  $p_1$  and  $p_2$  such that  $q_1 = p_1$  and  $q_2 = p_2$ . By induction on the structure of basic term  $p_1$  we prove that there exists a basic term  $p$  such that  $p_1 \cdot p_2 = p$ .
  - (a)  $p_1 \equiv \sum_{\bar{v}} a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then  $p_1 \cdot p_2 = (\sum_{\bar{v}} a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot p_2 \stackrel{\text{SUM5}}{=} \sum_{\bar{v}} (a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot p_2 \stackrel{\text{C6}}{=} \sum_{\bar{v}} a \cdot t \cdot p_2 \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot p_2 \stackrel{\text{ATA3,A7}}{=} \sum_{\bar{v}} a \cdot t \cdot p_2 \triangleleft b \triangleright \delta \cdot \mathbf{0}$ , which is a type 2 basic term.
  - (b)  $p_1 \equiv \sum_{\bar{v}} a \cdot t \cdot p'_1 \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . By induction we have the existence of a basic term  $p'$  such that  $p'_1 \cdot p_2 = p'$ . Then,  $p_1 \cdot p_2 = (\sum_{\bar{v}} a \cdot t \cdot p'_1 \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot p_2 \stackrel{\text{SUM5}}{=} \sum_{\bar{v}} (a \cdot t \cdot p'_1 \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot p_2 \stackrel{\text{C6}}{=} \sum_{\bar{v}} (a \cdot t \cdot p'_1) \cdot p_2 \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot p_2 \stackrel{\text{A5,ATA3,A7}}{=} \sum_{\bar{v}} a \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ , which is a type 2 basic term.
  - (c)  $p_1 \equiv p'_1 + p''_1$  for some basic terms  $p'_1$  and  $p''_1$ . By induction we have the existence of basic terms  $p'$  and  $p''$  such that  $p'_1 \cdot p_2 = p'$  and  $p''_1 \cdot p_2 = p''$ . Then  $p_1 \cdot p_2 = (p'_1 + p''_1) \cdot p_2 \stackrel{\text{A4}}{=} p'_1 \cdot p_2 + p''_1 \cdot p_2 = p' + p''$ , which is a type 3 basic term.
4.  $q \equiv \sum_v q'$  for some process-closed term  $q'$ . By induction we have the existence of basic term  $p'$  such that  $q' = p'$ . We proceed by induction on the structure of basic term  $p'$ . If  $p'$  is of type 1 or type 2, then obviously  $\sum_v p'$  is a basic term. If  $p'$  is of type 3, e.g.  $p' = p'_1 + p'_2$  for some basic terms  $p'_1$  and  $p'_2$ , then we have by induction the existence of basic terms  $p''_1$  and  $p''_2$  such that  $\sum_v p'_1 = p''_1$  and  $\sum_v p'_2 = p''_2$ . Then  $q = \sum_v q' = \sum_v p' = \sum_v (p'_1 + p'_2) \stackrel{\text{SUM4}}{=} \sum_v p'_1 + \sum_v p'_2 = p''_1 + p''_2$ , which is a basic term.

5.  $q \equiv q_1 \triangleleft b \triangleright q_2$  for some process-closed terms  $q_1$  and  $q_2$ . By induction we have the existence of basic terms  $p_1$  and  $p_2$  such that  $q_1 = p_1$  and  $q_2 = p_2$ . Then  $q = q_1 \triangleleft b \triangleright q_2 = p_1 \triangleleft b \triangleright p_2 \stackrel{C3}{=} p_1 \triangleleft b \triangleright \delta \cdot \mathbf{0} + p_2 \triangleleft \neg b \triangleright \delta \cdot \mathbf{0}$ . We prove that for every basic term  $r$  there exists a basic term  $s$  such that  $r \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = s$  by induction on the structure of basic term  $r$ .

- (a)  $r \equiv \sum_{\vec{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,  $r \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = (\sum_{\vec{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} \stackrel{SUM12'}{=} \sum_{\vec{v}} (a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} \stackrel{C4}{=} \sum_{\vec{v}} a^c t \triangleleft b \wedge \alpha \triangleright \delta \cdot \mathbf{0}$ , which is a basic term.
- (b)  $r \equiv \sum_{\vec{v}} a^c t \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,  $r \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = (\sum_{\vec{v}} a^c t \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} \stackrel{SUM12'}{=} \sum_{\vec{v}} (a^c t \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} \stackrel{C4}{=} \sum_{\vec{v}} a^c t \cdot r' \triangleleft b \wedge \alpha \triangleright \delta \cdot \mathbf{0}$ , which is a basic term.
- (c)  $r \equiv r_1 + r_2$ . By induction we have the existence of basic terms  $s_1$  and  $s_2$  such that both  $r_1 \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = s_1$  and  $r_2 \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = s_2$ . Then  $r \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = (r_1 + r_2) \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} \stackrel{C7}{=} r_1 \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} + r_2 \triangleleft \alpha \triangleright \delta \cdot \mathbf{0} = s_1 + s_2$ , which is a basic term.

6.  $q \equiv t \gg q'$  for some process-closed term  $q'$ . By induction we have the existence of basic term  $p'$  such that  $q' = p'$ . By induction on the structure of basic term  $p'$  we prove the existence of a basic term  $p''$  such that  $t \gg p' = p''$ .

- (a)  $p' \equiv \sum_{\vec{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned} t \gg p' &= t \gg (\sum_{\vec{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{ATB4}{=} \sum_{\vec{v}} t \gg (a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{ATB5}{=} \sum_{\vec{v}} (t \gg a^c u) \triangleleft b \triangleright (t \gg \delta \cdot \mathbf{0}) \\ &\stackrel{C3, SUM4, 2.4.2}{=} \sum_{\vec{v}} (t \gg a^c u) \triangleleft b \triangleright \delta \cdot \mathbf{0} + \sum_{\vec{v}} \delta \cdot t \triangleleft \neg b \triangleright \delta \cdot \mathbf{0}. \end{aligned}$$

The second summand is a basic term. We continue with the first summand:

$$\begin{aligned} &\sum_{\vec{v}} (t \gg a^c u) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{ATB1}{=} \sum_{\vec{v}} (a^c u \triangleleft t \leq u \triangleright \delta \cdot t) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{C3}{=} \sum_{\vec{v}} (a^c u \triangleleft t \leq u \triangleright \delta \cdot \mathbf{0} + \delta \cdot t \triangleleft t > u \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{C7, SUM4}{=} \sum_{\vec{v}} (a^c u \triangleleft t \leq u \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} + \sum_{\vec{v}} (\delta \cdot t \triangleleft t > u \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{C4}{=} \sum_{\vec{v}} a^c u \triangleleft t \leq u \wedge b \triangleright \delta \cdot \mathbf{0} + \sum_{\vec{v}} \delta \cdot t \triangleleft t > u \wedge b \triangleright \delta \cdot \mathbf{0}, \end{aligned}$$

which is a basic term.

- (b)  $p' \equiv \sum_{\vec{v}} a^c u \cdot p'_1 \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Similar to the previous case only axiom ATB3 has to be applied just before applying ATB1.
- (c)  $p' \equiv p'_1 + p'_2$ . By induction we have the existence of basic terms  $p''_1$  and  $p''_2$  such that  $t \gg p'_1 = p''_1$  and  $t \gg p'_2 = p''_2$ . Then,  $t \gg p' = t \gg (p'_1 + p'_2) \stackrel{ATB2}{=} t \gg p'_1 + t \gg p'_2 = p''_1 + p''_2$ , which is a basic term.

7.  $q \equiv q' \cdot t$  for some process-closed term  $q'$ . By induction we have the existence of basic term  $p'$  such that  $q' = p'$ . We prove that for every basic term  $p'$  there exists a basic term  $p''$  such that  $p' \cdot t = p''$  by induction on the structure of basic term  $p'$ .

(a)  $p' \equiv \sum_{\bar{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned}
p' \cdot t &= (\sum_{\bar{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot t \\
&\stackrel{\text{ATA4}}{=} \sum_{\bar{v}} (a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot t \\
&\stackrel{\text{ATA5}}{=} \sum_{\bar{v}} a^c u \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot t \\
&\stackrel{\text{ATA1,2.3}}{=} \sum_{\bar{v}} ((a^c u \triangleleft t \leq u \triangleright \delta \cdot u) \triangleleft u \leq t \triangleright \delta \cdot t) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\stackrel{\text{C3}}{=} \sum_{\bar{v}} ((a^c u \triangleleft t \leq u \triangleright \delta \cdot \mathbf{0} + \delta \cdot u \triangleleft t > u \triangleright \delta \cdot \mathbf{0}) \triangleleft u \leq t \triangleright \delta \cdot \mathbf{0} \\
&\quad + \delta \cdot t \triangleleft u > t \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\stackrel{\text{C7,SUM4}}{=} \sum_{\bar{v}} ((a^c u \triangleleft t \leq u \triangleright \delta \cdot \mathbf{0} + \delta \cdot u \triangleleft t > u \triangleright \delta \cdot \mathbf{0}) \triangleleft u \leq t \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\quad + \sum_{\bar{v}} (\delta \cdot t \triangleleft u > t \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\stackrel{\text{C7,SUM4}}{=} \sum_{\bar{v}} ((a^c u \triangleleft t \leq u \triangleright \delta \cdot \mathbf{0}) \triangleleft u \leq t \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\quad + \sum_{\bar{v}} ((\delta \cdot u \triangleleft t > u \triangleright \delta \cdot \mathbf{0}) \triangleleft u \leq t \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\quad + \sum_{\bar{v}} (\delta \cdot t \triangleleft u > t \triangleright \delta \cdot \mathbf{0}) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&\stackrel{\text{C4}}{=} \sum_{\bar{v}} a^c u \triangleleft t \leq u \wedge u \leq t \wedge b \triangleright \delta \cdot \mathbf{0} + \sum_{\bar{v}} \delta \cdot u \triangleleft t > u \wedge u \leq t \wedge b \triangleright \delta \cdot \mathbf{0} \\
&\quad + \sum_{\bar{v}} \delta \cdot t \triangleleft u > t \wedge b \triangleright \delta \cdot \mathbf{0} \\
&= \sum_{\bar{v}} a^c u \triangleleft eq(t, u) \wedge b \triangleright \delta \cdot \mathbf{0} + \sum_{\bar{v}} \delta \cdot u \triangleleft t > u \wedge b \triangleright \delta \cdot \mathbf{0} \\
&\quad + \sum_{\bar{v}} \delta \cdot t \triangleleft t < u \wedge b \triangleright \delta \cdot \mathbf{0},
\end{aligned}$$

which is a basic term.

- (b)  $p' \equiv \sum_{\bar{v}} a^c u \cdot p'_1 \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Similar to the previous case: only axiom ATA3 has to be applied just before applying ATA1.
- (c)  $p' \equiv p'_1 + p'_2$ . By induction we have the existence of basic terms  $p''_1$  and  $p''_2$  such that  $p'_1 \cdot t = p''_1$  and  $p'_2 \cdot t = p''_2$ . Then,  $p' \cdot t = (p'_1 + p'_2) \cdot t \stackrel{\text{ATA2}}{=} p'_1 \cdot t + p'_2 \cdot t = p''_1 + p''_2$ , which is a basic term.  $\square$

The main virtue of the basic term theorem is that in proving an identity for process-closed terms, we can restrict the proof to basic terms. Instead of using induction on the structure of process-closed terms we can use induction on the structure of basic terms. This limits the cases to be considered considerably. Another example of such a proof is the proof of the following lemma. In Section 3, we frequently use induction on basic terms.

**Lemma 2.6.** For process-closed terms  $p$  we have  $t \gg (t \gg p) = t \gg p$ .

**Proof:**

Without loss of generality we may assume that  $p$  is a basic term. We prove the lemma by induction on the structure of basic term  $p$ .

1.  $p \equiv \sum_{\bar{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . This case is similar to the next case albeit that axiom ATB3 does not have to be applied.

2.  $p \equiv \sum_{\bar{v}} a^c u \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned}
t \gg (t \gg p) &= t \gg (t \gg \sum_{\bar{v}} a^c u \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB4}}{=} \sum_{\bar{v}} t \gg (t \gg (a^c u \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0})) \\
&\stackrel{\text{ATB5}}{=} \sum_{\bar{v}} t \gg (t \gg a^c u \cdot p' \triangleleft b \triangleright t \gg (t \gg \delta \cdot \mathbf{0})) \\
&\stackrel{\text{ATB3}}{=} \sum_{\bar{v}} (t \gg (t \gg a^c u)) \cdot p' \triangleleft b \triangleright t \gg (t \gg \delta \cdot \mathbf{0}) \\
&\stackrel{2.4.1}{=} \sum_{\bar{v}} (t \gg a^c u) \cdot p' \triangleleft b \triangleright t \gg \delta \cdot \mathbf{0} \\
&\stackrel{\text{ATB3}}{=} \sum_{\bar{v}} t \gg a^c u \cdot p' \triangleleft b \triangleright t \gg \delta \cdot \mathbf{0} \\
&\stackrel{\text{ATB5}}{=} \sum_{\bar{v}} t \gg (a^c u \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB4}}{=} t \gg \sum_{\bar{v}} a^c u \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&= t \gg p.
\end{aligned}$$

3.  $p \equiv p' + p''$ . This part follows easily from the induction hypothesis and axiom ATB2.  $\square$

### 3. Semantics of timed $p$ CRL

In this section, we present a model in which processes parameterised with data can be interpreted. With respect to the abstract data types we assume the existence of a data algebra with universe  $\mathcal{D}_s$  for each sort symbol  $s$ . For instance, for the booleans we have that the universe contains the interpretation of  $t$  and the interpretation of  $f$ .

A valuation  $\nu$  is a function from the set of variables  $V$  to the set of values  $\bigcup_{s \in \mathcal{S}} \mathcal{D}_s$  such that  $\nu(v) \in \mathcal{D}_s$  if and only if  $v \in V_s$ . This means that a variable can only be mapped to a value of the right data algebra. For valuations  $\nu$  and  $\nu'$  and variable  $v \in V$ , we write  $\nu[v]\nu'$ , if for all  $u \in V$ ,  $u \neq v$  implies  $\nu(u) = \nu'(u)$ . Thus  $\nu[v]\nu'$  means that the valuations  $\nu$  and  $\nu'$  are the same for all variables except possibly for the variable  $v$ . Given a valuation  $\nu$ , we write  $\llbracket t \rrbracket^\nu$  for the interpretation of a term  $t$ .

The process terms are mapped to processes and on these we define a suitable notion of *strong timed bisimulation*. Soundness of the axioms is proved w.r.t. the model. Completeness of the axioms is proved under the assumption that the axiomatisation of the data is  $\omega$ -complete (i.e. complete for open terms) and adheres to certain other properties. This notion of completeness will be called *relative completeness*.

#### 3.1. Interpretation of process terms

**Definition 3.1.** Given some  $p\text{CRL}_t$  specification with a set of action declarations  $Act$ , the set of *actions* is defined by

$$\mathcal{A} = \{a(e_1, \dots, e_n) \mid a : s_1 \times \dots \times s_n \in Act, e_i \in \mathcal{D}_{s_i}\}.$$

The set  $\mathcal{A} \cup \{\delta\}$  is denoted by  $\mathcal{A}_\delta$ . The set  $\mathcal{P} = \bigcup_{i=0}^{\omega} \mathcal{P}^i$  of *processes* is obtained by the following recursion:

$$\begin{aligned}
\mathcal{P}^0 &= \mathcal{A}_\delta, \\
\mathcal{P}^{n+1} &= \mathcal{P}^n \cup \{p \cdot q, \sum P', p^c t, t \gg p \mid p, q \in \mathcal{P}^n, P' \neq \emptyset, P' \subseteq \mathcal{P}^n, t \in \mathcal{D}_T\}.
\end{aligned}$$

This summation operator, which takes a set of processes as its argument, is taken from [27, 57].

**Definition 3.2.** The interpretation of process-closed terms under a valuation  $\nu$  is inductively defined by: for  $\mathbf{a} : s_1 \times \dots \times s_n \in Act$  and  $d_i \in T_{s_i}$ , process-closed terms  $p, q \in T_P$ ,  $b \in T_B$ , and  $t \in T_T$

$$\begin{array}{ll}
\llbracket \mathbf{a}(d_1, \dots, d_n) \rrbracket^\nu &= \mathbf{a}(\llbracket d_1 \rrbracket^\nu, \dots, \llbracket d_n \rrbracket^\nu), & \llbracket t \gg p \rrbracket^\nu &= \llbracket t \rrbracket^\nu \gg \llbracket p \rrbracket^\nu, \\
\llbracket \delta \rrbracket^\nu &= \delta, & \llbracket p \triangleleft b \triangleright q \rrbracket^\nu &= \begin{cases} \llbracket p \rrbracket^\nu & \text{if } \llbracket b \rrbracket^\nu = \llbracket t \rrbracket^\nu, \\ \llbracket q \rrbracket^\nu & \text{if } \llbracket b \rrbracket^\nu = \llbracket f \rrbracket^\nu, \end{cases} \\
\llbracket p + q \rrbracket^\nu &= \sum \{ \llbracket p \rrbracket^\nu, \llbracket q \rrbracket^\nu \}, & \llbracket \sum_v p \rrbracket^\nu &= \sum \{ \llbracket p \rrbracket^{\nu'} \mid \nu[v]\nu' \}. \\
\llbracket p \cdot q \rrbracket^\nu &= \llbracket p \rrbracket^\nu \cdot \llbracket q \rrbracket^\nu, & & \\
\llbracket p \cdot t \rrbracket^\nu &= \llbracket p \rrbracket^\nu \cdot \llbracket t \rrbracket^\nu, & & 
\end{array}$$

### 3.2. Operational semantics and strong bisimulation

In this section, we define an operational semantics of processes in terms of the timed execution of actions. For this purpose we introduce *action relations*  $\_ \xrightarrow{a}_t \_ \checkmark$  and  $\_ \xrightarrow{a}_t \_ -$ . By  $p \xrightarrow{a}_t \checkmark$  we express that the process  $p$  can perform an action  $a$  at time  $t$ , and then terminate successfully at  $t$ . By  $p \xrightarrow{a}_t p'$  we express that the process  $p$  evolves into process  $p'$  by performing action  $a$  at time  $t$ .

**Definition 3.3.** The action relations  $\_ \xrightarrow{a}_t \_ \checkmark \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_T$  and  $\_ \xrightarrow{a}_t \_ - \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{D}_T \times \mathcal{P}$  are defined by the transition rules in Table 3.

$a \xrightarrow{a}_t \checkmark$	$\frac{p \xrightarrow{a}_t \checkmark}{\sum(\{p\} \cup P) \xrightarrow{a}_t \checkmark}$	$\frac{p \xrightarrow{a}_t p'}{\sum(\{p\} \cup P) \xrightarrow{a}_t p'}$	$\frac{p \xrightarrow{a}_t \checkmark}{p \cdot t \xrightarrow{a}_t \checkmark}$	$\frac{p \xrightarrow{a}_t p'}{p \cdot t \xrightarrow{a}_t p'}$
$\frac{p \xrightarrow{a}_t \checkmark}{p \cdot q \xrightarrow{a}_t t \gg q}$	$\frac{p \xrightarrow{a}_t p'}{p \cdot q \xrightarrow{a}_t p' \cdot q}$	$\frac{p \xrightarrow{a}_t \checkmark \quad t' \leq t}{t' \gg p \xrightarrow{a}_t \checkmark}$	$\frac{p \xrightarrow{a}_t p' \quad t' \leq t}{t' \gg p \xrightarrow{a}_t p'}$	

Table 3. Transition rules for the  $pCRL_t$  operators, where  $a \in \mathcal{A}$ ,  $p, p', q \in \mathcal{P}$ ,  $t, t' \in \mathcal{D}_T$  and  $P \subseteq \mathcal{P}$ .

The execution of an action does not take time: it is possible that the process  $p'$  can subsequently perform other actions at  $t$ , e.g.

$$a \cdot t \cdot b \cdot t \xrightarrow{a}_t t \gg b \cdot t \xrightarrow{b}_t \checkmark.$$

Given the action relations we have a good understanding of the steps which can be performed by a process. An obvious question is when two processes are equivalent. Such a notion is required to justify the axioms. Strong bisimulation is a commonly used process equivalence. Therefore, we provide a timed variant of strong bisimulation, called strong timed bisimulation. However, bisimulation cannot be defined in terms of actions only, e.g. the processes  $\delta \cdot t$  and  $\delta \cdot u$  are considered equivalent only if  $t = u$ . We introduce the *delay relation*  $U_t(p)$ , which expresses that  $p$  can at least idle until time  $t$ . It has the same purpose as the ultimate delay in [3, 5, 50].

$U_t(a)$	$\frac{U_t(p)}{U_t(p \cdot q)}$	$\frac{U_t(p)}{U_t(\sum P \cup \{p\})}$	$\frac{U_t(p) \quad t \leq t'}{U_t(p \circ t')}$	$\frac{U_t(p)}{U_t(t' \gg p)}$	$\frac{t \leq t'}{U_t(t' \gg p)}$
----------	---------------------------------	---	--	--------------------------------	-----------------------------------

Table 4. Transition rules for the delay relation, where  $a \in \mathcal{A}_\delta$ ,  $p, q \in \mathcal{P}$ ,  $t, t' \in \mathcal{D}_T$  and  $P \subseteq \mathcal{P}$ .

**Definition 3.4.** The delay relation  $U_\_(-) \subseteq \mathcal{D}_T \times \mathcal{P}$  is defined by the transition rules in Table 4.

**Definition 3.5.** A symmetric relation  $R \subseteq \mathcal{P} \times \mathcal{P}$  is a *strong timed bisimulation* if for all  $(p, q) \in R$  it holds that

1. if  $p \xrightarrow{a}_t \checkmark$ , then  $q \xrightarrow{a}_t \checkmark$ ,
2. if  $p \xrightarrow{a}_t p'$ , then there is a  $q'$  such that  $q \xrightarrow{a}_t q'$  and  $(p', q') \in R$ , and
3. if  $U_t(p)$ , then  $U_t(q)$ ,

for all  $a \in \mathcal{A}$ ,  $t \in \mathcal{D}_T$  and  $p' \in \mathcal{P}$ .

Processes  $p$  and  $q$  are *strongly timed bisimilar*, notation  $p \Leftrightarrow_t q$ , if and only if there is a strong timed bisimulation  $R$  such that  $(p, q) \in R$ . Observe that the union of any set of strong timed bisimulations is again a strong timed bisimulation.

**Lemma 3.1. (Congruence)**

Strong timed bisimilarity is a congruence with respect to the operators defined on  $\mathcal{P}$ .

**Proof:**

It is not hard to prove that strong timed bisimilarity is an equivalence. We omit the proof. Next, we give bisimulations that witness the congruence of strong timed bisimilarity with respect to the operators  $\gg$ ,  $\cdot$ ,  $\sum$ , and  $\circ$ . For  $\cdot$  we give a more detailed proof.

1.  $\gg$ . Let  $R : p \Leftrightarrow_t q$ . Let  $t \in \mathcal{D}_T$ . Obviously, the relation  $\{(t \gg p, t \gg q), (t \gg q, t \gg p)\} \cup R$  is a strong timed bisimulation.
2.  $\cdot$ . Let  $R_1 : p_1 \Leftrightarrow_t q_1$  and  $R_2 : p_2 \Leftrightarrow_t q_2$ . Define  $R = \{(p'_1 \cdot p_2, q'_1 \cdot q_2), (q'_1 \cdot q_2, p'_1 \cdot p_2) \mid (p'_1, q'_1) \in R_1\} \cup \{(t \gg p_2, t \gg q_2), (t \gg q_2, t \gg p_2) \mid t \in \mathcal{D}_T\} \cup R_2$ . We do not have to consider pairs in  $R$  that are also in  $R_2$  as for these we have already assumed that they are strongly timed bisimilar. We also do not have to consider the pairs in  $R$  of the form  $(t \gg p_2, t \gg q_2)$  and vice versa with  $(p_2, q_2) \in R_2$  as the strong timed bisimulation from the proof of congruence of  $\gg$  is contained in  $R$ . Thus, we only have to consider pairs of the form  $(p'_1 \cdot p_2, q'_1 \cdot q_2)$  with  $(p'_1, q'_1) \in R_1$ . For pairs of this form it is easy to prove that these satisfy the conditions of the definition of strong timed bisimulations. Hence the relation  $R$  is a strong timed bisimulation.
3.  $\sum$ . Let  $R$  be a strong timed bisimulation such that for all  $p \in P$  there exists  $q \in Q$  such that  $(p, q) \in R$  and vice versa for all  $q \in Q$  there exists  $p \in P$  such that  $(q, p) \in R$ . The relation  $R' = \{(\sum P, \sum Q), (\sum Q, \sum P)\} \cup R$  is obviously a strong timed bisimulation.

4.  $\Leftarrow$ . Let  $R : p \Leftarrow_t q$ . Let  $t \in \mathcal{D}_T$ . Obviously the relation  $\{(p \cdot t, q \cdot t), (q \cdot t, p \cdot t)\} \cup R$  is a strong timed bisimulation. □

The axioms are sound with respect to the model of closed terms modulo strong timed bisimilarity. This means that any two derivably equal process-closed terms are strongly timed bisimilar.

**Theorem 3.1. (Soundness)**

With respect to strong timed bisimulation  $p\text{CRL}_t$  is a sound axiom system.

Although the details of the soundness proof are omitted, in general, the following strategy is taken towards such a soundness proof. It must be shown that for any two derivably equal process-closed terms, the processes that are obtained by interpreting them are strongly timed bisimilar. However, as any derivable equality is based on applications of the axioms and proof rules, it suffices to prove that each of the axioms is sound. Note that the soundness of the proof rules follows from the fact that strong timed bisimilarity is a congruence with respect to all operators of timed  $p\text{CRL}$ . Soundness of a single axiom is obtained by giving a relation  $R$  between processes such that this relation relates all interpretations of (all instantiations of) that axiom and a proof that this relation is a strong timed bisimulation. Examples of soundness proofs that are given using this strategy (though for different process algebras) can be found in [10, 65, 68].

## 4. Relative completeness

In this section, we prove completeness of the axiomatisation with respect to the operational semantics modulo strong timed bisimilarity. For  $p\text{CRL}$  a completeness result is given in [27, 57], where it is shown that the axiomatization of  $p\text{CRL}$ , as presented in Appendix A, is complete under the following assumptions: (1) the axiomatisation of the data algebra is  $\omega$ -complete, (2) the data algebra has built-in equality for all data sorts, and (3) the data algebra has built-in Skolem functions for all data sorts. It is beyond the scope of this paper to give the complete proof of this result, but we will give some of the reasoning behind it. First, we define the notions *built-in equality* and *built-in Skolem functions*.

**Definition 4.1.** A data algebra  $\mathcal{D}$  has *built-in equality* if, for every sort  $s$ , it has a function declaration  $eq : s \times s \rightarrow B$ , such that for all terms  $t_1$  and  $t_2$  of sort  $s$ , it holds that  $\mathcal{D} \models t_1 = t_2$  if and only if  $\mathcal{D} \models eq(t_1, t_2) = \top$ . A data algebra  $\mathcal{D}$  has *built-in Skolem functions* if for every first-order formula  $\phi$  with free variables  $\{x, y_1, \dots, y_n\}$  there exists a term  $t_\phi(y_1, \dots, y_n)$  such that  $\mathcal{D} \models (\exists x)\phi$  implies  $\mathcal{D} \models \phi(t_\phi(y_1, \dots, y_n), y_1, \dots, y_n)$ .

In [27, 57], injections are given between an equation in  $p\text{CRL}$  and a formula in the first-order theory with the same data algebra in such a way that for every equation  $p = q$  there exists a formula  $\phi$  such that for all valuations  $\nu$  of the data variables:  $\nu \models p = q$  iff  $\mathcal{D}, \nu \models \phi$  and vice versa. This formula  $\phi$  is then used in the completeness proof through a Boolean expression  $b$  such that  $\mathcal{D}, \nu \models b = \top$  iff  $\mathcal{D}, \nu \models \phi$ . From the assumptions that the data algebra has built-in equality and built-in Skolem functions it follows that such a Boolean expression exists. For a more detailed treatment of these notions we refer to [27, 57].

As  $p\text{CRL}_t$  is an extension of  $p\text{CRL}$  we also need these assumptions about the data sorts for the completeness of  $p\text{CRL}_t$ . In fact, as  $p\text{CRL}_t$  is a syntactic extension of  $p\text{CRL}$ , the proof of completeness

for  $p\text{CRL}_t$  contains the proof of completeness of  $p\text{CRL}$ . We do not repeat the proof presented in [27, 57], but formulate our proof of completeness in such a way that the proof of completeness of  $p\text{CRL}$  is reused.

#### 4.1. Well-timedness and deadlock-saturation

The main difference between  $p\text{CRL}$  and  $p\text{CRL}_t$  is the addition of the at-operator. This can easily be seen by considering the basic terms. We can even conclude that the only difference is the application of the at-operator on atomic actions (including deadlock). To reuse the proof of completeness of  $p\text{CRL}$ , we translate timed process terms as untimed process terms in such a way that strongly timed bisimilar process terms are strongly bisimilar after translating them, and vice versa. Furthermore, two untimed process terms that are derivably equal in  $p\text{CRL}$  are also derivably equal after applying the inverse of the translation.

The only way to translate the timed process terms as untimed process terms is to consider the time assignment to atomic actions as an additional data parameter of the atomic action. For example, the timed atomic action  $\mathbf{a}(d)\cdot 3$  might be translated to the untimed atomic action  $\mathbf{a}(d, 3)$ . There are, however, two problems with such a translation.

First, there is the problem of ill-timedness. In  $p\text{CRL}_t$  we have the following identity:

$$\mathbf{a}(d)\cdot 3 \cdot (\mathbf{b}(e)\cdot 2 + \mathbf{c}(f)\cdot 4) = \mathbf{a}(d)\cdot 3 \cdot \mathbf{c}(f)\cdot 4.$$

Translating the timing assignment as an additional parameter yields the identity

$$\mathbf{a}(d, 3) \cdot (\mathbf{b}(e, 2) + \mathbf{c}(f, 4)) = \mathbf{a}(d, 3) \cdot \mathbf{c}(f, 4).$$

This identity does not hold in  $p\text{CRL}$ . The reason for this is that timing assignments in  $p\text{CRL}_t$  influence the behaviour in such a way that data parameters attached to atomic action cannot simulate in  $p\text{CRL}$ . Our solution to this problem is the introduction of the notion of well-timedness. We will prove that for every basic term there exists a derivably equal well-timed basic term.

Second, there is the more technical problem that deadlock cannot have a parameter in  $p\text{CRL}$ . Hence it is not clear how to translate  $\delta\cdot 3$ . We cannot simply translate  $\delta\cdot 3$  as  $\delta$  since in that case  $\delta\cdot 3$  and  $\delta\cdot 4$  are translated both as  $\delta$ . Our solution to this problem comes from the following observation. In a timed process term, there are many time deadlocks implicitly included. For example, the process term  $a\cdot 3$  implicitly has time deadlock summands  $\delta\cdot t$  for each  $t \leq 3$ . If we can succeed in making all implicit deadlocks explicitly visible then we can translate a time deadlock as a special atomic action with one data parameter representing the time assignment of that deadlock. That is, we will translate  $\delta\cdot 3$  as  $\Delta(3)$ , where  $\Delta$  is this new atomic action. Making all implicit time deadlocks explicitly available is called deadlock-saturation. We show that for each basic term a deadlock-saturated basic terms exists that is derivably equal, and hence, that the restriction to deadlock-saturated basic terms is allowed.

By restricting our attention to deadlock-saturated well-timed basic terms we can prove that the translation sketched above satisfies the criteria mentioned.

#### Definition 4.2. (Well-timedness)

The set of well-timed basic terms is inductively defined as follows:

1.  $\sum_{\vec{a}} a\cdot t \triangleleft b \triangleright \delta\cdot \mathbf{0}$  is well-timed;
2. if  $r$  is a well-timed basic term and  $t \gg r = r$ , then  $\sum_{\vec{a}} a\cdot t \cdot r \triangleleft b \triangleright \delta\cdot \mathbf{0}$  is well-timed;

3. if  $r$  and  $r'$  are well-timed basic terms, then  $r + r'$  is well-timed.

The following theorem states that every basic term is derivably equal to a well-timed basic term. Combined with the Basic Term Theorem (Theorem 2.1) we thus have that every process-closed term is derivably equal to a well-timed basic term. Thus, in cases where we want to prove an equality over process-closed terms, we can restrict ourselves to a proof that the equality holds for well-timed basic terms.

**Theorem 4.1. (Well-timedness)**

For every basic term  $r$  there exists a well-timed basic term  $s$  such that  $r = s$ .

**Proof:**

We prove this theorem by induction on the structure of basic term  $r$ .

1.  $r \equiv \sum_{\bar{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then  $r$  is well-timed by definition.
2.  $r \equiv \sum_{\bar{v}} a^c u \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . By induction we have the existence of a well-timed basic term  $s'$  such that  $r' = s'$ . First we prove by induction on the structure of well-timed basic term  $r$ , that for every well-timed basic term  $r$  and term  $t$  of sort  $T$ , there exists a well-timed basic term  $s$  such that  $t \gg r = s$ .

- (a)  $r \equiv \sum_{\bar{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned}
t \gg r &= t \gg (\sum_{\bar{v}} a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB4}}{=} \sum_{\bar{v}} t \gg (a^c u \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB5}}{=} \sum_{\bar{v}} (t \gg a^c u) \triangleleft b \triangleright (t \gg \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB1,2.4.2}}{=} \sum_{\bar{v}} (a^c u \triangleleft t \leq u \triangleright \delta \cdot t) \triangleleft b \triangleright \delta \cdot t \\
&\stackrel{\text{C4}}{=} \sum_{\bar{v}} a^c u \triangleleft t \leq u \wedge b \triangleright \delta \cdot t \\
&\stackrel{\text{C3,SUM4}}{=} \sum_{\bar{v}} a^c u \triangleleft t \leq u \wedge b \triangleright \delta \cdot \mathbf{0} + \sum_{\bar{v}} \delta \cdot t \triangleleft \neg(t \leq u \wedge b) \triangleright \delta \cdot \mathbf{0},
\end{aligned}$$

which is a well-timed basic term.

- (b)  $r \equiv \sum_{\bar{v}} a^c u \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . As  $r$  is well-timed we have  $u \gg r' = r'$ . Then,

$$\begin{aligned}
t \gg r &= t \gg (\sum_{\bar{v}} a^c u \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB4}}{=} \sum_{\bar{v}} t \gg (a^c u \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB5,ATB3}}{=} \sum_{\bar{v}} (t \gg a^c u) \cdot r' \triangleleft b \triangleright (t \gg \delta \cdot \mathbf{0}) \\
&\stackrel{\text{ATB1,2.4.2}}{=} \sum_{\bar{v}} (a^c u \triangleleft t \leq u \triangleright \delta \cdot t) \cdot r' \triangleleft b \triangleright \delta \cdot t \\
&\stackrel{\text{C6,ATA3,A7}}{=} \sum_{\bar{v}} (a^c u \cdot r' \triangleleft t \leq u \triangleright \delta \cdot t) \triangleleft b \triangleright \delta \cdot t \\
&\stackrel{\text{C4}}{=} \sum_{\bar{v}} a^c u \cdot r' \triangleleft t \leq u \wedge b \triangleright \delta \cdot t \\
&\stackrel{\text{C3,SUM4}}{=} \sum_{\bar{v}} a^c u \cdot r' \triangleleft t \leq u \wedge b \triangleright \delta \cdot \mathbf{0} + \sum_{\bar{v}} \delta \cdot t \triangleleft \neg(t \leq u \wedge b) \triangleright \delta \cdot \mathbf{0}.
\end{aligned}$$

Obviously, this is a well-timed basic term.

- (c)  $r \equiv r' + r''$ . By induction we have the existence of well-timed basic terms  $s'$  and  $s''$  such that  $t \gg r' = s'$  and  $t \gg r'' = s''$ . Then,  $t \gg r = t \gg (r' + r'') \stackrel{\text{ATB2}}{=} t \gg r' + t \gg r'' = s' + s''$  which is a well-timed basic term.

Now, we have the existence of a well-timed basic term  $s''$  such that  $u \gg s' = s''$ . Then,

$$\begin{aligned} r &= \sum_{\vec{v}} a^c u \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &= \sum_{\vec{v}} a^c u \cdot s' \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{\text{AT2}}{=} \sum_{\vec{v}} a^c u \cdot u \gg s' \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &= \sum_{\vec{v}} a^c u \cdot s'' \triangleleft b \triangleright \delta \cdot \mathbf{0}. \end{aligned}$$

By Lemma 2.6 we have that  $u \gg s'' = u \gg (u \gg s') = u \gg s' = s''$ , and hence the above term is well-timed.

3.  $r \equiv r' + r''$ . By induction we have the existence of well-timed basic terms  $s'$  and  $s''$  such that  $r' = s'$  and  $r'' = s''$ . Thus,  $r = r' + r'' = s' + s''$  which is a well-timed basic term.  $\square$

The fact that we can restrict ourselves to well-timed basic terms instead of arbitrary basic terms brings us closer to our goal, the completeness result. As explained before we wish to embed timed processes into untimed processes. One of the major difficulties in this embedding is that time restricts behaviour. For well-timed basic terms, however, time is reduced to just an annotation with an action term. In the embedding we will simply add it to the parameter list of a corresponding action term.

In this section, we will present an even more specific notion of basic terms. To each basic term as many deadlocks will be added as possible while keeping the result equal to the original. The reason for this deadlock-saturation is that later we will embed the timed process terms into untimed process terms by replacing timed deadlocks by a special atomic action with a parameter representing the time stamp.

#### Definition 4.3. (Deadlock-saturation)

The set of deadlock-saturated basic terms is inductively defined as follows:

1.  $\sum_{\vec{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0} + \sum_{\vec{v}, u} \delta^c u \triangleleft u \leq t \wedge b \triangleright \delta \cdot \mathbf{0}$  is a deadlock-saturated basic term;
2. if  $r$  is a deadlock-saturated basic term, then  $\sum_{\vec{v}} a^c t \cdot r \triangleleft b \triangleright \delta \cdot \mathbf{0} + \sum_{\vec{v}, u} \delta^c u \triangleleft u \leq t \wedge b \triangleright \delta \cdot \mathbf{0}$  is a deadlock-saturated basic term;
3. if  $r$  and  $r'$  are deadlock-saturated basic terms, then  $r + r'$  is a deadlock-saturated basic term.

The following result states that every basic term can be transformed into a deadlock-saturated basic term. Hence, it is allowed to restrict ourselves to deadlock-saturated basic terms in cases where we are only interested in closed terms.

**Theorem 4.2.** For every basic term  $p$  there exists a deadlock-saturated basic term  $q$  such that  $p\text{CRL}_t \vdash p = q$ .

#### Proof:

We prove this theorem by induction on the structure of basic term  $p$ .

1.  $p \equiv \sum_{\vec{v}} a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0}$ . Let  $q = \sum_{\vec{v}} a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_{\vec{v}, u} \delta^{\circ} u \triangleleft u \leq t \wedge b \triangleright \delta^{\circ} \mathbf{0}$  where  $u$  is a fresh variable. Then,

$$\begin{aligned}
q &= \sum_{\vec{v}} a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_{\vec{v}, u} \delta^{\circ} u \triangleleft u \leq t \wedge b \triangleright \delta^{\circ} \mathbf{0} \\
&\stackrel{\text{SUM4}}{=} \sum_{\vec{v}} (a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_u \delta^{\circ} u \triangleleft u \leq t \wedge b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{\text{C4}}{=} \sum_{\vec{v}} (a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_u (\delta^{\circ} u \triangleleft u \leq t \triangleright \delta^{\circ} \mathbf{0}) \triangleleft b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{\text{SUM12}'}{=} \sum_{\vec{v}} (a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + (\sum_u (\delta^{\circ} u \triangleleft u \leq t \triangleright \delta^{\circ} \mathbf{0})) \triangleleft b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{2.5}{=} \sum_{\vec{v}} (a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \delta^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{\text{C7}}{=} \sum_{\vec{v}} (a^{\circ} t + \delta^{\circ} t) \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&\stackrel{\text{ATA2}}{=} \sum_{\vec{v}} (a + \delta)^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&\stackrel{\text{A6}^-}{=} \sum_{\vec{v}} a^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&= p.
\end{aligned}$$

2.  $p \equiv \sum_{\vec{v}} a^{\circ} t \cdot p' \triangleleft b \triangleright \delta^{\circ} \mathbf{0}$ . By induction we have the existence of a deadlock-saturated basic term  $q'$  such that  $p\text{CRL}_t \vdash p' = q'$ . Let  $q = \sum_{\vec{v}} a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_{\vec{v}, u} \delta^{\circ} u \triangleleft u \leq t \wedge b \triangleright \delta^{\circ} \mathbf{0}$  where  $u$  is a fresh variable. Then,

$$\begin{aligned}
q &= \sum_{\vec{v}} a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_{\vec{v}, u} \delta^{\circ} u \triangleleft u \leq t \wedge b \triangleright \delta^{\circ} \mathbf{0} \\
&\stackrel{\text{SUM4}}{=} \sum_{\vec{v}} (a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_u \delta^{\circ} u \triangleleft u \leq t \wedge b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{\text{C4}}{=} \sum_{\vec{v}} (a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \sum_u (\delta^{\circ} u \triangleleft u \leq t \triangleright \delta^{\circ} \mathbf{0}) \triangleleft b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{\text{SUM12}'}{=} \sum_{\vec{v}} (a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + (\sum_u (\delta^{\circ} u \triangleleft u \leq t \triangleright \delta^{\circ} \mathbf{0})) \triangleleft b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{2.5}{=} \sum_{\vec{v}} (a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} + \delta^{\circ} t \triangleleft b \triangleright \delta^{\circ} \mathbf{0}) \\
&\stackrel{\text{C7}}{=} \sum_{\vec{v}} (a^{\circ} t \cdot q' + \delta^{\circ} t) \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&\stackrel{\text{ATA3, ATA2, A7, A4}}{=} \sum_{\vec{v}} ((a + \delta)^{\circ} t) \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&\stackrel{\text{A6}^-, \text{ATA3}}{=} \sum_{\vec{v}} a^{\circ} t \cdot q' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&= \sum_{\vec{v}} a^{\circ} t \cdot p' \triangleleft b \triangleright \delta^{\circ} \mathbf{0} \\
&= p.
\end{aligned}$$

3.  $p \equiv p' + p''$ . By induction we have the existence of deadlock-saturated basic terms  $q'$  and  $q''$  such that  $p\text{CRL}_t \vdash p' = q'$  and  $p\text{CRL}_t \vdash p'' = q''$ . Then,  $p\text{CRL}_t \vdash p = p' + p'' = q' + q''$ . Obviously,  $q' + q''$  is a deadlock-saturated basic term.  $\square$

As a consequence of the Theorems 4.1 and 4.2, for each basic term there exists a derivably equal deadlock-saturated well-timed basic term.

**Corollary 4.1.** For every basic term  $p$  there exists a deadlock-saturated, well-timed basic term  $q$  such that  $p\text{CRL}_t \vdash p = q$ .

**Proof:**

By Theorem 4.1 there exists a well-timed basic term  $p'$  such that  $p\text{CRL}_t \vdash p = p'$ . Obviously  $p'$  itself is a basic term. Hence, by Theorem 4.2, there exists a deadlock-saturated basic term  $q$  such that  $p\text{CRL}_t \vdash p' = q$ . By construction deadlock-saturation preserves well-timedness. Hence,  $q$  is both well-timed and deadlock-saturated.  $\square$

**4.2. Time-free abstraction**

Next, we present an embedding of well-timed basic terms into basic terms in the theory  $p\text{CRL}$ . Thereto, we assume that for every action declaration  $\mathbf{a} : s_1 \times \cdots \times s_n$  in  $p\text{CRL}_t$ , we have an action declaration  $\mathbf{a} : s_1 \times \cdots \times s_n \times T$  in  $p\text{CRL}$ . We prove that this embedding preserves strong bisimilarity in the sense that the embeddings of two strongly timed bisimilar process terms are strongly bisimilar.

**Definition 4.4. (Time-free abstraction)**

The mapping  $\pi_{\text{tf}}$  on basic terms is inductively defined as follows:

$$\begin{aligned} \pi_{\text{tf}}(\sum_{\vec{v}} \mathbf{a}(\vec{e}) \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) &= \sum_{\vec{v}} \mathbf{a}(\vec{e}, t) \triangleleft b \triangleright \delta, \\ \pi_{\text{tf}}(\sum_{\vec{v}} \delta \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) &= \sum_{\vec{v}} \Delta(t) \triangleleft b \triangleright \delta, \\ \pi_{\text{tf}}(\sum_{\vec{v}} \mathbf{a}(\vec{e}) \cdot t \cdot r \triangleleft b \triangleright \delta \cdot \mathbf{0}) &= \sum_{\vec{v}} \mathbf{a}(\vec{e}, t) \cdot \pi_{\text{tf}}(r) \triangleleft b \triangleright \delta, \\ \pi_{\text{tf}}(r + r') &= \pi_{\text{tf}}(r) + \pi_{\text{tf}}(r'). \end{aligned}$$

**Theorem 4.3.** For deadlock-saturated well-timed basic terms  $p$  and  $q$  and arbitrary valuations  $\nu$  and  $\xi$ : if  $\llbracket p \rrbracket^\nu \Leftrightarrow_t \llbracket q \rrbracket^\xi$ , then  $\llbracket \pi_{\text{tf}}(p) \rrbracket^\nu \Leftrightarrow \llbracket \pi_{\text{tf}}(q) \rrbracket^\xi$ .

**Proof:**

We prove this theorem by induction on the number of symbols of  $p$  and  $q$ . Suppose that  $\llbracket p \rrbracket^\nu \Leftrightarrow_t \llbracket q \rrbracket^\xi$  for some valuations  $\nu$  and  $\xi$ . Let  $R$  be defined as follows: if  $\llbracket p \rrbracket^\nu \Leftrightarrow_t \llbracket q \rrbracket^\xi$  for some deadlock-saturated well-timed basic terms  $p$  and  $q$  and arbitrary valuations  $\nu$  and  $\xi$ , then  $(\llbracket \pi_{\text{tf}}(p) \rrbracket^\nu, \llbracket \pi_{\text{tf}}(q) \rrbracket^\xi) \in R$ .

By induction on the structure of basic term  $p$  we can easily prove that the transitions of the process associated with basic term  $p$  under valuation  $\nu$  are of one of the following forms:

1.  $\llbracket p \rrbracket^\nu \xrightarrow{a}_t \checkmark$ ;
2.  $\llbracket p \rrbracket^\nu \xrightarrow{a}_t t \gg \llbracket p' \rrbracket^{\nu'}$  for some deadlock-saturated well-timed basic term  $p'$  and valuation  $\nu'$ .

Now, we first prove the following lemmata: for arbitrary deadlock-saturated well-timed basic terms  $p$  and  $p'$ ,  $\mathbf{a} \in Act$ ,  $t \in \mathcal{D}_T$ , and valuations  $\chi$  and  $\chi'$

1. if  $\mathbf{a} \neq \Delta$  and  $\llbracket \pi_{\text{tf}}(p) \rrbracket^\chi \xrightarrow{\mathbf{a}(\vec{d}, t)} \checkmark$ , then  $\llbracket p \rrbracket^\chi \xrightarrow{\mathbf{a}(\vec{d})}_t \checkmark$ ;
2. if  $\llbracket p \rrbracket^\chi \xrightarrow{\mathbf{a}(\vec{d})}_t \checkmark$ , then  $\llbracket \pi_{\text{tf}}(p) \rrbracket^\chi \xrightarrow{\mathbf{a}(\vec{d}, t)} \checkmark$ ;
3. if  $\llbracket \pi_{\text{tf}}(p) \rrbracket^\chi \xrightarrow{\Delta(t)} \checkmark$ , then  $U_t(\llbracket p \rrbracket^\chi)$ ;
4. if  $U_t(\llbracket p \rrbracket^\chi)$ , then  $\llbracket \pi_{\text{tf}}(p) \rrbracket^\chi \xrightarrow{\Delta(t)} \checkmark$ ;

5. if  $\llbracket \pi_{\text{tf}}(p) \rrbracket^{\chi} \xrightarrow{\mathbf{a}(\vec{d}, t)} \llbracket \pi_{\text{tf}}(p') \rrbracket^{\chi'}$ , then  $\llbracket p \rrbracket^{\chi} \xrightarrow{\mathbf{a}(\vec{d})} t \gg \llbracket p' \rrbracket^{\chi'}$  and  $t \gg \llbracket p' \rrbracket^{\chi'} \Leftrightarrow_t \llbracket p' \rrbracket^{\chi'}$ ;
6. if  $\llbracket p \rrbracket^{\chi} \xrightarrow{\mathbf{a}(\vec{d})} t \gg \llbracket p' \rrbracket^{\chi'}$ , then  $\llbracket \pi_{\text{tf}}(p) \rrbracket^{\chi} \xrightarrow{\mathbf{a}(\vec{d}, t)} \llbracket \pi_{\text{tf}}(p') \rrbracket^{\chi'}$ .

Next, we prove that  $R$  is a strong timed bisimulation. First, suppose that  $\llbracket \pi_{\text{tf}}(p) \rrbracket^{\nu} \xrightarrow{\mathbf{a}(\vec{d}, t)} \surd$ . If  $\mathbf{a} = \Delta$ , then  $U_t(\llbracket p \rrbracket^{\nu})$ . Therefore,  $U_t(\llbracket q \rrbracket^{\xi})$ . Thus,  $\llbracket \pi_{\text{tf}}(q) \rrbracket^{\xi} \xrightarrow{\Delta(t)} \surd$ . If  $\mathbf{a} \neq \Delta$ , then  $\llbracket p \rrbracket^{\nu} \xrightarrow{\mathbf{a}(\vec{d})} t \gg \surd$ . Therefore,  $\llbracket q \rrbracket^{\xi} \xrightarrow{\mathbf{a}(\vec{d})} t \gg \surd$ . Thus,  $\llbracket \pi_{\text{tf}}(q) \rrbracket^{\xi} \xrightarrow{\mathbf{a}(\vec{d}, t)} \surd$ .

Second, suppose that  $\llbracket \pi_{\text{tf}}(p) \rrbracket^{\nu} \xrightarrow{\mathbf{a}(\vec{d}, t)} \llbracket \pi_{\text{tf}}(p') \rrbracket^{\nu'}$  for some deadlock-saturated well-timed basic term  $p'$  and valuation  $\nu'$ . Then  $\llbracket p \rrbracket^{\nu} \xrightarrow{\mathbf{a}(\vec{d})} t \gg \llbracket p' \rrbracket^{\nu'}$ . Therefore,  $\llbracket q \rrbracket^{\xi} \xrightarrow{\mathbf{a}(\vec{d})} t \gg \llbracket q' \rrbracket^{\xi'}$  for some  $q'$  and  $\xi'$  such that  $t \gg \llbracket p' \rrbracket^{\nu'} \Leftrightarrow_t t \gg \llbracket q' \rrbracket^{\xi'}$ . Thus,  $\llbracket \pi_{\text{tf}}(q) \rrbracket^{\xi} \xrightarrow{\mathbf{a}(\vec{d}, t)} \llbracket \pi_{\text{tf}}(q') \rrbracket^{\xi'}$ . As  $p$  and  $q$  are well-timed we also have  $t \gg \llbracket p' \rrbracket^{\nu'} \Leftrightarrow_t \llbracket p' \rrbracket^{\nu'}$  and  $t \gg \llbracket q' \rrbracket^{\xi'} \Leftrightarrow_t \llbracket q' \rrbracket^{\xi'}$ . Thus,  $(\llbracket \pi_{\text{tf}}(p) \rrbracket^{\nu'}, \llbracket \pi_{\text{tf}}(q) \rrbracket^{\xi'}) \in R$ .  $\square$

Next, we present a mapping  $\pi_t$  from  $p\text{CRL}$ -terms to  $p\text{CRL}_t$ -terms and we show that this mapping is the inverse of the mapping  $\pi_{\text{tf}}$  for basic terms.

**Definition 4.5.** The mapping  $\pi_t$  is defined as follows:

$$\begin{array}{ll}
\pi_t(x) & = x, & \pi_t(\Delta) & = \delta \cdot \mathbf{0}, \\
\pi_t(\delta) & = \delta \cdot \mathbf{0}, & \pi_t(\Delta(t)) & = \delta \cdot t, \\
\pi_t(\mathbf{a}) & = \delta \cdot \mathbf{0}, & \pi_t(\Delta(\vec{d}, t)) & = \delta \cdot \mathbf{0}, \\
\pi_t(\mathbf{a}(\vec{d}, t)) & = \mathbf{a}(\vec{d}) \cdot t, & \pi_t(p \triangleleft b \triangleright q) & = \pi_t(p) \triangleleft b \triangleright \pi_t(q), \\
\pi_t(p + q) & = \pi_t(p) + \pi_t(q), & \pi_t(\sum_v p) & = \sum_v \pi_t(p). \\
\pi_t(p \cdot q) & = \pi_t(p) \cdot \pi_t(q), & & 
\end{array}$$

The mapping  $\pi_t$  replaces every atomic action that occurs in the process term to which it is applied by a similarly named atomic action with the last data parameter removed and added as a timing assignment. Thus,  $\pi_t(\mathbf{a}(\text{blue}, 7) \cdot \mathbf{b}(9)) = \mathbf{a}(\text{blue}) \cdot 7 \cdot \mathbf{b} \cdot 9$ . Observe that atomic actions without data parameters and  $\Delta$ -occurrences with more than one data parameter are replaced by  $\delta \cdot \mathbf{0}$ . This is only done to make the mapping  $\pi_t$  a total mapping. The mapping  $\pi_t$  is to be applied only to untimed process terms that result from applying  $\pi_{\text{tf}}$  to timed process terms. In such terms none of the mentioned atomic actions can occur. Also observe that as the deadlock constant can be added as an alternative to each untimed process term, it is necessary that the mapping  $\pi_t$  replaces it by the timed process term  $\delta \cdot \mathbf{0}$  since that is the neutral element for alternative composition in timed  $\mu\text{CRL}$ .

**Lemma 4.1.** For basic terms  $p$  we have  $p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p)) = p$ .

**Proof:**

We prove this lemma by induction on the structure of basic term  $p$ .

1.  $p \equiv \sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned}
p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p)) &= \pi_t(\pi_{\text{tf}}(\sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0})) \\
&= \pi_t(\sum_{\vec{v}} \mathbf{a}(\vec{d}, t) \triangleleft b \triangleright \delta) \\
&= \sum_{\vec{v}} \pi_t(\mathbf{a}(\vec{d}, t) \triangleleft b \triangleright \delta) \\
&= \sum_{\vec{v}} \pi_t(\mathbf{a}(\vec{d}, t)) \triangleleft b \triangleright \pi_t(\delta) \\
&= \sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&= p.
\end{aligned}$$

2.  $p \equiv \sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . By induction we have  $p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p')) = p'$ . Then,

$$\begin{aligned}
p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p)) &= \pi_t(\pi_{\text{tf}}(\sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0})) \\
&= \pi_t(\sum_{\vec{v}} \mathbf{a}(\vec{d}, t) \cdot \pi_{\text{tf}}(p') \triangleleft b \triangleright \delta) \\
&= \sum_{\vec{v}} \pi_t(\mathbf{a}(\vec{d}, t) \cdot \pi_{\text{tf}}(p') \triangleleft b \triangleright \delta) \\
&= \sum_{\vec{v}} \pi_t(\mathbf{a}(\vec{d}, t) \cdot \pi_{\text{tf}}(p')) \triangleleft b \triangleright \pi_t(\delta) \\
&= \sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \cdot \pi_t(\pi_{\text{tf}}(p')) \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&= \sum_{\vec{v}} \mathbf{a}(\vec{d}) \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&= p.
\end{aligned}$$

3.  $p \equiv p' + p''$ . By induction we have  $p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p')) = p'$  and  $p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p'')) = p''$ . Then,

$$\begin{aligned}
p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p)) &= \pi_t(\pi_{\text{tf}}(p' + p'')) \\
&= \pi_t(\pi_{\text{tf}}(p') + \pi_{\text{tf}}(p'')) \\
&= \pi_t(\pi_{\text{tf}}(p')) + \pi_t(\pi_{\text{tf}}(p'')) \\
&= p' + p'' \\
&= p.
\end{aligned}$$

□

The following result indicates that the mapping  $\pi_t$  preserves derivability.

**Lemma 4.2.** For  $p\text{CRL}$ -terms  $p$  and  $q$  we have  $p\text{CRL} \vdash p = q \implies p\text{CRL}_t \vdash \pi_t(p) = \pi_t(q)$ .

**Proof:**

As the proof systems underlying  $p\text{CRL}$  and  $p\text{CRL}_t$  are identical, all we have to prove is that the  $\pi_t$ -image of every axiom of  $p\text{CRL}$  is derivable from the axioms of  $p\text{CRL}_t$ . The axioms of  $p\text{CRL}$  are given in Table 10 in Appendix A. This is easily established. Consider for example axiom A7. We must prove that  $p\text{CRL}_t \vdash \pi_t(\delta \cdot x) = \pi_t(\delta)$ . This is not difficult:

$$\begin{aligned}
p\text{CRL}_t \vdash \pi_t(\delta \cdot x) &= \pi_t(\delta) \cdot \pi_t(x) \\
&= \delta \cdot \mathbf{0} \cdot x \stackrel{\text{ATA3}}{=} (\delta \cdot x) \cdot \mathbf{0} \stackrel{\text{A7}}{=} \delta \cdot \mathbf{0} \\
&= \pi_t(\delta).
\end{aligned}$$

As another example consider axiom Cond6:

$$\begin{aligned}
p\text{CRL}_t \vdash \pi_t((x \triangleleft b \triangleright \delta) \cdot y) &= \pi_t(x \triangleleft b \triangleright \delta) \cdot \pi_t(y) \\
&= (\pi_t(x) \triangleleft b \triangleright \pi_t(\delta)) \cdot \pi_t(y) \\
&= (x \triangleleft b \triangleright \delta \cdot \mathbf{0}) \cdot y \stackrel{\text{C6}}{=} x \cdot y \triangleleft \delta \cdot \mathbf{0} \cdot y \\
&\stackrel{\text{ATA3}}{=} x \cdot y \triangleleft (\delta \cdot y) \cdot \mathbf{0} \\
&\stackrel{\text{A7}}{=} x \cdot y \triangleleft b \triangleright \delta \cdot \mathbf{0} \\
&= \pi_t(x) \cdot \pi_t(y) \triangleleft b \triangleright \pi_t(\delta) \\
&= \pi_t(x \cdot y) \triangleleft b \triangleright \pi_t(\delta) \\
&= \pi_t(x \cdot y \triangleleft b \triangleright \delta).
\end{aligned}$$

□

#### Theorem 4.4. (Relative completeness)

With respect to strong timed bisimulation  $p\text{CRL}_t$  is a complete axiom system for process-closed process terms under the assumptions that the data algebra is  $\omega$ -completely axiomatised and that the data algebra has built-in equality and Skolem functions.

#### Proof:

Without loss of generality we may assume by Corollary 4.1 that  $p$  and  $q$  are deadlock-saturated well-timed basic terms. We use the completeness result of [27, 57] for untimed  $p\text{CRL}$ . Suppose that  $\llbracket p \rrbracket^\nu \Leftrightarrow_t \llbracket q \rrbracket^\nu$  for all valuations  $\nu$ . Then, by Theorem 4.3, we have  $\llbracket \pi_{\text{tf}}(p) \rrbracket^\nu \Leftrightarrow \llbracket \pi_{\text{tf}}(q) \rrbracket^\nu$  for all valuations  $\nu$ . Here we use the completeness of  $p\text{CRL}$  to obtain  $p\text{CRL} \vdash \pi_{\text{tf}}(p) = \pi_{\text{tf}}(q)$ . By Lemma 4.2 we thus have  $p\text{CRL}_t \vdash \pi_t(\pi_{\text{tf}}(p)) = \pi_t(\pi_{\text{tf}}(q))$ . Using Lemma 4.1 we have  $p\text{CRL}_t \vdash p = q$ . □

## 5. The axiom system $\mu\text{CRL}_t$

One of the major strengths of timed  $\mu\text{CRL}$  is that it is possible to specify time-dependent, parallel processes with data. In this section, we incorporate operators for parallelism in the language and introduce the axiom system  $\mu\text{CRL}_t$ .

### 5.1. Axioms for parallelism

Concurrency is described by the binary operator  $\parallel$ . A process  $p \parallel q$ , the parallel execution of  $p$  and  $q$ , can first perform an action of  $p$ , first perform an action of  $q$ , or start with a communication, or synchronisation, between  $p$  and  $q$ . Hence, the first axiom for this operator is  $x \parallel y = x \parallel y + y \parallel x + x \mid y$  (see axiom CM1 in Table 5), where the auxiliary *left merge* operators  $\parallel$  and the *communication merge*  $\mid$  are defined below. The process  $p \parallel q$  exists at time  $t$  only if both  $p$  and  $q$  exist at time  $t$ .

The process  $p \parallel q$  is as  $p \parallel q$ , but the first action that is performed comes from  $p$ . As was the case for parallel composition, the action can only be performed if the other party still exists at that time. For the axiomatisation of the left merge  $\parallel$  the auxiliary *before*-operator is defined;  $p \ll q$  should be interpreted as the part of process  $p$  that starts before  $q$  gets definitely disabled. Otherwise  $p \ll q$  becomes a time deadlock at time  $t_0$ . Although the operator differs slightly from the similarly named operator in [3], we have decided to give it the same symbol. A small example may facilitate the understanding of the

“ $\ll$ ”-operator. For example, the process represented by  $(a \cdot 2 + b \cdot 4) \ll c \cdot 3$  cannot choose to execute action  $b$  at time 4 since its first action must be executed not later than time 3. Using the axioms we can derive that  $(a \cdot 2 + b \cdot 4) \ll c \cdot 3 = a \cdot 2$ .

The process  $p \mid q$  also behaves as the process  $p \parallel q$ , except that the first action must be a communication between its left and right operand. Furthermore, these actions must occur at the same time. The action resulting from such a communication is defined by the binary, commutative and associative function  $\gamma$ , which is only defined on *action declarations*. In order for a communication to occur between action terms  $\mathbf{a}(d), \mathbf{a}'(e) \in AT$ ,  $\gamma(\mathbf{a}, \mathbf{a}')$  should be defined, and the data parameters  $d$  and  $e$  of the action terms should match according to axiom CF in Table 5.

CM1	$x \parallel y = x \parallel y + y \parallel x + x \mid y$
CM2	$a \cdot t \parallel x = (a \cdot t \ll x) \cdot x$
CM3	$a \cdot t \cdot x \parallel y = (a \cdot t \ll y) \cdot (t \gg x \parallel y)$
CM4	$(x + y) \parallel z = x \parallel z + y \parallel z$
SUM6	$(\sum_v p) \parallel x = \sum_v (p \parallel x)$
H18	$(x \triangleleft b \triangleright y) \parallel z = (x \parallel z) \triangleleft b \triangleright (y \parallel z)$
CF	$\mathbf{a}(\vec{d}) \mid \mathbf{a}'(\vec{e}) = \begin{cases} \gamma(\mathbf{a}, \mathbf{a}')(\vec{d}) \triangleleft eq(\vec{d}, \vec{e}) \triangleright \delta \\ \text{if } \gamma(\mathbf{a}, \mathbf{a}') \text{ defined} \\ \delta \text{ otherwise} \end{cases}$
CD1	$\delta \mid a = \delta$
CD2	$a \mid \delta = \delta$
CM5	$a \cdot x \mid a' = (a \mid a') \cdot x$
CM6	$a \mid a' \cdot x = (a \mid a') \cdot x$
CM7	$a \cdot x \mid a' \cdot y = (a \mid a') \cdot (x \parallel y)$
CM8	$(x + y) \mid z = x \mid z + y \mid z$
CM9	$x \mid (y + z) = x \mid y + x \mid z$
ATA7	$(x \mid y) \cdot t = x \cdot t \mid y$
ATA8	$(x \mid y) \cdot t = x \mid y \cdot t$
SUM7	$(\sum_v p) \mid x = \sum_v (p \mid x)$
SUM7'	$x \mid (\sum_v p) = \sum_v (x \mid p)$
H8	$x \mid (y \triangleleft b \triangleright z) = (x \mid y) \triangleleft b \triangleright (x \mid z)$
H8'	$(x \triangleleft b \triangleright y) \mid z = (x \mid z) \triangleleft b \triangleright (y \mid z)$

Table 5. Axioms for parallelism of  $\mu$ CRL<sub>t</sub>, where  $x, y, z \in V_P$ , process-closed  $p \in T_P$ ,  $a, a' \in AT_\delta$ ,  $\mathbf{a}, \mathbf{a}' \in Act$ ,  $b \in V_B$ ,  $t \in V_T$ ,  $v \in V$  and  $H \subseteq Act$ .

The axioms for these operators are given in Table 5 and Table 6. These axioms are designed to eliminate the parallel operators in favour of the alternative and the sequential operator.

Sometimes we want to express that certain actions cannot happen, and must be blocked. Generally, this is only done when we want to force actions into a communication. The encapsulation operator  $\partial_H$  ( $H \subseteq \text{Act}$ ) is specially designed for this task. In  $\partial_H(p)$  it prevents all actions of which the action declaration is mentioned in  $H$  from happening by renaming the corresponding action terms into  $\delta$ . Consider, for example, the process  $\mathbf{a}\|\mathbf{b} = \mathbf{a} \cdot \mathbf{b} + \mathbf{b} \cdot \mathbf{a} + \mathbf{c}$ . Often, this is not quite what is desired, as the intention generally is that  $\mathbf{a}$  and  $\mathbf{b}$  do not happen separately. Therefore, the encapsulation operator can be used. The process represented by  $\partial_{\{\mathbf{a}, \mathbf{b}\}}(\mathbf{a}\|\mathbf{b})$  is equal to  $\mathbf{c}$ .

DD	$\partial_H(\delta) = \delta$	
D1	$\partial_H(\mathbf{a}(\vec{d})) = \mathbf{a}(\vec{d})$	if $\mathbf{a} \notin H$
D2	$\partial_H(\mathbf{a}(\vec{d})) = \delta$	if $\mathbf{a} \in H$
D3	$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$	
D4	$\partial_H(x \cdot y) = \partial_H(x) \cdot \partial_H(y)$	
D5	$\partial_H(x \triangleleft b \triangleright y) = \partial_H(x) \triangleleft b \triangleright \partial_H(y)$	
D6	$\partial_H(\sum_v p) = \sum_v \partial_H(p)$	
D7	$\partial_H(x \cdot t) = \partial_H(x) \cdot t$	
ATC1	$x \ll a \cdot t = \sum_u x \cdot u \triangleleft u \leq t \triangleright x \cdot t$	
ATC2	$x \ll (y + z) = x \ll y + x \ll z$	
ATC3	$x \ll y \cdot z = x \ll y$	
ATC4	$x \ll \sum_v p = \sum_v x \ll p$	
ATC5	$x \ll (y \triangleleft b \triangleright z) = (x \ll y) \triangleleft b \triangleright (x \ll z)$	

Table 6. Time related axioms of  $\mu\text{CRL}_t$ , where  $x, y, z \in V_P$ , process-closed  $p \in T_P$ ,  $\mathbf{a} \in \text{Act}$ ,  $a \in AT_\delta$ ,  $b \in V_B$ ,  $t, u \in V_T$ ,  $v \in V$  and  $H \subseteq \text{Act}$ .

The axioms of  $\mu\text{CRL}_t$  are the axioms of  $p\text{CRL}_t$ , combined with the axioms in the Tables 5 and 6. The signature  $\Sigma(\mu\text{CRL}_t)$  is as  $\Sigma(p\text{CRL}_t)$ , extended with the operators for parallelism and the “ $\ll$ ”-operator. The various operators of  $\Sigma(\mu\text{CRL}_t)$  are listed in order of decreasing binding strength:

$$\begin{array}{c}
 \cdot \\
 \partial_H \\
 \cdot \\
 \gg \quad \ll \\
 \triangleleft \triangleright \quad \parallel \quad \perp \quad | \\
 \sum_v \\
 +.
 \end{array}$$

Parentheses are omitted from terms according to this convention.

Processes that are put in parallel can have time constraints on their actions that are used for communication. It may happen that these constraints are incompatible. For instance, a process can be able to perform a read action  $r$  at time 2, whereas the corresponding send action occurs at time 3. This is expressed by  $\partial_{\{s,r\}}(r \cdot 2 \parallel s \cdot 3)$ . This process is equivalent to  $\delta \cdot 2$ , expressing that up till time 2, the process can go on without violating any timing constraint, but to reach any moment in time larger than 2, a timing constraint of an individual process must be ignored. Hence, this process does not exist at time 3.

## 5.2. Fischer's mutual exclusion protocol

As an example we describe Fischer's mutual exclusion protocol [53, 68]. This protocol guarantees that only one single party can be in a critical section at any given time. The idea is that an application process can ask the protocol, using a `request` action, to organise exclusive access for some critical region. The protocol then executes a simple program, and will after some time respond with an `enter` action, indicating that the application process has now exclusive access. When the application process is ready, it issues a `leave` instruction, indicating to the protocol that it does not require exclusive access anymore.

We describe this protocol for only two parties, although it can be used for any arbitrary number. The main idea behind the protocol is that each party after being asked to access a critical region, checks whether the common variable  $x$  equals 0, indicating that no process has claimed access. It then quickly sets the variable to its own sequence number to claim the critical section, and checks whether the variable has still the same value after a sufficiently long delay, guaranteeing that no other process will get access to the critical section. When leaving the critical section, the protocol sets the variable  $x$  back to 0.

We omit the description of the standard datatypes, we assume that the sorts and functions are self evident. The processes asking for access to their critical sections use a common variable  $x$ , which is modelled as a process  $X$ . As can be seen, the variable  $x$  can be set to a new value, and can be tested for the value it contains:

$$X(x) = \sum_y \text{rec\_set}(y) \cdot X(y) + \text{send\_test}(x) \cdot X(x).$$

Next, we describe a process trying to gain access to a critical section. We assume that each such process has a unique natural number ( $n$ ) as its identification. Such a process first checks whether the variable  $x$  is equal to 0, and records the time at which this action successfully takes place in the variable  $t$ . Then, within  $d$  time, it sets  $x$  to value  $n$ , and subsequently after at least  $d$  time reads the value of  $x$  again. If this value is equal to  $n$ , the protocol terminates and grants the process access to the critical section. Otherwise, it starts the protocol again for a new attempt to gain access.

$$\begin{aligned} FP(n) &= \text{request}(n) \cdot Prot(n) \cdot \text{enter}(n) \cdot \text{leave}(n) \cdot \text{send\_set}(0) \cdot FP(n), \\ Prot(n) &= \sum_t \text{rec\_test}(0) \cdot t \cdot \sum_{t'} (\text{send\_set}(n) \cdot t' \triangleleft t' \leq t + d \triangleright \delta \cdot \mathbf{0}) \cdot \\ &\quad \sum_{t''} \sum_m \text{rec\_test}(m) \cdot t'' \cdot (\text{enter}(n) \triangleleft eq(m, n) \triangleright Prot(n)) \triangleleft t'' > t' + d \triangleright \delta \cdot \mathbf{0}. \end{aligned}$$

Communication between the processes is defined by

$$\begin{aligned} \gamma(\text{send\_set}, \text{rec\_set}) &= \text{set}, \\ \gamma(\text{send\_test}, \text{rec\_test}) &= \text{test}. \end{aligned}$$

The encapsulation operator is now used to block all send and receive actions that occur in isolation (i.e. have not synchronised). Let  $H = \{\text{send\_set}, \text{rec\_set}, \text{send\_test}, \text{rec\_test}\}$ . The process term

$$\partial_H(FP(1) \parallel FP(2) \parallel X(0))$$

now describes Fischer's protocol for two competing processes.

### 5.3. Elimination of parallel operators

In this section, we consider processes with parallel operators, and show that all process-closed terms are derivably equal to  $\Sigma(p\text{CRL}_t)$ -terms. In [37], the elimination of the parallel operators has been provided for the version of timed  $\mu\text{CRL}_t$ , as presented in [25]. Although this version of  $\mu\text{CRL}_t$  differs from the version presented here the results are similar.

**Lemma 5.1.** Let  $p$  and  $q$  be basic terms. Then,  $\partial_H(p)$  and  $p \ll q$  are derivably equal to basic terms.

**Proof:**

First, we prove the existence of a basic term that is derivably equal to  $\partial_H(p)$  by induction on the structure of basic term  $p$ .

1.  $p \equiv \sum_{\vec{v}} a:t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned} \partial_H(p) &= \partial_H(\sum_{\vec{v}} a:t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{\text{D6}}{=} \sum_{\vec{v}} \partial_H(a:t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{\text{D5}}{=} \sum_{\vec{v}} \partial_H(a:t) \triangleleft b \triangleright \partial_H(\delta \cdot \mathbf{0}) \\ &\stackrel{\text{D7}}{=} \sum_{\vec{v}} \partial_H(a) \cdot t \triangleleft b \triangleright \partial_H(\delta) \cdot \mathbf{0} \\ &\stackrel{\text{DD}}{=} \sum_{\vec{v}} \partial_H(a) \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}. \end{aligned}$$

Since  $\partial_H(a)$  is an action term or  $\delta$  we have obtained a process-closed  $p\text{CRL}_t$ -term. Hence a basic term exists for  $\partial_H(p)$ .

2.  $p \equiv \sum_{\vec{v}} a:t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . By induction we have the existence of basic term  $r'$  such that  $\partial_H(p') = r'$ . Then,

$$\begin{aligned} \partial_H(p) &= \partial_H(\sum_{\vec{v}} a:t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{\text{D6}}{=} \sum_{\vec{v}} \partial_H(a:t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{\text{D5}}{=} \sum_{\vec{v}} \partial_H(a:t \cdot p') \triangleleft b \triangleright \partial_H(\delta \cdot \mathbf{0}) \\ &\stackrel{\text{D4}}{=} \sum_{\vec{v}} \partial_H(a:t) \cdot \partial_H(p') \triangleleft b \triangleright \partial_H(\delta \cdot \mathbf{0}) \\ &\stackrel{\text{D7}}{=} \sum_{\vec{v}} \partial_H(a) \cdot t \cdot \partial_H(p') \triangleleft b \triangleright \partial_H(\delta) \cdot \mathbf{0} \\ &\stackrel{\text{DD}}{=} \sum_{\vec{v}} \partial_H(a) \cdot t \cdot r' \triangleleft b \triangleright \delta \cdot \mathbf{0}. \end{aligned}$$

Since  $\partial_H(a)$  is an action term or  $\delta$ , we have obtained a process-closed  $p\text{CRL}_t$ -term. Hence, a basic term exists.

3.  $p \equiv p_1 + p_2$ . By induction we have the existence of basic terms  $p'_1$  and  $p'_2$  such that  $\partial_H(p_1) = p'_1$  and  $\partial_H(p_2) = p'_2$ . Then,  $\partial_H(p) = \partial_H(p_1 + p_2) \stackrel{D3}{=} \partial_H(p_1) + \partial_H(p_2) = p'_1 + p'_2$ , which is a basic term.

Second, we prove the existence of basic term that is derivably equal to  $p \ll q$  by induction on the structure of basic term  $q$ .

1.  $q \equiv \sum_{\bar{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned} p \ll q &= p \ll \sum_{\bar{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{ATC4}{=} \sum_{\bar{v}} p \ll (a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{ATC5}{=} \sum_{\bar{v}} (p \ll a^c t) \triangleleft b \triangleright (p \ll \delta \cdot \mathbf{0}) \\ &\stackrel{ATC1}{=} \sum_{\bar{v}} (\sum_u p^c u \triangleleft u \leq t \triangleright p^c t) \triangleleft b \triangleright (\sum_u p^c u \triangleleft u \leq \mathbf{0} \triangleright p^c \mathbf{0}). \end{aligned}$$

This is a process-closed  $p\text{CRL}_t$ -term, and hence it is derivably equal to a basic term.

2.  $q \equiv \sum_{\bar{v}} a^c t \cdot q' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned} p \ll q &= p \ll \sum_{\bar{v}} a^c t \cdot q' \triangleleft b \triangleright \delta \cdot \mathbf{0} \\ &\stackrel{ATC4}{=} \sum_{\bar{v}} p \ll (a^c t \cdot q' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \\ &\stackrel{ATC5}{=} \sum_{\bar{v}} (p \ll a^c t \cdot q') \triangleleft b \triangleright (p \ll \delta \cdot \mathbf{0}) \\ &\stackrel{ATC3}{=} \sum_{\bar{v}} (p \ll a^c t) \triangleleft b \triangleright (p \ll \delta \cdot \mathbf{0}) \\ &\stackrel{ATC1}{=} \sum_{\bar{v}} (\sum_u p^c u \triangleleft u \leq t \triangleright p^c t) \triangleleft b \triangleright (\sum_u p^c u \triangleleft u \leq \mathbf{0} \triangleright p^c \mathbf{0}). \end{aligned}$$

This is a process-closed  $p\text{CRL}_t$ -term, and hence it is derivably equal to a basic term.

3.  $q \equiv q_1 + q_2$ . By induction we have the existence of basic terms  $p_1$  and  $p_2$  such that  $p \ll q_1 = p_1$  and  $p \ll q_2 = p_2$ . Then,  $p \ll q = p \ll (q_1 + q_2) \stackrel{ATC2}{=} p \ll q_1 + p \ll q_2 = p_1 + p_2$ , which is a basic term.

□

**Lemma 5.2.** Let  $p$  and  $q$  be well-timed basic terms. Then,  $p \ll q$ ,  $p | q$ , and  $p || q$  are derivably equal to a basic term.

**Proof:**

The three statements are proven simultaneously by induction on the total number of symbols of well-timed basic terms  $p$  and  $q$ . First, we consider the term  $p \ll q$ . We distinguish three cases based on the structure of basic term  $p$ :

1.  $p \equiv \sum_{\bar{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned} p \ll q &= (\sum_{\bar{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \ll q \\ &\stackrel{SUM6}{=} \sum_{\bar{v}} (a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) \ll q \\ &\stackrel{H18}{=} \sum_{\bar{v}} (a^c t \ll q) \triangleleft b \triangleright (\delta \cdot \mathbf{0} \ll q) \\ &\stackrel{CM2}{=} \sum_{\bar{v}} (a^c t \ll q) \cdot q \triangleleft b \triangleright (\delta \cdot \mathbf{0} \ll q) \cdot q. \end{aligned}$$

This is a  $\mu\text{CRL}_t$ -term in which none of the operators  $\parallel$ ,  $|$ , or  $\ll$  occurs, and hence it is derivably equal to a basic term using the previous lemma and Theorem 2.1.

2.  $p \equiv \sum_{\bar{v}} a \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$ . By induction we have the existence of basic term  $r'$  such that  $p' \parallel q = r'$ . As  $p$  is well-timed we have  $t \gg p' = p'$ . Then,

$$\begin{aligned}
p \parallel q &= (\sum_{\bar{v}} a \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \parallel q \\
&\stackrel{\text{SUM6}}{=} \sum_{\bar{v}} (a \cdot t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) \parallel q \\
&\stackrel{\text{H18}}{=} \sum_{\bar{v}} (a \cdot t \cdot p' \parallel q) \triangleleft b \triangleright (\delta \cdot \mathbf{0} \parallel q) \\
&\stackrel{\text{CM3,CM2}}{=} \sum_{\bar{v}} (a \cdot t \ll q) \cdot (t \gg p' \parallel q) \triangleleft b \triangleright (\delta \cdot \mathbf{0} \ll q) \cdot q \\
&= \sum_{\bar{v}} (a \cdot t \ll q) \cdot (p' \parallel q) \triangleleft b \triangleright (\delta \cdot \mathbf{0} \ll q) \cdot q \\
&= \sum_{\bar{v}} (a \cdot t \ll q) \cdot r' \triangleleft b \triangleright (\delta \cdot \mathbf{0} \ll q) \cdot q.
\end{aligned}$$

This is a  $\mu\text{CRL}_t$ -term in which none of the operators  $\parallel$ ,  $|$ , or  $\ll$  occurs, and hence it is derivably equal to a basic term using the previous lemma and Theorem 2.1.

3.  $p \equiv p' + p''$ . By induction we have the existence of basic terms  $r'$  and  $r''$  such that  $p' \parallel q = r'$  and  $p'' \parallel q = r''$ . Then,  $p \parallel q = (p' + p'') \parallel q \stackrel{\text{CM4}}{=} p' \parallel q + p'' \parallel q = r' + r''$ , which is a basic term.

Second, we consider the term  $p | q$ . We distinguish cases based on the structure of the well-timed basic terms  $p$  and  $q$ :

1.  $p \equiv \sum_{\bar{v}} a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}$  and  $q = \sum_{\bar{w}} a' \cdot t' \triangleleft b' \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned}
p | q &= (\sum_{\bar{v}} a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | (\sum_{\bar{w}} a' \cdot t' \triangleleft b' \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{SUM7,SUM7'}}{=} \sum_{\bar{v}, \bar{w}} (a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | (a' \cdot t' \triangleleft b' \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{H8}}{=} \sum_{\bar{v}, \bar{w}} ((a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | a' \cdot t') \triangleleft b' \triangleright ((a \cdot t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | \delta \cdot \mathbf{0}) \\
&\stackrel{\text{H8'}}{=} \sum_{\bar{v}, \bar{w}} ((a \cdot t | a' \cdot t') \triangleleft b \triangleright (\delta \cdot \mathbf{0} | a' \cdot t')) \triangleleft b' \triangleright ((a \cdot t | \delta \cdot \mathbf{0}) \triangleleft b \triangleright (\delta \cdot \mathbf{0} | \delta \cdot \mathbf{0})) \\
&\stackrel{\text{ATA7,ATA8}}{=} \sum_{\bar{v}, \bar{w}} ((a | a') \cdot t \cdot t' \triangleleft b \triangleright (\delta | a') \cdot \mathbf{0} \cdot t') \triangleleft b' \triangleright ((a | \delta) \cdot t \cdot \mathbf{0} \triangleleft b \triangleright (\delta | \delta) \cdot \mathbf{0} \cdot \mathbf{0}) \\
&\stackrel{\text{CD1,CD2}}{=} \sum_{\bar{v}, \bar{w}} ((a | a') \cdot t \cdot t' \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot t') \triangleleft b' \triangleright (\delta \cdot t \cdot \mathbf{0} \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot \mathbf{0}).
\end{aligned}$$

By axiom CF the process  $(a | a')$  is equal to an atomic action or  $\delta$ . Hence, all parallel operators can be eliminated from the term  $p | q$ . By Theorem 2.1 there exists a basic term  $r$  such that  $p | q = r$ .

2.  $p \equiv \sum_{\bar{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}$  and  $q \equiv \sum_{\bar{w}} a'^c t' \cdot q' \triangleleft b' \triangleright \delta \cdot \mathbf{0}$ . Then,

$$\begin{aligned}
p | q &= (\sum_{\bar{v}} a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | (\sum_{\bar{w}} a'^c t' \cdot q' \triangleleft b' \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{SUM7}, \text{SUM7}'}{=} \sum_{\bar{v}, \bar{w}} (a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | (a'^c t' \cdot q' \triangleleft b' \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{H8}}{=} \sum_{\bar{v}, \bar{w}} ((a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | a'^c t' \cdot q') \triangleleft b' \triangleright ((a^c t \triangleleft b \triangleright \delta \cdot \mathbf{0}) | \delta \cdot \mathbf{0}) \\
&\stackrel{\text{H8}'}{=} \sum_{\bar{v}, \bar{w}} ((a^c t | a'^c t' \cdot q') \triangleleft b \triangleright (\delta \cdot \mathbf{0} | a'^c t' \cdot q')) \triangleleft b' \triangleright \\
&\quad ((a^c t | \delta \cdot \mathbf{0}) \triangleleft b \triangleright (\delta \cdot \mathbf{0} | \delta \cdot \mathbf{0})) \\
&\stackrel{\text{ATA3}, \text{ATA7}, \text{ATA8}}{=} \sum_{\bar{v}, \bar{w}} ((a | a' \cdot q')^c t t' \triangleleft b \triangleright (\delta | a' \cdot q') \cdot \mathbf{0} t') \triangleleft b' \triangleright \\
&\quad ((a | \delta)^c t \cdot \mathbf{0} \triangleleft b \triangleright (\delta | \delta) \cdot \mathbf{0} \cdot \mathbf{0}) \\
&\stackrel{\text{CM6}, \text{CD1}, \text{CD2}, \text{A7}}{=} \sum_{\bar{v}, \bar{w}} ((a | a' \cdot q')^c t t' \triangleleft b \triangleright \delta \cdot \mathbf{0} t') \triangleleft b' \triangleright (\delta \cdot t \cdot \mathbf{0} \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot \mathbf{0}) \\
&\stackrel{\text{CM6}}{=} \sum_{\bar{v}, \bar{w}} (((a | a') \cdot q')^c t t' \triangleleft b \triangleright \delta \cdot \mathbf{0} t') \triangleleft b' \triangleright (\delta \cdot t \cdot \mathbf{0} \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot \mathbf{0}).
\end{aligned}$$

By axiom CF the process  $(a | a')$  is equal to an atomic action or  $\delta$ . Hence, all parallel operators can be eliminated from the term  $p | q$ . By Theorem 2.1 there exists a basic term  $r$  such that  $p | q = r$ .

3.  $p \equiv \sum_{\bar{v}} a^c t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$  and  $q \equiv \sum_{\bar{w}} a'^c t' \triangleleft b' \triangleright \delta \cdot \mathbf{0}$ . This case is similar to the previous case.
4.  $p \equiv \sum_{\bar{v}} a^c t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}$  and  $q \equiv \sum_{\bar{w}} a'^c t' \cdot q' \triangleleft b' \triangleright \delta \cdot \mathbf{0}$ . By induction we have the existence of a basic term  $r'$  such that  $p' || q' = r'$ . Then,

$$\begin{aligned}
p | q &= (\sum_{\bar{v}} a^c t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) | (\sum_{\bar{w}} a'^c t' \cdot q' \triangleleft b' \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{SUM7}, \text{SUM7}'}{=} \sum_{\bar{v}, \bar{w}} (a^c t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) | (a'^c t' \cdot q' \triangleleft b' \triangleright \delta \cdot \mathbf{0}) \\
&\stackrel{\text{H8}}{=} \sum_{\bar{v}, \bar{w}} ((a^c t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) | a'^c t' \cdot q') \triangleleft b' \triangleright ((a^c t \cdot p' \triangleleft b \triangleright \delta \cdot \mathbf{0}) | \delta \cdot \mathbf{0}) \\
&\stackrel{\text{H8}'}{=} \sum_{\bar{v}, \bar{w}} ((a^c t \cdot p' | a'^c t' \cdot q') \triangleleft b \triangleright (\delta \cdot \mathbf{0} | a'^c t' \cdot q')) \triangleleft b' \triangleright \\
&\quad ((a^c t \cdot p' | \delta \cdot \mathbf{0}) \triangleleft b \triangleright (\delta \cdot \mathbf{0} | \delta \cdot \mathbf{0})) \\
&\stackrel{\text{ATA3}, \text{ATA7}, \text{ATA8}}{=} \sum_{\bar{v}, \bar{w}} ((a \cdot p' | a' \cdot q')^c t t' \triangleleft b \triangleright (\delta | a' \cdot q') \cdot \mathbf{0} t') \triangleleft b' \triangleright \\
&\quad ((a \cdot p' | \delta)^c t \cdot \mathbf{0} \triangleleft b \triangleright (\delta | \delta) \cdot \mathbf{0} \cdot \mathbf{0}) \\
&\stackrel{\text{CM5}, \text{CM6}, \text{CM7}}{=} \sum_{\bar{v}, \bar{w}} (((a | a') \cdot (p' || q'))^c t t' \triangleleft b \triangleright ((\delta | a') \cdot q') \cdot \mathbf{0} t') \triangleleft b' \triangleright \\
&\quad (((a | \delta) \cdot p')^c t \cdot \mathbf{0} \triangleleft b \triangleright (\delta | \delta) \cdot \mathbf{0} \cdot \mathbf{0}) \\
&\stackrel{\text{CD1}, \text{CD2}, \text{A7}}{=} \sum_{\bar{v}, \bar{w}} (((a | a') \cdot (p' || q'))^c t t' \triangleleft b \triangleright \delta \cdot \mathbf{0} t') \triangleleft b' \triangleright (\delta \cdot t \cdot \mathbf{0} \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot \mathbf{0}) \\
&= \sum_{\bar{v}, \bar{w}} (((a | a') \cdot r')^c t t' \triangleleft b \triangleright \delta \cdot \mathbf{0} t') \triangleleft b' \triangleright (\delta \cdot t \cdot \mathbf{0} \triangleleft b \triangleright \delta \cdot \mathbf{0} \cdot \mathbf{0}).
\end{aligned}$$

By axiom CF the process  $(a | a')$  is equal to an atomic action or  $\delta$ . Hence, all parallel operators can be eliminated from the term  $p | q$ . By Theorem 2.1 there exists a basic term  $r$  such that  $p | q = r$ .

5.  $p \equiv p' + p''$ . By induction we have the existence of basic terms  $r'$  and  $r''$  such that  $p' | q = r'$  and  $p'' | q = r''$ . Then,  $p | q = (p' + p'') | q \stackrel{\text{CM8}}{=} p' | q + p'' | q = r' + r''$ , which is a basic term.
6.  $q \equiv q' + q''$ . Similar to case 5.

Finally, we base the elimination of  $\parallel$  on the elimination of  $\lll$  and  $|$  as follows: By the previous parts of the proof we have the existence of basic terms  $r_1$ ,  $r_2$ , and  $r_3$  such that  $p \lll q = r_1$ ,  $q \lll p = r_2$ , and  $p | q = r_3$ . Hence,  $p \parallel q = p \lll q + q \lll p + p | q = r_1 + r_2 + r_3$ , which is a basic term.  $\square$

**Theorem 5.1. (Elimination theorem)**

If  $q$  is a process-closed term over  $\Sigma(\mu\text{CRL}_t)$ , then there is a basic term  $p$  such that  $\mu\text{CRL}_t \vdash q = p$ .

**Proof:**

By Lemma 5.1 and Lemma 5.2 any process-closed term  $q$  over  $\Sigma(\mu\text{CRL}_t)$  with at most one operator from  $\{\lll, \partial_H, \parallel, \lll, |\}$  is derivably equal to some basic term. Obviously, any term with  $n + 1$  operators from this set contains subterms with only one such operator, such that after elimination of one of these, a term with  $n$  parallel operators results. By induction on  $n$  we conclude that all these operators can be eliminated, so that  $q$  is derivably equal to some basic term  $p$ .  $\square$

#### 5.4. Semantics of timed $\mu\text{CRL}$

**Definition 5.1.** The set  $\mathcal{P} = \bigcup_{i=0}^{\omega} \mathcal{P}^i$  of processes is obtained by the following recursion:

$$\begin{aligned} \mathcal{P}^0 &= \mathcal{A}_\delta \\ \mathcal{P}^{n+1} &= \mathcal{P}^n \\ &\cup \{ p \cdot q, \sum P', p \cdot t, t \gg p, p \parallel q, p \lll q, p | q, p \lll q, p \lll q, \partial_H(p) \\ &\quad | p, q \in \mathcal{P}^n, P' \neq \emptyset, P' \subseteq \mathcal{P}^n, t \in \mathcal{D}_T, H \subseteq \text{Act} \\ &\quad \}. \end{aligned}$$

**Definition 5.2.** The interpretation of process-closed terms under a valuation, from Definition 3.2, is extended with the following clauses: for process-closed  $p, q \in T_{\mathcal{P}}$ ,  $H \subseteq \text{Act}$ , and valuation  $\nu$

$$\begin{aligned} \lll p \lll q \rrr^\nu &= \lll p \rrr^\nu \lll \lll q \rrr^\nu, & \lll p | q \rrr^\nu &= \lll p \rrr^\nu | \lll q \rrr^\nu, \\ \lll p \parallel q \rrr^\nu &= \lll p \rrr^\nu \parallel \lll q \rrr^\nu, & \lll \partial_H(p) \rrr^\nu &= \partial_H(\lll p \rrr^\nu). \\ \lll p \lll q \rrr^\nu &= \lll p \rrr^\nu \lll \lll q \rrr^\nu, \end{aligned}$$

The operational semantics of these processes is described in terms of the action relations  $\xrightarrow{a}_t \surd$  and  $\xrightarrow{a}_t$  and the delay relation  $U_t$ . For the new operators the definitions of these relations are extended with the transition rules in the Tables 7, 8, and 9.

Although the set of processes that we consider in this section is larger than the set of processes we considered for  $p\text{CRL}_t$  we have no reason to change our definition of strong timed bisimilarity as it is defined in terms of the action relations and the delay predicate only.

**Theorem 5.2. (Congruence)**

Strong timed bisimilarity is a congruence with respect to the operators defined on  $\mathcal{P}$ .

**Proof:**

We give bisimulations that witness the congruence of strong timed bisimilarity with respect to the operators  $\lll$ ,  $\partial_H$ ,  $\parallel$ ,  $\lll$ , and  $|$ . In each of these cases it is straightforward to prove that the constructed set  $R$  is a strong timed bisimulation.

$\frac{p \xrightarrow{a}_{\rightarrow t} \checkmark \quad U_t(q)}{p \parallel q \xrightarrow{a}_{\rightarrow t} t \gg q}$	$\frac{q \xrightarrow{a}_{\rightarrow t} \checkmark \quad U_t(p)}{p \parallel q \xrightarrow{a}_{\rightarrow t} t \gg p}$	$\frac{p \xrightarrow{a}_{\rightarrow t} \checkmark \quad U_t(q)}{p \parallel\!\! \parallel q \xrightarrow{a}_{\rightarrow t} t \gg q}$	$\frac{p \xrightarrow{a}_{\rightarrow t} \checkmark \quad U_t(q)}{p \ll\!\! \ll q \xrightarrow{a}_{\rightarrow t} \checkmark}$
$\frac{p \xrightarrow{a}_{\rightarrow t} p' \quad U_t(q)}{p \parallel q \xrightarrow{a}_{\rightarrow t} p' \parallel t \gg q}$	$\frac{q \xrightarrow{a}_{\rightarrow t} q' \quad U_t(p)}{p \parallel q \xrightarrow{a}_{\rightarrow t} t \gg p \parallel q'}$	$\frac{p \xrightarrow{a}_{\rightarrow t} p' \quad U_t(q)}{p \parallel\!\! \parallel q \xrightarrow{a}_{\rightarrow t} p' \parallel t \gg q}$	$\frac{p \xrightarrow{a}_{\rightarrow t} p' \quad U_t(q)}{p \ll\!\! \ll q \xrightarrow{a}_{\rightarrow t} p'}$
$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} \checkmark \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} \checkmark \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \parallel q \xrightarrow{c(\vec{d})}_{\rightarrow t} \checkmark}$		$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} \checkmark \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} q' \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \parallel q \xrightarrow{c(\vec{d})}_{\rightarrow t} q'}$	
$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} p' \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} \checkmark \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \parallel q \xrightarrow{c(\vec{d})}_{\rightarrow t} p'}$		$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} p' \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} q' \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \parallel q \xrightarrow{c(\vec{d})}_{\rightarrow t} p' \parallel q'}$	
$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} \checkmark \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} \checkmark \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \mid q \xrightarrow{c(\vec{d})}_{\rightarrow t} \checkmark}$		$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} \checkmark \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} q' \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \mid q \xrightarrow{c(\vec{d})}_{\rightarrow t} q'}$	
$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} p' \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} \checkmark \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \mid q \xrightarrow{c(\vec{d})}_{\rightarrow t} p'}$		$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} p' \quad q \xrightarrow{b(\vec{d})}_{\rightarrow t} q' \quad \gamma(\mathbf{a}, \mathbf{b}) = \mathbf{c}}{p \mid q \xrightarrow{c(\vec{d})}_{\rightarrow t} p' \parallel q'}$	

Table 7. Transition rules for the parallel operators, where  $a \in \mathcal{A}$ ,  $\mathbf{a}, \mathbf{b}, \mathbf{c} \in Act$ ,  $\vec{d} \in \mathcal{D}_s$ ,  $p, q, p', q' \in \mathcal{P}$ , and  $t \in \mathcal{D}_T$ .

$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} \checkmark \quad \mathbf{a} \notin H}{\partial_H(p) \xrightarrow{a(\vec{d})}_{\rightarrow t} \checkmark}$	$\frac{p \xrightarrow{a(\vec{d})}_{\rightarrow t} q \quad \mathbf{a} \notin H}{\partial_H(p) \xrightarrow{a(\vec{d})}_{\rightarrow t} \partial_H(q)}$
---	---

Table 8. Transition rules for encapsulation, where  $p, q \in \mathcal{P}$ ,  $t \in \mathcal{D}_T$ ,  $H \subseteq Act$ ,  $\mathbf{a} \in Act$ , and  $\vec{d} \in \mathcal{D}_s$ .

$\frac{U_t(p) \quad U_t(q)}{U_t(p \ll q)}$	$\frac{U_t(p) \quad U_t(q)}{U_t(p \parallel q)}$	$\frac{U_t(p) \quad U_t(q)}{U_t(p \parallel\!\!\! \parallel q)}$	$\frac{U_t(p) \quad U_t(q)}{U_t(p \mid q)}$	$\frac{U_t(p)}{U_t(\partial_H(p))}$
--	--	--	---	-------------------------------------

Table 9. Transition rules for the delay relation, where  $p, q \in \mathcal{P}$ ,  $t \in \mathcal{D}_T$ , and  $H \subseteq \text{Act}$ .

1.  $\ll$ . Let  $R_1 : p_1 \Leftarrow_t q_1$  and  $R_2 : p_2 \Leftarrow_t q_2$ . Define  $R = \{(p_1 \ll p_2, q_1 \ll q_2), (q_1 \ll q_2, p_1 \ll p_2)\} \cup R_2$ .
2.  $\partial_H$ . Let  $R_1 : p \Leftarrow_t q$ . Define  $R = \{(\partial_H(p'), \partial_H(q')) \mid (p', q') \in R_1, H \subseteq \text{Act}\}$ .
3.  $\parallel$ . Let  $R_1 : p_1 \Leftarrow_t q_1$  and  $R_2 : p_2 \Leftarrow_t q_2$ . Define  $R'_1$  to be the smallest symmetrical relation that satisfies  $R_1 \subseteq R'_1$  and if  $(p', q') \in R'_1$ , then  $(t \gg p', t \gg q') \in R'_1$ . Similarly, define  $R'_2$  to be the smallest symmetrical relation that satisfies  $R_2 \subseteq R'_2$  and if  $(p', q') \in R'_2$ , then  $(t \gg p', t \gg q') \in R'_2$ . Define  $R$  to be the smallest symmetrical relation that satisfies  $R'_1 \subseteq R$ ,  $R'_2 \subseteq R$ , and if  $(p', q') \in R'_1$  and  $(p'', q'') \in R'_2$ , then  $(p' \parallel p'', q' \parallel q'') \in R$ . Then  $R$  is the bisimulation witnessing the bisimilarity of  $p_1 \parallel p_2$  and  $q_1 \parallel q_2$ .
4.  $\parallel\!\!\! \parallel$ . Let  $R_1 : p_1 \Leftarrow_t q_1$  and  $R_2 : p_2 \Leftarrow_t q_2$ . Then, using the relation  $R$  from the previous item,  $R' = R \cup \{(p_1 \parallel\!\!\! \parallel p_2, q_1 \parallel\!\!\! \parallel q_2), (q_1 \parallel\!\!\! \parallel q_2, p_1 \parallel\!\!\! \parallel p_2)\}$  is the bisimulation witnessing the bisimilarity of  $p_1 \parallel\!\!\! \parallel p_2$  and  $q_1 \parallel\!\!\! \parallel q_2$ .
5.  $\mid$ . Let  $R_1 : p_1 \Leftarrow_t q_1$  and  $R_2 : p_2 \Leftarrow_t q_2$ . Then, using the relation  $R$  from the third item,  $R' = R \cup \{(p_1 \mid p_2, q_1 \mid q_2), (q_1 \mid q_2, p_1 \mid p_2)\}$  is the bisimulation witnessing the bisimilarity of  $p_1 \mid p_2$  and  $q_1 \mid q_2$ .

□

## 5.5. Soundness and completeness

### Theorem 5.3. (Soundness)

With respect to strong timed bisimilarity,  $\mu\text{CRL}_t$  is a sound axiom system.

### Theorem 5.4. (Relative completeness)

With respect to strong timed bisimilarity,  $\mu\text{CRL}_t$  is a complete axiom system for process-closed terms under the assumptions that the data algebra is  $\omega$ -completely axiomatised and that the data algebra has built-in equality and Skolem functions.

#### Proof:

Consider arbitrary process-closed  $\mu\text{CRL}_t$ -terms  $p$  and  $q$ . Suppose that  $\llbracket p \rrbracket^\nu \Leftarrow_t \llbracket q \rrbracket^\nu$  for all valuations  $\nu$ . By the elimination theorem we have the existence of basic terms  $p'$  and  $q'$  such that  $\mu\text{CRL}_t \vdash p = p'$  and  $\mu\text{CRL}_t \vdash q = q'$ . By the soundness of the axioms we then also have  $\llbracket p' \rrbracket^\nu \Leftarrow_t \llbracket q' \rrbracket^\nu$  for all valuations  $\nu$ . By the completeness of  $p\text{CRL}_t$  we then have  $p\text{CRL}_t \vdash p' = q'$ . Using the fact that all axioms of  $p\text{CRL}_t$  are also axioms of  $\mu\text{CRL}_t$  we find  $\mu\text{CRL}_t \vdash p' = q'$ . Thus we have  $\mu\text{CRL}_t \vdash p = p' = q' = q$ . □

## References

- [1] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems, *Theoretical Computer Science*, **138**(1), February 1995, 3–34.
- [2] Alur, R., Dill, D.: A theory of timed automata, *Theoretical Computer Science*, **126**(2), April 1994, 183–235.
- [3] Baeten, J., Bergstra, J.: Real Time Process Algebra, *Formal Aspects of Computing*, **3**(2), 1991, 142–188.
- [4] Baeten, J., Bergstra, J.: Real Time Process Algebra with Infinitesimals, in: *Algebra of Communicating Processes, Utrecht 1994* (A. Ponse, C. Verhoef, S. van Vlijmen, Eds.), Workshops in Computing, 1995, 148–187.
- [5] Baeten, J., Bergstra, J.: Discrete Time Process Algebra, *Formal Aspects of Computing*, **8**(2), 1996, 188–208.
- [6] Baeten, J., Bergstra, J.: Discrete time process algebra: absolute time, relative time and parametric time, *Fundamenta Informaticae*, **29**(1-2), 1997, 51–76.
- [7] Baeten, J., Bergstra, J., Reniers, M.: Discrete Time Process Algebra with Silent Step, in: *Proof, Language, and Interaction: Essays in Honour of Robin Milner* (G. Plotkin, C. Stirling, M. Tofte, Eds.), Foundations of Computing Series, chapter 18, MIT Press, 2000, 535–569.
- [8] Baeten, J., Middelburg, C.: Process algebra with timing: real time and discrete time, in: *Handbook of Process Algebra* (J. Bergstra, A. Ponse, S. Smolka, Eds.), chapter 10, Elsevier Science Publishers B.V., 2001, 627–684.
- [9] Baeten, J., Reniers, M.: *Termination in timed process algebra*, Technical Report CSR 00-13, Eindhoven University of Technology, Department of Computing Science, 2000.
- [10] Baeten, J., Verhoef, C.: Concrete Process Algebra, in: *Semantic Modelling* (S. Abramsky, D. M. Gabbay, T. Maibaum, Eds.), vol. 4 of *Handbook of Logic in Computer Science*, Oxford University Press, 1995, 149–268.
- [11] Baeten, J., Weijland, W.: *Process Algebra*, vol. 18 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, 1990.
- [12] Bergstra, J., Klop, J.: Process Algebra for Synchronous Communication, *Information and Control*, **60**(1/3), 1984, 109–137.
- [13] Bezem, M., Bol, R., Groote, J.: Formalizing process algebraic verifications in the calculus of constructions, *Formal Aspects of Computing*, **9**(1), 1997, 1–48.
- [14] Bezem, M., Groote, J.: A correctness proof of a one bit sliding window protocol in  $\mu$ CRL, *The Computer Journal*, **37**(4), 1994, 289–307.
- [15] Bezem, M., Groote, J.: Invariants in process algebra with data, in: *CONCUR'94: Concurrency Theory* (B. Jonsson, J. Parrow, Eds.), vol. 836 of *Lecture Notes in Computer Science*, Springer-Verlag, Uppsala, Sweden, 1994, 401–416.
- [16] Bosscher, D., Ponse, A.: Translating a process algebra with symbolic data values to linear format, in: *Proceedings of the workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)* (U. Engberg, K. Larsen, A. Skou, Eds.), vol. NS-95-2 of *Brics Notes Series*, Aarhus, Denmark, 1995, 119–130.
- [17] Brinksma, E.: *On the design of extended LOTOS: a specification language for open distributed systems*, Ph.D. Thesis, Twente University, 1988.
- [18] Chen, L.: *Timed processes: models, axioms, and decidability*, Ph.D. Thesis, University of Edinburgh, 1992.

- [19] Dams, D., Groote, J.: *Specification and Implementation of Components of a  $\mu$ CRL Toolbox*, Technical Report 152, University Utrecht, Department of Philosophy, 1995.
- [20] Daniels, M.: Modelling real-time behaviour with an interval time calculus, in: *Formal Techniques in Real-Time and Fault-Tolerant Systems* (J. Vytöpil, Ed.), vol. 571 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991, 53–71.
- [21] D’Argenio, P.: *Algebras and Automata for Timed and Stochastic Systems*, Ph.D. Thesis, University of Twente, 1999.
- [22] Fokkink, W.: An elimination theorem for regular behaviours with integration, in: *CONCUR’93, International Conference on Concurrency Theory* (E. Best, Ed.), vol. 715 of *Lecture Notes in Computer Science*, 1993, 432–446.
- [23] Fokkink, W.: *Clock, Trees and Stars in Process Theory*, Ph.D. Thesis, University of Amsterdam, 1994.
- [24] Fredlund, L.-Å., Groote, J., Korver, H.: Formal verification of a leader election protocol in process algebra, *Theoretical Computer Science*, **177**(2), 1997, 459–486.
- [25] Groote, J.: *The syntax and semantics of timed  $\mu$ CRL*, Technical Report SEN-R9709, CWI, Amsterdam, 1997.
- [26] Groote, J., Korver, H.: Correctness proof of the bakery protocol in  $\mu$ CRL, in: *Algebra of Communicating Processes* (A. Ponse, C. Verhoef, S. van Vlijmen, Eds.), Workshops in Computing, Springer-Verlag, 1994, 63–86.
- [27] Groote, J., Luttkik, S.: *Undecidability and completeness results for process algebras with alternative quantification over data*, Technical Report SEN-R9806, CWI, Amsterdam, 1998, Available at <ftp://ftp.cwi.nl/pub/CWIreports/SEN/SEN-R9806.ps.Z>.
- [28] Groote, J., Mateescu, R.: Verification of temporal properties of processes in a setting with data, in: *Proceedings of the 7th International Conference on Algebraic Methodology and Software Technology AMAST’98, Amazonia, Brazil*, vol. 1548, January 1998, 74–90.
- [29] Groote, J., Monin, F., van de Pol, J.: Checking verifications of protocols and distributed systems by computer, in: *Proceedings CONCUR’98* (D. Sangiorgi, R. de Simone, Eds.), vol. 1466 of *Lecture Notes in Computer Science*, Springer-Verlag, 1998, 629–655.
- [30] Groote, J., van de Pol, J.: A bounded retransmission protocol for large data packets. A case study in computer checked verification, in: *Proceedings of AMAST’96* (M. Wirsing, M. Nivat, Eds.), vol. 1101 of *Lecture Notes in Computer Science*, Munich, Germany, 1996, 536–550.
- [31] Groote, J., Ponse, A.: Proof theory for  $\mu$ CRL: A language for processes with data, in: *Semantics of Specification Languages, Proceedings of the International Workshop on Semantics of Specification Languages, Utrecht, The Netherlands, 25-27 October 1993* (D. Andrews, J. Groote, C. Middelburg, Eds.), Workshops in Computing, Springer-Verlag, 1994, 232–251.
- [32] Groote, J., Ponse, A.: The Syntax and Semantics of  $\mu$ CRL, in: *Algebra of Communicating Processes, Utrecht 1994* (A. Ponse, C. Verhoef, S. v. Vlijmen, Eds.), Workshops in Computing, Springer-Verlag, 1995, 26–62.
- [33] Groote, J., Reniers, M.: Algebraic Process Verification, in: *Handbook of Process Algebra* (J. Bergstra, A. Ponse, S. Smolka, Eds.), chapter 17, Elsevier Science Publishers B.V., 2001, 1151–1208.
- [34] Groote, J., Sellink, M.: Confluence for process verification, *Theoretical Computer Science*, **170**(1-2), 1996, 47–81.
- [35] Groote, J., Springintveld, J.: Focus points and convergent process operators: a proof strategy for protocol verification, *Journal of Logic and Algebraic Programming*, **49**(1-2), 2001, 31–60.

- [36] Groote, J., van Vlijmen, S.: A modal logic for  $\mu$ CRL, in: *Modal Logic and Process Algebra, a Bisimulation Perspective* (A. Ponse, M. de Rijke, Y. Venema, Eds.), vol. 53 of *CSLI Lecture Notes*, Stanford, 1995, 131–150.
- [37] Groote, J., van Wamel, J.: Basic theorems for parallel processes in timed  $\mu$ CRL, in: *WDS'99*, 1999, Revised version of Technical Report SEN-R9808, CWI, 1999.
- [38] Groote, J., van Wamel, J.: Analysis of three hybrid systems in timed  $\mu$ CRL, *Science of Computer Programming*, **39**(2-3), 2001, 215–247.
- [39] Groote, J., van Wamel, J.: The parallel composition of uniform processes with data, *Theoretical Computer Science*, **266**(1-2), 2001, 631–652.
- [40] van Ham, F.: *Visualisation of State Transition Graphs*, Master Thesis, Eindhoven University of Technology, 2000.
- [41] Hennessy, M., Lin, H.: Symbolic bisimulations, *Theoretical Computer Science*, **138**(2), 1995, 353–389.
- [42] Hennessy, M., Regan, T.: A Process Algebra for Timed Systems, *Information and Computation*, **177**(2), 1995, 221–239.
- [43] Henzinger, T.: The Theory of Hybrid Automata, in: *Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, 1996, 278–292.
- [44] Henzinger, T., Ho, P., Wong-Toi, H.: HyTech: A model checker for hybrid systems, *International Journal on Software Tools for Technology Transfer*, **1**(1-2), 1997, 110–122.
- [45] Henzinger, T., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems, *Information and Computation*, **111**(2), 1994, 193–244.
- [46] Hoare, C.: *Communicating Sequential Processes*, International Series in Computer Science, Prentice-Hall International, 1985.
- [47] ITU-T: *Recommendation Z.100: Specification and Description Language (SDL)*, ITU-T, Geneva, June 1994.
- [48] Jeffrey, A.: A linear time process algebra, in: *Computer Aided Verification* (K. Larsen, A. Skou, Eds.), vol. 575 of *Lecture Notes in Computer Science*, Springer-Verlag, 1992, 433–442.
- [49] Kleijn, J., Reniers, M., Rooda, J.: A Process Algebra Based Verification of a Production System, *Second IEEE Conference on Formal Engineering Methods* (J. Staples, M. Hinchley, S. Liu, Eds.), IEEE Computer Society Press, Brisbane, Australia, December 1998, To appear in *Formal Methods in System Design*, 2002.
- [50] Klusener, A.: *Models and Axioms for a Fragment of Real Time Process Algebra*, Ph.D. Thesis, Eindhoven University of Technology, 1993.
- [51] Korver, H.: *Building a simulator in the  $\mu$ CRL toolbox. A case study in modern software engineering*, Technical Report CS-R9632, CWI, Amsterdam, 1996.
- [52] Korver, H., Springintveld, J.: A computer-checked verification of Milner's Scheduler, in: *Proceedings of the international symposium on Theoretical Aspects of Computer Software (TACS'94)* (M. Hagiya, J. Mitchell, Eds.), vol. 789 of *Lecture Notes in Computer Science*, Springer-Verlag, Sendai, Japan, 1994, 161–178.
- [53] Lamport, L.: A fast mutual exclusion algorithm, *ACM Transactions on Computer Systems*, **5**(1), 1987, 1–11.
- [54] Larsen, K., Pettersson, P., Yi, W.: Uppaal in a nutshell, *International Journal on Software Tools for Technology Transfer*, **1**(1-2), 1997, 134–152.
- [55] Léonard, L., Leduc, G.: An enhanced version of timed LOTOS and its application to a case study, in: *Formal Description Techniques IV* (R. Tenney, P. Amer, M. Uyar, Eds.), 1994, 483–498.

- [56] Luttkik, S.: On the expressiveness of the choice quantifier, To appear in *Annals of Pure and Applied Logic*.
- [57] Luttkik, S.: *Choice quantification in process algebra*, Ph.D. Thesis, University of Amsterdam, 2002.
- [58] Lynch, N., Segala, R., Vaandrager, F., Weinberg, H.: Hybrid I/O automata, in: *Hybrid Systems III: Verification and Control (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, October 1995)* (R. Alur, T. Henzinger, E. Sontag, Eds.), vol. 1066 of *Lecture Notes in Computer Science*, Elsevier Science Publishers B.V., 1996, 496–510.
- [59] Milner, R.: *A Calculus of Communicating Systems*, vol. 92 of *Lecture Notes in Computer Science*, Springer-Verlag, 1980.
- [60] Moller, F., Tofts, C.: A Temporal Calculus of Communicating Systems, in: *CONCUR'90 - Theories of Concurrency: Unification and Extension* (J. Baeten, J. Klop, Eds.), vol. 458 of *Lecture Notes in Computer Science*, Springer-Verlag, Amsterdam, 1990, 401–415.
- [61] Nicollin, X., Sifakis, J.: The Algebra of Timed Processes, ATP: Theory and Application, *Information and Computation*, **114**(1), 1994, 131–178.
- [62] Ponse, A.: Computable processes and bisimulation equivalence, *Formal Aspects of Computing*, **8**(6), 1996, 648–678.
- [63] Quemada, J., de Frutos, D., Azcorra, A.: TIC: A TImed Calculus, *Formal Aspects of Computing*, **5**(3), 1993, 224–252.
- [64] Reed, G., Roscoe, A.: A timed model for communicating sequential processes, *Theoretical Computer Science*, **58**(1-3), 1988, 249–261.
- [65] Reniers, M., Vereijken, J.: *Completeness in Discrete-Time Process Algebra*, Technical Report CSR 96/15, Eindhoven University of Technology, Department of Computing Science, 1996.
- [66] Shankland, C., van der Zwaag, M.: The tree identify protocol of IEEE 1394 in  $\mu$ CRL, *Formal Aspects of Computing*, **10**(5-6), 1998, 509–531.
- [67] Sighireanu, M.: *Contribution à la définition et à l'implémentation de la norme "Extended LOTOS"*, Ph.D. Thesis, Université Joseph Fourier, Grenoble, France, 1999.
- [68] Vereijken, J.: *Discrete-time process algebra*, Ph.D. Thesis, Eindhoven University of Technology, 1997.
- [69] Wang, Y.: Real-time behaviour of asynchronous agents, in: *CONCUR'90 - Theories of Concurrency: Unification and Extension* (J. Baeten, J. Klop, Eds.), vol. 458 of *Lecture Notes in Computer Science*, Springer-Verlag, Amsterdam, 1990, 502–520.
- [70] Willemse, T.: *The Analysis of a Conveyor Belt System a case study in Hybrid Systems and timed  $\mu$ CRL*, Technical Report CSR 99-10, Eindhoven University of Technology, Department of Computing Science, 1999.
- [71] Willemse, T.: *Interpretations of Automata*, Technical Report CSR 01-02, Eindhoven University of Technology, Department of Computing Science, 2001.
- [72] Yovine, S.: Kronos: A verification tool for real-time systems, *International Journal on Software Tools for Technology Transfer*, **1**(1-2), October 1997, 123–133.
- [73] van der Zwaag, M.: *Time-stamped actions in pCRL algebras*, Technical Report SEN-R0002, CWI, Amsterdam, 2000.
- [74] van der Zwaag, M.: The cones and foci proof technique for timed transition systems, *Information Processing Letters*, **80**(1), 2001, 33–40.

## A. Overview of $p$ CRL

In this appendix we give a short overview of the syntax and semantics of the process algebra  $p$ CRL. The reason for this is that our completeness result for  $p$ CRL<sub>t</sub> is based on the completeness of  $p$ CRL. The signature of  $p$ CRL consists of: a set of action declarations, a nullary function  $\delta$ , binary functions  $+$  and  $\cdot$ , a ternary function  $\langle \_ \triangleright \_ \triangleright \_ \rangle$ , and the quantifier  $\sum_v$ .

A1	$x + y = y + x$
A2	$x + (y + z) = (x + y) + z$
A3	$x + x = x$
A4	$(x + y) \cdot z = x \cdot z + y \cdot z$
A5	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
A6	$x + \delta = x$
A7	$\delta \cdot x = \delta$
PE <sub>a</sub>	$\mathbf{a}(\vec{x}) \langle eq(\vec{x}, \vec{y}) \triangleright \delta = \mathbf{a}(\vec{y}) \langle eq(\vec{x}, \vec{y}) \triangleright \delta$
Cond1	$x \langle \mathbf{t} \triangleright y = x$
Cond2	$x \langle \mathbf{f} \triangleright y = y$
Cond3	$x \langle b \triangleright y = x \langle b \triangleright \delta + y \langle \neg b \triangleright \delta$
Cond4	$(x \langle b_1 \triangleright \delta) \langle b_2 \triangleright \delta = x \langle b_1 \wedge b_2 \triangleright \delta$
Cond5	$x \langle b_1 \triangleright \delta + x \langle b_2 \triangleright \delta = x \langle b_1 \vee b_2 \triangleright \delta$
Cond6	$(x \langle b \triangleright \delta) \cdot y = x \cdot y \langle b \triangleright \delta$
Cond7	$(x + y) \langle b \triangleright \delta = x \langle b \triangleright \delta + y \langle b \triangleright \delta$
SCA	$(x \langle b \triangleright \delta) \cdot (y \langle b \triangleright \delta) = x \cdot y \langle b \triangleright \delta$
Sum1	$\sum_v y = y$
Sum3	$\sum_v p = \sum_v p + p$
Sum4	$\sum_v (p + q) = \sum_v p + \sum_v q$
Sum5	$(\sum_v p) \cdot x = \sum_v p \cdot x$
Sum12	$(\sum_v p) \langle b \triangleright \delta = \sum_v p \langle b \triangleright \delta$

Table 10. Axioms of  $p$ CRL.