An Elimination Theorem for Regular Behaviours with Integration

Willem Jan Fokkink
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
e-mail: wan@cwi.nl

Abstract

In this paper we consider we consider a variant of the process algebra ACP with rational time and integration. We shall indicate a subdomain of regular processes for which an Elimination Theorem holds: for each pair of processes p, q in this class there is a process z in this class such that $p \| q$ and z have the same behaviour. Furthermore, we indicate by some simple examples that if the subdomain is restricted or enlarged, then the elimination result is lost. The subdomain has a strong link with the model of timed automata of Alur and Dill.
1991 Mathematics Subject Classification: 68Q50, 68Q60.
1987 CR Categories: D.3.1, F.3.1.
Key Words & Phrases: ACP, relative time, integration, regular process, Elimination Theorem.
Note: This work is in the context of ESPRIT Basic Research Action no 7166, CONCUR 2.

1 Introduction

In recent years, many papers have appeared that study real-time aspects of systems. Most process algebras have been extended with constructs that mean to describe some notion of either discrete or dense time. Some examples are timed CCS [Wan90], timed ACP [BB91], ATP [NS90] and timed CSP [RR88].

This paper is based on the approach in [BB91]. Only, that paper focuses on absolute time, while here we work with relative time, i.e., we assume that an expression $a[r]$ denotes an action a that is executed exactly r time units after the previous action has been executed. However, all our definitions and results can be translated to absolute time without any complications.

In [BB91] the notion of *integration* has been introduced, which describes the possibility of an action happening within a dense interval in time. For example, the process $\int_{v \in (0,1)} a[v]$ executes action a somewhere between time

0 and time 1. In this paper we take a more restrictive view on integration than in [BB91], called *prefixed* integration, which originates from [Klu91]. In general, a (prefixed) integral is of the form $\int_{v \in V} a[v]$ or $\int_{v \in V} a[v] \cdot p$, where V is an interval and p is a process expression. Integration enables the description of time dependencies, i.e., the process p may contain the variable v as a 'free' variable. Such free occurrences of v in p are bound by the integral sign $\int_{v \in V}$.

This paper deals with *regular* processes. Traditionally, regularity is defined in the sense of a process having a finite number of states, or a finite number of transitions. However, here such a definition would not work, due to the presence of the integral construct, which causes even finite processes to have an infinite number of different transitions. Therefore, a regular process is defined to be the solution of a linear specification. This definition is based on the fact that regular processes in the untimed case are exactly the solutions of linear specifications.

For the sake of verification, it is important to have an Elimination Theorem for regular processes, which says that the parallel composition of two regular processes is again a regular process. Because a verification mostly deals with a process $\partial_H(p_1 \| \cdots \| p_k)$, where $p_1, ..., p_k$ are regular processes. Since there is an Elimination Theorem for regular processes in untimed ACP, and also in timed ACP without integration [Fok92], one can get rid of the merges during the verification in these formalisms.

In this paper we set out to deduce an Elimination Theorem for timed ACP with integration. The existence of time variables in the syntax of timed ACP is essential for the validity of such a result, because Godskesen and Larsen have proved that one cannot hope to find an Elimination Theorem in the absence of time variables [GL92].

However, in this paper we shall encounter some examples which show that in general one cannot eliminate the merge from regular processes in timed ACP with integration. Thus, one cannot use the common verification techniques for this algebra. Fortunately, it will turn out that one does obtain an Elimination Theorem for a subdomain of the class of regular processes. This subdomain is very specific; if it is restricted or enlarged in any obvious way, then we will see by some simple examples that the elimination result is lost. At first sight the syntactic restrictions for the subdomain may seem arbitrary, but if one studies the examples more closely, it will turn out that linear specifications which do not satisfy these restrictions tend to describe all kinds of irregular behaviour, such as accelerations (see Examples 3.1 and 3.2) and oscillations (see Example 3.3).

The subdomain for which we shall deduce an Elimination Theorem has a strong link with the class of *timed automata* of Alur and Dill [AD90]. However, we do not obtain a translation between the processes in our subdomain and timed automata, due to the requirement of non-Zeno behaviour and the pres-

ence of fairness restrictions for languages accepted by timed automata. But if these restrictions are discarded, the classes of strongly regular processes and of timed automata turn out to be equivalent. Hence, the algebra of strongly regular processes may be used as a syntax for timed automata, and moreover the operations of ACP may be used to compose smaller automata into larger ones. This compositionality is missing in existing timed automata work.

This paper is an extended abstract of [Fok93]. The main difference with the full version is that the long and technical proof of the Elimination Theorem has been replaced by a much shorter and more intuitive description of the proof.

For the sake of simplicity we will not include the encapsulation operator ∂_H to the syntax. However, it can be added to the syntax without any complications [Fok93].

Acknowledgements. Steven Klusener and anonymous referees are thanked for helpful comments, and Frits Vaandrager for suggesting the link with timed automata.

2 The Syntax and Semantics

This section contains a description of the syntax and operational semantics for ACP with relative rational time and integration, denoted by ACP_{rqI}, together with recursion.

2.1 The Alphabet

Assume an alphabet A of atomic actions, together with a special constant δ, representing deadlock. Furthermore, assume a communication function $| : A \cup \{\delta\} \times A \cup \{\delta\} \to A \cup \{\delta\}$ which is commutative and associative and has δ as zero element.

2.2 Bounds and Intervals

$TVar$ denotes a countably infinite set of *time variables*. Let $t \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$, $r \in \mathbb{Q}_{>0}$ and $v \in TVar$. The set of *bounds*, with typical element b, is defined by

$$b ::= t \mid v \mid b + b \mid b \dot{-} b \mid r \cdot b$$

where $\dot{-}$ denotes the monus function, i.e. if $t_0 \leq t_1$ then $t_0 \dot{-} t_1 = 0$. In the sequel $\lessdot$ and $\gtrdot$ are elements of $\{(, [\}$ and $\{),]\}$ respectively. An interval V is of the form $\lessdot b_1, b_2 \gtrdot$ with b_1, b_2 bounds.

For a bound b, the set of time variables occurring in b is denoted by $tvar(b)$. Of course $tvar(\lessdot b, c \gtrdot) = tvar(b) \cup tvar(c)$.

2.3 Process Terms

Let $a \in A \cup \{\delta\}$, $v \in TVar$, V an interval and b a bound. The set of *process terms*, with typical element p, is defined by

$$p ::= \int_{v \in V} a[v] \mid \int_{v \in V} (a[v] \cdot p) \mid p + p \mid p \cdot p \mid p \| p \mid \sigma_{-}^{b}(p)$$

The operator σ_- is called the *(negative) time shift*. It is an auxiliary operator that is needed in the operational semantics of the merge $\|$. The process $\sigma_{-}^{r}(p)$ denotes the process p that is shifted back r time units in time.

In the sequel $\int_{v \in [b,b]} a[v]$ is abbreviated by $a[b]$. Furthermore, we shall use a *scope convention*, saying that if we do not write scope brackets, then the scope is as large as possible. Thus we write $\int_{v \in V} a[v] \cdot p$ for $\int_{v \in V} (a[v] \cdot p)$.

2.4 Time-closed Processes

In general, one cannot attach a transition system to a process term containing time variables that are not 'guarded' by an integral sign. For example, what would be the behaviour of a process $\int_{v \in [x,y]} a[v]$. Therefore, the notion of a *time-closed* process is introduced.

First, define inductively the collection $FV(p)$ of time variables appearing in a process term p that are not bound by an integral sign, the so-called *free* variables:

$$\begin{array}{lcll} FV(\int_{v \in V} a[v]) & = & tvar(V) & \\ FV(\int_{v \in V} a[v] \cdot p) & = & (FV(p) \backslash \{v\}) \cup tvar(V) & \\ FV(p \Box q) & = & FV(p) \cup FV(q) & \Box \in \{+, \cdot, \|\} \\ FV(\sigma_{-}^{b}(p)) & = & FV(p) \cup tvar(b) & \end{array}$$

A term p is called *time-closed* if $FV(p) = \emptyset$. The model for ACP_{rqI} that we shall consider contains only the time-closed process terms.

A *substitution* is a mapping $\sigma : TVar \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$. We can extend σ to the collection of processes; $\sigma(p)$ denotes the process p with all free occurrences of a variable v replaced by $\sigma(v)$. Clearly, $\sigma(p)$ is a time-closed process, and we denote $\sigma(p)$ by $p[\sigma(v_1)/v_1, ..., \sigma(v_k)/v_k]$, where $v_1, ..., v_k$ are the free variables of p.

2.5 Ultimate Delay

The *ultimate delay* $U(p)$ of a time-closed process p is the latest moment in time till which p can idle without executing an initial action. Moller and Tofts have introduced a similar construct [MT90], called the maximum delay. It is

defined inductively as follows, where $a \in A \cup \{\delta\}$.

$$\begin{array}{lcl} U(\int_{v\in V} a[v]) & = & sup(V) \\ U(\int_{v\in V} a[v]\cdot p) & = & sup(V) \\ U(p+q) & = & max\{U(p),U(q)\} \\ U(p\cdot q) & = & U(p) \\ U(p\|q) & = & min\{U(p),U(q)\} \\ U(\sigma^r_-(p)) & = & U(p) \dot{-} r \end{array}$$

The ultimate delay enables to distinguish processes that only differ in their deadlock behaviour. For example, the processes $a(1)+\delta(1)$ and $a(1)+\delta(2)$ can only execute the a at 1, but they have different ultimate delays.

2.6 Operational Semantics

Table 1 contains an operational semantics for ACP_{rqI}, taken from [Klu91]. It is assumed in Table 1 that $a,b \in A$ and $r \in \mathbb{Q}_{>0}$.

The rules defining the communication operators are such that the merge does not result in arbitrary interleavings. For this would result in transitions such as $a[1] \parallel b[2] \xrightarrow{b[2]} \sigma^2_-(a[1])$, which means that the process gets into a deadlock. Such situations are avoided as follows. Suppose that p can execute an action $a[r]$. Then $p\|q$ can execute action $a[r]$ only if $r < U(q)$.

2.7 Bisimulation

We consider process expressions modulo *(strong) bisimulation.*

Definition 2.1 *Two process expressions p_0, q_0 are said to be* strongly bisimilar, *notation $p_0 \leftrightarrows q_0$, if there exists a symmetric, binary* bisimulation relation $\mathcal{R}$ *such that*

1. $p_0 \mathcal{R} q_0$.
2. *If $p \xrightarrow{a[r]} p'$ and $p\mathcal{R}q$, then $q \xrightarrow{a[r]} q'$ for some process q' with $p'\mathcal{R}q'$.*
3. *If $p \xrightarrow{a[r]} \surd$ and $p\mathcal{R}q$, then $q \xrightarrow{a[r]} \surd$.*
4. *If $p\mathcal{R}q$, then $U(p) = U(q)$.*

Strong bisimulation is a congruence, which can be seen by extending the operational semantics with rules defining predicates $U(p) = r$. Such *mix fix predicates* fit into the tyft/tyxt format extended with predicates of Baeten and Verhoef, called the *path* format [BV93]. Strong bisimulation defined by transition rules within this format is always a congruence, and the strong bisimulation that is induced by our extended operational semantics is exactly the one given in Definition 2.1.

$\int_{v\in V}$: If $r \in V\backslash\{0,\infty\}$, then

$$\int_{v\in V} a[v] \xrightarrow{a[r]} \surd \qquad \int_{v\in V} a[v]\cdot p \xrightarrow{a[r]} p[r/v]$$

$\cdot$:
$$\frac{p \xrightarrow{a[r]} \surd}{p\cdot q \xrightarrow{a[r]} q} \qquad \frac{p \xrightarrow{a[r]} p'}{p\cdot q \xrightarrow{a[r]} p'\cdot q}$$

$+$:
$$\frac{p \xrightarrow{a[r]} \surd}{p+q \xrightarrow{a[r]} \surd \quad q+p \xrightarrow{a[r]} \surd} \qquad \frac{p \xrightarrow{a[r]} p'}{p+q \xrightarrow{a[r]} p' \quad q+p \xrightarrow{a[r]} p'}$$

$\|$:
$$\frac{p \xrightarrow{a[r]} \surd \quad r < U(q)}{p\|q \xrightarrow{a[r]} \sigma^r_-(q) \quad q\|p \xrightarrow{a[r]} \sigma^r_-(q)}$$

$$\frac{p \xrightarrow{a[r]} p' \quad r < U(q)}{p\|q \xrightarrow{a[r]} p'\|\sigma^r_-(q) \quad q\|p \xrightarrow{a[r]} \sigma^r_-(q)\|p'}$$

If $a|b = c \neq \delta$, then

$$\frac{p \xrightarrow{a[r]} \surd \quad q \xrightarrow{b[r]} \surd}{p\|q \xrightarrow{c[r]} \surd} \qquad \frac{p \xrightarrow{a[r]} \surd \quad q \xrightarrow{b[r]} q'}{p\|q \xrightarrow{c[r]} q' \quad q\|p \xrightarrow{c[r]} q'}$$

$$\frac{p \xrightarrow{a[r]} p' \quad q \xrightarrow{b[r]} q'}{p\|q \xrightarrow{c[r]} p'\|q'}$$

σ^s_- :
$$\frac{p \xrightarrow{a[r]} \surd \quad s < r}{\sigma^s_-(p) \xrightarrow{a[r-s]} \surd} \qquad \frac{p \xrightarrow{a[r]} p' \quad s < r}{\sigma^s_-(p) \xrightarrow{a[r-s]} p'}$$

Table 1: Action rules for ACP_{rqI}

2.8 Recursion

We define what is a *recursive specification* E. Assume a set V_E of pairs (X, k), with X a recursion variable and k the number of its time parameters. Now E consists of a collection of equations

$$\{X(v_1, ..., v_k) = t_{(X,k)} \mid (X, k) \in V_E\}$$

where $v_1, ..., v_k$ denote time variables and $t_{(X,k)}$ a process expression, possibly containing expressions of the form $Y(b_1, ..., b_l)$, where $(Y, l) \in V_E$ and $b_1, ..., b_l$ bounds.

In this paper we shall only consider *finite* recursive specifications, i.e. it is assumed that for each recursive specification E the collection V_E is finite.

In order to define the collections $FV(t_{(X,k)})$, it is sufficient to extend the definition of free variables to expressions $X(b_1, ..., b_k)$:

$$FV(X(b_1, ..., b_k)) = tvar(b_1) \cup ... \cup tvar(b_k)$$

A recursive specification is called *well-defined* if all its equations $X(v_1, ..., v_k) = t_{(X,k)}$ satisfy $FV(t_{(X,k)}) \subseteq \{v_1, ..., v_k\}$.

The notion of a *process term* is extended by allowing expressions of the form $\langle X(b_1, ..., b_k) | E \rangle$ with E a well-defined specification and $(X, k) \in V_E$ and $b_1, ..., b_k$ bounds. By abuse of notation, $\langle X(b_1, ..., b_k) | E \rangle$ is often denoted by $X(b_1, ..., b_k)$.

In the following table the operational semantics for time-closed process terms is extended to recursion variables, by supplying $\langle X(r_1, ...r_k) | E \rangle$ with the action rules of $t_{(X,k)}[r_1/v_1, ..., r_k/v_k]$.

$$\frac{\langle t_{(X,k)}[r_1/v_1, ..., r_k/v_k] \mid E \rangle \xrightarrow{a[r]} \surd}{\langle X(r_1, ..., r_k) \mid E \rangle \xrightarrow{a[r]} \surd} \qquad \frac{\langle t_{(X,k)}[r_1/v_1, ..., r_k/v_k] \mid E \rangle \xrightarrow{a[r]} p}{\langle X(r_1, ..., r_k) \mid E \rangle \xrightarrow{a[r]} p}$$

Here, the term $\langle t_{(X,k)}[r_1/v_1, ..., r_k/v_k] \mid E \rangle$ denotes $t_{(X,k)}[r_1/v_1, ..., r_k/v_k]$ with each occurrence of expressions $Y(b_1, ..., b_l)$ replaced by $\langle Y(b_1, ..., b_l) | E \rangle$.

3 Eliminating the Merge

We prove an Elimination Theorem for a class of regular behaviours. Usually, regularity of a process is defined in terms of having a finite number of states. However, in the present setting this definition would backfire, due to the presence of the integral construct, which causes even finite processes to have an infinite number of states. Therefore, we use a different definition for regularity here.

3.1 Linear Recursive Specifications

In untimed process algebras one can prove, for suitable models, that a process is regular if and only if it is equivalent to a solution of a finite linear recursive specification [Mil84]. Here, we use this property as the definition of regularity.

A finite recursive specification over ACP_{rqI} shall be called *linear* if all its equations are of the form

$$X(v_1,...,v_k) = \sum_i \int_{v\in V_i} a_i[v] \cdot Y_i(b_{i1},...,b_{il_i}) + \sum_j \int_{v\in V'_j} a'_j[v] \qquad (1)$$

A process is called *regular* if it is bisimilar to a solution of a linear recursive specification.

3.2 A Counter-example

The following simple example shows that one cannot hope to find an Elimination Theorem for general regular processes.

Example 3.1 *Define*

$$\begin{aligned} X &= \textstyle\int_{v\in\langle 0,1\rangle} a[v] \cdot Y(v) \\ Y(v) &= \textstyle\int_{w\in[v,v]} a[w] \cdot Y(w) \end{aligned}$$

Consider the process $p = X \| b[1]$. *For convenience we put* $a|b = \delta$.

Each trace of the process X is of the form $a[r] \cdot a[r] \cdot a[r] \cdot ...$ with $r \in \langle 0,1\rangle$. Let $a[r]$ be the first transition that is executed by p. If $r \in \langle 1/(n+1), 1/n\rangle$ for certain n, then p will execute this $a[r]$ n times, followed by $b[1-nr]$, then $a[(n+1)r-1]$, and after that only $a[r]$'s. And if $r = 1/(n+1)$, then p will get into a deadlock, after n times executing $a[r]$.

So a linear specification describing the behaviour of p would have to contain infinitely many summands. Hence, p is not regular. *(End example)*

Thus, the class of solutions of linear specifications is too big for finding an Elimination Theorem. On the other hand, if no occurrences of time variables are allowed in the process terms, then the collection is clearly too small. Because in this class even equivalences such as

$$\int_{v\in\langle 0,1\rangle} a[v] \parallel \int_{w\in\langle 1,2\rangle} b[w] \leftrightarrows \int_{v\in\langle 0,1\rangle} a[v] \cdot \int_{w\in\langle 1\dot{-}v, 2\dot{-}v\rangle} b[w]$$

cannot be expressed anymore.

So, is there an algebra in between, for which an Elimination Theorem can be deduced? The answer is yes.

3.3 Strong Regularity

Define a process to be *strongly* regular if it is a solution of a linear specification (of the form 1) that satisfies the following requirements:

- the bounds in the V_i are all of the form r or $r \dot{-} v_j$, where $r \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$.
- the bounds b_{ij} are all of the form v or $v_j + v$, where $v_j \not\equiv v$.

We shall deduce an Elimination Theorem for this algebra. But first, we give two more examples to show that the elimination result would not hold for a less restrictive definition of strong regularity.

3.4 Two More Examples

Example 3.1 already showed that if one allows not only expressions r and $r \dot{-} v_j$, but also variables v_j as bounds in the intervals of strong regularity, then the elimination result is lost. The following example implies that neither can one allow variables v_j as bounds b_{ij}.

Example 3.2 *Define*

$$\begin{aligned} X &= \int_{v \in \langle 0,1 \rangle} a[v] \cdot Y(v) \\ Y(v) &= \int_{w \in [1 \dot{-} v, 1 \dot{-} v]} a[w] \cdot Y(v) \end{aligned}$$

The process $p = X \| b[2]$ (with $a|b = \delta$) is not a solution of a linear specification.

Each trace of the process X is of the form $a[r] \cdot a[1-r] \cdot a[1-r] \cdot \ldots$ with $r \in \langle 0,1 \rangle$. So if the first transition that p executes is $a[r]$ with $r \in \langle (n-2)/(n-1), (n-1)/n \rangle$ for some $n \geq 2$, then p will first execute $n+1$ a's, then a b and then only a's. And if $r = (n-2)/(n-1)$, then p will execute $n+1$ a's and get into a deadlock. *(End example)*

Finally, the following example shows that one cannot allow expressions $r \dot{-} s \cdot v$ as bounds in the intervals V_i, where $s \in \mathbb{Q}_{>0}$. This example is a bit more complicated than the previous ones.

Example 3.3 *Define*

$$\begin{aligned} X_1 &= \int_{v \in \langle 0,1 \rangle} a[v] \cdot X_2(v) & \qquad Y_1 &= b[\tfrac{1}{2}] \cdot Y_2 \\ X_2(v) &= \int_{w \in [\frac{3}{2} \dot{-} \frac{1}{2}v, \frac{3}{2} \dot{-} \frac{1}{2}v]} a[w] \cdot X_2(w) & \qquad Y_2 &= b[1] \cdot Y_2 \end{aligned}$$

The process $X_1 \| Y_1$ (with $a|b = \delta$) is not a solution of a linear specification.

Let us consider the processes X_1 and Y_1 in absolute time for a moment. An easy calculation tells that if X_1 executes its first action at time r, then its nth action will be executed at (absolute) time $n-(1-r)\alpha_n$, where

$$\alpha_n = \sum_{i=0}^{n-1}(-\frac{1}{2})^i$$

So if the first action of X_1 is $a[r]$, then for eliminating the merge from $X_1 \| Y_1$, it is essential to know whether $\alpha_n(1-r)$ is smaller or greater than $1/2$ for $n = 1, 2, ...$ Because this inequality implies whether the nth a-action is executed after or before the nth b-action. And if $(1-r)\alpha_n = 1/2$ for some n, then $X_1 \| Y_1$ will deadlock after $n-1$ transitions.

The equalities $r = 1 - 1/(2\alpha_n)$, $n = 1, 2, ...$ give an infinite partition of the interval $\langle 0, 1\rangle$. *(End example)*

3.5 The Elimination Theorem

For the algebra of strongly regular processes we have an Elimination Theorem.

Theorem 3.4 *(Elimination Theorem) For each pair of strongly regular processes p and q, there exists a strongly regular process z such that $p \| q \leftrightarrows z$.*

We do not give the full proof of this theorem, which is quite technical and takes many pages, but confine ourselves to an outline. For further details, the reader is referred to [Fok93].

First, we formulate a preliminary lemma. Assume a finite collection of bounds $\mathcal{B}$. An *ordering* α on $\mathcal{B}$ consists of an anti-symmetric relation $<_\alpha$ together with a symmetric relation $=_\alpha$ on $\mathcal{B}$, such that for each pair $b, b' \in \mathcal{B}$ either $b <_\alpha b'$ or $b' <_\alpha b$ or $b =_\alpha b'$.

For an ordering α on $\mathcal{B}$ and a substitution $\sigma : TVar \to \mathbb{Q}_{\geq 0}$, the set of relations $\sigma(\alpha)$ results to either true (if all (in)equalities are true) or false (otherwise). Let $[\alpha]$ denote the collection of substitutions σ for which $\sigma(\alpha)$ is true. Note that for each σ there is exactly one ordering α on $\mathcal{B}$ such that $\sigma \in [\alpha]$.

We consider a specific set of bounds. Let $b_1, ..., b_m$ denote bounds and $t \in \mathbb{Q}_{>0}$ and $N \in \mathbb{N}$. Then $\mathcal{B}_{t,N}(b_1, ..., b_m)$ is defined to be the following set of bounds:

$$\mathcal{B}_{t,N}(b_1, ..., b_m) \;=\; \{kt,\; kt \dot{-} b_i \mid k = 0, ..., N,\; i = 1, ..., m\} \cup \{\infty\}$$

Now the lemma is as follows. Let $x, x_1, ..., x_n$ denote time variables.

Lemma 3.5 *For each ordering α of $\mathcal{B}_{t,N}(x_1, ..., x_n) \cup \{x\}$ there is an ordering β of $\mathcal{B}_{t,N}(x, x_1 + x, ..., x_n + x)$ such that $[\alpha] \subseteq [\beta]$.*

The proof of this lemma is omitted. It consists of simply rewriting each possible relation in $\mathcal{B}_{t,N}(x, x_1 + x, ..., x_n + x)$ to a relation in $\mathcal{B}_{t,N}(x_1, ..., x_n) \cup \{x\}$.

Sketch of the proof of the Elimination Theorem:
Assume two strongly regular processes p and q. We construct a strongly regular process z such that $p \| q \leftrightarrows z$.

By definition, p and q are bisimilar to processes $\langle X_1(r_1, ..., r_{m(1)}) | E \rangle$ and $\langle Y_1(s_1, ..., s_{n(1)}) | E' \rangle$ with $r_i, s_i \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$, and the equations of E and E' are of the form

$$X_I(v_1, ..., v_{m(I)}) = \sum_{k \in K} \int_{v \in V_k} a_k[v] \cdot X_{I_k}(b_{k1}, ..., b_{km(I_k)}) + \sum_{l \in L} \int_{v \in W_l} b_l[v] \quad (2)$$

$$Y_J(w_1, ..., w_{n(J)}) = \sum_{k \in K'} \int_{v \in V'_k} a'_k[v] \cdot Y_{J_k}(b'_{k1}, ..., b'_{kn(J_k)}) + \sum_{l \in L'} \int_{v \in W'_l} b'_l[v] \quad (3)$$

where the bounds in the intervals V_k and W_l (respectively V'_k and W'_l) are expressions r or $r \dot{-} v_i$ (respectively $r \dot{-} w_i$) with $r \in \mathbb{Q}_{\geq 0} \cup \{\infty\}$, and the bounds b_{ki} (respectively b'_{ki}) are expressions v or $v_j + v$ (respectively $w_j + v$).

Let $\{t_1, ..., t_m\}$ be the collection of rationals that occur in E and E'. We can assume this set to be non-empty. Let t be the greatest common divisor of this collection, i.e., the greatest rational such that t_i/t is a natural number for each i. Define

$$N = max\{t_1/t, ..., t_m/t\}$$

Ensure, by applying α-conversion, that the v_i and the w_j are all different. In the linear specification to describe the behaviour of $p \| q$ consists of the following recursion variables (apart from the X_I and Y_J). For each recursion variable $W^\alpha(x_1, ..., x_n)$ we assume that α ranges over all orderings of $\mathcal{B}_{t,N}(x_1, ..., x_n)$ for which $[\alpha] \neq \emptyset$.

1 $\tilde{X}_I(x, x_1, ..., x_{m(I)})$

The expression $\tilde{X}_I(v, v_1 + v, ..., v_{m(I)} + v)$ describes the behaviour of the process $\sigma^v_-(X_I(v_1, ..., v_{m(I)}))$.

2 $\tilde{Y}_J(x, y_1, ..., y_{n(J)})$

The expression $\tilde{Y}_J(v, w_1 + v, ..., w_{n(J)} + v)$ describes the behaviour of the process $\sigma^v_-(Y_J(w_1, ..., w_{n(J)}))$.

3 $Z^\alpha_{IJ}(x_1, ..., x_{m(I)}, y_1, ..., y_{n(J)})$

This recursion variable describes the behaviour under condition α of the process $X_I(x_1, ..., x_{m(I)}) \| Y_J(y_1, ..., y_{n(J)})$.

4 $\tilde{Z}^{\alpha}_{IJ}(x, x_1, ..., x_{m(I)}, y_1, ..., y_{n(J)})$

This recursion variable describes the behaviour under condition α of the process $\tilde{X}_I(x, x_1, ..., x_{m(I)}) \| Y_J(y_1, ..., y_{n(J)})$.

5 $\bar{Z}^{\alpha}_{IJ}(x_1, ..., x_{m(I)}, x, y_1, ..., y_{n(J)})$

This recursion variable describes the behaviour under condition α of the process $X_I(x_1, ..., x_{m(I)}) \| \bar{Y}_J(x, y_1, ..., y_{n(J)})$.

We consider the process $X_I(v_1, ..., v_{m(I)}) \| Y_J(w_1, ..., w_{n(J)})$, to explain why we need all these recursion variables. The behaviour of this expression cannot be described by only one linear equation, due to the appearance of open variables. Therefore, we have introduced a collection of variables Z^{α}_{IJ}, where α ranges over orderings on $\mathcal{B}_{t,N}(v_1, ..., v_{m(I)}, w_1, ..., w_{n(J)})$ for which $[\alpha] \neq \emptyset$.

We give a short, intuitive description to explain the crux in constructing the equation for such an expression $Z^{\alpha}_{IJ}(v_1, ..., v_{m(I)}, w_1, ..., w_{n(J)})$. Suppose that it executes an initial action $a_k[v]$ with $v \in V_k$ (originating from X_I). The resulting behaviour is

$$X_{I_k}(b_{k1}, ..., b_{km(I_k)}) \| \sigma^v_-(Y_J(w_1, ..., w_{n(J)}))$$

which is described by $\bar{Z}^{\beta}_{I_kJ}(b_{k1}, ..., b_{km(I_k)}, v, w_1 + v, ..., w_{n(J)} + v)$. The only problem is, what is the condition β? Or in other words, does the condition $\alpha \wedge v \in V_k$ imply an ordering on $\mathcal{B}_{t,N}(b_{k1}, ..., b_{km(I_k)}, v, w_1 + v, ..., w_{n(J)} + v)$? According to Lemma 3.5, the answer is yes if $\alpha \wedge v \in V_k$ implies an ordering on $\mathcal{B}_{t,N}(v_1, ..., v_{m(I)}, w_1, ..., w_{n(J)}) \cup \{v\}$. And this can be ensured by partitioning the intervals V_k into sufficiently small subintervals. *(End sketch of proof)*

3.6 An Example

We study an example, to give some more intuition for the Elimination Theorem. It turns out that even for very simple strongly regular processes, the behaviour of their merge can only be described by a very complicated linear specification.

Example 3.6 *Define*

$$X = \int_{v \in (0,1)} a[v] \cdot X$$

Let $p = X \| b[k]$, where $k \in \mathbb{N}$ and $a|b = \delta$. The process p is strongly regular.

The behaviour of p can be described by the following linear specification, containing $k+3$ equations.

$$
\begin{aligned}
X_0 &= \int_{v\in\langle 0,1\rangle} a[v]\cdot X_1(v)\\
X_i(v) &= \int_{w\in\langle 0,i\dot{-}v]} a[w]\cdot X_i(v+w) + \int_{w\in\langle i\dot{-}v,1\rangle} a[w]\cdot X_{i+1}(v+w)\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i=1,\ldots,k-1\\
X_k(v) &= \int_{w\in\langle 0,k\dot{-}v\rangle} a[w]\cdot X_k(v+w) + \int_{w\in[k\dot{-}v,k\dot{-}v]} b[w]\cdot Y(w)\\
Y(v) &= \int_{w\in\langle 0,1\dot{-}v\rangle} a[w]\cdot X\\
X &= \int_{v\in\langle 0,1\rangle} a[v]\cdot X
\end{aligned}
$$

The idea behind this specification is quite easy. Process p will execute X until it reaches (absolute) time k, when it executes a b. After that it continues with X. Now, when has p the possibility of executing b at k? This is if p has executed an a after time $k-1$. So if this is the case, the linear specification must take into account the execution of b at k. And when can p execute an a after $k-1$? If it has executed an a after $k-2$. So if this is the case, the linear specification must take into account the execution of a after $k-1$, etc.

The equations for the X_i, with $i=1,\ldots,k-1$, register whether a is executed after time i or not. If so, then X_{i+1} is triggered, and otherwise X_i is repeated. Finally, X_k takes into account the execution of b. *(End example)*

3.7 Timed Automata

An *automaton* consists of a set of states S, a set of start states $S_0 \subseteq S$, a set of labels A and a set of transitions $E \subseteq S \times A \times S$. The *language* accepted by the automaton consists of all traces $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$ such that $(s_i, a_i, s_{i+1}) \in E$ for $i = 0, 1, 2, \ldots$ Furthermore, the trace must satisfy certain fairness requirements, e.g., that it reaches a specific state an infinite number of times.

The algebra of strongly regular processes can be linked to the class of *timed* automata of Alur and Dill [AD90]. For a timed automaton, the elements of E are supplied with timing constraints on 'clock variables'. These constraints are of the form $x < r$ or $x \leq r$ or $x > r$ or $x \geq r$ with x a clock variable and $r \in \mathbb{Q}_{\geq 0}$, and there is a construct $x := 0$, denoting that while executing this transition, clock x is set back to zero. A trace is only accepted by a timed automaton if its transitions are performed at times that all the clocks satisfy their constraints. Again, there is a fairness requirement for accepted traces. Furthermore, Zeno behaviour has been explicitly excluded from timed

automata, i.e., traces are only accepted if they progress beyond every moment in time.

The fairness restrictions, the non-Zeno requirement and the fact that only infinite traces are considered are obstacles for the translation between timed automata and strongly regular processes, since ACP_{rqI} does not take into account such semantic restrictions. However, if these restrictions are discarded, then the classes of strongly regular processes and of timed automata turn out to be equivalent.

We can translate strongly regular processes to the setting of timed automata as follows. A strongly regular processes executes an action $a[v]$ under restrictions of the form $v \Box r$ or $v \Box (r \dot{-} v_i)$, with $\Box \in \{<, >, \leq, \geq\}$ and $r \in \mathbb{Q}_{\geq 0}$. These last inequalities can be rewritten to the form $(v_i + v) \Box r$. The $v_i + v$ and v can be regarded as clocks. Since we work in relative time, the v has been set back to zero by $v := 0$ in the previous transition. The state that results after executing $a[v]$ is a recursive expression of the form $X(v, v_1 + v, ..., v_k + v)$. The $v_i + v$ and v are some kind of memories, which store the actual times of the clocks at the moment of the transition $a[v]$.

Conversely, the language accepted by a timed automaton (without semantic restrictions) can always be described by a strongly regular process. We give a simple example.

Example 3.7 *We consider a timed automaton with states s_0, s_1, of which s_0 is start state, and clock variables x, y. The timed automaton is defined by the following two transitions:*

- (s_0, a, s_1) *with time constraints* $x < 2,\ y := 0$,

- (s_1, b, s_0) *with time constraints* $x < 3,\ y < 2,\ x := 0$.

This automaton executes alternately a and b. We describe its behaviour by a linear specification.

$$\begin{aligned} X &= \int_{v \in (0,1]} a[v] \cdot Y_1 + \int_{v \in [1,2)} a[v] \cdot Y_2(v) \\ Y_1 &= \int_{v \in (0,2)} b[v] \cdot X \\ Y_2(v) &= \int_{w \in (0, 3 \dot{-} v)} b[w] \cdot X \end{aligned}$$

The process X describes the behaviour of the timed automaton.

(End example)

References

[AD90] R. Alur and D. Dill. Automata for modeling real-time behaviour. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, LNCS 443, pages 322–335. Springer-Verlag, 1990.

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

[BV93] J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. Report CSN-93/05, Eindhoven University of Technology, Eindhoven, 1993.

[Fok92] W.J. Fokkink. Regular processes with rational time and silent steps. Report CS-R9231, CWI, Amsterdam, 1992.

[Fok93] W.J. Fokkink. An elimination theorem for regular behaviours with integration. Technical report, CWI, Amsterdam, 1993.

[GL92] J.C. Godskesen and K.G. Larsen. Real-time calculi and expansion theorems. In R. Shyamasundar, editor, *Proceedings* 12^{th} *Conference on Foundations of Software Technology and Theoretical Computer Science,* New Delhi, India, LNCS 652, pages 302–315. Springer-Verlag, 1992.

[Klu91] A.S. Klusener. Completeness in real time process algebra. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, LNCS 527, pages 376–392. Springer-Verlag, 1991.

[Mil84] R. Milner. A complete inference system for a class of regular behaviours. *Journal of Computer and System Sciences*, 28:439–466, 1984.

[MT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings CONCUR 90,* Amsterdam, LNCS 458, pages 401–415. Springer-Verlag, 1990.

[NS90] X. Nicollin and J. Sifakis. ATP: An algebra for timed processes. Technical Report RT-C26, IMAG, Laboratoire de Génie informatique, Grenoble, 1990.

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

[Wan90] Y. Wang. *A Calculus of Real Time Systems.* PhD thesis, Chalmers University of Technology, Göteborg, 1990.