An Elimination Theorem for Regular Behaviours with Integration

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

Abstract

This chapter deals with an extension of the process algebra ACP with rational time and integration. We determine a proper subdomain of the regular processes for which an elimination theorem holds, namely, for each pair of processes p_0, p_1 in this class there is a process q in this class such that $p_0 \| p_1$ and q are bisimilar. Some simple examples show that if this subdomain is enlarged, then the elimination result is lost. The subdomain is equivalent to the model of timed automata from Alur and Dill.

1 Introduction

In recent years, process algebras such as CCS, CSP and ACP, have been extended with constructs that mean to describe some notion of either discrete or dense time. This chapter is based on the approach of Baeten and Bergstra [3], which extends ACP with real time. They introduced the notion of integration, which expresses the possibility that an action occurs somewhere within a time interval. The construct $\int_{v \in V} p$ executes the process p, where the behaviour of p may depend on the value of v in the time interval V. In this chapter, we restrict to prefix integration, and integration is parametrized by conditions, which consist of inequalities between linear expressions of variables. These notions originate from Klusener [10].

This chapter concerns regular processes. Traditionally, a process is regular if it consists of a finite number of states. 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 transitions. Therefore, a regular process is defined to be the solution of a linear specification, which is motivated by the fact that regular processes in the untimed case are exactly the solutions of linear specifications.

For the sake of verification in process algebra, it is important to have an elimination theorem which says that the parallel composition of two regular processes is again a regular process. Namely, in general a verification deals with a process $\partial_H(p_1 \| \cdots \| p_k)$,

where $p_1, ..., p_k$ are regular. Elimination theorems have been deduced for regular processes in untimed ACP (see [4]), and in timed ACP without integration [6]. In this chapter, we set out to deduce an elimination theorem for regular processes in timed ACP with integration. A simple example will show that in general, the merge can *not* be eliminated from processes in this algebra. We will determine a subclass of 'strongly' regular processes, for which an elimination theorem does hold. Furthermore, some more examples will show that the elimination result is lost if the subalgebra is enlarged in any obvious way.

At first sight, the syntactic restrictions for the subdomain of strongly regular processes may seem quite arbitrary. However, if one studies the examples more closely, then it turns out that linear specifications which do not satisfy these restrictions tend to describe different kinds of irregular behaviour, such as accelerations and oscillations. The subdomain seems to be sufficiently wide for practical purposes, see e.g. Hillebrand [9].

The subdomain of strongly regular processes can be linked to the class of timed automata from Alur and Dill [2]. It is not possible to obtain a precise translation between the processes in our subdomain and timed automata, due to the requirement of non-Zeno behaviour and the presence of fairness restrictions for languages accepted by timed automata. If these restrictions are discarded, then the classes of strongly regular processes and of timed automata turn out to be equivalent. Hence, the operations from ACP can be used to compose smaller timed automata into larger ones. This compositionality is missing in existing timed automata work.

The proof of the elimination theorem turns out to be deplorably complicated. In order to keep the exposition as simple as possible, the left merge $\between$, the communication operator $|$ and the encapsulation operator ∂_H will be excluded from to the syntax. The elimination result extends to the syntax which does incorporate these constructs without any complications.

Acknowledgements. Steven Klusener and an anonymous referee provided helpful comments, and Frits Vaandrager suggested the link with timed automata.

2 The Syntax and Semantics

This section contains a description of the syntax and semantics for ACP with relative time and integration, denoted by ACPrI, together with recursion. For the elimination result it is essential that we restrict the time domain to the rational numbers. For example, if $X = a[1] \cdot X$ and $Y = b[\sqrt{2}] \cdot Y$, then the merge cannot be eliminated from $X \| Y$.

2.1 Bounds and conditions

In the sequel, we assume a countably infinite set of time variables $TVar$.

Definition 2.1 *The set of* bounds*, with typical element b, is defined as follows, where $r \in \mathbb{Q}$ and $v \in TVar$.*

$$b \ ::= \ r \ \mid \ v \ \mid \ b + b \ \mid \ r \cdot b.$$

The expression $b_0 - b_1$ abbreviates $b_0 + (-1) \cdot b_1$.

Definition 2.2 *The set of* conditions*, with typical element ϕ, is defined by*

$$\phi \ ::= \ b_0 \leq b_1 \ \mid \ \phi \wedge \phi \ \mid \ \neg\phi.$$

In the sequel, we use the abbreviations $\phi \vee \psi$ for $\neg(\neg\phi \wedge \neg\psi)$, and tt for $\neg(1 \leq 0)$, and ff for $1 \leq 0$, and $b_0 < b_1$ for $\neg(b_1 \leq b_0)$, and $b_0 = b_1$ for $b_0 \leq b_1 \wedge b_1 \leq b_0$, and $b_0 < v < b_1$ for $b_0 < v \wedge v < b_1$.

2.2 Process terms

We assume a countable alphabet A of atomic actions, together with a special constant δ, representing deadlock. In the sequel, a and α denote elements of A and $A \cup \{\delta\}$ respectively. Furthermore, we assume a communication function $| : A \times A \to A$, which is commutative and associative and has δ as zero element.

Integration enables to express that the behaviour of a process may depend on the value of a time variable. If a process p depends on the value of v between 1 and 2, then we write

$$\int_{1<v<2} p.$$

Here, integration is parametrized by conditions, and we deal with *prefix* integration $\int_\phi \alpha[v] \cdot p$. If $\alpha \neq \delta$, then this process can execute the action $\alpha[r]$ under the condition that $\phi[r/v]$ is true, after which the process results to $p[r/v]$.

In ACPrI, process terms are constructed from prefix integration, the alternative composition $x + y$, the merge $x \| y$ and the time shift $(b)x$, where b is a bound. The time shift is an auxiliary operator that is needed in the operational semantics of the merge; the process $(b)p$ denotes the process p that is shifted forward b time units in time. As binding convention, merge, integration and time shift bind stronger than alternative composition.

Assume a finite set V_E of recursion variables, where each variable X in V_E is supplied with an arity $ar(X)$. The recursive specification E consists of a collection

$$\{X(v_1, ..., v_{ar(X)}) = t_X \mid X \in V_E\},$$

where $v_1, ..., v_{ar(X)}$ are distinct time variables, and t_X is a process term constructed from integration, the alternative composition, the merge, the time shift, and expressions of the form $Y(b_1, ..., b_{ar(Y)})$, where $Y \in V_E$ and $b_1, ..., b_{ar(Y)}$ bounds.

Definition 2.3 *E is called* well-defined *if each term t_X for $X \in V_E$ contains only the time variables $v_1, ..., v_{ar(X)}$.*

The set of process terms, with typical element p, is defined by

$$p \ ::= \ \int_{\phi} \alpha[v] \ \mid \ \int_{\phi} \alpha[v] \cdot p \ \mid \ p + p \ \mid \ p \| p \ \mid \ (b)p \ \mid \ \langle X(b_1, ..., b_{ar(X)}) | E \rangle.$$

where E is well-defined.

In the sequel, $\int_{v=b} \alpha[v]$ is often abbreviated to $\alpha[b]$, and $\langle X(b_1, ..., b_{ar(X)}) | E \rangle$ to $X(b_1, ..., b_{ar(X)})$.

2.3 Free variables and substitutions

In general, one cannot attach a transition system to a process term that contains time variables which are not bound by an integral sign. Therefore, we need the notion of a time-closed process. In the term $\int_{\phi} \alpha[v]$, occurrences of v in ϕ are bound, and in $\int_{\phi} \alpha[v] \cdot p$, occurrences of v in ϕ and in p are bound. A time variable in a process term is called free if it is not bound by any integral signs. The collection of free variables in a term p is denoted by $fvar(p)$. A term p is called *time-closed* if $fvar(p) = \emptyset$. As a model we take the collection of time-closed process terms, modulo bisimulation.

A substitution is a mapping from time variables to bounds. For b a bound and σ a substitution, $\sigma(b)$ denotes the bound that results from substituting $\sigma(v)$ for time variables v in b. Substitutions extend to conditions as expected. A substitution $\sigma : TVar \to \mathbb{Q}_{\geq 0}$ is called a *valuation*.

For a process term p, $\sigma(p)$ is obtained by replacing free occurrences of time variables v by $\sigma(v)$. In this definition however, there is one serious complication. Namely, if a free occurrence of v in p has been replaced by $\sigma(v)$, then variables w that occur in $\sigma(v)$ may suddenly be bound in subterms $\int_{\phi} \alpha[w] \cdot q$ of p. A solution for this problem, which originates from the λ-calculus, is to allow unrestricted substitution by renaming bound variables. In the sequel, process terms are considered modulo α-conversion, and when a substitution is applied, bound variables are renamed. Stoughton [12] presented a simple treatment of this technique.

If for a substitution σ there are finitely many variables $v_1, ..., v_n$ such that $\sigma(v_i) \neq v_i$, then often we will use the standard notation $p[\sigma(v_1)/v_1, ..., \sigma(v_n)/v_n]$ for $\sigma(p)$.

2.4 Operational semantics

Table 1 contains an operational semantics for time-closed processes, taken from [7]. We focus on 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. Time starts at zero, which means that actions with negative time stamps do not display any behaviour.

The timed deadlock $\delta[r]$ idles until time r. For example, the process $\int_{v<1} a[v] + \delta[3]$ either executes the a before time 1, or idles until time 3. On the other hand, the process $\int_{v<1} a[v] + \delta[1]$ will always execute the a before time 1. In order to distinguish

processes that only differ in their deadlock behaviour, we introduce the delay predicate $U_r(p)$, which holds if p can idle beyond time r. Processes that only differ in their deadlock behaviour have distinct delays. For example, $U_2(\int_{v<1} a[v] + \delta[3])$, but not $U_2(\int_{v<1} a[v] + \delta[1])$.

The rules which define the communication operators are such that the merge does not result in arbitrary interleavings. For this would result in transitions such as $a[1] \parallel b[2] \xrightarrow{b[2]} (-2)a[1]$, where $(-2)a[1]$ does not display any behaviour. Such situations are avoided by a side condition. Namely, if a process p can execute an action $a[r]$, then $p\|q$ can execute $a[r]$ only if $U_r(q)$.

In the action rules for recursion, the construct $\langle t_X[r_1/v_1, ..., r_{ar(X)}/v_{ar(X)}] \mid E\rangle$ denotes $t_X[r_1/v_1, ..., r_{ar(X)}/v_{ar(X)}]$ with each occurrence of expressions $Y(b_1, ..., b_{ar(Y)})$ replaced by $\langle Y(b_1, ..., b_{ar(Y)})|E\rangle$.

2.5 Bisimulation

Time-closed process terms are considered modulo *(strong) bisimulation.*

Definition 2.4 *Two time-closed process terms p_0 and q_0 are* bisimilar, *denoted by $p_0 \leftrightarrows q_0$, if there exists a symmetric binary* bisimulation relation $\mathcal{B}$ *on time-closed process terms such that*

- $p_0 \mathcal{B} q_0$,
- *if $p \xrightarrow{a[r]} p'$ and $p\mathcal{B}q$, then $q \xrightarrow{a[r]} q'$ for some process q' with $p'\mathcal{B}q'$,*
- *if $p \xrightarrow{a[r]} \surd$ and $p\mathcal{B}q$, then $q \xrightarrow{a[r]} \surd$,*
- *if $U_r(p)$ and $p\mathcal{B}q$, then $U_r(q)$.*

The rules in Table 1 can be fit into the congruence format for action rules with types, data and variable binding of Bloom and Vaandrager [5]. Strong bisimulation defined by transition rules within this format is always a congruence on the algebra of closed terms, which means that if $p \leftrightarrows p'$ and $q \leftrightarrows q'$, then $p + q \leftrightarrows p' + q'$ and $p\|q \leftrightarrows p'\|q'$ and $\int_\phi \alpha[v] \cdot p \leftrightarrows \int_\phi \alpha[v] \cdot p'$ and $(b)p \leftrightarrows (b)p'$.

3 An Elimination Theorem

3.1 Regular processes

Usually, a process is called regular if it has a finite number of states. In the present setting this definition would backfire, since the integral construct causes even finite processes to have an infinite number of states. In untimed ACP one can prove, for suitable models, that a process is regular if and only if it is equal to a solution of

$$\frac{\phi[r/v] \quad r>0}{\int_\phi a[v] \xrightarrow{a[r]} \surd} \qquad \frac{\phi[r/v] \quad r>0}{\int_\phi a[v]\cdot x \xrightarrow{a[r]} x[r/v]}$$

$$\frac{x \xrightarrow{a[r]} \surd}{x+y \xrightarrow{a[r]} \surd \xleftarrow{a[r]} y+x} \qquad \frac{x \xrightarrow{a[r]} x'}{x+y \xrightarrow{a[r]} x' \xleftarrow{a[r]} y+x}$$

$$\frac{x \xrightarrow{a[r]} \surd \quad U_r(y)}{x\|y \xrightarrow{a[r]} (-r)y \xleftarrow{a[r]} y\|x} \qquad \frac{x \xrightarrow{a[r]} x' \quad U_r(y)}{x\|y \xrightarrow{a[r]} x'\|(-r)y \quad y\|x \xrightarrow{a[r]} (-r)y\|x'}$$

$$\frac{x \xrightarrow{a[r]} \surd \quad y \xrightarrow{a'[r]} \surd}{x\|y \xrightarrow{(a|a')[r]} \surd} \qquad \frac{x \xrightarrow{a[r]} x' \quad y \xrightarrow{a'[r]} y'}{x\|y \xrightarrow{(a|a')[r]} x'\|y'}$$

$$\frac{x \xrightarrow{a[r]} \surd \quad y \xrightarrow{a'[r]} y'}{x\|y \xrightarrow{(a|a')[r]} y' \xleftarrow{(a|a')[r]} y\|x}$$

$$\frac{x \xrightarrow{a[r]} \surd \quad r+b=s>0}{(b)x \xrightarrow{a[s]} \surd} \qquad \frac{x \xrightarrow{a[r]} x' \quad r+b=s>0}{(b)x \xrightarrow{a[s]} x'}$$

$$\frac{\langle t_X[r_1/v_1,...,r_{ar(X)}/v_{ar(X)}] \mid E\rangle \xrightarrow{a[r]} \surd}{\langle X(r_1,...,r_{ar(X)}) \mid E\rangle \xrightarrow{a[r]} \surd} \qquad \frac{\langle t_X[r_1/v_1,...,r_{ar(X)}/v_{ar(X)}] \mid E\rangle \xrightarrow{a[r]} y}{\langle X(r_1,...,r_{ar(X)}) \mid E\rangle \xrightarrow{a[r]} y}$$

$$\frac{\phi[s/v] \quad 0<r<s}{U_r(\int_\phi \alpha[v]) \quad U_r(\int_\phi \alpha[v]\cdot x)} \qquad \frac{U_r(x)}{U_r(x+y) \quad U_r(y+x)}$$

$$\frac{U_r(x) \quad r+b=s>0}{U_s((b)x)} \qquad \frac{0<r<b}{U_r((b)x)}$$

$$\frac{U_r(x) \quad U_r(y)}{U_r(x\|y)} \qquad \frac{U_r(t_X[b_1/v_1,...,b_{ar(X)}/v_{ar(X)}])}{U_r(X(b_1,...,b_{ar(X)}))}$$

Table 1: Action rules for ACPrI

a linear recursive specification [11]. Here, we use this property as the definition of regularity. A recursive specification is linear if its equations are of the form

$$X(v_1, ..., v_{ar(X)}) = \sum_i \int_{\phi_i} a_i[w] \cdot Y_i(b_{i1}, ..., b_{iar(Y_i)}) + \sum_j \int_{\psi_j} \alpha_j[w].$$

A time-closed process is *regular* if it is bisimilar to a process $\langle X|E\rangle$, where E is linear.

3.2 A counter-example

We want to prove an elimination theorem for a class of regular behaviours. An easy example will learn that such a theorem does not hold for the full class. This example uses the following lemma.

Lemma 3.1 *Let $Q(p)$ denote the collection of rationals r for which there exists a transition $p \xrightarrow{a[r]} p'$ where p' can deadlock. If p is regular, then $Q(p)$ consists of a finite number of intervals.*

Proof. Omitted.

The following example shows that the merge of two regular processes is not always a regular process.

Example 3.2 *Define*

$$\begin{aligned} X &= \textstyle\int_{0<w<1} a[w] \cdot Y(w) \\ Y(v) &= \textstyle\int_{w=v} a[w] \cdot Y(w). \end{aligned}$$

The process $X \| a'[1]$ (with $a|a' = \delta$) is not regular.

Each trace of the process X is of the form $a[r] \cdot a[r] \cdot a[r] \cdot ...$ with $r \in \langle 0, 1\rangle$. If $r \in \langle 1/(n+1), 1/n\rangle$ for a natural n, then $X \| a'[1]$ will execute $a[r]$ n times, followed by $a'[1 - nr]$, then $a[(n+1)r - 1]$, and after that only $a[r]$'s. And if $r = 1/n$, then $X \| a'[1]$ will get into a deadlock after $n - 1$ times executing a. So according to Lemma 3.1, $X \| a'[1]$ is not regular. *(End example)*

3.3 Strongly regular processes

According to Example 3.2, the class of regular processes is too large for an elimination theorem. On the other hand, if no occurrences of of time dependencies are allowed, then the collection is too small. Because in this class equivalences such as

$$\int_{0<v<1} a[v] \parallel \int_{1<w<2} a'[w] \;\leftrightarrow\; \int_{0<v<1} a[v] \cdot \int_{1-v<w<2-v} a'[w]$$

cannot be expressed anymore. Godskesen and Larsen [8] provided a rigorous proof that time dependencies are essential in order to obtain an expansion theorem in a

timed setting. (Aceto and Murphy [1] proposed the notion of 'ill-timed' traces, in order to obtain an expansion theorem in the absence of time dependencies.)

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

Definition 3.3 *A time-closed process is* strongly regular *if it is bisimilar to a process* $\langle X_0|E\rangle$*, where* E *consists of equations*

$$X_I(v_1,...,v_{ar(X)}) = \sum_k \int_{\phi_k} a_k[w] \cdot X_{I_k}(b_{k1},...,b_{kar(Y_k)}) + \sum_l \int_{\psi_l} \alpha_l[w],$$

that satisfy the following requirements.

1. *The* ϕ_k *and the* ψ_l *are in the class of conditions that is defined by*

$$\phi \ ::= \ w \leq b \ \mid \ b \leq w \ \mid \ \phi \wedge \phi \ \mid \ \neg\phi,$$

   where b *is of the form* r *or* $r - v_i$*.*

2. *The bounds* b_{kj} *are of the form* r *or* $w + r$ *or* $v_i + w + r$*.*

Lemma 3.4 *Each strongly regular process can be described by a linear specification with the following more restrictive constraints.*

1. *The bounds* b *in the* ϕ_k *and the* ψ_l *are of the form* r *or* $r - v_i$ *with* $r \geq 0$*.*

2. *The bounds* b_{kj} *are of the form or* w *or* $v_i + w$*.*

Proof sketch.

1. If a bound b in the ϕ_k and the ψ_l is of the form r or $r - v_i$ with $r < 0$, then r can be replaced by 0, because actions only occur after time zero.

2. If a bound b_{kj} is of the form r or $w + r$ or $v_i + w + r$, then r can be removed by introducing a new recursion variable $\tilde{Y}_k(v_1,...,v_{ar(Y_k)})$, of which the linear equation is obtained by replacing expressions $s - v_j$ in the conditions of the equation of $Y_k(v_1,...,v_{ar(Y_v)})$ by $(s - r) - v_j$. □

We prove an elimination theorem for the algebra of strongly regular processes.

3.4 Two counter-examples

First, we present two more examples to show that this elimination result would get lost if the definition of strong regularity were less restrictive. Example 3.2 already indicated that this definition cannot be loosened by allowing not only expressions r and $r - v_i$, but also variables v_i as bounds b. The following example indicates that neither one can allow variables v_i as bounds b_{kj}.

Example 3.5 *Define*

$$\begin{aligned} X &= \int_{0<w<1} a[w] \cdot Y(w) \\ Y(v) &= \int_{w=1-v} a[w] \cdot Y(v). \end{aligned}$$

The process $X \| a'[2]$ (with $a|a' = \delta$) is not regular.

Each trace of the process X is of the form $a[r] \cdot a[1-r] \cdot a[1-r] \cdot ...$ with $r \in \langle 0, 1\rangle$. If $r \in \langle (n-2)/(n-1), (n-1)/n\rangle$ for some $n \geq 2$, then $X \| a'[2]$ will first execute $n+1$ a's, then an a' and then only a's. And if $r = (n-1)/n$, then $X \| a'[2]$ will get into a deadlock after $n+1$ times executing a. So according to Lemma 3.1, $X \| a'[2]$ is not regular. *(End example)*

Finally, the following example indicates that one cannot allow expressions $r - sv_i$ as bounds b, where $s \in \mathbb{Q}$. This example is more involved than the previous ones.

Example 3.6 *Define*

$$\begin{aligned} X_1 &= \int_{0<v<1} a[v] \cdot X_2(v) & \qquad Y_1 &= a'[\tfrac{1}{2}] \cdot Y_2 \\ X_2(v) &= \int_{w=\frac{3}{2}-\frac{1}{2}v} a[w] \cdot X_2(w) & \qquad Y_2 &= a'[1] \cdot Y_2. \end{aligned}$$

The process $X_1 \| Y_1$ (with $a|a' = \delta$) is not regular.

Clearly, the nth a'-action of Y_1 is executed at absolute time $n - 1/2$. An easy calculation learns that if X_1 executes its first action at time r, then its nth action will be executed at absolute time $n - (1-r)t_n$, where

$$t_n = \sum_{i=1}^{n} (-\frac{1}{2})^{i-1}.$$

So if $(1-r)t_n = 1/2$ for some n, then $X_1 \| Y_1$ will get into a deadlock after $n-1$ times executing a. The equalities $r = 1 - 1/(2t_n)$ for $n = 1, 2, ...$ yield an infinite partition of the interval $\langle 0, 1\rangle$, so according to Lemma 3.1 $X_1 \| Y_1$ is not regular.

(End example)

3.5 Orderings on bounds

Before giving the proof of the elimination theorem for strongly regular processes, first we present some definitions and results on orderings on bounds that shall be crucial ingredients in this proof.

Assume a finite collection B of bounds. An *ordering* $\mathcal{O}$ on B is determined by a partition $B_1, ..., B_n$ of non-empty subsets of B, where $\cup_{i=1}^{n} B_i = B$, and $B_i \cap B_j = \emptyset$ if $i \neq j$. The condition $\mathcal{O}$ consists of the conjunct of expressions $b = b'$ for $b, b' \in B_i$ and $b < b'$ for $b \in B_i$ and $b' \in B_j$ with $i < j$. We say that two conditions ϕ and ϕ' are

equivalent under $\mathcal{O}$ if for each valuation σ, $\sigma(\phi \wedge \mathcal{O})$ is true if and only if $\sigma(\phi' \wedge \mathcal{O})$ is true.

Fix a bound $b_i \in B_i$ for $i = 1, ..., n$, and fix a variable w which does not occur in bounds in B. We define conditions ψ_i which express all possible positions of w with respect to the bounds b_i under the ordering $\mathcal{O}$.

ψ_1	denotes	$w < b_1$	
ψ_{2i}	denotes	$w = b_i$	for $i = 1, ..., n$
ψ_{2i+1}	denotes	$b_i < w < b_{i+1}$	for $i = 1, ..., n-1$
ψ_{2n+1}	denotes	$b_n < w$	

Note that $\mathcal{O} \wedge \psi_i$ determines an ordering on $B \cup \{w\}$.

Consider the class C of conditions defined by

$$\phi \ ::= \ b \leq w \mid w \leq b \mid \phi \wedge \phi \mid \neg\phi,$$

where $b \in B$.

Lemma 3.7 *Under $\mathcal{O}$, each condition in C is equivalent to a condition $\vee_{i \in I} \psi_i$.*

Proof sketch. Let ϕ be in C. We apply induction on the size of ϕ. If ϕ is of the form $b \leq w$ with $b \in B_i$, then ϕ is equivalent to $\psi_{2i} \vee \cdots \vee \psi_{2n+1}$ under $\mathcal{O}$. If ϕ is of the form $w \leq b$ with $b \in B_i$, then ϕ is equivalent to $\psi_1 \vee \cdots \vee \psi_{2i}$ under $\mathcal{O}$.

Next, let ϕ be of the form $\phi_0 \wedge \phi_1$, where ϕ_0 is equivalent to $\vee_{i \in I_0} \psi_i$ and ϕ_1 is equivalent to $\vee_{i \in I_1} \psi_i$ under $\mathcal{O}$. Then ϕ is equivalent to $\vee_{i \in I_0 \cap I_1} \psi_i$ under $\mathcal{O}$.

Finally, let ϕ be of the form $\neg\phi_0$, where ϕ_0 is equivalent to $\vee_{i \in I} \psi_i$ under $\mathcal{O}$. Then ϕ is equivalent to $\vee_{i \in \{1,...,2n+1\} \setminus I} \psi_i$ under $\mathcal{O}$. □

Hence, for each $\phi \in C$ we can calculate a condition $u(\phi \wedge \mathcal{O})$ which describes the ultimate delay of $\int_{\phi \wedge \mathcal{O}} \alpha[w]$ as follows.

Definition 3.8 *Let ϕ be equivalent to $\vee_{i \in I} \psi_i$ under $\mathcal{O}$.*

- *If $I = \emptyset$, then put $u(\phi \wedge \mathcal{O})$ is ff.*
- *If I has maximum $2i-1$ or $2i$ for some $i = 1, ..., n$, then put $u(\phi \wedge \mathcal{O})$ is $w < b_i$.*
- *If I has maximum $2n+1$, then put $u(\phi \wedge \mathcal{O})$ is tt.*

For B a finite set of bounds, and $r \in \mathbb{Q}$ and $N \in \mathbb{N}$, let $\mathcal{B}_{r,N}(B)$ denote the finite set of bounds

$$\{kr, \ kr - b \mid k = 0, ..., N, \ b \in B\}.$$

Lemma 3.9 *For each ordering $\mathcal{O}$ of $\mathcal{B} = \mathcal{B}_{r,N}(v_1, ..., v_n) \cup \{w\}$, there is an ordering $\tilde{\mathcal{O}}$ of $\tilde{\mathcal{B}} = \mathcal{B}_{r,N}(w, v_1 + w, ..., v_n + w)$ such that if $\sigma(\mathcal{O})$ is true for a valuation σ, then $\sigma(\tilde{\mathcal{O}})$ is true.*

Proof sketch. Assume an ordering $\mathcal{O}$ on $\mathcal{B}$. We construct the desired ordering $\tilde{\mathcal{O}}$ on $\tilde{\mathcal{B}}$ by rewriting each possible relation in $\tilde{\mathcal{B}}$ to a relation in $\mathcal{B}$.

- The $\tilde{\mathcal{O}}$-order of $kr-(v_i+w)$ and $lr-(v_j+w)$ is the $\mathcal{O}$-order of $kr-v_i$ and $lr-v_j$.
- The $\tilde{\mathcal{O}}$-order of $kr-(v_i+w)$ and $lr-w$ is the $\mathcal{O}$-order of $kr-v_i$ and lr.
- If $k \geq l$, then the $\tilde{\mathcal{O}}$-order of $kr-(v_i+w)$ and lr is the $\mathcal{O}$-order of $(k-l)r-v_i$ and w. Otherwise, if $k<l$, then $kr-(v_i+w) <_{\tilde{\mathcal{O}}} lr$.
- The $\tilde{\mathcal{O}}$-order of $kr-w$ and $lr-w$ is the $\mathcal{O}$-order of kr and lr.
- If $k \geq l$, then the $\tilde{\mathcal{O}}$-order of $kr-w$ and lr is the $\mathcal{O}$-order of $(k-l)r$ and w. Otherwise, if $k<l$, then $kr-w <_{\tilde{\mathcal{O}}} lr$.
- The $\tilde{\mathcal{O}}$-order of kr and lr is the $\mathcal{O}$-order of kr and lr.

It follows immediately from the construction of $\tilde{\mathcal{O}}$ that if $\sigma(\mathcal{O})$ is true, then $\sigma(\tilde{\mathcal{O}})$ is true. □

3.6 The main theorem

Theorem 3.10 *For each pair of strongly regular processes p_0 and p_1, there exists a strongly regular process q such that $p_0 \| p_1 \leftrightarrows q$.*

Proof sketch. According to Lemma 3.4, p_0 and p_1 are bisimilar to processes $\langle X_0|E_0\rangle$ and $\langle X_1|E_1\rangle$, where the equations in E_0 and E_1 are of the form

$$X_I(v_{I1},...,v_{In(I)}) = \sum_{k\in K_I} \int_{\phi_k} a_k[w] \cdot X_{I_k}(b_{k1},...,b_{kn(I_k)}) + \sum_{l\in L_I} \int_{\phi'_l} \alpha_l[w],$$

such that

- the ϕ_k and the ϕ'_l are constructed from $w \leq b$ and $b \leq w$ and $\wedge$ and $\neg$, where b is of the form r or $r-v_{Ii}$ with $r \geq 0$,
- the bounds b_{kj} are of the form w or $v_{Ii}+w$.

We construct a suitable linear specification which describes the behaviour of $p_0 \| p_1$.

First, we introduce for each I a fresh recursion variable $Y_I(v, v_{I1},...,v_{In(I)})$. Its linear equation is obtained from the equation of $X_I(v_{I1},...,v_{In(I)})$ by replacing conditions in ϕ_k and ϕ'_l of the form $w \leq r$ or $r \leq w$ by $w \leq r-v$ or $r-v \leq w$ respectively. Let $\tilde{\phi}_k$ and $\tilde{\phi}'_l$ denote the resulting conditions, and put

$$Y_I(v, v_{I1},...,v_{In(I)}) = \sum_{k\in K_I} \int_{\tilde{\phi}_k} a_k[w] \cdot X_{I_k}(b_{k1},...,b_{kn(I_k)}) + \sum_{l\in L_I} \int_{\tilde{\phi}'_l} \alpha_l[w].$$

For $v \geq 0$, $Y_I(v, v_{I1}, ..., v_{In(I)})$ yields the behaviour of $(-v)X_I(v_{I1} - v, ..., v_{In(I)} - v)$.

Next, we introduce a fresh recursion variable $Z_{IJ}(v, v_{I1}, ..., v_{In(I)}, v_{J1}, ..., v_{Jn(J)})$ for each I and J, and construct its linear equation such that it describes the behaviour of $Y_I(v, v_{I1}, ..., v_{In(I)}) \| X_J(v_{J1}, ..., v_{Jn(J)})$. Only, it turns out that this behaviour cannot be described by a single linear equation, so we introduce a collection of variables $Z_{IJ}^{\mathcal{O}}$, where $\mathcal{O}$ ranges over the orderings on a collection of bounds $\mathcal{B}$.

This collection $\mathcal{B}$ is defined as follows. Ensure, by means of α-conversion, that v and the v_{Ii} and the v_{Ji} are distinct variables. Let R denote the finite collection of rationals that occur in E_0 or in E_1. Define $N = \max\{r/r_0 \mid r \in R\}$, where r_0 is the greatest rational such that r/r_0 is a natural number for each $r \in R$. Put

$$\mathcal{B} = \mathcal{B}_{r_0,N}(v, v_{I1}, ..., v_{In(I)}, v_{J1}, ..., v_{Jn(J)}).$$

Fix an ordering $\mathcal{O}$ on $\mathcal{B}$. The bounds b in the ϕ_k, ϕ'_l, $\tilde{\phi}_k$, $\tilde{\phi}'_l$ are of the form kr_0 or $kr_0 - v$ or $kr_0 - v_{Ii}$ or $kr_0 - v_{Ji}$ with $k = 0, ..., N$, so they are all in $\mathcal{B}$. Hence, according to Definition 3.8, we can define conditions u_I and $\tilde{u}_I$ which yield the ultimate delays of $X_I(v_{I1}, ..., v_{In(I)})$ and $Y_I(v, v_{I1}, ..., v_{In(I)})$ under $\mathcal{O}$ respectively, namely

$$\begin{aligned} u_I &= u((\vee_{k \in K_I} \phi_k \vee \vee_{l \in L_I} \phi'_l) \wedge \mathcal{O}), \\ \tilde{u}_I &= u((\vee_{k \in K_I} \tilde{\phi}_k \vee \vee_{l \in L_I} \tilde{\phi}'_l) \wedge \mathcal{O}). \end{aligned}$$

We construct the linear equation of $Z_{IJ}^{\mathcal{O}}$. The behaviour of $Y_I \| X_J$ originates from

(a) executing an initial action from Y_I,

(b) executing an initial action from X_J,

(c) executing a communication of initial actions from these two processes.

We describe these behaviours separately, under the ordering $\mathcal{O}$, which is determined by a partition $B_1, ..., B_n$ of $\mathcal{B}$. Fix $b_i \in B_i$ for $i = 1, ..., n$, and let $\psi_1, ..., \psi_{2n+1}$ denote the conditions $w < b_1$, $w = b_1$, $b_1 < w < b_n, ...,$ $b_n < w$ respectively.

(a) Recall that a process $p \| q$ can only execute an initial action from p before the ultimate delay of q. So initial actions from $Y_I(v, v_{I1}, ..., v_{In(I)})$ under $\mathcal{O}$ can only be executed under the condition u_J. After executing such an initial action $a_k[w]$ for some $k \in K_I$ under the condition $\phi_k \wedge u_J$, the resulting state is

$$X_{I_k}(b_{k1}, ..., b_{kn(I_k)}) \| (-w)X_J(v_{J1}, ..., v_{Jn(J)}).$$

This process is described by $Z_{JI_k}^{\tilde{\mathcal{O}}}(w, v_{J1} + w, ..., v_{Jn(J)} + w, b_{k1}, ..., b_{kn(I_k)})$ for some ordering $\tilde{\mathcal{O}}$ on $\tilde{\mathcal{B}} = \mathcal{B}_{r_0,N}(w, v_{J1} + w, ..., v_{Jn(J)} + w, b_{k1}, ..., b_{kn(I_k)})$. This ordering $\tilde{\mathcal{O}}$ can be determined as follows. The ordering $\mathcal{O}$ on $\mathcal{B}$ together with a condition ψ_i for $i = 1, ..., 2n + 1$ determine an ordering $\mathcal{O}_i$ on $\mathcal{B} \cup \{w\}$. Since $b_{k1}, ..., b_{kn(I_k)} \in$

$\{w, v_{I1}+w, ..., v_{In(I)}+w\}$, Lemma 3.9 says that there is an ordering $\tilde{\mathcal{O}}_i$ on $\tilde{\mathcal{B}}$ such that if $\sigma(\mathcal{O}_i)$ is true for a valuation σ, then $\sigma(\tilde{\mathcal{O}}_i)$ is true. Hence, we add the following subterms to the equation of $Z^{\mathcal{O}}_{IJ}$.

$$\sum_{i=1}^{2n+1} \int_{\tilde{\phi}_k \wedge u_J \wedge \psi_i} a_k[w] \cdot Z^{\tilde{\mathcal{O}}_i}_{JI_k}(w, v_{J1}+w, ..., v_{Jn(J)}+w, b_{k1}, ..., b_{kn(I_k)}).$$

After executing an initial $\alpha_l[w]$ for some $l \in L_I$ under the condition $\tilde{\phi}'_l \wedge u_J$, the resulting state is $(-w)X_J(v_{J1}, ..., v_{Jn(J)})$, which is described by $Y_J(w, v_{J1}+w, ..., v_{Jn(J)}+w)$. So we add the following subterm to the equation of $Z^{\mathcal{O}}_{IJ}$.

$$\int_{\tilde{\phi}'_l \wedge u_J} \alpha_l[w] \cdot Y_J(w, v_{J1}+w, ..., v_{Jn(J)}+w).$$

(b) Next, we construct the subterm that originates from executing an initial action of $X_J(v_{J1}, ..., v_{Jn(J)})$. Such initial actions can only be executed under the condition $\tilde{u}_I$. After executing an initial action $a_k[w]$ for some $k \in K_J$ under the condition $\phi_k \wedge \tilde{u}_I$, the resulting state is

$$(-w)Y_I(v, v_{I1}, ..., v_{In(I)}) \| X_{J_k}(b_{k1}, ..., b_{kn(J_k)}),$$

which is described by $Z^{\tilde{\mathcal{O}}}_{IJ_k}(v+w, v_{I1}+w, ..., v_{In(I)}+w, b_{k1}, ..., b_{kn(J_k)})$. Again, the ordering $\mathcal{O}$ on $\mathcal{B}$ together with a condition ψ_i for $i = 1, ..., 2n+1$ determine an ordering $\tilde{\mathcal{O}}_i$ on $\mathcal{B}_{r_0,N}(v+w, v_{I1}+w, ..., v_{In(I)}+w, b_{k1}, ..., b_{kn(J_k)})$. So we add the following subterms to the equation of $Z^{\mathcal{O}}_{IJ}$.

$$\sum_{i=1}^{2n+1} \int_{\phi_k \wedge \tilde{u}_I \wedge \psi_i} a_k[w] \cdot Z^{\tilde{\mathcal{O}}_i}_{IJ_k}(v+w, v_{I1}+w, ..., v_{In(I)}+w, b_{k1}, ..., b_{kn(J_k)}).$$

After executing an initial $\alpha_l[w]$ for some $l \in L_J$ under the condition $\phi'_l \wedge \tilde{u}_I$, the resulting state is $(-w)Y_I(v, v_{J1}, ..., v_{Jn(J)})$, which is described by $Y_I(v+w, v_{J1}+w, ..., v_{Jn(J)}+w)$. So we add the following subterm to the equation of $Z^{\mathcal{O}}_{IJ}$.

$$\int_{\phi'_l \wedge \tilde{u}_I} \alpha_l[w] \cdot Y_I(v+w, v_{J1}+w, ..., v_{Jn(J)}+w).$$

(c) Finally, we construct the subterms that originate from executing the communication of initial actions of Y_I and X_J.

Let $k \in K_I$ and $k' \in K_J$. After executing the action $(a_k|a_{k'})[w]$ under the condition $\tilde{\phi}_k \wedge \phi_{k'}$, the resulting state is $X_{I_k}(b_{k1}, ..., b_{kn(I_k)}) \| X_{J_{k'}}(b_{k'1}, ...b_{k'n(J_{k'})})$. Again, the ordering $\mathcal{O}$ on $\mathcal{B}$ together with a condition ψ_i for $i = 1, ..., 2n+1$ determine an ordering $\tilde{\mathcal{O}}_i$. So we add the following subterms to the equation of $Z^{\mathcal{O}}_{IJ}$.

$$\sum_{i=1}^{2n+1} \int_{\tilde{\phi}_k \wedge \phi_{k'} \wedge \psi_i} (a_k|a_{k'})[w] \cdot Z^{\tilde{\mathcal{O}}_i}_{I_k J_{k'}}(0, b_{k1}, ..., b_{kn(I_k)}, b_{k'1}, ..., b_{k'n(J_{k'})}).$$

For $k \in K_I$ and $l \in L_J$ we add the subterm

$$\int_{\tilde{\phi}_k \wedge \phi'_l} (a_k|\alpha_l)[w] \cdot Y_{I_k}(0, b_{k1}, ..., b_{kn(I_k)}).$$

For $l \in L_I$ and $k \in K_J$ we add the subterm

$$\int_{\tilde{\phi}'_l \wedge \phi_k} (\alpha_l|a_k)[w] \cdot Y_{J_k}(0, b_{k1}, ..., b_{kn(J_k)}).$$

Finally, for $l \in L_I$ and $l' \in L_J$, we add the subterm

$$\int_{\tilde{\phi}'_l \wedge \phi'_{l'}} (\alpha_l|\alpha_{l'})[w].$$

This finishes the construction of the linear equation of $Z^{\mathcal{O}}_{IJ}$.

Let E denote the recursive specification that consists of the linear equations of the Y_I and the $Z^{\mathcal{O}}_{IJ}$. Note that E satisfies the constraints in Definition 3.3 for strongly regular processes. Consider the strongly regular process $\langle Z^{\mathcal{O}}_{01}(0)|E\rangle$, where the ordering $\mathcal{O}$ on $\mathcal{B}_{r_{0,N}}(v)$ is true under the valuation that maps v to 0. The construction ensures that this process is bisimilar to $\langle X_0|E_0\rangle \| \langle X_1|E_1\rangle$, which finishes the proof of the elimination theorem. □

3.7 An example

We present an example of a merge of two simple strongly regular processes, which itself can only be described by a much more complicated linear specification.

Example 3.11 *Define*

$$X = \int_{0<v<1} a[v] \cdot X.$$

Let $p = X \| a'[k]$ *(with* $a|a' = \delta$*), where* $k \in \mathbb{N}$.

The behaviour of p can be described by the following linear specification.

$$\begin{aligned}
X_0 &= \textstyle\int_{0<v<1} a[v] \cdot X_1(v) \\
X_i(v) &= \textstyle\int_{0<w\leq i-v} a[w] \cdot X_i(v+w) + \int_{i-v<w<1} a[w] \cdot X_{i+1}(v+w) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1, ..., k-1 \\
X_k(v) &= \textstyle\int_{0<w<k-v} a[w] \cdot X_k(v+w) + \int_{w=k-v} a'[w] \cdot Y(w) \\
Y(v) &= \textstyle\int_{0<w<1-v} a[w] \cdot X \\
X &= \textstyle\int_{0<v<1} a[v] \cdot X.
\end{aligned}$$

The idea behind this specification is quite easy. Process p will execute X until it reaches (absolute) time k, when it executes a', after which it continues with X. The process p has the possibility of executing a' or at time k if it has executed an a after time $k-1$. So if this is the case, then the linear specification must take into account the execution of a' at k. Similarly, p can 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 of the X_i for $i = 1, ..., k-1$ 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 a'. *(End example)*

4 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 $s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} ...$ such that $(s_i, a_i, s_{i+1}) \in E$ for $i = 0, 1, 2, ...$ Furthermore, the trace may have to satisfy certain fairness requirements, e.g. that it reaches a specific state an infinite number of times.

In the extension of automata with time from Alur and Dill [2], the elements of E are supplied with time constraints on 'clock variables'. Such constraints are disjunctions of expressions of the form $x \square r$, with x a clock variable and $r \in \mathbb{Q}_{>0}$ and $\square \in \{<, >, \leq, \geq\}$. Moreover, there is a construct $x := 0$, which denotes that while executing a 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 clocks satisfy their constraints. Furthermore, accepted traces must satisfy the required fairness constraints. Finally, Zeno behaviour is excluded from timed automata, i.e. traces are only accepted if they progress beyond any moment in time.

The algebra of strongly regular processes can be linked to the class of timed automata. The fairness restrictions and the non-Zeno requirement are obstacles for the translation between timed automata and strongly regular processes, since ACPrI 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.

A strongly regular process can be translated to the setting of timed automata as follows. A strongly regular processes executes an action $a[w]$ under restrictions of the form $w \square r$ or $w \square (r - v_i)$, with $r > 0$ and $\square \in \{<, >, \leq, \geq\}$. These last inequalities can be rewritten to the form $(v_i + w) \square r$. The v_i and w can be regarded as clocks, where w has been set back to zero by $w := 0$ in the previous transition. The state that results after the execution of $a[w]$ is a recursive expression of the form $X(w, v_1 + w, ..., v_k + w)$. The $v_i + w$ and w store the actual times of the clocks at the moment of the transition $a[w]$.

By means of the converse translation, it is easy to see that 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 4.1 *Consider a timed automaton with states s_0, s_1, with start state s_0, and clock variables x, y, which is defined by the following two transitions:*

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

This timed automaton executes alternately a and a'. Define

$$\begin{aligned} X &= \int_{0<w\leq 1} a[w] \cdot Y_1 + \int_{1\leq w<2} a[w] \cdot Y_2(w) \\ Y_1 &= \int_{0<w<2} a'[w] \cdot X \\ Y_2(v) &= \int_{0<w<3-v} a'[w] \cdot X. \end{aligned}$$

The process X describes the behaviour of the automaton. *(End example)*

References

[1] L. Aceto and D. Murphy. On the ill-timed but well-caused. In E. Best, editor, *Proceedings 4th Conference on Concurrency Theory (CONCUR'93)*, Hildesheim, volume 715 of *LNCS*, pages 97–111. Springer, 1993.

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

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

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra.* Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, 1990.

[5] B. Bloom and F.W. Vaandrager. SOS rule formats for parameterized and state-bearing processes. Unpublished manuscript, 1995.

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

[7] W.J. Fokkink and A.S. Klusener. An effective axiomatization for real time ACP. *Information and Computation*, 122(2):286–299, 1995.

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

[9] J.A. Hillebrand. The ABP and the CABP – a comparison of performances in real time process algebra. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Proceedings 1st Workshop on the Algebra of Communicating Processes (ACP'94),* Utrecht, Workshops in Computing, pages 124–147. Springer, 1994.

[10] A.S. Klusener. Completeness in real time process algebra. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings 2nd Conference on Concurrency Theory (CONCUR'91),* Amsterdam, volume 527 of *LNCS*, pages 376–392. Springer, 1991.

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

[12] A. Stoughton. Substitution revisited. *Theoretical Computer Science*, 59:317–325, 1988.