Centrum voor Wiskunde en Informatica
**REPORT***RAPPORT*

# Undecidability and Completeness Results for Process Algebras with Alternative Quantification over Data

J.F. Groote, S.P. Luttik

Software Engineering (SEN)

**SEN-R9806 June 30, 1998**

Report SEN-R9806
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

# Undecidability and Completeness Results for Process Algebras with Alternative Quantification over Data

J.F. Groote[1,2] JanFriso.Groote@cwi.nl

S.P. Luttik[1,3] Bas.Luttik@cwi.nl

[1] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

[2] Computing Science Department, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

[3] Programming Research Group, University of Amsterdam,
Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands

ABSTRACT

We define a class of process algebras with a generalised operation $\sum$ that allows explicit treatment of *alternative quantification* over data, and investigate the specific subclass formed by the algebras of finite processes modulo strong bisimulation.

We prove that, in such algebras, equality between process terms is definable by means of a first-order data formula, and that, if the data is computable and has a built-in equality predicate, any $\Pi^0_4$ data formula is definable as an equation between ground process terms. From these results we work to the conclusion that equality in strong bisimulation algebras with a computable data part is $\Pi^0_4$-hard.

We also investigate a restricted version of alternative quantification: the input prefix mechanism of Parrow and Sangiorgi (1995) and Hennessy and Lin (1996). We show that this restriction yields a less expressive formalism if the data is computable and has a built-in equality predicate: equality between input prefix processes coincides with the universal fragment of first-order logic for the data. That is, the input prefix mechanism gives rise to strong bisimulation algebras for which equality is complete in $\Pi^0_1$.

Finally, we give a ground complete axiomatisation for those strong bisimulation algebras of which the data part has built-in equality and Skolem functions.

*1991 Mathematics Subject Classification:* 03D80; 03G25; 08A70; 68Q65; 68Q70
*1991 Computing Reviews Classification System:* D.1.3; F.1.1; F.4.1
*Keywords and Phrases:* Generalised Algebra, Process Algebra, Algebraic Specification, Alternative Quantification, Input Prefixing, Expressiveness of Strong Bisimulation.
*Note:* Research supported by the Netherlands Organization for Scientific Research (NWO) under contract SION 612-33-008. Work carried out under project SEN 2.1 Process Specification and Analysis.

## 1. Introduction

A *process algebra* is an algebra that contains a universe P and binary operations $+$ and $\cdot$ on P such that $\langle \mathrm{P}, + \rangle$ is a semilattice, $\langle \mathrm{P}, \cdot \rangle$ is a semigroup, and $\cdot$ distributes from the right over $+$. The operation $+$ is used to model *alternative composition*, a nondeterministic choice between its arguments, and $\cdot$ stands for *sequential composition*, performing its second argument upon completion of its first. A many-sorted process algebra that does not contain operations that take the sort of processes into some other sort, is often called a *process algebra with data*; the *data* being the reduct one obtains by removing the universe of processes and all operations involving this universe.

The algebras that we consider in this paper consist of a many-sorted algebra of data that contains a boolean algebra and a process algebra that has, besides the usual process algebraic operations $+$ and $\cdot$:

1. a constant $\boldsymbol{\delta}$ modelling unsuccesful termination (it is a unit for $\langle \mathrm{P}, + \rangle$ and additionally satisfies $\boldsymbol{\delta} \cdot \mathbf{p} = \boldsymbol{\delta}$ for all $\mathbf{p} \in \mathrm{P}$);

2. operations $\mathtt{a} : \mathrm{D}_1 \times \cdots \times \mathrm{D}_n \to \mathrm{A}$ that map an $n$-tuple ($n = 0$ is allowed) of data elements injectively onto a special set $\mathrm{A} \subseteq \mathrm{P}$, the elements of which are called *atomic actions*;

3. a ternary operation $\mathbf{p} \lhd \mathbf{b} \rhd \mathbf{q}$ that receives a boolean $\mathbf{b}$ and processes $\mathbf{p}$ and $\mathbf{q}$ and yields the process that performs a choice between $\mathbf{p}$ and $\mathbf{q}$, but one that depends on the evaluation of $\mathbf{b}$; the construction should be read "$\mathbf{p}$ if $\mathbf{b}$, else $\mathbf{q}$".

4. a generalised (or: infinitary) operation $\sum : (\mathsf{Pow}(\mathrm{P}) - \emptyset) \to \mathrm{P}$ (where $\mathsf{Pow}(\mathrm{P})$ refers to the set of all subsets of P), defined by $\mathrm{P}' \mapsto \sum \mathrm{P}'$.

Our major concern in this paper is with this latter operation: *alternative quantification* over data. We have given it a universal algebraic definition in the same way as Rasiowa and Sikorski (1963) generalise the binary joins ($\vee$) and meets ($\wedge$) of boolean algebras to obtain existential ($\exists$) and universal ($\forall$) quantification. In our signatures alternative quantification is present as an operation symbol that binds a data variable. That is, we shall write $\sum_{x:s} p$ ($p$ some process term and $x$ a variable of data sort $s$) to denote the (possibly infinite) expression

$$p[x := d_0] + p[x := d_1] + p[x := d_2] + \ldots,$$

where $d_0, d_1, d_2, \ldots$ is an enumeration of the universe associated with data sort $s$.

Alternative quantification is a very useful feature, since it can be used to model the notion of *input*: if $\mathtt{r}(n)$ refers to the atomic action of reading the natural number $n$, then $\sum_{x:\mathbb{N}} \mathtt{r}(x)$ can be used to refer to the process that can read any natural number; it is a choice between reading 0, 1, 2, ... .

We shall investigate alternative quantification in a popular class of process algebras: the algebras of transition systems modulo strong bisimulation. We shall prove that if one has alternative quantification as a primitive, then strong bisimulation on the set of *finite* processes (transition systems that are denoted by a process term) with computable data is in the Arithmetical Hierarchy and $\Pi^0_4$-hard: for any two finite processes there exists a first-order data formula that holds iff they are bisimilar, and conversely, for every $\Pi^0_4$ data formula one can find a pair of finite processes that are strongly bisimilar iff the formula holds for the data.

We shall also consider *input prefixing*. Roughly, a process term is an input prefix term if alternative quantification only occurs in it in the form $\sum_{\bar{x}:\bar{s}} \mathtt{a}(\bar{x}) \cdot p \lhd b \rhd \delta$, i.e., quantification is always over the data that defines the first atomic action following the operation $\sum$. It turns out that if one restricts alternative quantification to input prefixing, then the expressivity of bisimulation coincides with the *universal* fragment of first-order logic ($\Pi^0_1$): for any universal first-order data formula one can find a pair of input prefix processes that are bisimilar if, and only if, the formula holds, and for any two input prefix process terms there exists a universal first-order formula that holds if, and only if, they are bisimilar.

Consequently, there exists no general complete axiomatisation of strong bisimulation, not even if we restrict alternative quantification to input prefixing. We shall describe a general method to obtain a ground complete axiomatisation of the strong bisimulation algebra from an $\omega$-complete axiomatisation of the data, provided that it has a built-in equality predicate and Skolem functions.

This paper is organised as follows. We adapt the notion of algebra with generalised operations of Rasiowa and Sikorski (1963) to a many-sorted setting and we extend equational logic with a rule for generalised operations (§2). This enables us to provide the class of process algebras with alternative quantification over data with a precise universal algebraic definition (§3). In §4 we single out specific members of this class: the strong bisimulation algebras. In §5 we explore the connection between equality in these algebras and first-order logic. In §6 we give a ground complete axiomatisation for those strong bisimulation algebras of which the data part has built-in equality and Skolem functions.

**Related Work** Milner (1989) used the generalised operation $\sum$ to provide the input prefix mechanism with a semantical basis. In the process specification language $\mu$CRL (Groote and Ponse, 1994b) the operation appears also in the syntax. A $\mu$CRL-specification consists of a set of recursion equations that defines a set of processes over an equationally specified abstract datatype. The syntax and semantics of $\mu$CRL are based on the *Algebra of Communicating Processes* of Bergstra and Klop (1984), but an abstract algebraic characterisation of alternative quantification such as we carry out in §2–4, was still missing.

Ponse (1996) investigates the expressivity of a fragment of $\mu$CRL that includes conditionals and recursion, but excludes alternative quantification. It is shown in his paper that in this case strong bisimulation for computable data is complete in $\Pi^0_1$.

Groote and Ponse (1994a) devise a proof system for $\mu$CRL, based on natural deduction, and provide a set of nonlogical axioms that are all valid in the strong bisimulation algebras (they do not have a completeness result). To tie in with standard practice in universal algebra, we use a proof system that is based on equational logic to which we add a replacement rule for binders; the nonequational axiom scheme $p \approx q \to \sum_{x:s} p \approx \sum_{x:s} q$ of Groote and Ponse (1994a) figures in this paper as an instance of this rule. Also, by our choice of the proof system, we do not incorporate the *law of the excluded middle* ($b \approx \top \vee b \approx \bot$) and the axiom $\neg(\top \approx \bot)$, as Groote and Ponse do. Instead, we have added a number of axioms that are derivable from these two and express the relations between boolean operations and process algebraic operations (these are derivable in the system of Groote and Ponse (1994a)).

Hennessy and Lin (1995) consider process algebras that have input prefixing (or rather a restricted version of it) as a primitive, instead of arbitrary alternative quantification. They define an *early* and a *late* variant of strong bisimulation on symbolic representations of transition systems (the "early" variant corresponds with our definition). Since strong bisimulation is complete in $\Pi^0_1$ in these circumstances, one cannot attain a ground complete axiomatisation of it without extra assumptions. Hennessy and Lin (1996) present a proof system for both variants that is complete if it is combined with a complete proof system for the data. This presumed proof system for the data should allow the derivation of propositions of the form $b \rhd s = t$, with the interpretation that the identity $s = t$ holds for every valuation that satisfies the boolean expression $b$. Evidently, such a proof system only exists if the data algebra has an $\omega$-complete algebraic specification. We end §6 with an example that shows that their requirements are not sufficient to guarantee that strong bisimulation on the entire set of finite processes (i.e., with alternative quantification as an operation) is semidecidable. Hence, a stronger requirement (e.g., built-in Skolem functions) is necessary.

Parrow and Sangiorgi (1995) provide a complete set of algebraic laws for a class of name-passing calculi (such as the $\pi$-calculus) that also contain the notion of input prefixing. Their language has no facility for an explicit treatment of operations on names and equality on names is syntactical. Therefore, if it is decidable whether two names are equal, then strong bisimulation is decidable as well.

**Acknowledgements** We thank Alban Ponse and Piet Rodenburg for their careful reading of draft versions of this paper and Jan Bergstra for many discussions on the subjects of this paper.

## 2. Generalised Algebra

We adapt the concept of algebras with generalised (or: infinitary) operations of Rasiowa and Sikorski (1963) to a many-sorted setting.

**Generalised Signatures** We assume a class $\mathcal{N}$ of *names* that are not related in any set-theoretical sense.

Let $\mathcal{S}$ be a subset of $\mathcal{N}$. A *function declaration over* $\mathcal{S}$ is a triple $F = \langle f, \bar{s}, s' \rangle$ such that $f \in \mathcal{N}$, $\bar{s}$ a finite sequence of elements of $\mathcal{S}$, and $s' \in \mathcal{S}$; we write $F\colon \bar{s} \to s'$. A *binder declaration over* $\mathcal{S}$ is a pair

$B = \langle b, s\rangle$ such that $b \in \mathcal{N}$ and $s \in \mathcal{S}$; we write $B\colon s$. A *(generalised) signature* is a set $\Sigma = \mathcal{S} \cup \mathcal{F} \cup \mathcal{B}$ with $\mathcal{F}$ a set of function declarations over $\mathcal{S}$ and $\mathcal{B}$ a set of binder declarations over $\mathcal{S}$. The elements of $\mathcal{S}$ are called *sort symbols* and $\Sigma$ is called $\mathcal{S}$-sorted.

If $\Sigma$ and $\Sigma'$ are signatures and $\Sigma \subseteq \Sigma'$, then we call $\Sigma$ a *subsignature* of $\Sigma'$. A signature is *first-order* if it does not contain binder declarations. We shall also speak of the *first-order part* of $\Sigma$, meaning the largest subsignature of $\Sigma$ that does not contain any binder declarations.

**Generalised Algebras** Let $\Sigma$ be an $\mathcal{S}$-sorted generalised signature.

A $\Sigma$-*algebra* is an assignment $\mathfrak{A}$ of a nonempty set $\mathfrak{A}(s)$ to every $s \in \mathcal{S}$, an $n$-ary operation (or: *first-order* operation)

$$\mathfrak{A}(F) : \mathfrak{A}(s_1) \times \cdots \times \mathfrak{A}(s_n) \to \mathfrak{A}(s')$$

to every $(F\colon s_1 \cdots s_n \to s') \in \Sigma$ and a *generalised* operation

$$\mathfrak{A}(B) : \mathsf{dom}_B \to \mathfrak{A}(s), \text{ with } \mathsf{dom}_B \subseteq \mathsf{Pow}(\mathfrak{A}(s))$$

to every $(B\colon s) \in \Sigma$. The sets $D \in \mathsf{dom}_B$ are called the *admissible sets* for $B$ in $\mathfrak{A}$. If $\mathsf{dom}_B$ consists of all nonempty subsets of $\mathfrak{A}(s)$ for all $(B\colon s) \in \Sigma$, then $\mathfrak{A}$ is said to be a *complete generalised algebra*.

If $\Sigma'$ is a subsignature of $\Sigma$, then the restriction $\mathfrak{A}{\restriction_{\Sigma'}}$ of $\mathfrak{A}$ to $\Sigma'$ is called the $\Sigma'$-*reduct* of $\mathfrak{A}$. If $\Sigma'$ happens to be the first-order part of $\Sigma$, then we also call $\mathfrak{A}{\restriction_{\Sigma'}}$ the *first-order reduct of* $\mathfrak{A}$.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\Sigma$-algebras. We call $\mathfrak{A}$ a *subalgebra* of $\mathfrak{B}$ (notation: $\mathfrak{A} \subseteq \mathfrak{B}$) if $\mathfrak{A}(s) \subseteq \mathfrak{B}(s)$ for all sorts $s \in \Sigma$, $\mathfrak{A}(F) = \mathfrak{B}(F){\restriction_{(\mathfrak{A}(s_1)\times\cdots\times\mathfrak{A}(s_n))}}$ (the restriction of the operation $\mathfrak{B}(F)$ to $\mathfrak{A}(s_1) \times \cdots \times \mathfrak{A}(s_n)$) for every function declaration $(F\colon s_1 \cdots s_n \to s') \in \Sigma$, and $\mathfrak{A}(B) = \mathfrak{B}(B){\restriction_{(\mathsf{dom}_B \cap \mathsf{Pow}(\mathfrak{A}(s)))}}$ for every binder declaration $(B\colon s) \in \Sigma$.

A *homomorphism* from $\mathfrak{A}$ to $\mathfrak{B}$ is a family $h = \langle h_s \mid s \in \mathcal{S}\rangle$ of functions $h_s : \mathfrak{A}(s) \to \mathfrak{B}(s)$ that constitute a homomorphism from the first-order reduct of $\mathfrak{A}$ to the first-order reduct of $\mathfrak{B}$ and preserve the generalised operations, i.e., if $(B\colon s) \in \Sigma$ and $S \subseteq \mathsf{Pow}(\mathfrak{A}(s))$ is an admissible set for $B$ in $\mathfrak{A}$, then $h_s(S)$ is admissible for $B$ in $\mathfrak{B}$ and

$$h_s(\mathfrak{A}(B)(S)) = \mathfrak{B}(B)(h_s(S)).$$

A congruence $\theta = \langle \theta_s \mid s \in \mathcal{S}\rangle$ on the first-order reduct of $\mathfrak{A}$ is a *congruence* on $\mathfrak{A}$ if for all $(B\colon s) \in \Sigma$ and sets $S, S'$ admissible for $B$ in $\mathfrak{A}$

$$S/\theta_s = S'/\theta_s \text{ implies } \mathfrak{A}(B)(S)/\theta_s = \mathfrak{A}(B)(S')/\theta_s.$$

**Term formation** We fix disjoint infinite sets $\mathcal{V}$ and $\Xi$, both disjoint from $\mathcal{N}$.

Let $\Sigma$ be an $\mathcal{S}$-sorted generalised signature. The set of *bound variables* for $\Sigma$ consists of all pairs $\xi = \langle b, s\rangle$ where $b$ is an element of $\Xi$ and $s \in \mathcal{S}$; $\xi$ is called a bound variable of sort $s$. Let $X = \langle X_s \subseteq \mathcal{V} \mid s \in \mathcal{S}\rangle$ be some $\mathcal{S}$-indexed family of pairwise disjoint sets; we call the elements of the $X_s$ *free variables (of sort $s$)*. We construct the $\mathcal{S}$-sorted family $T_\Sigma[X] = \langle T_\Sigma[X]^s \mid s \in \mathcal{S}\rangle$ of $\Sigma$-*terms* in $X$ as the least family that satisfies

1. $X_s \subseteq T_\Sigma[X]^s$, and
2. $F(t_1, \ldots, t_n) \in T_\Sigma[X]^s$ if $(F\colon s_1 \cdots s_n \to s') \in \Sigma$ and $t_i \in T_\Sigma[X]^{s_i}$ for $1 \le i \le n$; and
3. $B_\xi(t[x := \xi]) \in T_\Sigma[X]^s$ if $(B\colon s) \in \Sigma$, $x \in X_{s'}$, $t \in T_\Sigma[X]^s$, and $t[x := \xi]$ is obtained from $t$ by replacing all occurrences of the free variable $x$ by the bound variable $\xi$ of sort $s'$.

The elements of $T_\Sigma[X]^s$ are called terms of sort $s$. We denote by $FV(t)$ the set of free variables that occur in $t$ and we write $t = t(x_1, \ldots, x_n)$ if $FV(t) \subseteq \{x_1, \ldots, x_n\}$. A term $t$ is $s$-*ground* if it does not contain variables of sort $s$, and *ground* if it is $s$-ground for every sort $s \in \Sigma$ (that is, if $FV(t) = \emptyset$).

Terms are considered modulo $\alpha$-conversion: $t$ and $t'$ are $\alpha$-congruent (notation: $t \sim t'$) iff $t'$ can be obtained from $t$ by a series of renamings of bound variables.

Let $\Sigma'$ denote the first-order part of $\Sigma$. The family of $\Sigma$-terms constitutes a $\Sigma'$-algebra and $\sim$ is a congruence on this algebra. We refer by $\mathfrak{T}_\Sigma[X]$ (or by $\mathfrak{T}$ if $\Sigma$ and $X$ are clear) to the $\Sigma'$-algebra of $\Sigma$-terms modulo $\alpha$-conversion, defined as follows

i. $\mathfrak{T}(s) = T_\Sigma[X]^s/\!\sim$; and

ii. if $(F\colon s_1 \cdots s_n \to s') \in \Sigma$ and $t_i/\!\sim \in T_\Sigma[X]^{s_i}/\!\sim$ for $1 \leq i \leq n$, then

$$\mathfrak{T}(F)(t_1/\!\sim, \ldots, t_n/\!\sim) = F(t_1, \ldots, t_n)/\!\sim.$$

Note that $\sim$ also respects binder declarations, i.e., if $t \sim t'$, then also $B_\xi(t[x := \xi]) = B_\xi(t'[x := \xi])$. Unless otherwise stated, if we write a term in the sequel, we shall mean its $\sim$-congruence class in $\mathfrak{T}$.

**Remark 2.1** Rasiowa and Sikorski (1963) describe two approaches that both lead to generalised term algebras that are free in the class of complete $\Sigma$-algebras. We choose not to define them here to avoid notational complications that are not essential for the rest of this paper.

Let $\mathfrak{A}$ be a $\Sigma$-algebra, and $X = \langle X_s \mid s \in \mathcal{S}\rangle$ a family of variables. A family $\alpha = \langle \alpha_s \mid s \in \mathcal{S}\rangle$ of mappings $\alpha_s : X_s \to \mathfrak{A}(s)$ is a *valuation* of $X$ in $\mathfrak{A}$. Each valuation extends to a homomorphism $\bar{\alpha} : \mathfrak{T} \to \mathfrak{A}\restriction_{\Sigma'}$ (where $\mathfrak{A}\restriction_{\Sigma'}$ is the first-order reduct of $\mathfrak{A}$) in the following way

i. $\bar{\alpha}(x) = \alpha_s(x)$ for $x \in X_s$;

ii. $\bar{\alpha}(\mathfrak{T}(F)(t_1, \ldots, t_n)) = \mathfrak{A}(F)(\bar{\alpha}(t_1), \ldots, \bar{\alpha}(t_n))$ for every $F\colon s_1 \cdots s_n \to s'$ and $t_i \in \mathfrak{T}(s_i)$ $(1 \leq i \leq n)$; and

iii. $\bar{\alpha}(B_\xi(t[x := \xi])) = \mathfrak{A}(B)(\{\bar{\alpha}_{[x:=a]}(t) \mid a \in \mathfrak{A}(s')\})$ for every $B\colon s$, $t \in \mathfrak{T}(s)$ and $x \in X_{s'}$, where $\alpha_{[x:=a]}(x) = a$ and $\alpha_{[x:=a]}(y) = \alpha(y)$ if $y \neq x$.

A substitution $\sigma$ is a valuation of $X$ in $\mathfrak{T}_\Sigma[X]$. The extension of $\sigma$ to a homomorphism $\bar{\sigma}$ is similar to that for valuations, except that we should replace the third clause by

iii′. $\bar{\sigma}(B_\xi(t[x := \xi])) = B_\xi(\bar{\sigma}_{[x:=x]}(t)[x := \xi])$.

We shall often write $t^\sigma$ for $\bar{\sigma}(t)$.

A $\Sigma$-*equation* in $X$ is an expression of the form $t \approx t'$, with $t, t' \in \mathfrak{T}_\Sigma[X](s)$ for some $s \in \mathcal{S}$. If $\alpha$ is a valuation of $X$ in $\mathfrak{A}$ such that $\bar{\alpha}(t) = \bar{\alpha}(t')$, then we say that $\alpha$ *satisfies* $t \approx t'$ *in* $\mathfrak{A}$ (notation: $\mathfrak{A}, \alpha \models t \approx t'$); if $\mathfrak{A}, \alpha \models t \approx t'$ for all valuations $\alpha$ of $X$ in $\mathfrak{A}$ we say that $\mathfrak{A}$ *satisfies* $t \approx t'$ (notation: $\mathfrak{A} \models t \approx t'$). We denote by $EqTh(\mathfrak{A})$ the set of all $\Sigma$-equations that $\mathfrak{A}$ satisfies.

**Generalised Equational Logic** Let $E$ be a set of $\Sigma$-equations in $X$. We write $E \vdash e$, for $e$ some $\Sigma$-equation in $X$, if $e$ is derivable from $E$ by means of the rules for *generalised equational logic*, depicted in Table 1. The pair $\langle \Sigma, E\rangle$ determines an *equational theory*, a congruence $\bar{E} = \langle \bar{E}_s \mid s \in \mathcal{S}\rangle$ on $\mathfrak{T} = \mathfrak{T}_\Sigma[X]$, where

$$\bar{E}_s = \{\langle t, t'\rangle \mid E \vdash t \approx t' \text{ and } t, t' \in \mathfrak{T}(s)\ \}.$$

**Proposition 2.2 (Soundness)** Let $\mathfrak{A}$ be a $\Sigma$-algebra and let $E$ be a set of $\Sigma$-equations in $X$. If $\mathfrak{A} \models E$ and $E \vdash t_1 \approx t_2$, then $\mathfrak{A} \models t_1 \approx t_2$.

Proof. Induction on the length of the derivation of $t_1 \approx t_2$. For conventional equational logic this is well-known (see e.g. Burris and Sankappanavar (1981)), so we only deal with the rule for binder declarations. If $\mathfrak{A} \models t_1 \approx t_2$, then in particular $\bar{\alpha}_{[x:=a]}(t_1) = \bar{\alpha}_{[x:=a]}(t_2)$ for any valuation $\alpha$ and for any $a \in \mathfrak{A}(s')$, so $\{\bar{\alpha}_{[x:=a]}(t_1) \mid a \in \mathfrak{A}(s')\} = \{\bar{\alpha}_{[x:=a]}(t_2) \mid a \in \mathfrak{A}(s')\}$. Hence

$$t \approx t' \text{ for every } t \approx t' \in E$$

$$\frac{s \approx t}{\bar{\sigma}(s) \approx \bar{\sigma}(t)} \text{ for every substitution } \sigma$$

$$\frac{t_1 \approx t_1' \cdots t_n \approx t_n'}{F(t_1, \ldots, t_n) \approx F(t_1', \ldots, t_n')} \text{ for } (F\colon s_1 \cdots s_n \to s') \in \Sigma$$

$$\frac{t \approx t'}{B_\xi(t[x := \xi]) \approx B_\xi(t'[x := \xi])} \text{ for } (B\colon s) \in \Sigma,\ x \text{ a variable of sort } s' \in \Sigma$$

$$t \approx t \qquad \frac{t \approx t'}{t' \approx t} \qquad \frac{t_1 \approx t_2 \quad t_2 \approx t_3}{t_1 \approx t_3}$$

Table 1: Generalized Equational Logic.

$$\begin{aligned} \bar{\alpha}(B_\xi(t_1[x := \xi])) &= \mathfrak{A}(B)(\{\bar{\alpha}_{[x:=a]}(t_1) \mid a \in \mathfrak{A}(s')\}) \\ &= \mathfrak{A}(B)(\{\bar{\alpha}_{[x:=a]}(t_2) \mid a \in \mathfrak{A}(s')\}) \\ &= \bar{\alpha}(B_\xi(t_2[x := \xi])) \end{aligned}$$

□

# 3. Process Algebras with Data

In this section we will define a class of process algebras with data, which is called $p\mathsf{CRL}$. We shall introduce $p\mathsf{CRL}$-signatures, $p\mathsf{CRL}$-theories and $p\mathsf{CRL}$-algebras, and state some elementary results, including a representation result (Lemma 3.3) for terms over a $p\mathsf{CRL}$-signature.

## 3.1 Signature

In the sequel, we reserve names $\mathbb{b}$ and $\mathbb{p}$ as sort symbols referring to *booleans* and *processes*, respectively. We shall denote by $\Sigma_{\mathbb{b}}$ the signature consisting of $\mathbb{b}$, nullary function declarations $\top\colon \lambda \to \mathbb{b}$ and $\bot\colon \lambda \to \mathbb{b}$ ($\lambda$ refers to the empty sequence), a unary function declaration $\neg\colon \mathbb{b} \to \mathbb{b}$, and binary function declarations $(\_ \wedge \_)\colon \mathbb{b}\mathbb{b} \to \mathbb{b}$ and $(\_ \vee \_)\colon \mathbb{b}\mathbb{b} \to \mathbb{b}$. If $\Sigma_{\mathbb{b}} \subseteq \Sigma$, then we say that $\Sigma$ *contains booleans*; elements of $\mathfrak{T}_\Sigma[X](\mathbb{b})$ are called *boolean terms*. As usual we abbreviate $\neg b_1 \vee b_2$ by $b_1 \to b_2$, and $b_1 \to b_2 \wedge b_2 \to b_1$ by $b_1 \leftrightarrow b_2$.

A *first-order* signature $\Sigma$ is called a *data signature* if it contains booleans, but $\mathbb{p} \notin \Sigma$; terms over a data signature we refer to as *data terms*.

Let $\Delta$ be an $\mathcal{S}$-sorted data signature. An *action declaration* for $\Delta$ is a function declaration $\mathsf{a}\colon s_1 \cdots s_n \to \mathbb{p}$, where $s_i \in \mathcal{S}$ for $1 \leq i \leq n$; we usually suppress the sort name $\mathbb{p}$ and write $\mathsf{a}\colon s_1 \cdots s_n$.

Let $\Delta$ be an $\mathcal{S}$-sorted data signature and $\mathcal{A}$ a set of action declarations for $\Delta$. The $p\mathsf{CRL}$-*signature* for $\Delta$ and $\mathcal{A}$ is the smallest signature that contains $\Delta$ and $\mathcal{A}$, and

1. a nullary function declaration $\delta\colon \lambda \to \mathbb{p}$;
2. binary function declarations $(\_ + \_)\colon \mathbb{p}\mathbb{p} \to \mathbb{p}$ and $(\_ \cdot \_)\colon \mathbb{p}\mathbb{p} \to \mathbb{p}$;
3. a ternary function declaration $(\_ \lhd \_ \rhd \_)\colon \mathbb{p}\mathbb{b}\mathbb{p} \to \mathbb{p}$; and
4. a binder declaration $\sum\colon \mathbb{p}$.

Let $\Sigma$ be a $p$CRL-signature and let $X$ be a family of free variables for $\Sigma$. The elements of $\mathfrak{T}_\Sigma[X](\mathbb{p})$ we shall refer to as *process terms*. A process term $a$ of the form $\mathtt{a}(\bar{t})$ with $\mathtt{a}{:}\bar{s}$ is called an *action term*; we shall refer to the leading action declaration of an action term by the name of the action term in typewriter font, e.g., $\mathtt{a}$ refers to the leading action declaration of $a$.

We adopt some notational conventions regarding boolean terms and process terms. Function declarations are usually written in mixfix notation and brackets are omitted where possible. We give the following precedence to the operators:

$$(\_ + \_) < \textstyle\sum < (\_ \lhd \_ \rhd \_) < (\_ \cdot \_).$$

Terms of the form $p \cdot q$ are usually written $pq$. If $p = p(x_1, \ldots, x_n)$ is a process term, $x_i \in X_{s_i}$ (for $1 \leq i \leq n$) and $\xi$ is a bound variable of sort $s_j$, then we shall use $\sum_{x_j:s_j} p$ as an abbreviation for the term $\sum_\xi p[x_j := \xi]$. Note that by $\alpha$-conversion the specific choice of the bound variable $\xi$ is immaterial and that the free variable $x_j$ does not occur in $\sum_{x_j:s_j} p$. Thus, provided that $X_s$ is infinite for all sorts $s \in \Delta$, we may choose $x_j$ different from all other free variables that occur in some mathematical context; e.g., if $p = \sum_{x_1:s_1} p'$ and $q = \sum_{x_2:s_2} q'$, then we may assume without loss of generality that $x_1 \neq x_2$, $x_1 \notin FV(q)$ and $x_2 \notin FV(p)$. This assumption will often be implicitly present. We use $\sum_{x_1 \cdots x_n:s_1 \cdots s_n} p$ as an abbreviation of $\sum_{x_1:s_n} \cdots \sum_{x_n:s_n} p$.

## 3.2 The Theory $p$CRL

| | | | |
|---|---|---|---|
| A1 | $x + y$ | $\approx$ | $y + x$ |
| A2 | $x + (y + z)$ | $\approx$ | $(x + y) + z$ |
| A3 | $x + x$ | $\approx$ | $x$ |
| A4 | $(x + y)z$ | $\approx$ | $xz + yz$ |
| A5 | $x(yz)$ | $\approx$ | $(xy)z$ |
| A6 | $x + \delta$ | $\approx$ | $x$ |
| A7 | $\delta x$ | $\approx$ | $\delta$ |

| | | | |
|---|---|---|---|
| COND1 | $x \lhd \top \rhd y$ | $\approx$ | $x$ |
| COND2 | $x \lhd \bot \rhd y$ | $\approx$ | $y$ |
| COND3 | $x \lhd b \rhd y$ | $\approx$ | $x \lhd b \rhd \delta + y \lhd \neg b \rhd \delta$ |
| COND4 | $(x \lhd b_1 \rhd \delta) \lhd b_2 \rhd \delta$ | $\approx$ | $x \lhd b_1 \wedge b_2 \rhd \delta$ |
| COND5 | $(x \lhd b_1 \rhd \delta) + (x \lhd b_2 \rhd \delta)$ | $\approx$ | $x \lhd b_1 \vee b_2 \rhd \delta$ |
| COND6 | $(x \lhd b \rhd \delta)y$ | $\approx$ | $xy \lhd b \rhd \delta$ |
| COND7 | $(x + y) \lhd b \rhd \delta$ | $\approx$ | $x \lhd b \rhd \delta + y \lhd b \rhd \delta$ |

| | | | |
|---|---|---|---|
| SUM1 | $\sum_{x:s} y$ | $\approx$ | $y$ |
| SUM3 | $\sum_\xi p[x := \xi]$ | $\approx$ | $\sum_\xi p[x := \xi] + p$ |
| SUM4 | $\sum_{x:s}(p + q)$ | $\approx$ | $\sum_{x:s} p + \sum_{x:s} q$ |
| SUM5 | $(\sum_{x:s} p)y$ | $\approx$ | $\sum_{x:s} py$ |
| SUM12 | $(\sum_{x:s} p) \lhd b \rhd \delta$ | $\approx$ | $\sum_{x:s} p \lhd b \rhd \delta$ |

Table 2: The axioms of $\Pi_\Sigma$ for a given $p$CRL-signature $\Sigma$; the SUM-axioms are schemes ranging over $p, q \in \mathfrak{T}(\mathbb{p})$; the symbols $x, y, b, b_1, b_2$ are variables. (We have kept our numbering consistent with Groote and Ponse (1994a); in their setting SUM2 defines $\alpha$-conversion.)

Let $\Sigma$ be a $p$CRL-signature, then the $p$CRL-*theory* $\Pi_\Sigma$ for $\Sigma$ consists of the axioms depicted in Table 2 and the axioms for boolean algebras depicted in Table 3. A $\Sigma$-algebra $\mathfrak{A}$ is called a $p$CRL-*algebra* if $\mathfrak{A} \models \Pi_\Sigma$.

| | | |
|---|---|---|
| B1 | $x \wedge y \approx y \wedge x$ | $x \vee y \approx y \vee x$ |
| B2 | $x \wedge (y \wedge z) \approx (x \wedge y) \wedge z$ | $x \vee (y \vee z) \approx (x \vee y) \vee z$ |
| B3 | $x \wedge x \approx x$ | $x \vee x \approx x$ |
| B4 | $x \wedge (x \vee y) \approx x$ | $x \vee (x \wedge y) \approx x$ |
| B5 | $(x \wedge y) \vee (x \wedge z) \approx x \wedge (y \vee z)$ | $(x \vee y) \wedge (x \vee z) \approx x \vee (y \wedge z)$ |
| B6 | $x \wedge \bot \approx \bot$ | $x \vee \top \approx \top$ |
| B7 | $x \wedge \neg x \approx \bot$ | $x \vee \neg x \approx \top$ |

Table 3: The axioms for boolean algebras.

**Proposition 3.1** The following equalities are derivable from $\Pi_\Sigma$

i. $x \lhd b \rhd x \approx x$;

ii. $\sum_{\bar{x}_1:\bar{s}_1} \sum_{\bar{x}_2:\bar{s}_2} p \approx \sum_{\bar{x}_2:\bar{s}_2} \sum_{\bar{x}_1:\bar{s}_1} p$.

Proof.

i. We derive

$$\begin{aligned} x \lhd b \rhd x &\approx x \lhd b \rhd \delta + x \lhd \neg b \rhd \delta && \text{(Cond3)} \\ &\approx x \lhd b \vee \neg b \rhd \delta && \text{(Cond5)} \\ &\approx x. && \text{(B7, Cond1)} \end{aligned}$$

ii. By Sum3 we have $(*)$ $\sum_{\bar{x}_1\bar{x}_2:\bar{s}_1\bar{s}_2} p \approx (\sum_{\bar{x}_1\bar{x}_2:\bar{s}_1\bar{s}_2} p) + p$, whence

$$\begin{aligned} \textstyle\sum_{\bar{x}_1\bar{x}_2:\bar{s}_1\bar{s}_2} p &\approx \textstyle\sum_{\bar{x}_2\bar{x}_1:\bar{s}_2\bar{s}_1} \sum_{\bar{x}_1\bar{x}_2:\bar{s}_1\bar{s}_2} p && \text{(Sum1)} \\ &\approx \textstyle\sum_{\bar{x}_1\bar{x}_2:\bar{s}_1\bar{s}_2} p + \sum_{\bar{x}_2\bar{x}_1:\bar{s}_2\bar{s}_1} p && (*, \text{Sum4, Sum1}) \\ &\approx \textstyle\sum_{\bar{x}_2\bar{x}_1:\bar{s}_2\bar{s}_1} p + \sum_{\bar{x}_1\bar{x}_2:\bar{s}_1\bar{s}_2} p && \text{(A1)} \\ &\approx \textstyle\sum_{\bar{x}_2\bar{x}_1:\bar{s}_2\bar{s}_1} p. \end{aligned}$$

□

**Definition 3.2** Let $\Sigma$ be a $p$CRL-signature and $X$ a family of free variables for $\Sigma$. Let $A$ be the set of action terms over $\Sigma$ in $X$, $B$ the set of boolean terms over $\Sigma$ in $X$. We inductively define the set $\mathcal{B}_\Sigma[X]$ of $\Sigma$-*basic terms* in $X$ as follows:

1. $\delta \in \mathcal{B}_\Sigma[X]$, and $\sum_{\bar{x}:\bar{s}} a \lhd b \rhd \delta \in \mathcal{B}_\Sigma[X]$, for $a \in A$ and $b \in B$;
2. if $p \in \mathcal{B}_\Sigma[X]$, then $\sum_{\bar{x}:\bar{s}} a \cdot p \lhd b \rhd \delta \in \mathcal{B}_\Sigma[X]$, for $a \in A$ and $b \in B$; and
3. if $p, q \in \mathcal{B}_\Sigma[X]$, then $p + q \in \mathcal{B}_\Sigma[X]$.

**Lemma 3.3 (Basic Term Lemma)** For every $\mathbb{p}$-ground process term $p$ there exists a basic term $q \in \mathcal{B}_\Sigma[X]$ such that $\Pi_\Sigma \vdash p \approx q$.

Proof. Straightforward by induction on the number of symbols in $p$. □

For $\mathcal{B}_\Sigma[X]$ a set of basic terms, let $\mathcal{B}^\epsilon_\Sigma[X] = \mathcal{B}_\Sigma[X] \cup \{\epsilon\}$, where $\epsilon$ acts as a unit for $\cdot$. That is, we assume $p \cdot \epsilon = \epsilon \cdot p = p$. We call a basic term *simple* if it is of the form

$$\textstyle\sum_{\bar{x}:\bar{s}} a \cdot p \lhd b \rhd \delta,$$

with $p \in \mathcal{B}^\epsilon_\Sigma[X]$. Note that if we define $\delta = \sum_{i \in \emptyset} p_i$, then every basic term is of the form

$$\sum_{i \in I} p_i, \qquad p_i \text{ a simple basic term for all } i \in I \ (I \text{ finite}).$$

$$\mathbf{a} \xrightarrow{\mathbf{a}} \epsilon \qquad \text{for all } \mathbf{a} \in \mathrm{A}$$

$$\frac{\mathbf{p} \xrightarrow{\mathbf{a}} \mathbf{p}'}{\mathbf{p}\cdot\mathbf{q} \xrightarrow{\mathbf{a}} \mathbf{p}'\cdot\mathbf{q}} \qquad \frac{\mathbf{p} \xrightarrow{\mathbf{a}} \epsilon}{\mathbf{p}\cdot\mathbf{q} \xrightarrow{\mathbf{a}} \mathbf{q}} \qquad \mathbf{a} \in \mathrm{A} \text{ and } \mathbf{p}, \mathbf{p}', \mathbf{q} \in \mathrm{P}$$

$$\frac{\mathbf{p} \xrightarrow{\mathbf{a}} \mathbf{q}}{\sum \mathrm{P}' \xrightarrow{\mathbf{a}} \mathbf{q}} \qquad \mathbf{a} \in \mathrm{A},\ \mathbf{p} \in \mathrm{P}' \subseteq \mathrm{P} \text{ and } \mathbf{q} \in (\mathrm{P} \cup \{\epsilon\})$$

Table 4: The transition system specification for $\mathfrak{P}$.

# 4. The Strong Bisimulation Algebra

For the rest of this paper we fix a data signature $\Delta$ and a $\Delta$-algebra $\mathfrak{D}$ such that $\mathfrak{D}\restriction_{\Delta_{\mathsf{b}}}$ is the *two-element* boolean algebra; we denote $\mathfrak{D}(\top)$ by $\mathsf{T}$ and $\mathfrak{D}(\bot)$ by $\bot$. Also, we fix a set of action declarations $\mathcal{A}$ for $\Delta$ and denote by $\Sigma$ the $p\mathsf{CRL}$-signature for $\Delta$ and $\mathcal{A}$.

We shall construct an algebra of *processes* $\mathfrak{P}$ and define a congruence $\leftrightarrows$ on this algebra. The quotient $\mathfrak{P}/\leftrightarrows$ will turn out to be a $p\mathsf{CRL}$-algebra that we shall call the *strong bisimulation algebra* for $\mathfrak{D}$ and $\mathcal{A}$.

**Processes** First, define a set A of *atomic actions* by

$$\mathrm{A} = \{\mathsf{a}\langle d_1, \ldots, d_n\rangle \mid \mathsf{a}{:}s_1 \cdots s_n \in \mathcal{A} \text{ and } d_i \in \mathfrak{D}(s_i) \text{ for } 1 \le i \le n\ \}.$$

The set $\mathrm{P} = \bigcup_{n\in\omega} \mathrm{P}^n$ of *processes* is obtained by the following recursion

$$\begin{array}{lcl} \mathrm{P}^0 & = & \mathrm{A} \cup \{\boldsymbol{\delta}\} \\ \mathrm{P}^{n+1} & = & \mathrm{P}^n \cup \{\mathbf{p}\cdot\mathbf{q},\ \sum \mathrm{P}' \mid \mathbf{p}, \mathbf{q} \in \mathrm{P}^n,\ \emptyset \neq \mathrm{P}' \subseteq \mathrm{P}^n\}; \end{array}$$

we shall write $\mathbf{p} + \mathbf{q}$ for $\sum\{\mathbf{p}, \mathbf{q}\}$.

Let $\mathfrak{P}$ be the $\Sigma$-algebra such that $\mathfrak{P}\restriction_\Delta = \mathfrak{D}$, $\mathfrak{P}(\mathbb{p}) = \mathrm{P}$ and operations defined by

$$\begin{array}{lcl} \mathfrak{P}(\mathsf{a})(d_1, \ldots, d_n) & = & \mathsf{a}\langle d_1, \ldots, d_n\rangle \quad \text{for each } \mathsf{a}{:}s_1 \cdots s_n \in \Sigma; \\ \mathfrak{P}(\delta) & = & \boldsymbol{\delta}; \\ \mathfrak{P}(+)(\mathbf{p}, \mathbf{q}) & = & \mathbf{p} + \mathbf{q}; \\ \mathfrak{P}(\cdot)(\mathbf{p}, \mathbf{q}) & = & \mathbf{p}\cdot\mathbf{q}; \\ \mathfrak{P}(\_ \lhd \_ \rhd \_)(\mathbf{p}, \mathbf{b}, \mathbf{q}) & = & \begin{cases} \mathbf{p} & \text{if } \mathbf{b} = \mathsf{T}; \\ \mathbf{q} & \text{if } \mathbf{b} = \bot; \text{ and} \end{cases} \\ \mathfrak{P}(\sum)(\mathrm{P}') & = & \sum \mathrm{P}' \qquad \text{for each } \emptyset \neq \mathrm{P}' \subseteq \mathrm{P}. \end{array}$$

**Operational Semantics and Bisimulation** The rules in Table 4 define a *transition relation* $\longrightarrow \subseteq \mathrm{P} \times \mathrm{A} \times (\mathrm{P} \cup \{\epsilon\})$ on $\mathfrak{P}$.

For convenience of notation we define $\mathrm{P}^\epsilon = \mathrm{P} \cup \{\epsilon\}$ and, for any binary relation $\mathcal{R}$ on P, $\mathcal{R}^\epsilon = \mathcal{R} \cup \{\langle\epsilon, \epsilon\rangle\}$. Also, we shall tacitly assume that $\mathbf{p}$ ranges over P and $\mathbf{p}'$ ranges over $\mathrm{P}^\epsilon$ in $\mathbf{p} \xrightarrow{\mathbf{a}} \mathbf{p}'$.

**Definition 4.1 (Bisimulation)** A binary relation $\mathcal{R} \subseteq \mathrm{P} \times \mathrm{P}$ is called a *bisimulation relation* if it is symmetric and $\mathcal{R}^\epsilon$ satisfies

$$\text{if } \langle \mathbf{p}, \mathbf{q}\rangle \in \mathcal{R} \text{ and } \mathbf{p} \xrightarrow{\mathbf{a}} \mathbf{p}', \text{ then there exists } \mathbf{q}' \in \mathrm{P}^\epsilon \text{ such that } \mathbf{q} \xrightarrow{\mathbf{a}} \mathbf{q}' \text{ and } \langle \mathbf{p}', \mathbf{q}'\rangle \in \mathcal{R}^\epsilon.$$

**Lemma 4.2** If $\langle \mathcal{R}_i \mid i \in I \rangle$ is a family of bisimulation relations, then $\mathcal{R} = \bigcup\{\mathcal{R}_i \mid i \in I\}$ is a bisimulation relation as well.

Proof. If $\langle \mathbf{p}, \mathbf{q} \rangle \in \mathcal{R}$ and $\mathbf{p} \xrightarrow{\mathbf{a}} \mathbf{p}'$, then $\langle \mathbf{p}, \mathbf{q} \rangle \in \mathcal{R}_i$ for some $i \in I$, so there exists $\mathbf{q}'$ such that $\mathbf{q} \xrightarrow{\mathbf{a}} \mathbf{q}'$ and $\langle \mathbf{p}', \mathbf{q}' \rangle \in \mathcal{R}_i^\epsilon$, whence $\langle \mathbf{p}', \mathbf{q}' \rangle \in \mathcal{R}^\epsilon$. □

If $\mathbf{p}, \mathbf{q} \in \mathrm{P}$ and there is a bisimulation relation that contains the pair $\langle \mathbf{p}, \mathbf{q} \rangle$, then $\mathbf{p}$ and $\mathbf{q}$ are called *bisimilar* (notation: $\mathbf{p} \leftrightarrows \mathbf{q}$).

**Lemma 4.3** The family $\leftrightarrows = \langle \theta_s \mid s \in \mathcal{S} \rangle$ with $\theta_{\mathbb{p}} = \leftrightarrows$ and $\theta_s = \mathsf{id}(\mathfrak{P}(s))$ for $s \neq \mathbb{p}$ is a congruence on $\mathfrak{P}$.

Proof. It is trivial that, for $s \neq \mathbb{p}$, the relation $\theta_s$ is an equivalence relation on $\mathfrak{P}(s)$ and that it has the substitution property for all operations.
The identity relation on P is a bisimulation relation that contains $\langle \mathbf{p}, \mathbf{p} \rangle$; any bisimulation relation that contains $\langle \mathbf{p}, \mathbf{q} \rangle$ also contains $\langle \mathbf{q}, \mathbf{p} \rangle$ by definition; and if $\mathcal{R}_1$ and $\mathcal{R}_2$ are bisimulation relations containing $\langle \mathbf{p}, \mathbf{q} \rangle$ and $\langle \mathbf{q}, \mathbf{r} \rangle$, respectively, then $\mathcal{R}_1 \circ \mathcal{R}_2 \cup \mathcal{R}_2 \circ \mathcal{R}_1$ is a bisimulation relation that contains $\langle \mathbf{p}, \mathbf{r} \rangle$. So $\leftrightarrows$ is an equivalence relation on P.
Clearly, $\leftrightarrows$ also has the substitution property for the function declarations in $\Delta$ and $\mathcal{A}$. Since all $p\mathsf{CRL}$-operations are defined from $\cdot$ and $\sum$, it remains to show that these operations have the substitution property.
If $\langle \mathbf{p}_1, \mathbf{q}_1 \rangle \in \mathcal{R}_1$ and $\langle \mathbf{p}_2, \mathbf{q}_2 \rangle \in \mathcal{R}_2$, with $\mathcal{R}_1, \mathcal{R}_2$ bisimulation relations, then the relation

$$\mathcal{R} = \{\langle \mathbf{p} \cdot \mathbf{q}_1, \mathbf{q} \cdot \mathbf{q}_2 \rangle, \langle \mathbf{p} \cdot \mathbf{q}_1, \mathbf{q} \cdot \mathbf{q}_2 \rangle \mid \langle \mathbf{p}, \mathbf{q} \rangle \in \mathcal{R}_1\} \cup \mathcal{R}_2$$

is a bisimulation relation; so if $\mathbf{p}_1 \leftrightarrows \mathbf{p}_2$ and $\mathbf{q}_1 \leftrightarrows \mathbf{q}_2$, then $\mathbf{p}_1 \cdot \mathbf{q}_1 \leftrightarrows \mathbf{p}_2 \cdot \mathbf{q}_2$.
If $\emptyset \neq \mathrm{P}', \mathrm{P}'' \subseteq \mathrm{P}$ and $\mathrm{P}'/\leftrightarrows = \mathrm{P}''/\leftrightarrows$, then for all $\mathbf{p} \in \mathrm{P}'$ there exists $\mathbf{q} \in \mathrm{P}''$ and a bisimulation $\mathcal{R}_{\mathbf{p}}$ that contains $\langle \mathbf{p}, \mathbf{q} \rangle$ and for all $\mathbf{q} \in \mathrm{P}''$ there exists $\mathbf{p} \in \mathrm{P}'$ and a bisimulation $\mathcal{R}_{\mathbf{q}}$ that contains $\langle \mathbf{q}, \mathbf{p} \rangle$. Then, using Lemma 4.2, it is easy to verify that also the relation

$$\{\langle \textstyle\sum \mathrm{P}', \sum \mathrm{P}'' \rangle, \langle \sum \mathrm{P}'', \sum \mathrm{P}' \rangle\} \cup \bigcup\{\mathcal{R}_{\mathbf{p}} \mid \mathbf{p} \in \mathrm{P}'\} \cup \bigcup\{\mathcal{R}_{\mathbf{q}} \mid \mathbf{q} \in \mathrm{P}''\}$$

is a bisimulation relation.
We conclude that $\leftrightarrows$ is a congruence on $\mathfrak{P}$. □

**Lemma 4.4** The algebra $\mathfrak{P}/\leftrightarrows$ satisfies $\Pi_\Sigma$.

Proof. As in the proof of the previous lemma we should find suitable bisimulation relations that witness the validity of the axioms in $\Pi_\Sigma$. We omit the straightforward, but lengthy verification. □

Thus, the algebra $\mathfrak{P}/\leftrightarrows$ is a $p\mathsf{CRL}$-algebra and we call it the *strong bisimulation algebra* for $\mathfrak{D}$ and $\mathcal{A}$.
Let $\mathbf{p}, \mathbf{q} \in \mathrm{P}$. We say that $\mathbf{q}$ *simulates* $\mathbf{p}$ if there exists a bisimulation relation $\mathcal{R}$ that satisfies: if $\mathbf{p} \xrightarrow{\mathbf{a}} \mathbf{p}'$, then there exists $\mathbf{q}' \in \mathrm{P}^\epsilon$ such that $\mathbf{q} \xrightarrow{\mathbf{a}} \mathbf{q}'$ and $\langle \mathbf{p}', \mathbf{q}' \rangle \in \mathcal{R}^\epsilon$; we call $\mathcal{R}$ a *simulation* of $\mathbf{p}$ by $\mathbf{q}$.

**Proposition 4.5** If $\mathbf{p}, \mathbf{q} \in \mathrm{P}$, then $\mathbf{q}$ simulates $\mathbf{p}$ iff $\mathbf{q} \leftrightarrows \mathbf{q} + \mathbf{p}$.

Proof. If $\mathcal{R}$ is a simulation of $\mathbf{p}$ by $\mathbf{q}$, then

$$\mathcal{R} \cup \{\langle \mathbf{q}, \mathbf{q} + \mathbf{p} \rangle,\ \langle \mathbf{q} + \mathbf{p}, \mathbf{q} \rangle\} \cup \mathsf{id}(\mathrm{P})$$

is a bisimulation witnessing $\mathbf{q} \leftrightarrows \mathbf{q} + \mathbf{p}$.
For the converse note that any bisimulation relation containing $\langle \mathbf{q}, \mathbf{q} + \mathbf{p} \rangle$ is a simulation of $\mathbf{p}$ by $\mathbf{q}$ as well. □

# 5. Strong bisimulation and first-order logic

Let $X$ be a family of variables for $\Delta$. The set $F_\Delta[X]$ of *first-order $\Delta$-formulae* is the smallest set that contains the $\Delta$-equations and is closed under the connectives $\neg$, $\wedge$, and $\forall$ of first-order logic ($\vee$, $\rightarrow$ and $\exists$ are defined from these in the usual way). If $\varphi \in F_\Delta[X]$, then we shall denote by $FV(\varphi)$ the set of free variables occurring in $\varphi$, where it is understood that $\forall$ and $\exists$ are binders (i.e. the variable $x \in X_s$ does not occur free in $(\forall x{:}s)\varphi$ and $(\exists x{:}s\varphi)$. A formula $\varphi$ is called *closed* if $FV(\varphi) = \emptyset$.

For $\Delta$-algebras $\mathfrak{D}$ and valuations $\alpha$ of $X$ in $\mathfrak{D}$, satisfaction of first-order $\Delta$-formulae is recursively defined from satisfaction of $\Delta$-equations in the following manner:

1. $\mathfrak{D}, \alpha \models \neg\varphi$ if, and only if, not $\mathfrak{D}, \alpha \models \varphi$,
2. $\mathfrak{D}, \alpha \models \varphi \wedge \psi$ if, and only if, both $\mathfrak{D}, \alpha \models \varphi$ and $\mathfrak{D}, \alpha \models \psi$,
3. $\mathfrak{D}, \alpha \models (\forall x{:}s)\varphi$ if, and only if, $\mathfrak{D}, \alpha_{[x:=a]} \models \varphi$ for every $a \in \mathfrak{D}(s)$.

If $\mathfrak{D}, \alpha \models \varphi$, then we say that $\alpha$ *satisfies* $\varphi$ *in* $\mathfrak{D}$; if $\mathfrak{D}, \alpha \models \varphi$ for all valuations $\alpha$ of $X$ in $\mathfrak{D}$, then we say that $\mathfrak{D}$ *satisfies* $\varphi$ (we write $\mathfrak{D} \models \varphi$).

We shall now show that equations between $\mathbb{p}$-ground process terms are logically equivalent with first-order $\Delta$-formulas.

**Theorem 5.1** If $X_s$ is infinite for all sorts $s \in \Delta$, then for any $\mathbb{p}$-ground process terms $p$ and $q$ there exists a first-order $\Delta$-formula $\varphi$ such that $\mathfrak{D}, \alpha \models \varphi$ iff $\mathfrak{P}/\underline{\leftrightarrow}, \alpha \models p \approx q$, for all valuations $\alpha$ of $X$ in $\mathfrak{D}$.

Proof. By Lemmas 3.3 and 4.4 we may assume without loss of generality that $p$ and $q$ are basic terms, i.e. let $I$ and $J$ be disjoint finite sets such that

$$p = \sum_{i \in I} p_i, \qquad q = \sum_{j \in J} q_j, \qquad \text{with } p_i \text{ and } q_j \text{ simple.}$$

We shall prove the theorem by strong induction on the sum of the complexities $|p|$ and $|q|$ of $p$ and $q$ respectively, defined as the maximal nesting of $\cdot$, with $|\delta| = 0$ and $|a| = 1$ for all action terms $a$.
If $|p| + |q| = 0$, then $p = q = \delta$, so $\bar{\alpha}(p) = \bar{\alpha}(q)$ for all valuations $\alpha$ of $X$ in $\mathfrak{D}$, and we can take for $\varphi$ the formula $(\top \approx \top)$ (or any other tautology).
Suppose $|p| > 0$, $|q| = 0$ and $p_i = \sum_{\bar{x}_i:\bar{s}_i} a_i p_i' \lhd b_i \rhd \delta$.
Since $q = \delta$, $\bar{\alpha}(p) = \bar{\alpha}(q)$ if, and only if $\bar{\alpha}(p) = \boldsymbol{\delta}$, i.e. if, and only if, $\bar{\alpha}_{[\bar{x}_i:=\bar{d}_i]}(b_i) = \bot$ for all $\bar{d}_i$ and for all $i \in I$. Hence define

$$\varphi = \bigwedge_{i \in I} ((\forall \bar{x}_i{:}\bar{s}_i)\neg(b_i \approx \top))$$

By symmetry, the case where $|p| = 0$ and $|q| > 0$ follows, so only the case where $|p|, |q| > 0$ remains; let $q_j = \sum_{\bar{x}_j:\bar{s}_j} a_j q_j' \lhd b_j \rhd \delta$.
First, we consider action terms $a_i = \mathtt{a}_i(t_1^i, \ldots, t_{n_i}^i)$ and $a_j = \mathtt{a}_j(t_1^j, \ldots, t_{n_j}^j)$. Notice that, for any valuation $\alpha$, $\bar{\alpha}(a_i) = \bar{\alpha}(a_j)$ iff $\mathtt{a}_i = \mathtt{a}_j$ and $\bar{\alpha}(t_k^i) = \bar{\alpha}(t_k^j)$ for all $1 \leq k \leq n_i = n_j$. Hence, the equation $a_i \approx a_j$ is logically equivalent to a first-order $\Delta$-formula:

$$a_i \approx a_j \leftrightarrow \begin{cases} \neg(\top \approx \top) & \text{if } \mathtt{a}_i \neq \mathtt{a}_j \\ (t_1^i \approx t_1^j) \wedge \cdots \wedge (t_{n_i}^i \approx t_{n_j}^j) & \text{otherwise} \end{cases} \tag{5.1}$$

It now suffices to find formulae $\varphi_i$ such that $\mathfrak{D}, \alpha \models \varphi_i$ iff $\bar{\alpha}(q)$ simulates $\bar{\alpha}(p_i)$, for any valuation $\alpha$; for, a symmetric argument yields formulae $\varphi_j$ such that $\varphi_j$ iff $\bar{\alpha}(p)$ simulates $\bar{\alpha}(q_j)$ and we can define $\varphi = (\bigwedge_{i \in I} \varphi_i) \wedge (\bigwedge_{j \in J} \varphi_j)$.

By the induction hypothesis we find formulae $\varphi_{ij}$ such that $\mathfrak{D}, \alpha \models \varphi_{ij}$ iff $\mathfrak{P}/\leftrightarrows, \alpha \models p'_i \approx q'_j$, for any valuation $\alpha$ of $X$ in $\mathfrak{D}$ (we put $\varphi_{ij} = (\top \approx \top)$ if $p'_i, q'_j = \epsilon$ and $\varphi_{ij} = \neg(\top \approx \top)$ if one of $p'_i, q'_j$ equals $\epsilon$ but not both). We define

$$\varphi_i = (\forall \bar{x}_i{:}\bar{s}_i)((b_i \approx \top) \to \bigvee_{j \in J} (\exists \bar{x}_j{:}\bar{s}_j)((b_j \approx \top) \wedge (a_i \approx a_j) \wedge \varphi_{ij})),$$

and prove that $\mathfrak{D}, \alpha \models \varphi_i$ iff $\bar{\alpha}(q)$ simulates $\bar{\alpha}(p_i)$.

($\Rightarrow$) Suppose $\bar{\alpha}(p_i) \xrightarrow{\mathbf{a}} \mathbf{p}$. Then there exists a sequence $\bar{d}_i$ of elements of $\mathfrak{D}$ such that

$$\bar{\alpha}_{[\bar{x}_i := \bar{d}_i]}(a_i) = \mathbf{a},\ \bar{\alpha}_{[\bar{x}_i := \bar{d}_i]}(p'_i) = \mathbf{p},\ \text{and}\ \bar{\alpha}_{[\bar{x}_i := \bar{d}_i]}(b_i) = \mathsf{T}.$$

Hence there exists a $j \in J$ and a sequence $\bar{d}_j$ such that

$$\bar{\alpha}_{[\bar{x}_j := \bar{d}_j]}(b_j) = \mathsf{T},\ \bar{\alpha}_{[\bar{x}_j := \bar{d}_j]}(a_j) = \mathbf{a},\ \text{and}\ \bar{\alpha}_{[\bar{x}_j := \bar{d}_j]}(q'_j) = \mathbf{p}.$$

Thus $\bar{\alpha}(q) \xrightarrow{\mathbf{a}} \mathbf{p}$ and we conclude that $\bar{\alpha}(q)$ simulates $\bar{\alpha}(p_i)$.

($\Leftarrow$) Fix arbitrary an arbitrary sequence $\bar{d}_i$ of elements of $\mathfrak{D}$, and assume $\bar{\alpha}_{[\bar{x}_i := \bar{d}_i]}(b_i) = \mathsf{T}$. Then $\bar{\alpha}(p_i) \xrightarrow{\mathbf{a}} \mathbf{p}$, with $\mathbf{a} = \bar{\alpha}_{[\bar{x}_i := \bar{d}_i]}(a_i)$ and $\mathbf{p} = \bar{\alpha}_{[\bar{x}_i := \bar{d}_i]}(p'_i)$, so $\bar{\alpha}(q) \xrightarrow{\mathbf{a}} \mathbf{p}$. Hence, there exists a sequence $\bar{d}_j$ of elements of $\mathfrak{D}$ such that

$$\bar{\alpha}_{[\bar{x}_j := \bar{d}_j]}(b_j) = \mathsf{T},\ \bar{\alpha}_{[\bar{x}_j := \bar{d}_j]}(a_j) = \mathbf{a},\ \text{and}\ \bar{\alpha}_{[\bar{x}_j := \bar{d}_j]}(q'_j) = \mathbf{p}.$$

Hence $\mathfrak{D}, \alpha \models \varphi_i$, and the theorem follows. □

We now consider a restricted form of alternative quantification.

**Definition 5.2 (input prefix terms)** An *input prefix term* is a basic term of the form

$$\sum_{i \in I} \sum_{\bar{x}_i : \bar{s}_i} \mathtt{a}_i(\bar{x}_i) p_i \lhd b_i \rhd \delta + \sum_{j \in J} \mathtt{a}_j(\bar{t}_j) p_j \lhd b_j \rhd \delta$$

with $\mathtt{a}_i{:}\bar{s}_i$ and $\mathtt{a}_j{:}\bar{s}_j$ action declarations, $\bar{t}_j$ a sequence of terms, $\bar{x}_i$ a sequence of variables, and $p_i, p_j$ input prefix terms, for each $i \in I$, $j \in J$.

We shall say that an input prefix term is *unconditional* if $\bar{x}_i \neq \lambda$ implies $b_i = \top$ and recursively all the $p_i, p_j$ are unconditional, for all $i \in I$ and $j \in J$.

Let $\Sigma$ be the $p$CRL-signature $\mathfrak{D}$ and $\mathcal{A}$. The set of input prefix terms is not closed under the operations, but it generates (together with $X$) a subalgebra $\mathfrak{I}_\Sigma[X] \subseteq \mathfrak{T}_\Sigma[X]$ that we shall call it the *algebra of input prefix terms*. Similarly we obtain the *algebra of unconditional input prefix terms* $\mathfrak{U}_\Sigma[X] \subseteq \mathfrak{T}_\Sigma[X]$ as the subalgebra generated by the set of unconditional input prefix terms. Notice that $\mathfrak{U}_\Sigma[X]$ is also a subalgebra of $\mathfrak{I}_\Sigma[X]$.

**Lemma 5.3** For every $\mathbb{p}$-ground process term $p \in \mathfrak{I}_\Sigma[X]$ ($p \in \mathfrak{U}_\Sigma[X]$) there exists an (unconditional) input prefix term $q$ such that $\Pi_\Sigma \vdash p \approx q$.

Proof. Routine by structural induction on $p$. □

If alternative quantification is replaced by the input prefix mechanism, as is done in Parrow and Sangiorgi (1995); Hennessy and Lin (1996), then we obtain a considerably sharpened version of Theorem 5.1.

**Corollary 5.4** If $X_s$ is infinite for all $s \in \Delta$, then for any $\mathbb{p}$-ground input prefix terms $p$ and $q$ there exists a universal first-order $\Delta$-formula $\varphi$ such that $\mathfrak{D}, \alpha \models \varphi$ iff $\mathfrak{P}/\leftrightarrows, \alpha \models p \approx q$, for all valuations $\alpha$ of $X$ in $\mathfrak{D}$.

PROOF. We need to show that the existential quantifiers occurring in the $\varphi_i$ of the proof of Theorem 5.1 are redundant.

So suppose

$$p_i = \sum_{i\in I} \sum_{\bar{x}_i:\bar{s}_i} \mathtt{a}_i(\bar{t}_i)p_i' \lhd b_i \rhd \delta, \qquad \text{where } \bar{x}_i = \lambda \text{ or } \bar{x}_i = \bar{t}_i,$$

and for all $j \in J$

$$q_j = \sum_{j\in J} \sum_{\bar{x}_j:\bar{s}_j} \mathtt{a}_j(\bar{t}_j)q_j' \lhd b_j \rhd \delta, \qquad \text{where } \bar{x}_j = \lambda \text{ or } \bar{x}_j = \bar{t}_j.$$

Defining $J_i = \{j \in J \mid \mathtt{a}_i = \mathtt{a}_j\}$, we find by (5.1) from the proof of Theorem 5.1 and some standard logical reasoning that $\varphi_i$ is equivalent to the formula

$$(\forall \bar{x}_i{:}\bar{s}_i)(b_i \to \bigvee_{j\in J_i} (\exists \bar{x}_j{:}\bar{s}_j)((b_j \approx \top) \wedge (\bar{t}_j \approx \bar{x}_i) \wedge \varphi_{ij})).$$

The existential quantifiers can be eliminated recursively from this formula by applying a consequence of the Equality Theorem (cf. Shoenfield (1967)):

$$(\exists x{:}s)((x \approx t) \wedge \varphi) \leftrightarrow \varphi[x := t] \qquad \text{provided that } x \notin FV(t).$$

Since all other formulae occurring in the proof of Theorem 5.1 are universal, this completes the proof of the corollary. □

## Arithmetical classification for computable data

We shall call a formula *open* if it does not contain occurrences of quantifiers ($\exists$,$\forall$). A formula $\varphi$ is in *prenex-form* if $\varphi = (\mathcal{Q}x_1{:}s_1)\ldots(\mathcal{Q}x_n{:}s_n)\psi$, where each $(\mathcal{Q}x_i{:}s_i)$ is either $(\exists x_i{:}s_i)$ or $(\forall x_i{:}s_i)$, $x_1, \ldots, x_n$ are all distinct, and $\psi$ is an open formula; the sequence $(\mathcal{Q}x_1{:}s_1)\ldots(\mathcal{Q}x_n{:}s_n)$ is called the *prefix* of $\varphi$. The *number of alternations* of a prefix is the number of pairs of adjacent but unlike quantifiers. A $\Sigma_n$*-prefix* is a prefix that begins with an existential quantifier ($\exists$) and has $n-1$ alternations; a $\Pi_n$*-prefix* is a prefix that begins with a universal quantifier ($\forall$) and has $n-1$ alternations (for instance, the prenex-form $(\forall x_1{:}s_1)(\exists x_2{:}s_2)(\exists x_3{:}s_3)(\forall x_4{:}s_4)\varphi$, with $\varphi$ open, has a $\Pi_3$-prefix).

We denote by $\Sigma_n^0(\Delta)$ ($\Pi_n^0(\Delta)$) the set of first-order $\Delta$-formulas that are logically equivalent to a prenex-form with a $\Sigma_n$-prefix ($\Pi_n$-prefix). We may always add "dummy" quantifiers in front of a prenex-form, so

$$(\Sigma_n^0(\Delta) \cup \Pi_n^0(\Delta)) \subseteq (\Sigma_{n+1}^0(\Delta) \cap \Pi_{n+1}^0(\Delta)).$$

Any first-order $\Delta$-formula has a logically equivalent prenex-form (cf. Shoenfield (1967)), so for every $\Delta$-formula $\varphi$ there exists an $n$ such that $\varphi \in \Sigma_n^0(\Delta) \cup \Pi_n^0(\Delta)$.

Let us call a first-order $\Delta$-formula $\varphi$ *bisimulation expressible* if there exist $\mathbb{p}$-ground process terms $p$ and $q$ such that $\mathfrak{D}, \alpha \models \varphi$ iff $\mathfrak{P}/\underline{\leftrightarrow}, \alpha \models p \approx q$, for every valuation $\alpha$ of $X$ in $\mathfrak{D}$. We shall now see that for certain data algebras any formula in $\Pi_4^0(\Delta)$ is bisimulation expressible.

**PROPOSITION 5.5 ($\exists$-INTRODUCTION)** If $\varphi$ is bisimulation expressible by $\mathbb{p}$-ground process terms $p_\varphi$ and $q_\varphi$, $\{\bar{x}\} \cap FV(q_\varphi) = \emptyset$ and $\mathcal{A} \neq \emptyset$, then $(\exists \bar{x}{:}\bar{s})\varphi$ is bisimulation expressible.

PROOF. We define $\mathbb{p}$-ground process terms $\lambda_\exists$ and $\rho_\exists$ by

$$\lambda_\exists = \sum_{\bar{x}:\bar{s}} ap_\varphi \qquad \rho_\exists = aq_\varphi + \lambda_\exists \qquad (a \text{ an action term, } \{\bar{x}\} \cap FV(a) = \emptyset.)$$

By Proposition 4.5 it is immediate that $\bar{\alpha}(\rho_\exists)$ simulates $\bar{\alpha}(\lambda_\exists)$, for every valuation $\alpha$ of $X$ in $\mathfrak{D}$, so it suffices to show that $\mathfrak{D},\alpha \models (\exists \bar{x}{:}\bar{s})\varphi$ iff $\bar{\alpha}(\lambda_\exists)$ simulates $\bar{\alpha}(aq_\varphi)$. So we consider the single transition $\bar{\alpha}(aq_\varphi) \xrightarrow{\mathtt{a}} \bar{\alpha}(q_\varphi)$ of $\bar{\alpha}(aq_\varphi)$. Using that $\{\bar{x}\} \cap FV(aq_\varphi) = \emptyset$ we find that

$$\bar{\alpha}(q_\varphi) = \bar{\alpha}_{[\bar{x}:=\bar{d}]}(q_\varphi) \text{ and } \bar{\alpha}(\lambda_\exists) \xrightarrow{\mathtt{a}} \bar{\alpha}_{[\bar{x}:=\bar{d}]}(p_\varphi), \qquad \text{for any sequence } \bar{d} \in \mathfrak{D}(\bar{s}).$$

If $\mathfrak{D},\alpha \models (\exists \bar{x}{:}\bar{s})\varphi$, then $\bar{\alpha}(\lambda_\exists)$ simulates $\bar{\alpha}(aq_\varphi)$, for there exists a sequence $\bar{d}$ such that $\bar{\alpha}_{[\bar{x}:=\bar{d}]}(p_\varphi) = \bar{\alpha}_{[\bar{x}:=\bar{d}]}(q_\varphi) = \bar{\alpha}(q_\varphi)$; and conversely, if $\mathfrak{D},\alpha \not\models (\exists \bar{x}{:}\bar{s})\varphi$, then there exists no such sequence, whence $\bar{\alpha}(\lambda_\exists)$ does not simulate $\bar{\alpha}(aq_\varphi)$. □

**Proposition 5.6 (∀-introduction)** If $\varphi$ is bisimulation expressible by $\mathbb{p}$-ground process terms $p_\varphi$ and $q_\varphi$ and $\mathcal{A}$ contains an action declaration $\mathtt{a}{:}s$, then $(\forall x{:}s)\varphi$ is bisimulation expressible.

Proof. We define $\mathbb{p}$-ground process terms $\lambda_\forall$ and $\rho_\forall$ by

$$\lambda_\forall = \textstyle\sum_{x:s} \mathtt{a}(x)p_\varphi \qquad \rho_\forall = \textstyle\sum_{x:s} \mathtt{a}(x)q_\varphi.$$

Fix an arbitrary valuation $\alpha$ of $X$ in $\mathfrak{D}$ and let $\mathtt{a}(d)$, for any $d \in \mathfrak{D}(s)$, refer to $\bar{\alpha}_{[x:=d]}(\mathtt{a}(x))$. Then $\bar{\alpha}(\lambda_\forall)$ and $\bar{\alpha}(\rho_\forall)$ have the following transitions:

$$\bar{\alpha}(\lambda_\forall) \xrightarrow{\mathtt{a}(d)} \bar{\alpha}_{[x:=d]}(p_\varphi) \qquad \bar{\alpha}(\rho_\forall) \xrightarrow{\mathtt{a}(d)} \bar{\alpha}_{[x:=d]}(q_\varphi) \qquad \text{for every } d \in \mathfrak{D}(s).$$

Clearly, $\bar{\alpha}(\lambda_\forall) = \bar{\alpha}(\rho_\forall)$ iff $\bar{\alpha}_{[x:=d]}(p_\varphi) = \bar{\alpha}_{[x:=d]}(q_\varphi)$ for every $d \in \mathfrak{D}(s)$; so $\mathfrak{P}/\leftrightarrows \models \lambda_\forall \approx \rho_\forall$ iff $\mathfrak{D},\alpha \models (\forall x{:}s)\varphi$. □

**Corollary 5.7** If $\varphi$ is bisimulation expressible by $\mathbb{p}$-ground process terms $p_\varphi$ and $q_\varphi$, $\{\bar{x}_2\} \cap FV(q_\varphi) = \emptyset$ and $\mathcal{A}$ contains an action declaration $\mathtt{a}_s{:}s$ for every $s \in \{\bar{s}_2\}$, then

$$(\forall \bar{x}_1{:}\bar{s}_1)(\exists \bar{x}_2{:}\bar{s}_2)(\forall \bar{x}_3{:}\bar{s}_3)\varphi$$

is bisimulation expressible.

Proof. By a sequence of applications of Proposition 5.6 we obtain $\mathbb{p}$-ground process terms $\lambda_\forall$ and $\rho_\forall$ that express $(\forall \bar{x}_3{:}\bar{s}_3)\varphi$. Notice that if $\{\bar{x}_2\} \cap FV(q_\varphi) = \emptyset$, then $\{\bar{x}_2\} \cap FV(\rho_\forall) = \emptyset$, so we may apply Proposition 5.5 to show that $(\exists \bar{x}_2{:}\bar{s}_2)(\forall \bar{x}_3{:}\bar{s}_3)\varphi$ is bisimulation expressible. Another sequence of applications of Proposition 5.6 completes the proof. □

If $\Delta$ contains an *equality function* for every sort $s \in \Delta$, i.e. a function declaration $[\_ =_s \_]{:}\, ss \to \mathbb{b}$, then $\Delta$ is a many-sorted first-order language; the function declarations with target sort $\mathbb{b}$ being the predicate symbols. The following definition is somewhat more liberal: we allow the equality function to be a binary term operation (a term with precisely two free variables).

**Definition 5.8 (Built-in Equality)** A $\Delta$-algebra $\mathfrak{D}$ has *built-in equality* for sort $s$ iff there exists a boolean $\Delta$-term $[x =_s y]$ in variables $x$ and $y$ of sort $s$ such that for all valuations $\alpha$ of $X$ in $\mathfrak{D}$:

$$\bar{\alpha}([x =_s y]) = \begin{cases} \top & \text{if } \alpha(x) = \alpha(y) \\ \bot & \text{otherwise} \end{cases}$$

A data algebra has *built-in equality* if it has built-in equality for all its sorts.

Note that any two-element boolean algebra has built-in equality: we set

$$[x =_{\mathbb{b}} y] = x \leftrightarrow y.$$

If $\bar{x} = x_1 \cdots x_n$, $\bar{y} = y_1 \cdots y_n$ and $x_i, y_i \in X_{s_i}$ for $1 \le i \le n$ and $\mathfrak{D}$ has built-in equality for $s_1, \ldots, s_n$, then we use $[\bar{x} = \bar{y}]$ as a shorthand for $[x_1 =_{s_1} y_1] \wedge \cdots \wedge [x_n =_{s_n} y_n]$. If $\mathfrak{D}$ has built-in equality for every sort $s \in \Delta$ occurring in the least signature containing $\mathcal{A}$, then we shall say that $\mathfrak{D}$ has built-in equality for $\mathcal{A}$.

If $\mathfrak{D}$ has built-in equality, then any open formula $\varphi$ can be translated to a boolean term $\varphi^{\mathbb{b}}$ such that $\varphi \leftrightarrow (\varphi^{\mathbb{b}} \approx \top)$, simply by replacing every equation $t \approx t'$ by $[t = t']$.

**Proposition 5.9** If $\varphi$ is an open formula, then there exists a boolean term $\varphi^{\mathbb{b}}$ such that $\mathfrak{D}, \alpha \models \varphi$ iff $\mathfrak{D}, \alpha \models \varphi^{\mathbb{b}} \approx \top$, for any valuation $\alpha$ of $X$ in $\mathfrak{D}$.

If $\mathfrak{D}$ has built-in equality, then any universal $\Delta$-formula is bisimulation expressible by means of $\mathbb{p}$-ground input prefix terms.

**Corollary 5.10** If $\mathfrak{D}$ has built-in equality and $\mathcal{A}$ contains for every sort $s \in \Delta$ an action declaration $\mathtt{a}_s{:}s$, then any formula in $\Pi_1^0(\Delta)$ is bisimulation expressible by means of input prefix terms.

Proof. Let $\varphi$ be any open $\Delta$-formula, and let $a$ be some action term not containing variables (e.g., $\mathtt{a}_{\mathbb{b}}(\top)$). Clearly, $\mathfrak{P}/\leftrightarrows, \alpha \models a \lhd \varphi^{\mathbb{b}} \rhd \delta \approx a$ iff $\mathfrak{D}, \alpha \models \varphi$. Any number of applications of Proposition 5.6 to these terms obviously yield input prefix terms. □

**Theorem 5.11** If $\mathfrak{D}$ has built-in equality and $\mathcal{A}$ contains for every sort $s \in \Delta$ an action declaration $\mathtt{a}_s{:}s$, then any formula in $\Pi_4^0(\Delta)$ is bisimulation expressible.

Proof. By Corollary 5.7 it suffices to show that for any first-order $\Delta$-formula of the form $(\exists \bar{x}{:}\bar{s})\varphi$, with $\varphi$ open, there exist $\mathbb{p}$-ground process terms $p$ and $q$ such that $FV(q) = \emptyset$ and $\mathfrak{D}, \alpha \models \varphi$ iff $\mathfrak{P}/\leftrightarrows, \alpha \models p \approx q$, for every valuation $\alpha$ of $X$ in $\mathfrak{D}$.
Let $\varphi^{\mathbb{b}}$ be a boolean term such that $\varphi \leftrightarrow (\varphi^{\mathbb{b}} \approx \top)$, the existence of which follows from Proposition 5.9. We define $p$ and $q$ by

$$p = \textstyle\sum_{\bar{x}:\bar{s}} a \lhd \varphi^{\mathbb{b}} \rhd \delta \qquad q = a,$$

where $a$ an action term not containing variables. The verification that, indeed, $\bar{\alpha}(p) \leftrightarrows \bar{\alpha}(q)$ iff $\mathfrak{D}, \alpha \models \varphi$, is trivial. □

If $\mathfrak{D}$ is a computable algebra (see Bergstra and Tucker (1995) for a definition), then every open $\Delta$-formula defines a recursive relation on the natural numbers. Thus Theorem 5.1 states the relation $\leftrightarrows$ on the set of *finite processes* (those elements of P that are interpretations of ground process terms) is in the *Arithmetical Hierarchy*, it can be defined from a recursive relation by means of a finite sequence of complementation and projection operations (cf. Rogers, Jr. (1992)). Moreover, on the set of *input prefix processes* (those elements of P that are interpretations of ground input prefix terms) strong bisimulation is in $\Pi_1^0$, by Corollary 5.4.

Now, let $\mathfrak{D}$ consist of a two-element boolean algebra, the field $\langle \mathbb{Z}, +, \times, 0, 1 \rangle$ of integers and a built-in equality function $[\_ = \_]\colon \mathbb{Z}^2 \to \mathbb{b}$; $\mathfrak{D}$ is a computable algebra (cf. Stoltenberg-Hansen and Tucker (1995)). It was shown by Matijasevič (1970) that for every $i \in \omega$ there exists $j \in \omega$ and a polynomial $U_i$ with integer coefficients such that

$$\langle x, y_1, \ldots, y_i \rangle \in W_n \text{ iff } (\exists y_{i+1} \ldots y_j)(U_i(n, x, y_1, \ldots, y_j) \approx 0),$$

where $W_0, W_1, \ldots$ is the list of all $(i+2)$-ary recursive enumerable relations (Matijasevič and Robinson (1975) proved that, in fact, one can choose $j \leq i + 13$).

Hence, by Kleene's Enumeration Theorem (cf. Rogers, Jr. (1992)), the binary predicates $E_i$, defined by

$$E_i(x, n) \leftrightarrow \begin{cases} (\forall y_1)(\exists y_2) \ldots (\forall y_i \ldots y_j) \neg (U_i(n, x, y_1, \ldots, y_j) \approx 0) & \text{if } i \text{ is odd} \\ (\forall y_1)(\exists y_2) \ldots (\exists y_i \ldots y_j)(U_i(n, x, y_1, \ldots, y_j) \approx 0) & \text{if } i \text{ is even} \end{cases}$$

are complete in $\Pi_i^0$.

Thus we get the following theorem by applications of Corollary 5.10 and Theorem 5.11, respectively.

**Theorem 5.12** For the class of strong bisimulation algebras with a computable data algebra:

i. equality on the set of input prefix processes is complete in $\Pi_1^0$; and

ii. equality on the set of finite processes is $\Pi_4^0$-hard.

**Remark 5.13** Observe that the formula $U_i(n, x, y_1, \ldots, y_j) \approx 0$ can also be expressed by

$$a(U_i(n, x, y_1, \ldots, y_j)) \leftrightarrows a(0).$$

So even if equality is not present in the data algebras, Corollary 5.7 can still be employed to show that equality on the set of interpretations of ground process terms is $\Pi^0_2$-hard.

We have not been able to settle the following question:

**Question 5.1** Is equality on finite processes complete in $\Pi^0_4$?

# 6. Relative Completeness

Theorem 5.12 indicates that there exist strong bisimulation algebras with a computable data algebra that have no ground complete axiomatisation. In this section we shall identify a subclass of computable data algebras such that each strong bisimulation algebra over any member of this subclass has a complete axiomatisation. Membership of this subclass is determined by the properties of having built-in equality, and allowing the elimination of quantifiers in the following sense:

**Definition 6.1 (Built-in Skolem Functions)** A data algebra $\mathfrak{D}$ has *built-in Skolem functions* if for every $\Delta$-formula $\varphi$ with $FV(\varphi) = \{x, y_1, \ldots, y_n\}$ there exists a term $t_\varphi = t_\varphi(y_1, \ldots, y_n)$ such that

$$\mathfrak{D} \models (\exists x{:}s)\varphi \quad \text{implies} \quad \mathfrak{D} \models \varphi(t_\varphi(y_1, \ldots, y_n), y_1, \ldots, y_n).$$

If $\mathfrak{D}$ is a computable algebra, then the set $ClEqTh(\mathfrak{D})$ of variable-free identities that hold in $\mathfrak{D}$ is decidable. Moreover, if $\mathfrak{D}$ has built-in Skolem functions and $X_s$ is nonempty, then $T_\Delta[X]^s$ contains a variable-free term (take for instance the "Skolem function" for the tautology $(\exists x{:}s)(x \approx x)$), for every sort $s \in \Delta$. Thus if $\mathfrak{D}$ is a computable algebra with built-in equality and Skolem functions, then $EqTh(\mathfrak{D})$ is decidable: if $t = t(\bar{x})$ and $t' = t'(\bar{x})$, then

$$t \approx t' \leftrightarrow \neg(\exists \bar{x}{:}\bar{s})\neg(t \approx t') \leftrightarrow \neg(\exists \bar{x}{:}\bar{s})([t = t'] \approx \bot),$$

and the existential formulas can be eliminated by substituting Skolem functions; this will yield variable-free terms $u$ and $u'$ such that $t \approx t' \in EqTh(\mathfrak{D})$ iff $[u = u'] \approx \top \in ClEqTh(\mathfrak{D})$.

The axiomatisation that we shall prove to be complete consists of a recursive (but infinite) set of identities between process terms and the recursive set $EqTh(\mathfrak{D})$ of $\Delta$-equations that are valid in $\mathfrak{D}$ (Bergstra and Heering (1994) show that such data algebras possess a *finite* $\omega$-complete algebraic specification if hidden sorts and functions are allowed).

It is immediate by the proof of Theorem 5.1 that for any two process terms $p$ and $q$ we can find a first-order $\Delta$-formula $\varphi$ such that $\mathfrak{D}, \alpha \models \varphi$ iff $\bar{\alpha}(q)$ simulates $\bar{\alpha}(p)$. If $\mathfrak{D}$ has built-in Skolem functions, then we can eliminate all quantifiers from $\varphi$; so the following proposition is an immediate consequence of Proposition 5.9.

**Proposition 6.2** If $\mathfrak{D}$ has built-in equality and Skolem functions, then there exists a boolean term $b$ such that $\bar{\alpha}(b) = \mathsf{T}$ iff $\bar{\alpha}(q)$ simulates $\bar{\alpha}(p)$, for all valuations $\alpha$ of $X$ in $\mathfrak{P}/\leftrightarrows$.

We call such a boolean term a *simulator* for $p$ by $q$.

Let us abbreviate $\Pi_\Sigma \cup EqTh(\mathfrak{D})$ by $\Pi_\Sigma(\mathfrak{D})$.

**Lemma 6.3 (Split Lemma)** If $\mathfrak{D}$ has built-in equality and Skolem functions and $X_s$ is infinite for all sorts $s \in \Delta$, then for all bisimilar $\mathfrak{p}$-ground basic terms $p, q \in \mathfrak{T}_\Sigma[X](\mathfrak{p})$ there exist a finite set $K$ and simple basic terms $p_k$ and $q_k$ such that for each $k \in K$

$$\Pi_\Sigma \cup EqTh(\mathfrak{D}) \vdash \sum_{k \in K} p_k \approx p, \quad \Pi_\Sigma \cup EqTh(\mathfrak{D}) \vdash \sum_{k \in K} q_k \approx q, \text{ and } \quad p_k \leftrightarrows q_k.$$

PROOF. Since $p$ and $q$ are basic terms there exist finite disjoint sets $I$ and $J$ such that

$$p = \sum_{i\in I} p_i, \qquad q = \sum_{j\in J} q_j, \qquad \text{and } p_i,\ q_j \text{ simple.}$$

Let $p_i = \sum_{\bar{x}_i:\bar{s}_i} a_i p_i' \lhd b_i \rhd \delta$ and let $b_{ij}^p$ be a simulator for $a_i p_i' \lhd b_i \rhd \delta$ by $q_j$. Define $p_{ij} = \sum_{\bar{x}_i:\bar{s}_i} a_i p_i' \lhd b_i \wedge b_{ij}^p \rhd \delta$.

**Claim:** $\Pi_\Sigma(\mathfrak{D}) \vdash \sum_{j\in J} p_{ij} \approx p_i$.

Since $p \leftrightarrows q$, for any valuation $\alpha$ there exists $j \in J$ such that $\bar{\alpha}(q_j)$ simulates $\bar{\alpha}(a_i p_i' \lhd b_i \rhd \delta)$. Hence $\Pi(\mathfrak{D}) \vdash \bigvee \{b_{ij}^p \mid j \in J\} \approx \top$, so by COND5 and some straightforward applications of the axioms for the booleans we derive

$$\begin{aligned}\sum_{j\in J} a_i p_i' \lhd b_i \wedge b_{ij}^p \rhd \delta &\approx a_i p_i' \lhd b_i \wedge \bigvee \{b_{ij}^p \mid j \in J\} \rhd \delta \\ &\approx a_i p_i' \lhd b_i \wedge \top \rhd \delta \\ &\approx a_i p_i' \lhd b_i \rhd \delta\end{aligned}$$

and by SUM4

$$\begin{aligned}\sum_{j\in J} \textstyle\sum_{\bar{x}_i:\bar{s}_i} a_i p_i' \lhd b_i \wedge b_{ij}^p \rhd \delta &\approx \textstyle\sum_{\bar{x}_i:\bar{s}_i} \left( \sum_{j\in J} a_i p_i' \lhd b_i \wedge b_{ij}^p \rhd \delta \right) \\ &\approx \textstyle\sum_{\bar{x}_i:\bar{s}_i} a_i p_i' \lhd b_i \rhd \delta.\end{aligned}$$

By a symmetric argument we find simulators $b_{ij}^q$ and simple basic terms

$$q_{ij} = \textstyle\sum_{\bar{x}_j:\bar{s}_j} a_j q_j' \lhd b_j \wedge b_{ij}^q \rhd \delta$$

such that $\Pi_\Sigma(\mathfrak{D}) \vdash \sum_{j\in J} q_{ij} \approx q_j$.
It remains to show that $p_{ij} \leftrightarrows q_{ij}$. Since $X_s$ is infinite for all sorts $s \in \Delta$, we may assume without losing generality that for all $i \in I$ and $j \in J$

$$\{\bar{x}_i\} \cap \{\bar{x}_j\} = \emptyset \qquad \{\bar{x}_i\} \cap FV(q_{ij}) = \{\bar{x}_j\} \cap FV(p_{ij}) = \emptyset.$$

Let $\alpha$ be any valuation such that $\bar{\alpha}(p_{ij}) \xrightarrow{\mathbf{a}} \mathbf{p}'$; there exist $\bar{d}_i$ such that $\bar{\alpha}_{[\bar{x}_i:=\bar{d}_i]}(b_{ij}^p) = \mathsf{T}$, so we find $\bar{\alpha}(q_j) \xrightarrow{\mathbf{a}} \mathbf{q}'$ and $\mathbf{p}' \leftrightarrows \mathbf{q}'$ or $\mathbf{p}' = \mathbf{q}' = \epsilon$. Hence there exist $\bar{d}_j$ such that $\bar{\alpha}_{[\bar{x}_j:=\bar{d}_j]}(a_j q_j' \lhd b_j \rhd \delta) \xrightarrow{\mathbf{a}} \mathbf{q}'$. Moreover, $\bar{\alpha}(p_i) \xrightarrow{\mathbf{a}} \mathbf{p}'$, whence $\bar{\alpha}_{[\bar{x}_j:=\bar{d}_j]}(p_i) \xrightarrow{\mathbf{a}} \mathbf{p}'$. So $\bar{\alpha}_{[\bar{x}_j:=\bar{d}_j]}(b_{ij}^q) = \mathsf{T}$, and $\bar{\alpha}(q_{ij}) \xrightarrow{\mathbf{a}} \mathbf{q}'$. We conclude that $\bar{\alpha}(q_{ij})$ simulates $\bar{\alpha}(p_{ij})$ for all valuations $\alpha$.
Conversely, by interchanging $p$ and $q$ in the preceding argument $\bar{\alpha}(p_{ij})$ simulates $\bar{\alpha}(q_{ij})$ for all valuations $\alpha$, so $p_{ij} \leftrightarrows q_{ij}$. □

If $\mathtt{a}{:}s_1 \cdots s_n$ is an action declaration, $b$ is a boolean term, and $t_i$, $t_i'$ are data terms of sort $s_i$ such that $\bar{\alpha}(b) = \mathsf{T} \Rightarrow \bar{\alpha}(t_i) = \bar{\alpha}(t_i')$ for all valuations $\alpha$, then

$$\mathtt{a}(t_1, \ldots, t_n) \lhd b \rhd \delta \leftrightarrows \mathtt{a}(t_1', \ldots, t_n') \lhd b \rhd \delta,$$

so to be able to prove equalities between process terms that originate from conditional equalities between data terms, we need to axiomatise this interplay between data and processes. To this end, we assume that $\mathfrak{D}$ has built-in equality for $\mathcal{A}$ and define for every $\mathtt{a}{:}\bar{s} \in \mathcal{A}$ an axiom

$$(\mathrm{AE}_{\mathtt{a}}) \quad \mathtt{a}(\bar{x}) \lhd [\bar{x} = \bar{y}] \rhd \delta \approx \mathtt{a}(\bar{y}) \lhd [\bar{x} = \bar{y}] \rhd \delta.$$

If the boolean algebra of $\mathfrak{D}$ has two elements, then conditionals distribute over $\cdot$:

$$(\mathrm{SCA}) \quad (xy \lhd b \rhd \delta) \approx (x \lhd b \rhd \delta)(y \lhd b \rhd \delta).$$

We call this axiom the *Static Condition Axiom*, because it distinguishes our notion of strong bisimulation from the notion of bisimulation that occurs in, e.g., Bergstra and Ponse (1997), where for each conditional all valuations of the free variables are taken into account.

**Lemma 6.4** The algebra $\mathfrak{P}/\leftrightarrows$ satisfies $\Pi_\Sigma(\mathfrak{D})$+AE+SCA.

Proof. By Lemma 4.4 it suffices to prove that $\mathfrak{P}/\leftrightarrows \models \mathrm{SCA}$ and $\mathfrak{P}/\leftrightarrows \models \mathrm{AE}_a$ for all $a{:}s_1 \cdots s_n \in \mathcal{A}$.

For the first part notice that if $(\mathbf{p}\cdot\mathbf{q} \lhd \mathbf{b} \rhd \boldsymbol{\delta}) \xrightarrow{\mathbf{a}} \mathbf{p}'$, then $\mathbf{b} = \mathbf{T}$, so $\mathbf{q} \lhd \mathbf{b} \rhd \boldsymbol{\delta} \leftrightarrows \mathbf{q}$. For the second part notice that if $\mathfrak{D}([\_ =_{s_i} \_])(d_i, d_i') = \top$ for all $1 \le i \le n$, then $a\langle d_1, \ldots, d_n\rangle = a\langle d_1', \ldots, d_n'\rangle$. □

Built-in Skolem functions can be used to eliminate redundant occurrences of $\sum$.

**Proposition 6.5** Suppose $\mathfrak{D}$ has built-in Skolem functions. If $b = b(\bar{x}, \bar{y})$ is a boolean term such that $\mathfrak{D} \models (\exists \bar{x}{:}\bar{s})b \approx \top$ and $p$ is a process term such that $\{\bar{x}\} \cap FV(p) = \emptyset$, then

$$\Pi_\Sigma(\mathfrak{D}) \models \textstyle\sum_{x:s} p \lhd b \rhd \delta \approx p.$$

Proof. Since $\mathfrak{D}$ has built-in Skolem functions, we get by induction on the length of $\bar{x}$ a sequence of $\Delta$-terms $\bar{t}_b = \bar{t}_b(\bar{y})$ such that

$$\mathfrak{D} \models (\exists \bar{x}{:}\bar{s})b \approx \top \quad \text{implies} \quad \mathfrak{D} \models b(\bar{t}_b(\bar{y}), \bar{y}) \approx \top,$$

so Cond1 yields $p \lhd b(\bar{t}_b(\bar{y}), \bar{y}) \rhd \delta \approx p$, and, by Sum3

$$\begin{aligned} \textstyle\sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta &\approx \textstyle\sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta + p \lhd b(\bar{t}_b(\bar{y}), \bar{y}) \rhd \delta \\ &\approx \textstyle\sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta + p. \end{aligned} \tag{6.1}$$

On the other hand, by means of Cond1, B7, Cond5, A2, and A3 we can derive

$$p \approx p \lhd b(\bar{x}, \bar{y}) \rhd \delta + p,$$

so with Sum1, Sum4, and Sum1 we find

$$p \approx \textstyle\sum_{\bar{x}:\bar{s}} p \approx \sum_{\bar{x}:\bar{s}} (p \lhd b \rhd \delta + p) \approx \sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta + \sum_{\bar{x}:\bar{s}} p \approx \sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta + p. \tag{6.2}$$

So we get

$$\textstyle\sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta \overset{(6.1)}{\approx} \sum_{\bar{x}:\bar{s}} p \lhd b \rhd \delta + p \overset{(6.2)}{\approx} p.$$ □

**Theorem 6.6** If $\mathfrak{D}$ has built-in equality and Skolem functions and $X_s$ is infinite for all $s \in \Delta$, then $\mathfrak{P}/\leftrightarrows \models p \approx q$ iff $\Pi_\Sigma(\mathfrak{D})$+AE+SCA $\vdash p \approx q$, for all $\mathfrak{p}$-ground process terms $p$ and $q$.

Proof. The implication from right to left follows from Lemma 6.4; we prove the implication from left to right by strong induction on the sum of the complexities $|p|$ and $|q|$ of $p$ and $q$ respectively, defined as the maximal nesting of $\cdot$ with $|\delta| = 0$ and $|a| = 1$ for all action terms $a$.

By Lemma 6.3 it suffices to prove the theorem for simple basic terms; let

$$p = \textstyle\sum_{\bar{x}_1:\bar{s}_1} a_1 p_1 \lhd b_1 \rhd \delta \quad \text{and} \quad q = \sum_{\bar{x}_2:\bar{s}_2} a_2 p_2 \lhd b_2 \rhd \delta.$$

If $\bar{\alpha}(b_1), \bar{\alpha}(b_2) = \bot$ for all valuations $\alpha$, then $b_1, b_2 \approx \bot$ and by Cond2 and Sum1 we find $p, q \approx \delta$. The remaining case is that $b_1, b_2 \not\approx \bot$. Then $a_1$ and $a_2$ must agree on their leading action declaration, i.e. let $a_1 = \mathtt{a}(\bar{t}_1)$ and $a_2 = \mathtt{a}(\bar{t}_2)$ for some $\mathtt{a}{:}s_1 \cdots s_n \in \mathcal{A}$.

Define $b = b_{12} \wedge b_{21}$, where $b_{12}$ is a simulator for $a_1 p_1$ by $a_2 p_2$ and $b_{21}$ a simulator for $a_2 p_2$ by $a_1 p_1$; we shall first prove

$$\Pi_\Sigma(\mathfrak{D})\text{+AE+SCA} \vdash a_1 p_1 \lhd b \rhd \delta \approx a_2 p_2 \lhd b \rhd \delta \tag{6.3}$$

For all valuations $\alpha$, $\bar{\alpha}(b) = \mathsf{T}$ implies $\bar{\alpha}(\mathtt{a}(\bar{t}_1)) = \bar{\alpha}(\mathtt{a}(\bar{t}_2))$, so $b \wedge [\bar{t}_1 = \bar{t}_2] \approx b$ and

$$\begin{aligned} a_1 \lhd b \rhd \delta &\approx (a_1 \lhd [\bar{t}_1 = \bar{t}_2] \rhd \delta) \lhd b \rhd \delta && \text{(Cond4)} \\ &\approx (a_2 \lhd [\bar{t}_1 = \bar{t}_2] \rhd \delta) \lhd b \rhd \delta && \text{(instance of } \mathrm{AE}_{\mathtt{a}}\text{)} \\ &\approx a_2 \lhd b \rhd \delta && \text{(Cond4).} \end{aligned} \tag{6.4}$$

If $p_1, p_2 = \epsilon$, then (6.3) trivially follows; otherwise $p_1, p_2 \neq \epsilon$, so the induction hypothesis yields

$$\Pi_\Sigma(\mathfrak{D})+\mathrm{AE}+\mathrm{SCA} \vdash p_1 \lhd b \rhd \delta \approx p_2 \lhd b \rhd \delta. \tag{6.5}$$

and we can derive (6.3):

$$\begin{aligned} a_1 p_1 \lhd b \rhd \delta &\approx (a_1 \lhd b \rhd \delta)(p_1 \lhd b \rhd \delta) && \text{(SCA)} \\ &\approx (a_2 \lhd b \rhd \delta)(p_2 \lhd b \rhd \delta) && \text{(6.4), (6.5)} \\ &\approx a_2 p_2 \lhd b \rhd \delta && \text{(SCA)} \end{aligned}$$

Observe that $b_1 \to (b_2 \wedge b)$ is a simulator for $a_1 p_1 \lhd b_1 \rhd \delta$ by $a_2 p_2 \lhd b_2 \rhd \delta$. Since $\sum_{\bar{x}_2:\bar{s}_2} a_2 p_2 \lhd b_2 \rhd \delta$ simulates $\sum_{\bar{x}_1:\bar{s}_1} a_1 p_1 \lhd b_1 \rhd \delta$,

$$\mathfrak{D} \models \exists \bar{x}_2{:}\bar{s}_2 (b_1 \to (b_2 \wedge b)),$$

so by Proposition 6.5, Cond4 and some straightforward applications of the axioms for booleans

$$\begin{aligned} a_1 p_1 \lhd b_1 \rhd \delta &\approx \textstyle\sum_{\bar{x}_2:\bar{s}_2} (a_1 p_1 \lhd b_1 \rhd \delta) \lhd b_1 \to (b_2 \wedge b) \rhd \delta \\ &\approx \textstyle\sum_{\bar{x}_2:\bar{s}_2} a_1 p_1 \lhd b_1 \wedge b_2 \wedge b \rhd \delta. \end{aligned} \tag{6.6}$$

Moreover, if we interchange the subscripts in the preceding argument we get

$$\textstyle\sum_{\bar{x}_1:\bar{s}_1} a_2 p_2 \lhd b_2 \wedge b_1 \wedge b \rhd \delta \approx a_2 p_2 \lhd b_2 \rhd \delta. \tag{6.7}$$

We obtain $\Pi_\Sigma(\mathfrak{D})+\mathrm{AE}+\mathrm{SCA} \vdash p \approx q$ by the following derivation

$$\begin{aligned} p &= \textstyle\sum_{\bar{x}_1:\bar{s}_1} a_1 p_1 \lhd b_1 \rhd \delta \\ &\approx \textstyle\sum_{\bar{x}_1:\bar{s}_1} \sum_{\bar{x}_2:\bar{s}_2} a_1 p_1 \lhd b_1 \wedge b_2 \wedge b \rhd \delta && \text{(6.6)} \\ &\approx \textstyle\sum_{\bar{x}_1:\bar{s}_1} \sum_{\bar{x}_2:\bar{s}_2} a_2 p_2 \lhd b_2 \wedge b_1 \wedge b \rhd \delta && \text{(Cond4), (6.3)} \\ &\approx \textstyle\sum_{\bar{x}_2:\bar{s}_2} \sum_{\bar{x}_1:\bar{s}_1} a_2 p_2 \lhd b_2 \wedge b_1 \wedge b \rhd \delta && \text{(Prop. 3.1(ii))} \\ &\approx \textstyle\sum_{\bar{x}_2:\bar{s}_2} a_2 p_2 \lhd b_2 \rhd \delta && \text{(6.7)} \\ &= q \end{aligned}$$

□

Hennessy and Lin (1996) consider the subclass of data algebras that have built-in equality and an $\omega$-complete specification, and give a complete proof system for their input prefix mechanism on the basis of these restrictions. We conclude this paper with an example showing that these restrictions are not sufficient in the case of alternative quantification.

Consider the single-sorted, first-order signature $\Sigma_{\mathbb{t}}$ defined by

$$\Sigma_{\mathbb{t}} = \{\mathbb{t}, \varepsilon{:}\,\lambda \to \mathbb{t}, A{:}\,\mathbb{t} \to \mathbb{t}, B{:}\,\mathbb{t} \to \mathbb{t}, F{:}\,\mathbb{t}\mathbb{t}\mathbb{t} \to \mathbb{t}\},$$

and let $\Delta$ consist of $\Sigma_{\mathbb{b}}$, $\Sigma_{\mathbb{t}}$ and two binary function declarations $[\_ =_{\mathbb{t}} \_]{:}\,\mathbb{t}\mathbb{t} \to \mathbb{b}$ and $[\_ \leq_{\mathbb{t}} \_]{:}\,\mathbb{t}\mathbb{t} \to \mathbb{b}$. Let $\mathfrak{D}$ be the $\Delta$-algebra such that

1. $\mathfrak{D}\restriction_{\Sigma_{\mathbb{b}}}$ is a two-element boolean algebra;
2. $\mathfrak{D}\restriction_{\Sigma_{\mathbb{t}}} = \mathfrak{T}_{\Sigma_{\mathbb{t}}}[\emptyset]$ (the algebra of ground $\Sigma_{\mathbb{t}}$-terms);
3. $\mathfrak{D}([\_ =_{\mathbb{t}} \_])$ is the equality function on $\mathfrak{D}(\mathbb{t})$; and
4. $\mathfrak{D}([\_ \leq_{\mathbb{t}} \_])(t, t') = \mathsf{T}$ if $t$ is a subterm of $t'$ and $\bot$ otherwise.

Note that $\mathfrak{D}$ has built-in equality. Venkataraman (1987) has shown that the existential fragment of the first-order theory of ground term algebras with subterm predicate is decidable, so $EqTh(\mathfrak{D})$ is decidable and may serve as an (infinite) $\omega$-complete specification of $\mathfrak{D}$.

On the other hand, Venkataraman (1987) gives an effective method for reducing any instance of Post's correspondence problem over the alphabet $\{A, B\}$ to a $\Sigma_2^0(\Delta)$-formula. Since Post's correspondence problem is complete in $\Sigma_1^0$, we conclude that any problem in $\Pi_2^0$ can be effectively reduced to a $\Pi_3^0(\Delta)$-formula, which is bisimulation expressible by our Corollary 5.7 provided that the set of action declarations $\mathcal{A}$ contains actions $\mathtt{a}_{\mathbb{b}}{:}\mathbb{b}$ and $\mathtt{a}_{\mathbb{t}}{:}\mathbb{t}$. Thus, equality on the set of finite processes in the strong bisimulation algebra over $\mathfrak{D}$ and $\mathcal{A}$ is $\Pi_2^0$-hard, whence cannot posses a ground complete axiomatisation.

# References


Bergstra, J. A. and Heering, J. (1994). Which data types have $\omega$-complete initial algebra specifications. *Theoretical Computer Science*, **124**, 149–168.

Bergstra, J. A. and Klop, J. W. (1984). Process algebra for synchronous communication. *Information and Control*, **60**(1–3), 109–137.

Bergstra, J. A. and Ponse, A. (1997). Process algebra with four-valued logic. Report P9724, Programming Research Group, University of Amsterdam.

Bergstra, J. A. and Tucker, J. V. (1995). Equational specifications, complete term rewriting systems, and computable and semicomputable algebras. *Journal of the Association for Computing Machinery*, **42**(6), 1194–1230.

Burris, S. and Sankappanavar, H. P. (1981). *A Course in Universal Algebra*. Number 78 in Graduate Texts in Mathematics. Springer-Verlag, New York Heidelberg Berlin.

Groote, J. F. and Ponse, A. (1994a). Proof theory for $\mu$CRL: A language for processes with data. In D. J. Andrews, J. F. Groote, and C. A. Middelburg, editors, *Proceedings of the International Workshop on Semantics of Specification Languages*, Workshops in Computing, pages 232–251, Utrecht, The Netherlands. Springer-Verlag.

Groote, J. F. and Ponse, A. (1994b). The syntax and semantics of $\mu$CRL. In A. Ponse, C. Verhoef, and S. F. M. van Vlijmen, editors, *Algebra of Communicating Processes*, Workshops in Computing, pages 26–62, Utrecht, The Netherlands. Springer-Verlag.

Hennessy, M. and Lin, H. (1995). Symbolic bisimulations. *Theoretical Computer Science*, **138**(2), 353–389.

Hennessy, M. and Lin, H. (1996). Proof systems for message-passing process algebras. *Formal Aspects of Computing*, **8**(4), 379–407.

Matijasevič, Y. (1970). Enumerable sets are diophantine. *Soviet Math. Dokl.*, **11**, 345–358.

Matijasevič, Y. and Robinson, J. (1975). Reduction of an arbitrary diophantine equation to one in 13 unknowns. *Acta Arithmetica*, **27**, 521–553.

Milner, R. (1989). *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs.

Parrow, J. and Sangiorgi, D. (1995). Algebraic theories for name-passing calculi. *Information and Computation*, **120**(2), 174–197.

Ponse, A. (1996). Computable processes and bisimulation equivalence. *Formal Aspects of Computing*, **8**, 648–678.

Rasiowa, H. and Sikorski, R. (1963). *The mathematics of metamathematics*. Państowe wydawnictwo naukowe, Warszawa, Poland.

Rogers, Jr., H. (1992). *Theory of Recursive Functions and Effective Computability*. The MIT Press. Paperback edition. Original edition published by McGraw-Hill Book Company, 1967.

Shoenfield, J. R. (1967). *Mathematical Logic*. Addison-Wesley Publishing Company.

Stoltenberg-Hansen, V. and Tucker, J. V. (1995). Effective algebras. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Semantic Modelling*, volume 4 of *Handbook of Logic in Computer Science*, pages 357–526. Oxford Science Publications.

Venkataraman, K. N. (1987). Decidability of the purely existential fragment of the theory of term algebras. *Journal of the Association for Computing Machinery*, **34**(2), 492–510.