ELSEVIER

Information Processing Letters 80 (2001) 59–65

Information Processing Letters

www.elsevier.com/locate/ipl

Equivalence of recursive specifications in process algebra

Alban Ponse [a,b,*], Yaroslav S. Usenko [a]

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

[b] *University of Amsterdam, Programming Research Group, P.O. Box 41882, 1009 DB Amsterdam, The Netherlands*

Received 28 August 2000; received in revised form 21 December 2000

Abstract

We define an equivalence relation on recursive specifications in process algebra that is model-independent and does not involve an explicit notion of solution. Then we extend this equivalence to the specification language μCRL. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: μCRL; Process algebra; Equivalence of recursive specifications

1. Introduction

In process algebra, infinite behavior is usually specified by means of recursive equations. [1] A simple example is $X = a \cdot X$, modeling a process that repeatedly executes action a. It is often convenient to consider a *system* of interdependent recursive equations. For instance, a communication protocol can be specified such that each of its parallel components (sender, receiver, etc.) is modeled by one or more equations. In the following we will use the terminology 'system of recursive equations' to denote a set of one or more equations in the sense sketched above.

Although the specification of processes by means of systems of recursive equations serves its purpose well, proof theory for this type of specification is not entirely trivial, and goes with various particular ingredients. For instance, we often want to assert that such a system represents a particular process in some intended model as the *unique* solution for one of its variables. As an example, the recursive equation $X = X$ has (in any model) any process as its solution, and the equation $X = X + a$ (where $+$ models choice) has many solutions (in many models), whereas $X = a \cdot X$ has no solution in models that represent only finite processes. In the case that a system of recursive equations has a unique solution (per variable) in some intended model, we say that this system is a recursive *specification*: some intended process is specified by means of recursive equations. Often, establishing the uniqueness of solutions is intertwined with verification purposes. If one can show that each solution for some distinguished variable in a system of recursive equations is also a solution for a smaller and simpler system (or vice versa), and both systems have unique solutions per variable, then both systems specify the same process, one focusing on 'implementation details', and the other abstracting from these and focusing on the external behavior of the whole system. Comparing solutions of systems of

* Corresponding author.

E-mail address: alban.ponse@cwi.nl (A. Ponse).

[1] An alternative method of specification is the use of recursive operations, such as the Kleene star [4], or the use of fixpoint operators [16].

PII: S0020-0190(01)00218-6

recursive equations often plays a major role in process verification.

In this paper we introduce an equivalence on recursively specified processes that is based on the *preservation* of solutions. This equivalence results from the theory of equivalences for regular systems of equations and applicative program schemes, as developed, among others, by Courcelle in [5,6]. Systems of (recursive) equations are considered with respect to their full sets of solutions in all models. As noted in [3], considering such a notion of equivalence avoids certain drawbacks of other methods used in process algebras, such as the restriction of the process domains to the ordered ones and considering the least solutions of recursive systems, or the restriction to systems that are guarded, and considering only the domains where all such systems have unique solutions (see [2]). In many cases, especially when data parameters are involved, such restrictions can be difficult to handle. We use the specification language μCRL [13,12] to describe our approach. This language comprises an extension of process algebra with features that involve data for the specification of processes: actions, recursive specification, communication, summation, and conditionals can all be data-parametric. For instance, it is not possible to justify transformations of recursive systems in value passing process algebras like μCRL using the method of restricting to syntactically guarded systems. For many models of processes (resembling different equivalences, see, e.g., [8,9]) guardedness becomes a more involved notion. Therefore it is useful to consider a model-independent equivalence of recursive systems in process algebra, and to use model specific equivalences only in cases where the former one is not sufficiently strong.

Typical for our approach is that we separate the question to unique solutions from the question how solutions of systems of recursive equations can be compared. This splitting of notions is worthwhile: properties of solutions are interesting for verification purposes, whereas comparison of systems of recursive equations is a fundamental notion that in itself can be applied in a model-independent way, only adhering to the axioms of process algebra. The comparison of systems of recursive equations plays a major role in the tool support for μCRL because such systems are transformed into linear form while preserving all their solutions [14]. Several kinds of optimizational transformations of μCRL specifications, see, e.g., [10], can be proved to be correct using the equivalence relations presented in this paper.

Structure of the paper. In order to give a simple exposition, we start out from the well-known process algebra system BPA (Basic Process Algebra) in Section 2. We characterize the equivalence mentioned, and consider some examples. Then, in Section 3 we generalize our equivalence to the setting of μCRL. The paper is ended with some conclusions.

2. Equivalence of BPA systems

Recall that the axioms of BPA (Basic Process Algebra, see, e.g., [2,7]) are the following:

$$\text{(A1)} \quad x + y = y + x$$

$$\text{(A2)} \quad x + (y + z) = (x + y) + z$$

$$\text{(A3)} \quad x + x = x$$

$$\text{(A4)} \quad (x + y) \cdot z = x \cdot z + y \cdot z$$

$$\text{(A5)} \quad (x \cdot y) \cdot z = x \cdot (y \cdot z)$$

where $+$ models alternative composition and $\cdot$ sequential composition. We further omit brackets in repeated applications of $+$ and $\cdot$.

For terms t, u over the signature of BPA we write

$$\text{BPA} \vdash t = u$$

if $t = u$ can be proved from BPA in equational logic. Furthermore, let $\vec{x} = x_1, \ldots, x_n$ be a sequence of variables. Then we write $t(\vec{x})$ if all free variables of t are in $\vec{x}$. In this section we consider systems of (recursive) equations over the signature of BPA. As a convention for this section we shall use capital letters for the variables in such systems, in order to distinguish these from the variables in the BPA axioms.

Let n fresh variables $\mathsf{X}_1, \ldots, \mathsf{X}_n = \vec{\mathsf{X}}$ and terms $t_1(\vec{\mathsf{X}}), \ldots, t_n(\vec{\mathsf{X}})$ over BPA be fixed. Then we call

$$G = \left\{ \mathsf{X}_i = t_i(\vec{\mathsf{X}}) \mid i = 1, \ldots, n \right\}$$

a *system of process equations* over BPA. A simple example is $\{\mathsf{X}_1 = \mathsf{a} \cdot \mathsf{X}_2,\ \mathsf{X}_2 = \mathsf{b} \cdot \mathsf{X}_1\}$. Furthermore, we call (X_i, G) for a particular i a *process definition* (this terminology is syntax-oriented; the question whether

(X_i, G) really 'defines' a process is a model dependent one).

Let $\mathcal{M}$ be a model of BPA with domain M. Then $(m_1, \ldots, m_n) \in M^n$ is a *solution of G in $\mathcal{M}$* if for all $i = 1, \ldots, n$ and interpretation functions $\mathcal{I}$ satisfying $\mathcal{I}(\mathsf{X}_i) = m_i$,

$$\mathcal{M}, \mathcal{I} \models \mathsf{X}_i = t_i(\vec{\mathsf{X}}). \tag{1}$$

We further abbreviate (statements like) (1) to

$$\mathcal{M} \models m_i = t_i(\vec{m}).$$

In this case we say that m_i is a solution of (X_i, G) in $\mathcal{M}$. Finally, given G as above, i.e.,

$$G = \{\mathsf{X}_i = t_i(\vec{\mathsf{X}}) \mid i = 1, \ldots, n\},$$

we define for term sequence $\vec{v} = v_1, \ldots, v_n$,

$$G(\vec{v}) \triangleq \bigwedge_{i=1}^{n} v_i = t_i(\vec{v}).$$

We now turn to the preservation of solutions. Let

$$H = \{\mathsf{Y}_j = u_j(\vec{\mathsf{Y}}) \mid j = 1, \ldots, k\}$$

be a system of k process equations over BPA such that $\vec{\mathsf{X}}$ and $\vec{\mathsf{Y}} = \mathsf{Y}_1, \ldots, \mathsf{Y}_k$ do not share any variable. The preservation of solutions refers to designated process definitions of G and H, usually (X_1, G) and (Y_1, H), respectively.

Definition 1. Let $\overline{G}$ refer to the setting where $\mathsf{X}_1, \ldots, \mathsf{X}_n = \vec{\mathsf{X}}$ are regarded as constants and the equations in G as additional axioms. We say that (X_1, G) *implies* (Y_1, H), notation

$$(\mathsf{X}_1, G) \Rightarrow (\mathsf{Y}_1, H),$$

if there exist terms $w_1, \ldots, w_k = \vec{w}$ with $w_i = w_i(\vec{\mathsf{X}})$ such that

$$\text{BPA} \cup \overline{G} \vdash \mathsf{X}_1 = w_1, \quad \text{and}$$
$$\text{BPA} \cup \overline{G} \vdash w_j = u_j(\vec{w}) \quad \text{for all } j = 1, \ldots, k.$$

In the case of BPA, $\Rightarrow$ characterizes the preservation of solutions. This can be seen as follows: we say that (X_1, G) *is preserved by* (Y_1, H), notation

$$(\mathsf{X}_1, G) \preceq (\mathsf{Y}_1, H),$$

if in each model of BPA, each solution of (X_1, G) is also a solution of (Y_1, H). So, $(\mathsf{X}_1, G) \preceq (\mathsf{Y}_1, H)$ if in each model $\mathcal{M}$ of BPA, say with domain M, $\forall \vec{m} \in M^n (\mathcal{M} \models G(\vec{m}) \Rightarrow \exists \vec{n} \in M^k (\mathcal{M} \models H(\vec{n}) \wedge m_1 = n_1))$. Thus, a characterization of $(\mathsf{X}_1, G) \preceq (\mathsf{Y}_1, H)$ in first order logic is the following:

$$\text{BPA} \models \forall \vec{\mathsf{X}} \big(G(\vec{\mathsf{X}}) \rightarrow \exists \vec{\mathsf{Y}} (H(\vec{\mathsf{Y}}) \wedge \mathsf{X}_1 = \mathsf{Y}_1)\big). \tag{2}$$

We proceed to show that $\preceq$ is characterized by $\Rightarrow$, i.e., to derive from (2) necessary and sufficient proof obligations in *equational* logic. Let the symbol $\vdash_{fol}$ refer to derivability in first order logic. By the completeness of first order logic, (2) is equivalent with

$$\text{BPA} \vdash_{fol} \forall \vec{\mathsf{X}} \big(G(\vec{\mathsf{X}}) \rightarrow \exists \vec{\mathsf{Y}} (H(\vec{\mathsf{Y}}) \wedge \mathsf{X}_1 = \mathsf{Y}_1)\big),$$

and thus with

$$\text{BPA} \vdash_{fol} \forall \vec{\mathsf{X}} \exists \vec{\mathsf{Y}} \big(G(\vec{\mathsf{X}}) \rightarrow H(\vec{\mathsf{Y}}) \wedge \mathsf{X}_1 = \mathsf{Y}_1\big).$$

Because the variables $\mathsf{X}_1, \ldots, \mathsf{X}_n$ do not occur in the axioms of BPA, the latter statement is equivalent with:

$$\text{BPA} \vdash_{fol} \exists \vec{\mathsf{Y}} \big(G(\vec{\mathsf{X}}) \rightarrow H(\vec{\mathsf{Y}}) \wedge \mathsf{X}_1 = \mathsf{Y}_1\big).$$

The above statement is in turn equivalent with:

There exist terms $w_1, \ldots, w_k = \vec{w}$
with $w_i = w_i(\vec{\mathsf{X}})$ such that
$\text{BPA} \vdash_{fol} G(\vec{\mathsf{X}}) \rightarrow H(\vec{w}) \wedge \mathsf{X}_1 = w_1$,

and finally also equivalent with:

There exist terms $w_1, \ldots, w_k = \vec{w}$
with $w_i = w_i(\vec{\mathsf{X}})$ such that
$\text{BPA} \cup \overline{G} \vdash_{fol} H(\vec{w}) \wedge \mathsf{X}_1 = w_1$.

(This last equivalence follows from the Deduction Theorem on open formulae, see, e.g., [17, pp. 33, 34].)

Now we have transformed our logical characterization of the preservation of solutions into the setting of equational logic:

Theorem 2. *Let (X_1, G) and (Y_1, H) be process definitions over BPA. Then $(\mathsf{X}_1, G) \preceq (\mathsf{Y}_1, H)$ iff $(\mathsf{X}_1, G) \Rightarrow (\mathsf{Y}_1, H)$.*

Proof. $(\mathsf{X}_1, G) \preceq (\mathsf{Y}_1, H)$ iff (2) holds. As argued above, this is the case iff there exist terms $w_1, \ldots, w_k = \vec{w}$ with $w_i = w_i(\vec{\mathsf{X}})$ such that $G \vdash_{fol} H(\vec{w}) \wedge \mathsf{X}_1 = w_1$ is derivable from $\text{BPA} \cup \overline{G}$ in first order logic, which in turn is the case iff each conjunct is derivable, or in other words, iff $(\mathsf{X}_1, G) \Rightarrow (\mathsf{Y}_1, H)$. (By the

completeness of first order logic and of equational logic, for any set Γ of equations $\Gamma \vdash t = u$ iff $\Gamma \vdash_{fol} t = u$.) □

Implication between process definitions induces the following equivalence between process definitions:

$(\mathsf{X}_1, G) = (\mathsf{Y}_1, H)$

if $(\mathsf{X}_1, G) \Rightarrow (\mathsf{Y}_1, H)$ and $(\mathsf{Y}_1, H) \Rightarrow (\mathsf{X}_1, G)$. Evidently, this is an equivalence. (We do not treat its possible congruence property: we do not have any use for that.)

Some examples. If $G = \{\mathsf{X} = \mathsf{X} + \mathsf{a} + \mathsf{b}\}$ and $H = \{\mathsf{Y} = \mathsf{Y} + \mathsf{a}\}$, then $(\mathsf{X}, G) \Rightarrow (\mathsf{Y}, H)$ but not vice versa.

If $G = \{\mathsf{X}_1 = \mathsf{a} \cdot \mathsf{X}_2,\ \mathsf{X}_2 = \mathsf{b} \cdot \mathsf{X}_1\}$ and $H = \{\mathsf{Y} = \mathsf{a} \cdot \mathsf{b} \cdot \mathsf{Y}\}$, then $(\mathsf{X}_1, G) = (\mathsf{Y}, H)$, the proof of which we leave to the reader.

The systems $G = \{\mathsf{X} = \mathsf{a} \cdot \mathsf{X}\}$ and $H = \{\mathsf{Y} = \mathsf{a} \cdot \mathsf{Y} \cdot \mathsf{b}\}$ are incomparable: in the model with domain $\mathbb{Z}$, and with $+$ interpreted as maximum, $\cdot$ as addition, and a as the value -1 and b as the value 1, there is no solution for X and many for Y. The converse holds in case a is interpreted as 0 and b as 1.

If $G = \{\mathsf{X}_1 = \mathsf{a} + \mathsf{X}_1 \cdot \mathsf{a},\ \mathsf{X}_2 = \mathsf{a} \cdot \mathsf{X}_2\}$ and $H = \{\mathsf{Y} = \mathsf{a} + \mathsf{Y} \cdot \mathsf{a}\}$, then $(\mathsf{X}_1, G) \Rightarrow (\mathsf{Y}, H)$, but the reverse implication does not hold. Consider the model where processes are trees with finite paths, but possibly infinite branching, taken modulo bisimulation equivalence In this model (Y, H) has a solution which is the class of trees representing the process $\sum_{i \in Nat} a^{i+1}$. But G has no solutions in this model because of its second equation, which requires an infinite path. See [2, p. 33] and [1, p. 153] for more information about this counterexample.

3. Up to μCRL

The language μCRL [13,12] is an extension of ACP-style process algebra with data-parametric actions, alternative composition over data domain, value-passing communication, and conditions. Furthermore, recursion in μCRL allows to specify data-parametric processes by means of systems of data-parametric process equations.

The axioms of μCRL define two sorts, Booleans *Bool* and processes *Proc*, which are part of any μCRL specification. Other data types, like natural numbers, integers, lists, queues, stacks, and domain specific data types can be defined by algebraic specifications.

Let $\vec{f} = f_1, \ldots, f_n$ be a sequence of typed function symbols $f_1 : \overrightarrow{D_{f_1}} \to \mathit{Proc}, \ldots, f_n : \overrightarrow{D_{f_n}} \to \mathit{Proc}$ for given data types $\overrightarrow{D_{f_i}}$, and $\vec{d}$ be a sequence of (typed) data variables. Then we write $t(\vec{f}, \vec{d})$ for a term t over the signature of μCRL extended with $\vec{f}$, if all its free data variables are in $\vec{d}$.

Let n fresh typed function symbols $\mathsf{X}_1, \ldots, \mathsf{X}_n$ be fixed. Then we call

$$G = \left\{ \mathsf{X}_i(\overrightarrow{d_{\mathsf{X}_i} : D_{\mathsf{X}_i}}) = t_i(\vec{\mathsf{X}}, \overrightarrow{d_{\mathsf{X}_i}}) \mid i = 1, \ldots, n \right\}$$

a *system of process equations* over μCRL. Each function symbol X_i is called a *process name of* G. Furthermore, we call each pair $(\mathsf{X}_i(\overrightarrow{dt}), G)$, for some appropriately typed sequence of data terms $\overrightarrow{dt}$, a *process definition*. As an example,

$$G = \left\{ \mathsf{X}(b : \mathit{Bool}) = \mathsf{a}(b) \cdot \mathsf{X}(\neg b) \right\}$$

is a system of process equations, and $(\mathsf{X}(\mathbf{t}), G)$ and $(\mathsf{X}(b), G)$ where $\mathbf{t}$ stands for "true" and b is a Boolean variable, both are process definitions.

Process definitions in μCRL comprise a restricted form of recursive applicative program schemes as defined in [5,6]. The restrictions are that all unknowns (process names) have the same range *Proc* and that there are no functions from *Proc* to other sorts. On the other hand, process definitions extend recursive applicative program schemes with binders (because the sum operators of μCRL are binders), and therefore require a more refined approach for a formal treatment, such as generalized equational logic [11].

Let $\mathcal{M}$ be a model of μCRL and the data types used in G and $\vec{t}$, with domains P for processes, B for Booleans and $\mathsf{D}_{\mathsf{X}_i}$ for D_{X_i}. A *solution* of G in $\mathcal{M}$ is a tuple $(f_1, \ldots, f_n)$ of functions $f_i : \overrightarrow{\mathsf{D}_{\mathsf{X}_i}} \to P$ such that for all $i = 1, \ldots, n$

$$\mathcal{M} \models f_i(\overrightarrow{d_{\mathsf{X}_i}}) = t_i(\vec{f}, \overrightarrow{d_{\mathsf{X}_i}}).$$

In this case $f_i(\overrightarrow{t_{\mathcal{M}}})$ is a solution of $(\mathsf{X}_i(\vec{t}), G)$ in $\mathcal{M}$.

Given G as above, let

$$H = \left\{ \mathsf{Y}_j(\overrightarrow{d_{\mathsf{Y}_j} : D_{\mathsf{Y}_j}}) = u_j(\vec{\mathsf{Y}}, \overrightarrow{d_{\mathsf{Y}_j}}) \mid j = 1, \ldots, k \right\}$$

be a fresh system of process equations over μCRL. We say that $(\mathsf{X}_1(\vec{t}), G)$ *is preserved by* $(\mathsf{Y}_1(\vec{u}), H)$, notation

$$\left(\mathsf{X}_1(\vec{t}), G\right) \preceq \left(\mathsf{Y}_1(\vec{u}), H\right),$$

if in each model $\mathcal{M}$ of μCRL and the data we have

$$\forall \vec{f}\big((\forall i\ \mathcal{M} \models f_i(\overrightarrow{d_{\mathsf{X}_i}}) = t_i(\vec{f}, \overrightarrow{d_{\mathsf{X}_i}})\big)$$
$$\Rightarrow$$
$$\exists \vec{g}\big(\forall j\ \mathcal{M} \models g_j(\overrightarrow{d_{\mathsf{Y}_j}}) = u_i(\vec{g}, \overrightarrow{d_{\mathsf{Y}_j}}) \wedge \mathcal{M} \models f_1(\vec{t}) = g_1(\vec{u})\big)\big),$$

where $f_i : \overrightarrow{D_{\mathsf{X}_i}} \to P$ and $g_j : \overrightarrow{D_{\mathsf{Y}_j}} \to P$.

We now define implication between process definitions in the setting of μCRL. Let the symbol $\vdash$ stand for derivability in generalized equational logic. We say that $(\mathsf{X}_1(\overrightarrow{dt}), G)$ *conditionally implies* $(\mathsf{Y}_1(\overrightarrow{dt'}), H)$, notation

$$\big(\mathsf{X}_1(\overrightarrow{dt}), G\big) \Rightarrow_c \big(\mathsf{Y}_1(\overrightarrow{dt'}), H\big),$$

if there exist terms $w_j(\overrightarrow{d_{\mathsf{Y}_j}}) = w_j(\vec{\mathsf{X}}, \overrightarrow{d_{\mathsf{Y}_j}})$ such that

$$\mu\text{CRL} \cup \mathit{DATA} \cup \overline{G} \vdash \mathsf{X}_1(\overrightarrow{dt}) = w_1(\overrightarrow{dt'}),$$

and for all $j = 1, \ldots, k$,

$$\mu\text{CRL} \cup \mathit{DATA} \cup \overline{G} \vdash w_j(\overrightarrow{d_{\mathsf{Y}_j}}) = u_j(\vec{w}, \overrightarrow{d_{\mathsf{Y}_j}}).$$

Here *DATA* represents the specification of the data types involved in both systems and in $\overrightarrow{dt}$ and $\overrightarrow{dt'}$. Furthermore, $\overline{G}$ refers to the setting where the equations in G are considered to define additional axioms.

We continue with an example. As before, let $G = \{\mathsf{X}(b : \mathit{Bool}) = \mathsf{a}(b) \cdot \mathsf{X}(\neg b)\}$ and, with *Nat* a specification of the naturals, $H = \{\mathsf{Y}(n : \mathit{Nat}) = \mathsf{a}(\mathit{even}(n)) \cdot \mathsf{Y}(S(n))\}$. We show that

$$\big(\mathsf{X}(\mathbf{t}), G\big) \Rightarrow_c \big(\mathsf{Y}(0), H\big)$$

by choosing $w(n) = \mathsf{X}(\mathit{even}(n))$. In this case we need to show that $\mathsf{X}(\mathbf{t}) = w(0)$ (this follows from $\mathit{even}(0) = \mathbf{t}$, which we assume to be derivable from *DATA*) and that $\mathsf{X}(\mathit{even}(n)) = \mathsf{a}(\mathit{even}(n)) \cdot \mathsf{X}(\mathit{even}(S(n)))$. This latter identity follows from $\mathsf{X}(b) = \mathsf{a}(b) \cdot \mathsf{X}(\neg b)$ and the necessarily derivable data identity $\mathit{even}(S(n)) = \neg \mathit{even}(n)$. If we assume the existence of a function $f : \mathit{Bool} \to \mathit{Nat}$, defined by $f(\mathbf{t}) = 0$ and $f(\mathbf{f}) = 1$ (where **f** stands for "false"), we can also prove that

$$\big(\mathsf{X}(b), G\big) \Rightarrow_c \big(\mathsf{Y}(f(b)), H\big)$$

using the same term $w(n)$ and the data identities $\mathit{even}(f(b)) = b$ and $\mathit{even}(S(f(b))) = \neg b$, both of which seem reasonable. We do not have any of the reverse implications: consider the model with carrier set *Nat*, in which $\mathsf{a}(b)$ is interpreted as 1, and sequential composition as $+$. Then $\mathsf{Y}(0)$ has many solutions, whereas $\mathsf{X}(\mathbf{t})$ has none.

Theorem 3. *Let* $(\mathsf{X}_1(\overrightarrow{dt}), G)$ *and* $(\mathsf{Y}_1(\overrightarrow{dt'}), H)$ *be process definitions over* μCRL. *If* $(\mathsf{X}_1(\overrightarrow{dt}), G) \Rightarrow_c (\mathsf{Y}_1(\overrightarrow{dt'}), H)$, *then* $(\mathsf{X}_1(\overrightarrow{dt}), G) \preceq (\mathsf{Y}_1(\overrightarrow{dt'}), H)$.

Proof. Let $\mathcal{M}$ be a model for μCRL with data theories for the data types used in G, H, $\overrightarrow{dt}$, and $\overrightarrow{dt'}$, and let $w_j(\overrightarrow{d_j})$ be such that $(\mathsf{X}_1(\overrightarrow{dt}), G) \Rightarrow_c (\mathsf{Y}_1(\overrightarrow{dt'}), H)$. Now assume that G has a solution in $\mathcal{M}$. So for $i = 1, \ldots, n$ there are $f_i(\overrightarrow{d_i})$ such that $\mathcal{M} \models f_i(\overrightarrow{d_i}) = t_i(\vec{f}, \overrightarrow{d_i})$ where f_i is the interpretation of X_i. Then, by $\mathcal{M} \models f_1(\overrightarrow{dt}) = w_1(\overrightarrow{dt'})$ and $\mathcal{M} \models w_j(\overrightarrow{d_{\mathsf{Y}_j}}) = u_j(\vec{w}, \overrightarrow{d_{\mathsf{Y}_j}})$ the theorem follows. □

The question whether our definition of $\Rightarrow_c$ is complete in the sense that it *characterizes* the preservation of solutions (in all models) is hard to answer. In general, proof principles such as induction are intertwined with such a question (cf. the example below).

We define *conditional equivalence*, notation

$$\big(\mathsf{X}_1(\overrightarrow{dt}), G\big) =_c \big(\mathsf{Y}_1(\overrightarrow{dt'}), H\big),$$

by requiring conditional implications in both directions. Indeed, conditional equivalence is an equivalence relation.

An example. Let *NAT* be a specification of the naturals comprising induction schemes (see, e.g., [15]), and let G and H be the following systems of equations:

$$G = \left\{ \begin{array}{l} \mathsf{X}_1(n : \mathit{Nat}) = \big(\mathsf{a} \cdot \mathsf{X}_2(n-1) + \mathsf{X}_1(n-1)\big) \lhd n > 0 \rhd \mathsf{a}, \\ \mathsf{X}_2(n : \mathit{Nat}) = \mathsf{a} \cdot \mathsf{X}_2(n-1) \lhd n > 0 \rhd \mathsf{a} \end{array} \right\}$$

$$H = \left\{ \begin{array}{l} \mathsf{Y}_1(n : \mathit{Nat}) = \big(\mathsf{a} + \mathsf{Y}_1(n-1) \cdot \mathsf{a}\big) \lhd n > 0 \rhd \mathsf{a}, \\ \mathsf{Y}_2(n : \mathit{Nat}) = \mathsf{a} \cdot \mathsf{Y}_2(n-1) \lhd n > 0 \rhd \mathsf{a} \end{array} \right\}$$

We show that $(\mathsf{X}_i(n), G) =_c (\mathsf{Y}_i(n), H)$ for $i = 1, 2$. For both implications $\Rightarrow_c$ and $\Leftarrow_c$ we choose the terms w_1 and w_2 to be trivial, namely, in the first

case $w_1(n) = \mathsf{X}_1(n)$, $w_2(n) = \mathsf{X}_2(n)$, and in the second case $w_1(n) = \mathsf{Y}_1(n)$, $w_2(n) = \mathsf{Y}_2(n)$. The proofs then reduce to showing that

$$\mu\mathrm{CRL} \cup \mathit{NAT} \cup \overline{G} \vdash \mathsf{X}_1(n) = \big(\mathsf{a} + \mathsf{X}_1(n-1) \cdot \mathsf{a}\big) \lhd n > 0 \rhd \mathsf{a}$$

and

$$\mu\mathrm{CRL} \cup \mathit{NAT} \cup \overline{H} \vdash \mathsf{Y}_1(n) = \big(\mathsf{a} \cdot \mathsf{Y}_2(n-1) + \mathsf{Y}_1(n-1)\big) \lhd n > 0 \rhd \mathsf{a}$$

First we show by induction on n that $\mu\mathrm{CRL} \cup \mathit{NAT} \cup \overline{G} \vdash \mathsf{a} \cdot \mathsf{X}_2(n) = \mathsf{X}_2(n) \cdot \mathsf{a}$. The case $n = 0$ is trivial. In the other case we get:

$$\mathsf{a} \cdot \mathsf{X}_2(n+1) = \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{X}_2(n) \overset{\text{IH}}{=} \mathsf{a} \cdot \mathsf{X}_2(n) \cdot \mathsf{a} = \mathsf{X}_2(n+1) \cdot \mathsf{a}$$

Similarly, $\mu\mathrm{CRL} \cup \mathit{NAT} \cup \overline{H} \vdash \mathsf{a} \cdot \mathsf{Y}_2(n) = \mathsf{Y}_2(n) \cdot \mathsf{a}$.

Next, we show by induction on n that $\mu\mathrm{CRL} \cup \mathit{NAT} \cup \overline{G} \vdash \mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{X}_1(n) = \mathsf{a} + \mathsf{X}_1(n) \cdot \mathsf{a}$. Again, for $n = 0$ we get $\mathsf{a} \cdot \mathsf{a} + \mathsf{a}$ in both sides. In the other case we get:

$$\begin{aligned}&\mathsf{a} \cdot \mathsf{X}_2(n+1) + \mathsf{X}_1(n+1)\\ &\quad = \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{X}_1(n)\end{aligned}$$

and

$$\begin{aligned}&\mathsf{a} + \mathsf{X}_1(n+1) \cdot \mathsf{a}\\ &\quad = \mathsf{a} + \big(\mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{X}_1(n)\big) \cdot \mathsf{a}\\ &\quad = \mathsf{a} + \mathsf{a} \cdot \mathsf{X}_2(n) \cdot \mathsf{a} + \mathsf{X}_1(n) \cdot \mathsf{a}\\ &\quad = \big(\mathsf{a} + \mathsf{X}_1(n) \cdot \mathsf{a}\big) + \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{X}_2(n)\\ &\quad \overset{\text{IH}}{=} \mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{X}_1(n) + \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{X}_2(n)\\ &\quad = \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{a} \cdot \mathsf{X}_2(n) + \mathsf{X}_1(n)\end{aligned}$$

Next, we show that a similar identity is derivable from H, namely $\mu\mathrm{CRL} \cup \mathit{NAT} \cup \overline{H} \vdash \mathsf{a} \cdot \mathsf{Y}_2(n) + \mathsf{Y}_1(n) = \mathsf{a} + \mathsf{Y}_1(n) \cdot \mathsf{a}$. Again, the case $n = 0$ is trivial, and in the other case we have:

$$\begin{aligned}&\mathsf{a} \cdot \mathsf{Y}_2(n+1) + \mathsf{Y}_1(n+1)\\ &\quad = \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{Y}_2(n) + \mathsf{a} + \mathsf{Y}_1(n) \cdot \mathsf{a}\\ &\quad = \mathsf{a} + \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{Y}_2(n) + \mathsf{Y}_1(n) \cdot \mathsf{a}\end{aligned}$$

and

$$\begin{aligned}&\mathsf{a} + \mathsf{Y}_1(n+1) \cdot \mathsf{a}\\ &\quad = \mathsf{a} + \big(\mathsf{a} + \mathsf{Y}_1(n) \cdot \mathsf{a}\big) \cdot \mathsf{a}\\ &\quad \overset{\text{IH}}{=} \mathsf{a} + \big(\mathsf{a} \cdot \mathsf{Y}_2(n) + \mathsf{Y}_1(n)\big) \cdot \mathsf{a}\\ &\quad = \mathsf{a} + \mathsf{a} \cdot \mathsf{Y}_2(n) \cdot \mathsf{a} + \mathsf{Y}_1(n) \cdot \mathsf{a}\\ &\quad = \mathsf{a} + \mathsf{a} \cdot \mathsf{a} \cdot \mathsf{Y}_2(n) + \mathsf{Y}_1(n) \cdot \mathsf{a}\end{aligned}$$

The last two identities imply the conditional equality we are proving.

It is important to note that the equation for Y_2 is not needed for the preservation of solutions. If H' is the system H with only the first equation, then $(\mathsf{Y}_1(n), H') \preceq (\mathsf{Y}_1(n), H)$. This is due to the fact that the equation for Y_2 has a solution in each model of $\mu\mathrm{CRL}$ and NAT, namely the function $f : \mathit{Nat} \to P$ such that $f(0) = \mathsf{a}$ and $f(n+1) = \mathsf{a} \cdot f(n)$. This differs with the necessity of the equation for Y_2 in the last example of Section 2.

4. Conclusions

We have defined equivalence between (recursive) process definitions over BPA, and showed that this equivalence emerges from a logical characterization of the preservation of solutions. Furthermore, we have presented a straightforward generalization of this equivalence to the data-parametric setting of μCRL. The main motivation to write this paper is to show that reasoning about equality (or implication) between solutions of systems of equations can be separated from considerations about unique solutions. As a consequence, transformation algorithms (e.g., for tools as now are available for μCRL) can be easily proved correct (cf. [14]).

We note that our counter examples concern models that do not satisfy the *left cancellation* property, i.e.,

$$\mathsf{a} \cdot x = \mathsf{b} \cdot y \to \mathsf{a} = \mathsf{b} \wedge x = y,$$

where a and b are actions. The left cancellation property holds in common process semantics.

Future work could focus on defining transformations of recursive systems in process algebras and μCRL that preserve the equivalence or the implication defined in this paper. Such transformations were studied in [5] in a general setting, but applying them to process algebras may be useful for optimization and verification purposes.

Acknowledgements

We thank Bas Luttik for careful proofreading of the manuscript and useful comments.

References

[1] J.C.M. Baeten, J.A. Bergstra, J.W. Klop, On the consistency of Koomen's fair abstraction rule, Theoret. Comput. Sci. 51 (1/2) (1987) 129–176.

[2] J.C.M. Baeten, W.P. Weijland, Process Algebra, Cambridge Tracts in Theoretical Computer Science, Vol. 18, Cambridge Univ. Press, Cambridge, 1990.

[3] D.B. Benson, I. Guessarian, Algebraic solutions to recursion schemes, J. Comput. System Sci. 35 (1987) 365–400.

[4] J.A. Bergstra, I. Bethke, A. Ponse, Process algebra with iteration and nesting, Comput. J. 37 (4) (1994) 243–258.

[5] B. Courcelle, Equivalences and transformations of regular systems — applications to recursive program schemes and grammars, Theoret. Comput. Sci. 42 (1986) 1–122.

[6] B. Courcelle, Recursive applicative program schemes, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. B, Elsevier, Amsterdam, 1990, pp. 459–492, Chapter 9.

[7] W.J. Fokkink, Introduction to Process Algebra, Texts in Theoretical Computer Science, An EATCS Series, Springer, Berlin, 2000.

[8] R.J. van Glabbeek, The linear time–branching time spectrum, II. The semantics of sequential systems with silent moves, Manuscript. Preliminary version available by ftp at ftp://boole.stanford.edu/pub/spectrum.ps.gz, 1993. Extended abstract, in: E. Best (Ed.), Proc. CONCUR'93, 4th Internat. Conf. on Concurrency Theory, Hildesheim, Germany, Lecture Notes in Comput. Sci., Vol. 715, Springer, Berlin, August 1993, pp. 66–81.

[9] R.J. van Glabbeek, The linear time–branching time spectrum. I. The semantics of concrete, sequential processes, in: J.A. Bergstra, A. Ponse, S.A. Smolka (Eds.), Handbook of Process Algebra, Chapter 1. Elsevier, to appear. Available at http://boole.stanford.edu/pub/spectrum1.ps.gz.

[10] J.F. Groote, B. Lisser, Computer assisted manipulation of algebraic process specifications, Technical Report, CWI, Amsterdam, To appear.

[11] J.F. Groote, S.P. Luttik, Undecidability and completeness results for process algebras with alternative quantification over data, Report SEN-R9806, CWI, The Netherlands, July 1998. Available from http://www.cwi.nl/~luttik/; submitted for publication.

[12] J.F. Groote, A. Ponse, Proof theory for μCRL: A language for processes with data, in: D.J. Andrews, J.F. Groote, C.A. Middelburg (Eds.), Semantics of Specification Languages, Workshop in Computing Series, Springer, Berlin, 1994, pp. 232–251.

[13] J.F. Groote, A. Ponse, The syntax and semantics of μCRL, in: A. Ponse, C. Verhoef, S.F.M. van Vlijmen (Eds.), Algebra of Communicating Processes 1994, Workshop in Computing Series, Springer, Berlin, 1995, pp. 26–62.

[14] J.F. Groote, A. Ponse, Y.S. Usenko, Linearization in parallel pCRL, Technical Report SEN-R0019, CWI, July 2000.

[15] J.F. Groote, J.J. van Wamel, Algebraic data types and induction in μCRL, Technical Report P9409, University of Amsterdam, Programming Research Group, 1994.

[16] R. Milner, Communication and Concurrency, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[17] J.R. Shoenfield, Mathematical Logic, Addison-Wesley, Reading, MA, 1967.