

The Value-Passing Calculus

YUXI FU

Shanghai Jiao Tong University

A value-passing calculus is a process calculus in which the contents of communications are values chosen from some data domain, and the propositions appearing in the conditionals are formulas constructed from a logic. Previous studies treat the domain models, as well as the logic theories, as unspecified oracles. The open-ended approach leaves open some fundamental issues unanswered. The paper provides a more formal account of the value-passing calculi. The new treatment is self-contained in that the logic theory a value-passing calculus refers to is formally defined. A value-passing calculus consists of a first order theory with boolean completeness and an operational model that makes use of the terms and the boolean expressions of the theory. A systematic investigation into the theory of the value-passing calculi is carried out. A particular value-passing calculus, \mathbb{VPC} , is shown to be the least expressive among all the Turing complete value-passing calculi.

Categories and Subject Descriptors: ... [...]: ...

General Terms: Theory, Languages

Additional Key Words and Phrases: Value-passing calculus, bisimulation, axiomatization

1. INTRODUCTION

Process calculus offers one approach to study interactions between computing objects. The process models can be classified by the type of the entities exchanged over interactions. The pioneering process calculus, the CCS of Milner [1989a], abstracts away the contents of communications. For this reason, it serves as a benchmark model for process. Although the pure CCS falls short of being a very interesting model from the point of view of expressiveness [Fu 2011b], the basic theory of CCS does generalize to many process calculi. The value-passing CCS [Milner 1989a] adds to CCS the capacity to pass data values between processes. In addition to the simple mechanism of synchronization, communications of values render it possible to control the interaction flow by testing the received data values. This additional control power significantly enhances the expressive power of the value-passing calculi. The name-passing calculus of Milner et al. [1992], the π -calculus, adopts the policy that the messages sent and received in communications can only be channel names. The exclusive focus on the names has achieved both simplicity and expressiveness. It is difficult to extend the name-passing mechanism to get a strictly

Author's postal address: BASICS, Department of Computer Science, Shanghai Jiaotong University, 800 Dong Chuan Road, Shanghai 200240, China.

Author's email address: fu-yx@cs.sjtu.edu.cn.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2011 ACM 0000-0000/2011/0000-0001 \$5.00

stronger model. The process-passing calculi, or the higher order calculi [Sangiorgi 1993; Thomsen 1989; 1993; 1995], have typically processes as the contents of communications. This seemingly powerful communication mechanism turns out to be much less expressive than the π -calculus [Fu 2011b]. To make use of what have been received through communications, variables (value variables, name variables, process variables) have to be introduced to act as placeholders.

From a logic point of view, the name-passing calculi and the process-passing calculi are preferable since they are closed models in the sense that the syntax and the semantics of these calculi are independent of any models or logics. In contrary the value-passing calculi, with strong motivation from practice, are distinguished by the fact that they must refer to an ‘oracle’, be it a domain model or a logic theory. The traditional treatment to the value-passing calculi are not self-contained. The attentions have largely been on the process aspect of the story. The oracles have never been formally defined. The under-specification of the value domain is not welcome from a foundational viewpoint, nor is it really useful in practice.

There are serious drawbacks about the lightweight treatment of the oracle models/logics. We shall mention three of them.

- (1) Deep theoretical investigations are inevitably hindered by the open-ended approach. For example there is no way to compare the expressiveness of a value-passing calculus to a different interaction model, say the π -calculus. An encoding of the former into the latter would require that the value terms and the logic expressions be fully specified.
- (2) For the same reason there is no way to implement a value-passing calculus, no matter what is meant by an implementation.
- (3) An equivalence checking algorithm is out of the question since the existence of such an algorithm depends on the algorithmic aspect of the oracle model and/or the oracle logic, which is not available in the open-ended framework.

Apart from these deficiencies, there are also a number of related subtleties that have to be taken into account when designing a value-passing calculus. Let’s illustrate these points by scrutinizing the process $M(x)$ given by the following equation.

$$M(x) = \text{if } \varphi(x) \text{ then } \bar{a}(f(t)).\mathbf{0} \text{ else } M(x + 1). \quad (1)$$

There are at least five questions one may ask about the process defined in (1). The first is concerned with the nature of the oracle. Where are $\varphi(x)$ and t coming from? There are basically two answers.

- (1) The first answer is model theoretical. The value term t and the logical expression $\varphi(x)$ are constructed from the elements of the universe of a model, the functions and the relations on the universe, and the variables that range over the universe. The logical expression φ must be evaluated in the model before process (1) fires an action. If φ contains free variables, the evaluation is done with respect to an assignment. Under the model theoretical interpretation, one expects that the process expression *if* $y = y$ *then* P *else* Q can be immediately put into action since it is fireable under all assignments. On the other hand the behavior of the process expression *if* $y = z$ *then* P *else* Q depends on particular assignments.

- (2) The second answer is proof theoretical. The value term t is defined inductively from a vocabulary and the logical expression φ is legitimate in a first order theory on top of the vocabulary. The process expression in (1) can fire if φ is a theorem of the theory.

In principle, a proof theoretical approach to oracle design should definitely be preferred. The point is that all implementations of an oracle are proof theoretical in nature. The set of the statements true in a model is typically not recursive enumerable [Rogers 1987]. But an implemented system can only generate a recursive enumerable set of theorems. Moreover the proof theoretical approach fits very well with the operational nature of process calculi.

The second question is about the expressiveness of the logical expressions appearing in the value-passing processes. According to Gödel's Incompleteness Theorem, there is in general no effective procedure to decide the theoremhood of a formula. If a theory is reasonably expressive, there exists some first order logical expression ψ such that neither ψ nor $\neg\psi$ is provable. This is definitely unacceptable from a programming point of view. To avoid the embarrassment caused by Gödel's Incompleteness Theorem, the logical expressions admitted in a value-passing calculus are confined to the quantifier free formulas. But this restriction does not entirely eliminate the problem on decidability. If φ contains the free variables x_1, \dots, x_n , then proving the theoremhood of φ is equivalent to proving the theoremhood of $\forall x_1 \dots \forall x_n. \varphi$, which is normally undecidable. There are other situations, for example in the equivalence checking algorithm design, where formulas with free variables must be dealt with. So we need to look for the logical theories with the nice property that the theoremhood of the quantifier free formulas are decidable. This is a starting point to regard a value-passing calculus as a programming language.

The third question is about the expressive power of the value terms admitted in a value-passing calculus. Our value-passing calculus would be too strong if the f appearing in (1) could be a non-computable function. The Church-Turing Thesis asserts that all the functions definable in a value-passing calculus are computable functions if the functions produced by the oracles are computable. It would be reasonable to disown those oracles that are capable of delivering non-computable functions. Now here is the twist, if all the recursive functions can be defined within a value-passing calculus, is it necessary to have an oracle that produces functions short-cutting the role of the definable functions? A negative answer would imply that the oracle should only supply constructors for the value terms; it should not introduce any functions that compute on the value terms.

The fourth question is about the functional separation between the calculi and the oracles. The standard semantics of the value-passing calculi demands that the value term $f(t)$ must be calculated to a canonical value before it is exported at the name a . This additional calculating machinery is not very appealing from the point of view of an interaction model. In process calculi all calculations should be achieved by interactions. In other words, the procedure of the calculation should be explicitly specified in a process, not implicitly done by an oracle. It is interesting to notice that the negative answer to the previous question is also an answer to the present question since it trivializes the issue.

The fifth question is about the level of abstraction of the value-passing calculi.

If n is the least natural number such that $\varphi(n)$ holds according to an oracle, then $M(0)$ emits $f(t)$ at the channel named a ; otherwise $M(0)$ is inactive. If $M(0)$ ever interacts, it need to consult the oracle for a finite number of times. If $M(0)$ never interacts, it must consult the oracle to evaluate $\varphi(0), \varphi(1), \varphi(2), \dots$ consecutively in a non-stop fashion. This phenomenon is familiar to higher order programming languages, it is however alien to process calculi. The process $M(0)$ has abstracted away too many computational and interactional activities, the explicit descriptions of which are precisely what is expected of a process calculus. If $M(0)$ never interacts, the execution of $M(0)$ in a higher order programming language would results in a loop, a computational behaviour that is quite different from that of the command *skip*. However in the standard semantics of the value-passing calculi $M(0)$ is strongly bisimilar to $\mathbf{0}$. This is a strong argument that the process defined in (1) should be banned.

The above discussions lead to the following design principle. A value-passing calculus consists of a first order theory and a labeled transition system. The former provides both the value terms and the boolean expressions. The latter defines the semantics of the value-passing processes. The first order theory and the labeled transition system are designed by taking the followings into consideration.

- (1) To make sure that the first order theory provides a right support to the operational semantics, the theory is supposed to be complete for the set of the quantifier free theorems.
- (2) To guarantee that the first order theory does not interfere with the computations/interactions defined by the labeled transition system, the formulas admissible in a value-passing calculus should only contain constructors that generate the universe of values; they should not contain any functions that compute on the elements of the universe.
- (3) To keep the value-passing calculi at the right level of abstraction, it is better not to define recursions by recursive definitions parameterized over value variables. In both theory and practice the replication operator is sufficient.

The aim of this paper is to develop a rigid theory of the value-passing calculi designed with the above remarks in mind. The theory is general enough so that it can be readily applied to any particular value-passing calculus. It is also formal enough so that many questions about the value-passing calculi can be addressed.

The paper is structured as follows. Section 2 reviews the relevant terminologies in mathematical logic. A boolean completeness result is proved, which provides the basis for a particular value-passing calculus studied in the paper. Section 3 studies the operational and the observational semantics of the value-passing calculi. The absolute equality is proposed as *the* equality for all value-passing calculi. Section 4 takes a look at symbolic approximation to the absolute equality. The symbolic equivalence reveals more structural aspects of the observation theory. Section 5 provides a proof system for the finite terms. The system is independent of any particular value-passing calculus. Section 6 discusses the expressiveness requirement for the value-passing calculi. Section 7 applies the methodology to the value-passing calculus \mathbb{VPC} defined over the Peano arithmetics. The model is shown to be minimal among all the Turing complete value-passing calculi. Section 8 concludes with discussions on future research.

2. LOGIC AND MODEL

In this section we review some preliminary concepts from mathematical logic necessary to carry out our program.

2.1 Theory and Theorem

Let \mathbf{N} be the set of the natural numbers. A *vocabulary* $\Sigma = (F, R, a)$ consists of two disjoint nonempty countable sets and one function: F is the set of *function symbols*; R is the set of *relation symbols*; and $a : F \cup R \rightarrow \mathbf{N}$ is the *arity function* that maps an element of F onto a natural number and an element of R onto a nonzero natural number. A symbol in $F \cup R$ is *k-ary* if it is mapped onto k under a . A *constant* is a 0-ary function symbol. It is always assumed that F contains at least one constant and that R contains the binary relation $=$ called *equality relation*.

For each vocabulary Σ there is a countable set $V_\Sigma = \{x, y, z, \dots\}$ of Σ -variables. The set T_Σ of Σ -terms, ranged over by r, s, t , is inductively defined as follows:

- $V_\Sigma \subseteq T_\Sigma$.
- If f is a k -ary function symbol and t_1, \dots, t_k are Σ -terms, then $f(t_1, \dots, t_k)$ is a Σ -term.

Clearly T_Σ contains all the constants. A Σ -term is *closed* if it does not contain any Σ -variable, it is *open* otherwise. The set of the closed Σ -terms is denoted by T_Σ^0 .

The set E_Σ of Σ -expressions, ranged over by ϕ, φ, ψ , is inductively defined as follows:

- The logical *false* \perp is a Σ -expression.
- If r is a k -ary relation symbol and t_1, \dots, t_k are Σ -terms, then $r(t_1, \dots, t_k)$ is an *atomic* Σ -expression.
- If φ, ψ are Σ -expressions, then $\varphi \Rightarrow \psi$ is a Σ -expression.
- If ϕ is a Σ -expression and x is a Σ -variable, then $\forall x.\phi$ is a Σ -expression, where \forall is the *universal quantifier*.

The Σ -variable x in $\forall x.\phi$ is *bound*. A Σ -variable is *free* if it is not bound. A Σ -sentence is a Σ -expression that does not contain any free Σ -variables. The set of the Σ -sentences is denoted by E_Σ^0 . A *boolean* Σ -expression is a quantifier free Σ -expression. A *boolean* Σ -sentence is a quantifier free Σ -sentence. In sequel we shall freely use the derived propositional connectives $\top, \neg, \wedge, \vee, \Leftrightarrow$ and the existential quantifier \exists .

The *first order logic over* Σ is the recursive set of the *first order logical axioms* defined in Fig. 1. The axiom schema BA actually stands for a recursive set of *boolean axioms*, each obtained from a boolean tautology by replacing all the propositional variables in the tautology by some atomic Σ -expressions. The EQ-axioms are about the equivalence property, while the CG-axioms formalize the congruence property. The FO-axioms state the provability of the universally quantified Σ -expressions. In both BA and FO1 the standard meta operation *substitution* is used.

Given a recursive enumerable set Γ of Σ -expressions, a *proof* of ψ from Γ is a finite sequence (ϕ_1, \dots, ϕ_n) of Σ -expressions such that ϕ_n is ψ and one of the following properties holds for each $i \leq n$:

- ϕ_i is a logical axiom;

BA	$P\{r_1(t_1^1, \dots, t_{k_1}^1)/X_1, \dots, r_n(t_1^n, \dots, t_{k_n}^n)/X_n\}$	P a tautology
EQ1	$t = t$	
EQ2	$s = t \Rightarrow t = s$	
EQ3	$r = s \wedge s = t \Rightarrow r = t$	
CG1	$t_1 = t'_1 \wedge \dots \wedge t_k = t'_k \Rightarrow f(t_1, \dots, t_k) = f(t'_1, \dots, t'_k)$	f a k -ary function
CG2	$t_1 = t'_1 \wedge \dots \wedge t_k = t'_k \Rightarrow r(t_1, \dots, t_k) \Rightarrow r(t'_1, \dots, t'_k)$	r a k -ary relation
FO1	$\forall x. \phi \Rightarrow \phi\{t/x\}$	
FO2	$\phi \Rightarrow \forall x. \phi$	x not in ϕ
FO3	$(\forall x. (\varphi \Rightarrow \psi)) \Rightarrow (\forall x. \varphi \Rightarrow \forall x. \psi)$	

Fig. 1. Logical Axioms of Σ .

— $\phi_i \in \Gamma$;

— There are two Σ -expressions φ and $\varphi \Rightarrow \phi_i$ in the proof $(\phi_1, \dots, \phi_{i-1})$.

A Σ -expression ψ is a Γ -*theorem*, notation $\Gamma \vdash \psi$, if there is a proof of ψ from Γ , and it is a *theorem*, notation $\vdash \psi$, if Γ is the empty set. A set Γ of Σ -expressions is *inconsistent* if $\Gamma \vdash \perp$; it is *consistent* otherwise. We sometimes write $s =_{\Gamma} t$ for $\Gamma \vdash s = t$.

A *first order theory over Σ* is a consistent recursive enumerable set Th of Σ -sentences, the elements of Th being the *nonlogical axioms*.

2.2 Truth and Boolean Completeness

Suppose $\Sigma = (\mathbf{F}, \mathbf{R}, \mathbf{a})$ is a vocabulary. A *model* $\mathfrak{M} = (U, \llbracket _ \rrbracket)$ of type Σ consists of a set U , the *universe* of the model, and a map $\llbracket _ \rrbracket$, the *interpretation function*. The domain of the interpretation function is $\mathbf{F} \cup \mathbf{R}$ and the range of the interpretation function is a set of functions and relations on U . More precisely,

— A k -ary function symbol f is interpreted by a k -ary function $\llbracket f \rrbracket$ on U ;

— A k -ary relation symbol r is interpreted by a k -ary relation $\llbracket r \rrbracket$ on U .

In particular a constant is interpreted by an element of U and the equality relation $=$ is interpreted by the binary relation $\{(u, u) \mid u \in U\}$.

An *assignment* is a function $\rho : \mathbf{V}_{\Sigma} \rightarrow U$ that assigns an element of U to each Σ -variable. Given an assignment ρ , a Σ -term t can be interpreted in the model \mathfrak{M} by the following structural induction:

$$\llbracket t \rrbracket_{\rho} \stackrel{\text{def}}{=} \begin{cases} \rho(x), & \text{if } t \equiv x \\ \llbracket f \rrbracket(\llbracket t_1 \rrbracket_{\rho}, \dots, \llbracket t_k \rrbracket_{\rho}), & \text{if } t \equiv f(t_1, \dots, t_k) \end{cases}$$

Using the assignment ρ we may define the *satisfaction* relation between the model \mathfrak{M} and a Σ -expression. We write $\mathfrak{M} \models_{\rho} \psi$ if one of the followings is true.

— If $\psi \equiv r(t_1, \dots, t_k)$, then $\mathfrak{M} \models_{\rho} \psi$ if $(\llbracket t_1 \rrbracket_{\rho}, \dots, \llbracket t_k \rrbracket_{\rho}) \in \llbracket r \rrbracket$.

— If $\psi \equiv \varphi \Rightarrow \phi$, then $\mathfrak{M} \models_{\rho} \psi$ if $\mathfrak{M} \models_{\rho} \phi$ whenever $\mathfrak{M} \models_{\rho} \varphi$.

— If $\psi \equiv \forall x. \varphi$, then $\mathfrak{M} \models_{\rho} \psi$ if $\mathfrak{M} \models_{\rho[x \leftarrow u]} \varphi$ for every $u \in U$, where the assignment $\rho[x \leftarrow u]$ differs from ρ *only* in that the former assigns u to x .

The satisfaction for the other forms of the Σ -expressions is defined in the standard manner. The model \mathfrak{M} satisfies ψ , notation $\mathfrak{M} \models \psi$, if $\mathfrak{M} \models_{\rho} \psi$ for every ρ .

PA1	$\forall x.(\mathfrak{s}(x) \neq 0)$
PA2	$\forall xy.(\mathfrak{s}(x) = \mathfrak{s}(y) \Rightarrow x = y)$
PA3	$\forall x.(x = 0 \vee \exists y.\mathfrak{s}(y) = x)$
PA4	$\forall x.(x < \mathfrak{s}(x))$
PA5	$\forall xy.(x < y \Rightarrow \mathfrak{s}(x) \leq y)$
PA6	$\forall xy.(\neg(x < y) \Leftrightarrow y \leq x)$
PA7	$\forall xy.((x < y) \wedge (y < z) \Rightarrow x < z)$

Fig. 2. First Order Peano Theory PA.

The Σ -expression ψ is said to be *valid*, notation $\models \psi$, if $\mathfrak{M} \models \psi$ for every model \mathfrak{M} of type Σ . Now suppose Γ is a recursive enumerable set of Σ -expressions. We write $\Gamma \models \psi$ if, for every model \mathfrak{M} of type Σ , $\mathfrak{M} \models \psi$ whenever $\mathfrak{M} \models \varphi$ for all $\varphi \in \Gamma$. We say that ψ is Γ -*valid* if $\Gamma \models \psi$.

For a given vocabulary Σ , we have introduced two kinds of judgement. The assertion $\Delta \models \psi$ is about the universal truth, whereas $\Delta \vdash \psi$ is about the power of the first order logic. A fundamental question asks if the first order logic is so strong that it can demonstrate all the universal truths about Σ . Gödel's Completeness Theorem confirms that this is actually the case.

For a *specific* model \mathfrak{M} of type Σ , is there a first order theory Th such that all the truths in \mathfrak{M} can be proved from Th ? The celebrated result of Gödel, the Incompleteness Theorem, asserts that such a first order theory does not exist if \mathfrak{M} is reasonably expressive. The fundamental barrier to achieving the completeness is that $\{\varphi \mid \varphi \in \mathbf{E}_\Sigma^0, \text{Th} \vdash \varphi\}$ cannot enumerate all the truths of \mathfrak{M} since the set of all the truths in \mathfrak{M} is normally not recursive enumerable. A completeness result is only possible for a proper subset of $\{\varphi \mid \varphi \in \mathbf{E}_\Sigma^0, \mathfrak{M} \models \varphi\}$. In particular it is possible for the set of the Σ -sentences in *universal prenex form*. A Σ -sentence is in the universal prenex form if it is of the shape $\forall x_1 \dots \forall x_n. \phi$ for some boolean Σ -expression ϕ . We say that a first order theory Th is *boolean complete* if, for every Σ -sentence ψ in the universal prenex form, either $\text{Th} \vdash \psi$ or $\text{Th} \vdash \neg\psi$.

PROPOSITION 2.1. *Suppose Th is a boolean complete first order theory over Σ . Then the Th -theoremhood of a boolean Σ -expression is decidable.*

PROOF. A theorem generator for Th is guaranteed to deliver an answer. \square

2.3 Peano Arithmetic

A well-known first order theory is *Peano Arithmetic* PA whose vocabulary Σ_{PA} contains one constant 0, one unary function symbol \mathfrak{s} , one binary relation symbol $<$ and of course the equality relation $=$. The arithmetic function symbols $+$, \times are omitted in our treatment for the reasons explained in Section 1. The axioms of PA are given in Fig. 2, in which $x \neq y$ stands for $\neg(x = y)$ and $x \leq y$ for $x = y \vee x < y$. For a natural number i , let \underline{i} denote the closed Σ_{PA} -term

$$\underbrace{\mathfrak{s}(\dots \mathfrak{s}(0) \dots)}_{i \text{ times}}$$

called *numerals*. Similarly we write $\mathfrak{s}^i(x)$ for the open Σ_{PA} -term

$$\underbrace{\mathfrak{s}(\dots \mathfrak{s}(x) \dots)}_{i \text{ times}}.$$

Axiom PA3 says that every Σ_{PA} -term is in one of the above two forms. Axiom PA2 is equivalent to saying that $\mathfrak{s}(x) = \mathfrak{s}(y)$ if and only if $x = y$. Axioms P4 and P5 (and its contrapositive) imply that

$$x < y \Leftrightarrow \mathfrak{s}(x) < \mathfrak{s}(y). \quad (2)$$

Axiom PA6 is equivalent to $x < y \vee x = y \vee y < x$, which is equivalent to

$$x \neq y \Leftrightarrow (x < y \vee y < x), \quad (3)$$

which is in turn equivalent to

$$x = y \Leftrightarrow (x < \mathfrak{s}(y) \wedge y < \mathfrak{s}(x)). \quad (4)$$

These exercises have made it clear that we could have introduced just one binary relation in PA. Using the PA-axioms one could derive the following PA-theorem

$$x = \underline{0} \vee x = \underline{1} \vee \dots \vee x = \underline{n} \vee \underline{n} < x,$$

from which one could prove the following PA-theorem

$$\varphi(x) \Leftrightarrow \varphi(\underline{0}) \wedge \varphi(\underline{1}) \wedge \dots \wedge \varphi(\underline{n}) \wedge (\underline{n} < x \Rightarrow \varphi(x)). \quad (5)$$

It is clear how to convert \mathbf{N} to a model of PA. Due to our particular formulation of the vocabulary Σ_{PA} , the following proposition admits an easy proof.

THEOREM 2.2. *PA is boolean complete.*

PROOF. First of all we prove that $\mathbf{N} \models \psi$ implies $\text{PA} \vdash \psi$ for all boolean Σ_{PA} -expressions. The proof is carried out by induction on the number of the free Σ_{PA} -variables appearing in ψ . We have already seen that the assertion holds for the boolean Σ_{PA} -sentences. For the induction step let x_0, x_1, \dots, x_n be the Σ_{PA} -variables in ψ . Since ψ is quantifier free, we may convert it into a conjunctive normal form $\psi_1 \wedge \dots \wedge \psi_l$. But then $\mathbf{N} \models \psi$ if and only if $\mathbf{N} \models \psi_j$ for all $j \in \{1, \dots, l\}$. So in view of (3), (4) and PA6 we may as well assume that ψ is a disjunction of atomic Σ_{PA} -expressions of the form $t_i < t'_i$. Using (2) we can convert all the terms that contain x_0 to $\mathfrak{s}^p(x_0)$ for some $p \geq 0$. Thus ψ can be rewritten to the following equivalent form

$$\left(\bigvee_{i=1}^{m_0} t_i^0 < \mathfrak{s}^p(x_0) \right) \vee \left(\bigvee_{i=1}^{m_1} \mathfrak{s}^p(x_0) < t_i^1 \right) \vee \psi' \quad (6)$$

such that x_0 does not appear in ψ' and $m_0 + m_1 \leq m$. Moreover we can assume that x_0 appears neither in t_i^0 nor in t_i^1 . Otherwise either we can remove $t_i^0 < \mathfrak{s}^p(x_0)$ (or $\mathfrak{s}^p(x_0) < t_i^1$), or we can conclude that $\text{PA} \vdash \psi$.

For Σ_{PA} -variables u, v, w , one has the following inferences:

$$\begin{aligned} u < v \wedge u < w &\Rightarrow (u < v \vee v < w) \wedge u < w, \\ v \leq u \wedge u < w &\Rightarrow v < w \wedge u < w \Rightarrow (u < v \vee v < w) \wedge u < w. \end{aligned}$$

It follows immediately that

$$u < w \Leftrightarrow (u < v \vee v < w) \wedge u < w.$$

An instance of the above equivalence is

$$t_k^0 < t_h^1 \Leftrightarrow (t_k^0 < \mathfrak{s}^p(x_0) \vee \mathfrak{s}^p(x_0) < t_h^1) \wedge t_k^0 < t_h^1, \quad (7)$$

where $k \in \{1, \dots, m_0\}$ and $h \in \{1, \dots, m_1\}$. Using the following theorem

$$\left(\bigwedge_{k=1..m_0}^{h=1..m_1} t_h^1 \leq t_k^0 \right) \vee \left(\bigvee_{k=1..m_0}^{h=1..m_1} t_k^0 < t_h^1 \right)$$

and the equivalence (7), it is easy to see that (6) is equivalent to

$$\varphi \vee \psi'',$$

where

$$\begin{aligned} \varphi &\equiv \left(\left(\bigvee_{i=1}^{m_0} t_i^0 < \mathfrak{s}^p(x_0) \right) \vee \left(\bigvee_{i=1}^{m_1} \mathfrak{s}^p(x_0) < t_i^1 \right) \right) \wedge \left(\bigwedge_{k=1..m_0}^{h=1..m_1} t_h^1 \leq t_k^0 \right), \\ \psi'' &\equiv \left(\bigvee_{k=1..m_0}^{h=1..m_1} t_k^0 < t_h^1 \right) \vee \psi'. \end{aligned}$$

We are now in a position to consider the satisfaction relation. Since $\mathbf{N} \models \psi$ and ψ is logically equivalent to $\varphi \vee \psi''$, it must be the case that

$$\mathbf{N} \models \varphi \vee \psi''.$$

To proceed, we need to consider, for each $i \in \{1, \dots, m_0\}$, the shape of the term t_i^0 . If t_i^0 is \underline{j} for some $j < p$, then $\text{PA} \vdash t_i^0 < \mathfrak{s}^p(x_0)$. In this case $\varphi \vee \psi''$ is equivalent to

$$\left(\bigwedge_{k=1..m_0}^{h=1..m_1} t_h^1 \leq t_k^0 \right) \vee \left(\bigvee_{k=1..m_0}^{h=1..m_1} t_k^0 < t_h^1 \right) \vee \psi',$$

which does not contain the Σ_{PA} -variable x_0 . By induction hypothesis we get that the above Σ_{PA} -expression is a PA -theorem. Hence $\text{PA} \vdash \psi$ in this case. It remains to consider the situation in which the following holds:

(†) Every closed Σ_{PA} -term $t \in \{t_i^0\}_{i=1..m_0}$ satisfies $\underline{p} \leq t$.

For each $x_i \in \{x_1, \dots, x_n\}$, we may apply (5) to get the following equivalence:

$$\varphi \vee \psi'' \Leftrightarrow (\varphi \vee \psi'')\{\underline{0}/x_i\} \wedge (\varphi \vee \psi'')\{\underline{1}/x_i\} \wedge \dots \wedge (\varphi \vee \psi'')\{\underline{p}/x_i\} \wedge (\underline{p} < x_i \Rightarrow (\varphi \vee \psi'')).$$

Now $\mathbf{N} \models \psi$ if and only if all the followings are valid.

$$\begin{aligned} \mathbf{N} &\models (\varphi \vee \psi'')\{\underline{0}/x_i\}, \\ &\vdots \\ \mathbf{N} &\models (\varphi \vee \psi'')\{\underline{p}/x_i\}, \\ \mathbf{N} &\models (\underline{p} < x_i \Rightarrow (\varphi \vee \psi'')). \end{aligned}$$

Since none of $(\varphi \vee \psi'')\{\underline{0}/x_i\}, \dots, (\varphi \vee \psi'')\{\underline{p}/x_i\}$ contains the variable x_i , we can apply the induction hypothesis to conclude that

$$\begin{aligned} \text{PA} &\vdash (\varphi \vee \psi'')\{\underline{0}/x_i\}, \\ &\vdots \\ \text{PA} &\vdash (\varphi \vee \psi'')\{\underline{p}/x_i\}. \end{aligned}$$

In other words,

$$\begin{aligned} \text{PA} &\vdash \psi\{\underline{0}/x_i\}, \\ &\vdots \\ \text{PA} &\vdash \psi\{\underline{p}/x_i\}. \end{aligned}$$

Continuing in this way, we may finally reduce our task to showing

$$\text{PA} \vdash (\underline{p} < x_1 \wedge \dots \wedge \underline{p} < x_n \Rightarrow (\varphi \vee \psi'')) \quad (8)$$

under the assumption

$$\mathbf{N} \models (\underline{p} < x_1 \wedge \dots \wedge \underline{p} < x_n \Rightarrow (\varphi \vee \psi'')). \quad (9)$$

Now let ρ be an arbitrary assignment such that

$$\mathbf{N} \models_{\rho} \underline{p} < x_1 \wedge \dots \wedge \underline{p} < x_n. \quad (10)$$

If $\mathbf{N} \not\models_{\rho} \left(\bigwedge_{k=1..m_0}^{h=1..m_1} t_h^1 \leq t_k^0 \right)$, then we must have

$$\mathbf{N} \models_{\rho} \psi''. \quad (11)$$

Suppose $\mathbf{N} \models_{\rho} \left(\bigwedge_{k=1..m_0}^{h=1..m_1} t_h^1 \leq t_k^0 \right)$. Let k be the following value

$$\min \{ \llbracket t_i^0 \rrbracket_{\rho} \}_{i \in \{1, \dots, m_0\}} - p.$$

This is well defined in view of (†) and (10). It is not difficult to see that

$$\mathbf{N} \not\models_{\rho[x_0 \leftarrow k]} \left(\bigvee_{i=1}^{m_0} t_i^0 < s^p(x_0) \right).$$

Moreover $\mathbf{N} \models_{\rho} \left(\bigwedge_{k=1..m_0}^{h=1..m_1} t_h^1 \leq t_k^0 \right)$ implies

$$\mathbf{N} \not\models_{\rho[x_0 \leftarrow k]} \left(\bigvee_{i=1}^{m_1} s^p(x_0) < t_i^1 \right).$$

It follows that $\mathbf{N} \not\models_{\rho[x_0 \leftarrow k]} \varphi$. We conclude that (11) holds for ρ anyway. It follows from the induction hypothesis that $\text{PA} \vdash \psi''$. Hence $\text{PA} \vdash \psi$.

Now suppose $\mathbf{N} \models \exists x. \psi$ such that x is the only variable in ψ . By definition $\mathbf{N} \models \psi\{\underline{n}/x\}$ for some \underline{n} . But then $\text{PA} \vdash \psi\{\underline{n}/x\}$. Hence $\text{PA} \vdash \exists x. \psi$. By induction, we may prove that $\mathbf{N} \models \exists x_1 \dots \exists x_n. \psi$ if and only if $\text{PA} \vdash \exists x_1 \dots \exists x_n. \psi$. Using this fact, it is easy to see that either $\text{PA} \vdash \forall x_1 \dots \forall x_n. \psi$ or $\text{PA} \vdash \neg(\forall x_1 \dots \forall x_n. \psi)$. \square

3. VALUE-PASSING CALCULUS

According to our discussions in Section 1, we shall focus on the value-passing calculi defined in terms of the first order boolean complete theories. Throughout this paper we assume that Th is a first order boolean complete theory of type Σ . The value-passing calculus defined on top of Th is denoted by VPC_{Th} . If Th is PA , the subscript in VPC_{Th} is omitted. The abbreviation will be justified in Section 7.

All process calculi are defined in terms of *names*. The set \mathcal{N} of names is ranged over by a, b, c, d, e, f, g, h . The set $\bar{\mathcal{N}}$ of conames is $\{\bar{a} \mid a \in \mathcal{N}\}$. Two meta theoretical operations are often used in postfix manner. A substitution is a partial map $\sigma : \mathbb{V}_{\Sigma} \rightarrow \mathbb{T}_{\Sigma}$ whose domain of definition is finite. An assignment is a partial map $\rho : \mathbb{V}_{\Sigma} \rightarrow \mathbb{T}_{\Sigma}^0$ whose domain of definition is cofinite. An assignment or a substitution is applicable to T if it is undefined on the bound variables of T . In sequel we shall always assume that ρ , or σ , is applicable to T whenever we write $T\rho$, or respectively $T\sigma$. The notations $\rho[x \leftarrow t]$ and $\sigma[x \leftarrow t]$ are understood in the standard interpretation. A substitution is often denoted explicitly by $\{t_1/x_1, \dots, t_n/x_n\}$.

The set $\mathcal{T}_{\text{VPC}_{\text{Th}}}$ of the VPC_{Th} -terms, ranged over by R, S, T and their decorated forms, is defined by the following BNF:

$$T := \sum_{i \in I} \varphi_i a(x).T_i \mid \sum_{i \in I} \varphi_i \bar{a}(t_i).T_i \mid T \mid T' \mid (c)T \mid \varphi T \mid !a(x).T \mid !\bar{a}(t).T,$$

where φ_i is a boolean Σ -expression, and I is a finite indexing set. The notation $\sum_{i \in \{1, \dots, n\}} \varphi_i \lambda_i.T_i$ stands for either $\sum_{i \in I} \varphi_i a(x).T_i$ or $\sum_{i \in I} \varphi_i \bar{a}(t_i).T_i$. The prefix $a(x)$ is an input primitive that binds the Σ -variable (henceforth just variable) x , and the prefix $\bar{a}(t)$ is an output primitive. We write $fv(\cdot)$, respectively $bv(\cdot)$, for the function that returns the set of the free variables, respectively the bound variables; and let $v(\cdot)$ be $fv(\cdot) \cup bv(\cdot)$. A VPC_{Th} -term is *closed* if it does not contain any free variables. Otherwise it is called an *open* VPC_{Th} -term. A closed VPC_{Th} -term is also called a VPC_{Th} -process. We write $\mathcal{P}_{\text{VPC}_{\text{Th}}}$ for the set of the VPC_{Th} -processes, ranged over by L, M, N, O, P, Q . For clarity we shall write $A(x, y) \stackrel{\text{def}}{=} T$ for instance to indicate that $A(x, y)$ is a shorthand for T with x, y as the only free variables. The notation $A(s, t)$ denotes $T\{s/x, t/y\}$. The *composition* $T \mid T'$ and the *localization* $(c)T$ are standard constructions. We write $\prod_{1 \leq i \leq n} T_i$ for the composition $T_1 \mid T_2 \mid \dots \mid T_n$. The VPC_{Th} -term $\sum_{i \in I} \varphi_i \lambda_i.T_i$ is a *conditional guarded choice*, and for each $i \in I$ the component $\varphi_i \lambda_i.T_i$ is a *summand*. The condition φ_i is often omitted if it is \top . There is essentially a unique guarded choice, noted $\mathbf{0}$, whose index set is the empty set. Often we write $\varphi_1 \lambda_1.T_1 + \dots + \varphi_n \lambda_n.T_n$ for $\sum_{i \in \{1, \dots, n\}} \varphi_i \lambda_i.T_i$. It should be remarked that in $\sum_{i \in I} \varphi_i \lambda_i.T_i$ the constructor is $\sum_{i \in I} \varphi_i \lambda_i \cdot$. The *conditional* φT is often written as *if* φ *then* T . The two leg conditional *if* φ *then* S *else* T can be defined by $\varphi S \mid \neg \varphi T$. The VPC_{Th} -term $!\nu.T$ is a *guarded replication*. We shall freely use the guarded fixpoint terms of the form $\mu X.E$ where X is a process variable whose occurrences in E are all under some prefixes. The fixpoint construction $\mu X.E$ can be encoded by $(c)(E\{c(z).\mathbf{0}/X\} \mid !\bar{c}(r).E\{c(z).\mathbf{0}/X\})$, where c is fresh and r is a closed term. So the guarded fixpoint operator does not introduce extra expressive power [Fu and Lu 2010]. A VPC_{Th} -term is *finite* if it does not contain any occurrences of the replication operator; it is a *finite control* term if it contains only the conditional guarded choice operator and the fixpoint operator.

Action

$$\frac{}{\sum_{i \in I} \varphi_i a(x).T_i \xrightarrow{a(t)} T_i\{t/x\}} \quad \begin{array}{l} i \in I, \\ t \in \mathbb{T}_{\Sigma}^0, \\ \text{Th} \vdash \varphi_i. \end{array} \quad \frac{}{\sum_{i \in I} \varphi_i \bar{a}(t_i).T_i \xrightarrow{\bar{a}(t_i)} T_i} \quad \begin{array}{l} i \in I, \\ t_i \in \mathbb{T}_{\Sigma}^0, \\ \text{Th} \vdash \varphi_i. \end{array}$$

Composition

$$\frac{S \xrightarrow{\lambda} S'}{S|T \xrightarrow{\lambda} S'|T} \quad \frac{S \xrightarrow{a(t)} S' \quad T \xrightarrow{\bar{a}(t)} T'}{S|T \xrightarrow{\tau} S'|T'}$$

Localization

$$\frac{T \xrightarrow{\lambda} T'}{(c)T \xrightarrow{\lambda} (c)T'} \quad c \text{ is not in } \lambda.$$

Condition

$$\frac{T \xrightarrow{\lambda} T'}{\varphi T \xrightarrow{\lambda} T'} \quad \text{Th} \vdash \varphi.$$

Recursion

$$\frac{}{!a(x).T \xrightarrow{a(t)} T\{t/x\} | !a(x).T} \quad t \in \mathbb{T}_{\Sigma}^0. \quad \frac{}{!\bar{a}(t).T \xrightarrow{\bar{a}(t)} T | !\bar{a}(t).T} \quad t \in \mathbb{T}_{\Sigma}^0.$$

Fig. 3. Concrete Semantics.

3.1 Concrete Semantics

In this section we define the so-called *concrete semantics*, which is given by the labeled transition system in Fig. 3, where the symmetric versions of two composition rules have been omitted. Although the semantics is defined for all \mathbb{VPC}_{Th} -terms, only the behaviors of the \mathbb{VPC}_{Th} -processes are completely characterized. If Th is the Peano Arithmetic PA , then under our semantics the \mathbb{VPC}_{Th} -term *if* $\underline{0} \leq x$ *then* $\bar{a}(\underline{0})$ can perform an action, but the obviously equivalent \mathbb{VPC}_{Th} -term *if* $x = \underline{0}$ *then* $\bar{a}(\underline{0})$ *else* $\bar{a}(\underline{0})$ cannot do anything.

The reader must have noticed that the rule for the output prefix in our concrete semantics appears different from the standard treatment. In the value-passing calculi defined in terms of a model [Milner 1989a], the term in an output prefix must be calculated to a value, an element of the universe, before it is exported. In our approach however the Σ -term is exported as it is. There are two reasons. One is that at our abstract model, there is no way to talk about calculation of terms. But the much more important reason, alluded in Section 1, is that the functional power of the oracle is undesirable. The first order theory provides a universe of values, whereas the value-passing calculus does the calculation. If we maintain a separation between the Σ -terms and the calculations of the Σ -terms, there is no need to calculate any closed Σ -terms since every closed Σ -term is already a ‘value’.

Let us see two examples. For the first example, let A be the following process

definable in \mathbb{VPC} .

$$(a)(\bar{a}(\underline{0}) \mid \mu X.a(x).(\bar{a}(s(x)) \mid (\tau.X + \bar{b}(x))))).$$

A typical action sequence of A is

$$A \xrightarrow{\tau} \xrightarrow{\tau} \dots \xrightarrow{\tau} \xrightarrow{\tau} \xrightarrow{\bar{b}(n)} \mathbf{0},$$

$\underbrace{\hspace{10em}}_{2n+1 \text{ times}}$

where $n \geq 0$. As this example shows, the calculation of the numeral is explicitly demonstrated. The second example is about the encoding of the minimization operator. It is given by the following process, also definable in \mathbb{VPC} .

$$(a)(\bar{a}(\underline{0}) \mid \mu X.a(x).(\bar{a}(s(x)) \mid \text{if } \varphi \text{ then } \bar{b}(x) \text{ else } \tau.X)).$$

Notice that if no numerals satisfy φ then the process diverges.

We write \Longrightarrow for the reflexive and transitive closure of $\xrightarrow{\tau}$, and $\xRightarrow{\lambda}$ for the composition $\Longrightarrow \xrightarrow{\lambda} \Longrightarrow$.

3.2 Absolute Equality

A first attempt to define the bisimulations for \mathbb{VPC}_{Th} is to reiterate the definition from the theory of CCS for all \mathbb{VPC}_{Th} -terms. This would require that $S \xrightarrow{\lambda} S'$ should be bisimulated by $T \xRightarrow{\hat{\lambda}} T'$ whenever S is bisimilar to T . Consider however the \mathbb{VPC} -terms

$$\bar{a}(\underline{0}).S$$

and

$$\text{if } x = \underline{0} \text{ then } \bar{a}(\underline{0}).S \text{ else } \bar{a}(\underline{0}).S.$$

Intuitively these two \mathbb{VPC} -terms are equivalent. But the action $\bar{a}(\underline{0}).S \xrightarrow{\bar{a}(\underline{0})} S$ cannot be simulated by any action of $\text{if } x = \underline{0} \text{ then } \bar{a}(\underline{0}).S \text{ else } \bar{a}(\underline{0}).S$. There are two ways to bypass the problem. One is to confine our attention to processes. This would be a reasonable choice if the operational semantics is formulated in a concrete manner. The other is to apply a symbolic approach, which would of course fit very well with the symbolic operational semantics. Before we take a look at these two solutions, we shall apply to \mathbb{VPC}_{Th} the model independent approach developed in [Fu 2011b]. The equality so obtained provides not only the intuition, but also a standard to compare against.

Process equivalences are observational. The first thing to get it right is to work out what it means for a process to be observable.

Definition 3.1. A process P is *observable*, notation $P \Downarrow$, if $P \xRightarrow{\lambda} P'$ for some P' and some $\lambda \neq \tau$.

In other words, a process is observable if it may interact with another process. Now whatever an observational equivalence is, it must not identify an observable process with an unobservable process. Hence the next definition.

Definition 3.2. A binary relation \mathcal{R} on $\mathcal{P}_{\mathbb{VPC}_{\text{Th}}}$ is *equipollent* if $P \Downarrow \Leftrightarrow Q \Downarrow$ whenever PRQ .

Now suppose two processes P, Q are observationally equivalent. A third process, say O , cannot detect any difference between P, Q by interacting with them. If we think of it, the fact that O cannot tell P, Q apart is best interpreted as saying that $P|O$ and $Q|O$ are observationally equivalent. Now trivially, P and Q cannot be distinguished by any process that does not interact at a particular channel name, say c . If one looks at the same thing from another angle, one easily sees that $(c)P$ must be observationally equivalent to $(c)Q$ as well.

Definition 3.3. A binary relation \mathcal{R} on $\mathcal{P}_{\text{VPC}_{\text{Th}}}$ is *extensional* if the following statements are valid.

- (1) If LRM and PRQ then $(L|P) \mathcal{R} (M|Q)$.
- (2) If PRQ then $(c)P \mathcal{R} (c)Q$ for every $c \in \mathcal{N}$.

If two processes are equivalent, they should be able to maintain the equivalence after one thousand years. The minimal condition making sure that this can be achieved is the bisimulation property of Milner [1989a] and Park [1981]. Following the idea of Fu [2011b], we actually will use a stronger version of the bisimulation introduced by van Glabbeek and Weijland [1989]. In the following definition, the notation \mathcal{R}^{-1} stands for the inverse of \mathcal{R} .

Definition 3.4. A binary relation \mathcal{R} on $\mathcal{P}_{\text{VPC}_{\text{Th}}}$ is a *bisimulation* if it validates the following statements.

- (1) If $QR^{-1}P \xrightarrow{\tau} P'$ then one of the following statements is valid.
 - (a) $Q \implies Q'\mathcal{R}^{-1}P'$ and $Q'\mathcal{R}^{-1}P$ for some Q' .
 - (b) $Q \implies Q''\mathcal{R}^{-1}P$ for some Q'' such that $Q'' \xrightarrow{\tau} Q'\mathcal{R}^{-1}P'$ for some Q' .
- (2) If $PRQ \xrightarrow{\tau} Q'$ then one of the following statements is valid.
 - (a) $P \implies P'\mathcal{R}Q'$ and $P'\mathcal{R}Q$ for some P' .
 - (b) $P \implies P''\mathcal{R}Q$ for some P'' such that $P'' \xrightarrow{\tau} P'\mathcal{R}Q'$ for some P' .

Since a τ -action is an internal action, the bisimulation property can be interpreted as saying how internal actions should simulate each other. If one internal action sequence can reach a state in which it may input or output a value at some name a whereas another internal action sequence cannot do it, then the two internal action sequences are inequivalent.

A basic assumption in the theory of computation is that a divergent computation is different from a computation that terminates. Often time the probability for a real program to diverge is zero. For such a program, divergence is a potential, not an inevitability. A condition that takes into account of this potentiality while upholding the bisimulation property is what we call codivergence requirement. It was first proposed by Priese [1978].

Definition 3.5. A binary relation \mathcal{R} on $\mathcal{P}_{\text{VPC}_{\text{Th}}}$ is *codivergent* if the following statements are valid whenever PRQ .

- If $Q \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} Q_n \xrightarrow{\tau} \dots$ is an infinite internal action sequence, then there must be some $k \geq 1$ and P' such that $P \xrightarrow{\tau} P' \mathcal{R} Q_k$.
- If $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} P_n \xrightarrow{\tau} \dots$ is an infinite internal action sequence, then there must be some $k \geq 1$ and Q' such that $Q \xrightarrow{\tau} Q' \mathcal{R} P_k$.

We have introduced four conditions for the equivalences on the evolving processes. These conditions are minimal from the point of view of interaction as well as computation. We are now turning these minimal conditions into defining properties of process equality.

Definition 3.6. The *absolute equality* $=_{\text{Th}}$ is the largest reflexive, equipollent, extensional, codivergent bisimulation on $\mathcal{P}_{\text{VPC}_{\text{Th}}}$.

The well-definedness of Definition 3.6 is due to the fact that its defining properties are stable under set unions. The definition is completely model independent as long as we only consider those models that have the composition and localization operators and enjoy a dichotomy between the internal actions and the external interactions. From the point of view of equality reasoning, the absolute equality is too abstract. It would be very helpful to work out an external characterization of the absolute equality. This is what we are going to do next for VPC_{Th} . Before that, we state a useful lemma about computation, the Bisimulation Lemma [Fu 2011b]. The property stated in the lemma is called *X-property* by De Nicola et al. [1990].

LEMMA 3.7. *If $P \Longrightarrow P' =_{\text{Th}} Q$ and $Q \Longrightarrow Q' =_{\text{Th}} P$, then $P =_{\text{Th}} Q$.*

Once the equality relation has been defined, a formal classification of the internal actions can be given. We say that S evolves to T in a *computation step*, notation $S \rightarrow T$, if $S \xrightarrow{\tau} T$ and $S =_{\text{Th}} T$, and that S evolves to T in a *change-of-state* internal action, notation $S \xrightarrow{\iota} T$, if $S \xrightarrow{\tau} T$ and $S \neq_{\text{Th}} T$. A change-of-state is a dramatic event of a system since the system can never go back to any of its previous states after the action. The reflexive and transitive closure of \rightarrow will be denoted by \rightarrow^* .

The bisimulation property can now be defined in a more informative way. If $P =_{\text{Th}} Q \xrightarrow{\tau} Q'$, then the simulation by P could be vacuous if $Q =_{\text{Th}} Q'$; otherwise it must take the form $P \rightarrow^* P'' \xrightarrow{\iota} P'$ such that $P'' =_{\text{Th}} Q$ and $P' =_{\text{Th}} Q'$. Similarly if $P =_{\text{Th}} Q \xrightarrow{\bar{a}(t)} Q'$, then the simulation of the output action must take the form $P \rightarrow^* P'' \xrightarrow{\bar{a}(t)} P'$ such that $P'' =_{\text{Th}} Q$ and $P' =_{\text{Th}} Q'$. This is the external bisimulation property we shall define in the next section.

3.3 External Bisimulation

External bisimulations are meant to give an alternative characterization of the absolute equality in terms of explicit simulation of *every* action. In addition to the property of Definition 3.4, external bisimulations must explain how the external actions are bisimulated.

Definition 3.8. A codivergent bisimulation \mathcal{R} on $\mathcal{P}_{\text{VPC}_{\text{Th}}}$ is a VPC_{Th} -*bisimulation* if the following statements are valid for every $\lambda \neq \tau$.

- (1) If $QR^{-1}P \xrightarrow{\lambda} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\lambda} Q'R^{-1}P'$ and PRQ'' for some Q', Q'' .
- (2) If $PRQ \xrightarrow{\lambda} Q'$ then $P \Longrightarrow P'' \xrightarrow{\lambda} P'RQ'$ and $P''RQ$ for some P', P'' .

The VPC_{Th} -*bisimilarity* \simeq_{Th} is the largest VPC_{Th} -bisimulation.

By constructing the relation inductively from \simeq_{Th} that closes up under the composition and the localization operations, one can easily prove the following lemma.

LEMMA 3.9. *The external bisimilarity \simeq_{T_h} is extensional.*

The above lemma and the Bisimulation Lemma is the only thing we need to establish Proposition 3.10, which proves the correctness of Definition 3.8.

PROPOSITION 3.10. *The external bisimilarity \simeq_{T_h} coincides with the absolute equality $=_{\text{T}_h}$.*

PROOF. The inclusion $\simeq_{\text{T}_h} \subseteq =_{\text{T}_h}$ is immediate from Lemma 3.9. The proof of the reverse inclusion is standard using Bisimulation Lemma. Processes of the form $a(x).if\ x = _ \text{ then } \bar{c}(_) \text{ else } \bar{d}(_)$ and of the form $\bar{a}(_) + \bar{a}(_).\bar{c}(_)$, with the names c, d chosen properly, are crucial to deriving the external bisimulation property. A detailed proof of a similar result in π -calculus is given in [Fu and Zhu 2011]. \square

Both the absolute equality and the external bisimilarity are relations on the processes. They can be extended to the VPC_{T_h} -terms in the standard manner.

Definition 3.11. $S \simeq_{\text{T}_h} T$ if and only if $S\rho \simeq_{\text{T}_h} T\rho$ for every assignment ρ whose domain of definition is disjoint from $bv(S \mid T)$.

A standard argument suffices to show that the relation \simeq_{T_h} , and consequently the relation $=_{\text{T}_h}$ as well, is closed under all the process operations.

PROPOSITION 3.12. *The absolute equality is both an equivalence relation and a congruence relation.*

4. SYMBOLIC SEMANTICS

The absolute equality is not very convenient if one is interested in its decidable subsets. Hennessy and Lin [1995] address the issue by introducing symbolic bisimulations. The advantage of the symbolic approach is that it allows one to make full use of the decidable fragments of the logics of the value-passing calculi when constructing equivalence checking algorithms. In [Hennessy and Lin 1995] the symbolic bisimilarity is defined as general as possible so that a coincidence result could be achieved. We shall be less ambitious in this paper. We regard the symbolic bisimilarity as a decidable approximation of the absolute equality. Our motivation for the symbolic bisimilarity is two folds:

- (1) The symbolic bisimilarity should be sound, meaning that it should be a subset of the absolute equality.
- (2) Ideally the symbolic bisimilarity is complete on the decidable subsets of the absolute equality.

We shall see that (2) is much more difficult to come by than (1).

To define the symbolic bisimulations, we need to introduce the symbolic operational semantics. This would not be a big issue had we dropped the conditionals. So the key is to define for instance the semantics of the VPC_{T_h} -term *if φ then $\bar{a}(\underline{0})$ else $\bar{b}(\underline{0})$* where the boolean Σ -expression φ contains free variables. The *symbolic semantics* of Hennessy and Lin [1995; Hennessy and Lin [1996] solves the problem by introducing conditional actions. The syntax of a transition in the symbolic semantics is a tuple of the form $T \xrightarrow{\lambda}_{\varphi} T'$, formalizing the idea that T

Action

$$\frac{}{\sum_{i \in I} \varphi_i \lambda_i . T_i \xrightarrow{\lambda_i}_{\varphi_i} T_i} \quad i \in I.$$

Composition

$$\frac{S \xrightarrow{\lambda}_{\varphi} S'}{S | T \xrightarrow{\lambda}_{\varphi} S' | T} \quad \frac{S \xrightarrow{a(x)}_{\varphi} S' \quad T \xrightarrow{\bar{a}(t)}_{\psi} T'}{S | T \xrightarrow{\tau}_{\varphi\psi} S' \{t/x\} | T'}$$

Localization

$$\frac{T \xrightarrow{\lambda}_{\varphi} T'}{(c)T \xrightarrow{\lambda}_{\varphi} (c)T'} \quad c \text{ is not in } \lambda.$$

Condition

$$\frac{T \xrightarrow{\lambda}_{\varphi} T'}{\phi T \xrightarrow{\lambda}_{\phi\varphi} T'}$$

Recursion

$$\frac{}{!a(x).T \xrightarrow{a(x)}_{\top} T \mid !a(x).T} \quad \frac{}{!\bar{a}(t).T \xrightarrow{\bar{a}(t)}_{\top} T \mid !\bar{a}(t).T}$$

Fig. 4. Symbolic Semantics.

may perform the action λ under the condition that the boolean Σ -expression φ is a Th-theorem. The set of the action labels for the symbolic semantics is

$$\{a(x), \bar{a}(t) \mid a \in \mathcal{N}, x \in \mathbf{V}_{\Sigma}, t \in \mathbf{T}_{\Sigma}\} \cup \{\tau\},$$

also ranged over by λ . The labeled transition system is defined in Fig. 4. Again the symmetric versions of the composition rules have been omitted. The treatment of the input prefix is in a late style, which is better suited for algorithmic investigations. This point will become clearer when we study the proof systems. The symbolic semantics is stable under substitution in the sense of the following lemma.

LEMMA 4.1. *If $S \xrightarrow{\lambda}_{\varphi} T$ then $S\sigma \xrightarrow{\lambda\sigma}_{\varphi\sigma} T\sigma$ for every substitution σ . On the other hand, if $S\sigma \xrightarrow{\lambda'}_{\varphi'} T'$ for some substitution σ , then there must exist some λ, φ, T such that $\lambda' = \lambda\sigma, \varphi' = \varphi\sigma, T' \equiv T\sigma$ and $S \xrightarrow{\lambda}_{\varphi} T$.*

We abbreviate $\xrightarrow{\tau}_{\varphi_1} \dots \xrightarrow{\tau}_{\varphi_n}$ to $\Longrightarrow_{\varphi_1 \dots \varphi_n}$, and $\Longrightarrow_{\varphi} \xrightarrow{\lambda}_{\varphi'} \Longrightarrow_{\varphi''}$ to $\xRightarrow{\lambda}_{\varphi\varphi''}$. We remark that $T \Longrightarrow_{\top} T$.

It's time to look at some examples. The following three terms are in \mathbb{VPC} :

$$\begin{aligned} L(y) &\stackrel{\text{def}}{=} \text{if } \underline{0} \leq y \text{ then } \bar{b}(\underline{0}), \\ M(y) &\stackrel{\text{def}}{=} (a)(\bar{a}(\underline{0}) \mid \mu X.a(x).(\bar{a}(s(x)) \mid (\tau.X + \text{if } y = x \text{ then } \bar{b}(\underline{0})))), \\ N(y) &\stackrel{\text{def}}{=} (a)(\bar{a}(\underline{0}) \mid \mu X.a(x).(\bar{a}(s(x)) \mid (\tau.X + \text{if } y = x \text{ then } \bar{b}(y)))). \end{aligned}$$

The \mathbb{VPC} -term $L(y)$ may perform only one conditional action:

$$L(y) \xrightarrow{\bar{b}(\underline{0})}_{0 \leq y} \mathbf{0}.$$

The \mathbb{VPC} -term $M(y)$ has the following interesting action sequence

$$M(y) \xrightarrow{\tau}_{\top} \xrightarrow{\bar{b}(\underline{0})}_{y=\underline{n}} \mathbf{0}.$$

Similarly the \mathbb{VPC} -term $N(y)$ has a similar action sequence

$$N(y) \xrightarrow{\tau}_{\top} \xrightarrow{\bar{b}(y)}_{y=\underline{n}} \mathbf{0}.$$

In the second and third examples the set of the possible conditions is

$$\{y=\underline{0}, y=\underline{1}, y=\underline{2}, \dots, y=\underline{n}, \dots\}.$$

What the second example tells us is that even with a finite description and a finite set of potential external actions, the set of the conditions that enable an action of a \mathbb{VPC} -term, like $\bar{b}(\underline{0})$, could be infinite.

The symbolic semantics is a correct extension of the concrete semantics. This is established in the next two lemmas whose proofs are given by induction on derivation.

LEMMA 4.2. *The symbolic semantics is sound with respect to the concrete semantics in the following sense:*

- (i) If $S \xrightarrow{\tau}_{\varphi} T$ for some Th-theorem φ then $S \xrightarrow{\tau} T$.
- (ii) If $S \xrightarrow{\bar{a}(t)}_{\varphi} T$ for some Th-theorem φ and some $t \in \mathcal{T}_{\Sigma}^0$ then $S \xrightarrow{\bar{a}(t)} T$.
- (iii) If $S \xrightarrow{a(x)}_{\varphi} T$ for some Th-theorem φ then $S \xrightarrow{a(t)} T\{t/x\}$ for every $t \in \mathcal{T}_{\Sigma}^0$.

LEMMA 4.3. *The symbolic semantics is complete with respect to the concrete semantics in the following sense:*

- (i) If $S \xrightarrow{\tau} T$ then $S \xrightarrow{\tau}_{\varphi} T$ for some Th-theorem φ .
- (ii) If $S \xrightarrow{\bar{a}(t)} T$ then $S \xrightarrow{\bar{a}(t)}_{\varphi} T$ for some Th-theorem φ .
- (iii) If $S \xrightarrow{a(t)} T$ then $S \xrightarrow{a(x)}_{\varphi} T'$ for some Th-theorem φ and some x, T' such that $T \equiv T'\{t/x\}$.

4.1 Symbolic Bisimulation

In the symbolic approach, is it reasonable to require that $S \xrightarrow{\tau}_{\varphi} S'$ be simulated by $T \xRightarrow{\varphi'} T'$ for bisimilar S and T such that $\text{Th} \vdash \varphi \Rightarrow \varphi'$? Again let's take a look at the \mathbb{VPC} -terms $\varphi\tau.S$ and *if* $x = \underline{0}$ *then* $\varphi\tau.S$ *else* $\varphi\tau.S$, where $x \notin \text{fv}(\varphi)$. The two \mathbb{VPC} -terms are equivalent by all reasonable criteria. But $\varphi\tau.S \xrightarrow{\tau}_{\varphi} S$ cannot be simulated by *if* $x = \underline{0}$ *then* $\varphi\tau.S$ *else* $\varphi\tau.S$ in the above required manner since in the latter term the τ -action can only be fired under a condition strictly stronger than φ . But notice that $(x = \underline{0}) \wedge \varphi \vee (x \neq \underline{0}) \wedge \varphi \Leftrightarrow \varphi$ is a PA-theorem. Under either $(x = \underline{0}) \wedge \varphi$ or $(x \neq \underline{0}) \wedge \varphi$ the \mathbb{VPC} -term *if* $x = \underline{0}$ *then* $\varphi\tau.S$ *else* $\varphi\tau.S$ may evolve into S . This leads to the idea of finding a collection of boolean expressions such that the disjunction of the collection is weaker than φ . If under each of the conditions the simulation can be done, then there is a simulation. A first attempt to define a symbolic counterpart of the Milner-Park bisimilarity could be as follows:

A symmetric relation \mathcal{R} on $\mathcal{T}_{\mathbb{VPC}_{\top\text{h}}}$ is a ? -bisimulation if the following condition is met whenever $S\mathcal{R}T$:

If $S \xrightarrow{\lambda}_{\varphi} S'$ then there is a class $\{T \xrightarrow{\widehat{\lambda}_i}_{\varphi_i} T'_i\}_{i \in I}$ such that $\top\text{h} \vdash \varphi\varphi_i \Rightarrow \lambda = \lambda_i$ and $(\varphi\varphi_i S', \varphi\varphi_i T'_i) \in \mathcal{R}$ for every $i \in I$.

Let $\simeq^?$ be the largest ? -bisimulation.

In the above definition, $\lambda = \lambda_i$ stands for \top if λ, λ_i are syntactically the same; it is $t = t'$ if $\lambda \equiv \bar{a}(t)$ and $\lambda_i \equiv \bar{a}(t')$ for some name a ; otherwise $\lambda = \lambda_i$ stands for \perp . The relation $\simeq^?$ is not a useful one. Consider the \mathbb{VPC} -processes $R(y), S(y), T(y)$ defined as follows:

$$\begin{aligned} R(y) &\stackrel{\text{def}}{=} (b(\bar{b}(y) \mid \mu X.b(z).\bar{r}(z).\text{if } \underline{0} < z \text{ then } (\bar{b}(\mathfrak{p}(z)) \mid X)), \\ S(y) &\stackrel{\text{def}}{=} \tau.T(y) + \text{if } \underline{0} \leq y \text{ then } \tau.R(y), \\ T(y) &\stackrel{\text{def}}{=} (a(\bar{a}(\underline{0}) \mid \mu X.a(x)).(\bar{a}(s(x)) \\ &\quad \mid (\tau.X + \text{if } \underline{0} \leq y \leq x \text{ then } \tau.(\text{if } y = x \text{ then } \tau.R(y) \text{ else } \bar{e}(\underline{0}))))). \end{aligned}$$

In the definition of $R(y)$ the term $\mathfrak{p}(z)$ is the predecessor of z . The predecessor function can be implemented in \mathbb{VPC} . The details can be found in Section 6. The behavior of $R(y)$ is captured by the following action sequence:

$$R(\underline{n}) \xrightarrow{\tau}_{\top} \xrightarrow{\bar{r}(\underline{n})}_{\top} \xrightarrow{\tau}_{\underline{0} < \underline{n}} \xrightarrow{\bar{r}(\underline{n}-1)}_{\top} \xrightarrow{\tau}_{\underline{0} < \underline{n}-1} \dots \xrightarrow{\tau}_{\underline{0} < \underline{1}} \xrightarrow{\bar{r}(\underline{1})}_{\top} \dots$$

It is obvious that $R(\underline{n})$ and $R(\underline{n}')$ are inequivalent whenever $n \neq n'$. The processes $S(y)$ and $T(y)$ are bisimilar since the action

$$S(y) \xrightarrow{\tau}_{\underline{0} \leq y} R(y) \tag{12}$$

can be simulated by $T(y)$ as long as y is instantiated by a numeral. On the other hand it is easy to see that $S(y) \not\approx^? T(y)$. The only way to simulate (12) is by the collection

$$\{T(y) \xrightarrow{\tau}_{\underline{0} \leq y \leq \underline{n}} \text{if } y = \underline{n} \text{ then } \tau.R(y) \text{ else } \bar{e}(\underline{0})\}_{n \in \mathbb{N}}.$$

But notice that the \mathbb{VPC} -term

$$\text{if } \underline{0} \leq y \leq \underline{n} \text{ then } R(y) \tag{13}$$

is *not* bisimilar to the \mathbb{VPC} -term

$$\text{if } \underline{0} \leq y \leq \underline{n} \text{ then } (\text{if } y = \underline{n} \text{ then } \tau.R(y) \text{ else } \bar{e}(\underline{0})). \tag{14}$$

The relation $\simeq^?$ has to be abandoned because it is not transitive! Fig. 5 offers a counter example. One has $U(y) \simeq^? V(y)$ and $V(y) \simeq^? W(y)$ but not $U(y) \simeq^? W(y)$. The non-bisimilarity of (13) and (14) explains why $U(y) \not\approx^? W(y)$. These exercises are helpful in improving our understanding of the symbolic semantics.

The symbolic bisimulations need be defined in a more subtle way. The key idea of Hennessy and his collaborators is that the partition of the condition under which the simulated action is fired should not depend on the conditions under which the simulations are done. In their definition a partition of ψ is a collection $\{\psi_i\}_{i \in I}$ such that $\psi \Leftrightarrow \bigvee_{i \in I} \psi_i$, where the indexing set I could be as large as the size of the model. The reason that they may resort to the all-powerful operator \bigvee is that

$$\begin{aligned}
U(y) &\stackrel{\text{def}}{=} \tau.W(y) + \text{if } \underline{0} \leq y \leq \underline{m} \text{ then } \tau.R(y), \\
V(y) &\stackrel{\text{def}}{=} \tau.W(y) + \text{if } y = \underline{0} \text{ then } \tau.R(y) + \dots + \text{if } y = \underline{m} \text{ then } \tau.R(y), \\
W(y) &\stackrel{\text{def}}{=} \text{if } \underline{0} = y \text{ then } \tau.(\text{if } y = \underline{0} \text{ then } \tau.R(y) \text{ else } \bar{e}(\underline{0})) \\
&\quad + \text{if } \underline{0} \leq y \leq \underline{1} \text{ then } \tau.(\text{if } y = \underline{1} \text{ then } \tau.R(y) \text{ else } \bar{e}(\underline{0})) \\
&\quad + \dots \\
&\quad + \text{if } \underline{0} \leq y \leq \underline{m} \text{ then } \tau.(\text{if } y = \underline{m} \text{ then } \tau.R(y) \text{ else } \bar{e}(\underline{0})).
\end{aligned}$$

Fig. 5. Non-transitivity of $\simeq^?$.

they are using a meta logic about the oracle model. We will use a more restricted approach. We insist that the symbolic semantics of a value-passing calculus should only make use of the first order logic by which the logical theory of the calculus is defined. Recall that the motivation of the symbolic approach is to support mechanical manipulations [Hennessy and Lin 1995]. Decidability is therefore an important issue for the symbolic semantics. Our selection of the logical theories for the value-passing calculi has been such that it is possible to develop a symbolic observational theory that supports algorithmic studies. To achieve decidability, not only that a partition has to be finite, but also that the Σ -expressions that constitute a partition must be boolean.

Definition 4.4. Suppose Th is a theory over Σ , V is a finite set of variables, and $fv(\varphi) \subseteq V$. A *boolean Th-partition* of φ on V is a finite set of boolean Σ -expressions $\{\varphi_i\}_{i \in I}$ such that $fv(\bigvee_{i \in I} \varphi_i) \subseteq V$, $\text{Th} \vdash \varphi_i \wedge \varphi_j \Rightarrow \perp$ if $i \neq j$, and $\text{Th} \vdash \varphi \Leftrightarrow \bigvee_{i \in I} \varphi_i$.

Let's now turn to divergence. Consider the following VPC -term

$$K(y) = (a)(\bar{a}(\underline{0}) \mid \mu X.a(x).(X \mid \text{if } x \leq y \text{ then } \bar{a}(s(x))))$$

The symbolic semantics admits the following infinite sequence

$$K(y) \xrightarrow{\tau} \top \xrightarrow{\tau} \underline{0} \leq y \xrightarrow{\tau} \underline{1} \leq y \dots \xrightarrow{\tau} \underline{n} \leq y \dots \quad (15)$$

Although every finite subset of $\{\top, \underline{0} \leq y, \underline{1} \leq y, \dots, \underline{n} \leq y, \dots\}$ is consistent, the infinite sequence (15) is a fake. For every numeral \underline{n} the process $K(\underline{n})$ terminates. From the point of view of the model \mathbf{N} , no assignment can satisfy all the Σ_{PA} -expressions in $\{\top, \underline{0} \leq y, \underline{1} \leq y, \dots, \underline{n} \leq y, \dots\}$. From the point of view of the first order logic, no satisfiable Σ_{PA} -expression implies all the Σ_{PA} -expressions in $\{\top, \underline{0} \leq y, \underline{1} \leq y, \dots, \underline{n} \leq y, \dots\}$. A faithful symbolic interpretation of divergence would make use of the infinite conjunction, which as we have argued, is not in line with the philosophy of the symbolic approach. To get a glimpse of the complication caused by the codivergence, take a look at the following VPC -term

$$H(y) = (a)(\bar{a}(\underline{0}) \mid \mu X.a(x).(X \mid \text{if } y \leq x \text{ then } \bar{a}(s(x))))$$

which is a slight modification of the previous VPC -term. This term also admit an infinite sequence of τ -actions:

$$H(y) \xrightarrow{\tau} \top \xrightarrow{\tau} y \leq \underline{0} \xrightarrow{\tau} y \leq \underline{1} \dots \xrightarrow{\tau} y \leq \underline{n} \dots \quad (16)$$

The infinite sequence (16) is not real for $H(1)$, $H(2)$, $H(3)$, \dots . But it is real for $H(0)$. These examples suggest that it is difficult to give a nice symbolic description of the codivergence property at a general level. It should be noticed though that the divergence is intrinsically an undecidable property. It is not that important for the symbolic semantics since the symbolic approach is of an algorithmic nature. Our definition of symbolic bisimulation ignores the divergence issue.

Definition 4.5. A symmetric relation \mathcal{R} on $\mathcal{T}_{\mathbb{VPC}_{\text{Th}}}$ is a *symbolic bisimulation* if the followings hold whenever $S\mathcal{R}T$ and $V = fv(S|T)$:

- (1) If $S \xrightarrow{\tau}_{\varphi} S'$ and $\text{Th} \cup \{\varphi\}$ is consistent, then there are a boolean Th-partition $\{\varphi_i\}_{i \in I}$ of φ on V and a collection $\{T \Longrightarrow_{\psi_i} T_i\}_{i \in I}$ such that, for each $i \in I$, $\text{Th} \vdash \varphi_i \Rightarrow \psi_i$ and $\varphi_i S \mathcal{R} \varphi_i T_i$, and one of the following properties holds:
 - (a) $\varphi_i S' \mathcal{R} \varphi_i T_i$;
 - (b) $T_i \xrightarrow{\tau}_{\psi'_i} T'_i$ for some ψ'_i, T'_i such that $\text{Th} \vdash \varphi_i \Rightarrow \psi'_i$, and $\varphi_i S' \mathcal{R} \varphi_i T'_i$.

- (2) If $S \xrightarrow{\bar{a}(t)}_{\varphi} S'$ and $\text{Th} \cup \{\varphi\}$ is consistent, then there are a boolean Th-partition $\{\varphi_i\}_{i \in I}$ of φ on V and a collection $\{T \Longrightarrow_{\psi_i} T_i \xrightarrow{\bar{a}(t_i)}_{\psi'_i} T'_i\}_{i \in I}$ such that

$$\text{Th} \vdash (\varphi_i \Rightarrow \psi_i \psi'_i) \wedge (\varphi_i \Rightarrow t = t_i)$$

and, for every $i \in I$, both $\varphi_i S \mathcal{R} \varphi_i T_i$ and $\varphi_i S' \mathcal{R} \varphi_i T'_i$ are valid.

- (3) If $S \xrightarrow{a(x)}_{\varphi} S'$ and $\text{Th} \cup \{\varphi\}$ is consistent, then there are a boolean Th-partition $\{\varphi_i\}_{i \in I}$ of φ on V and, for each $i \in I$, a boolean Th-partition $\{\phi_i^j\}_{j \in J_i}$ of \top on $\{x\}$, and moreover a collection

$$\left\{ T \Longrightarrow_{\psi_{ij}} T_{ij} \xrightarrow{a(x)}_{\psi'_{ij}} T'_{ij} \right\}_{i \in I, j \in J_i}$$

such that $\text{Th} \vdash \varphi_i \Rightarrow \psi_{ij} \psi'_{ij}$, $\varphi_i S \mathcal{R} \varphi_i T_{ij}$ and $\varphi_i \phi_i^j S' \mathcal{R} \varphi_i \phi_i^j T'_{ij}$ for every $i \in I$ and for all $j \in J_i$.

The *symbolic bisimilarity* \simeq_{Th}^s is the largest symbolic bisimulation. \square

In the above definition the requirement that $\text{Th} \cup \{\varphi\}$ being consistent is important. Without the condition the \mathbb{VPC} -term *if $x \neq x$ then $\bar{a}(0)$* would be wrongfully distinguished from $\mathbf{0}$.

The main algebraic properties of \simeq_{Th}^s are summarized in the next proposition.

PROPOSITION 4.6. *The following statements about \simeq_{Th}^s are valid.*

- (1) *The relation \simeq_{Th}^s is an equivalence.*
- (2) *The relation \simeq_{Th}^s is a congruence.*
- (3) *If $S \simeq_{\text{Th}}^s T$ then $S\sigma \simeq_{\text{Th}}^s T\sigma$ for every substitution σ .*

PROOF. (1) The proof of transitivity is routine and tedious.

(2) The proof of the congruence property is standard after demonstrating that, for every boolean expression φ , $S \simeq_{\text{Th}}^s T$ implies $\varphi S \simeq_{\text{Th}}^s \varphi T$.

(3) Using Lemma 4.1 it is easy to show that $\{(S\sigma, T\sigma) \mid S \simeq_{\text{Th}}^s T\}$ is a symbolic bisimulation. We need to make use of a meta theoretical result asserting that $\text{Th} \vdash \varphi\sigma$ for every substitution σ whenever $\text{Th} \vdash \varphi$, which is an easy consequence of FO1 and FO2. \square

The next technical lemma, whose simple proof is omitted, helps understand the above definition. It is the symbolic version of the Stuttering Lemma of van Glabbeek and Weijland [1989].

LEMMA 4.7. *Suppose $T \xrightarrow{\tau}_{\varphi_1} T_1 \xrightarrow{\tau}_{\varphi_2} T_2 \dots \xrightarrow{\tau}_{\varphi_n} T_n$ and $\varphi \Rightarrow \varphi_1 \dots \varphi_n$. Then $\varphi T \simeq_{\text{Th}}^s \varphi T_n$ implies $\varphi T_1 \simeq_{\text{Th}}^s \varphi T_2 \simeq_{\text{Th}}^s \dots \simeq_{\text{Th}}^s \varphi T_n$.*

Now let's take a look at some examples. Consider the following VPC -terms:

$$\begin{aligned} E(y) &\stackrel{\text{def}}{=} \mu X. \text{if } y = \underline{0} \vee y = \underline{1} \text{ then } \bar{b}(y).X, \\ A(y) &\stackrel{\text{def}}{=} E(y) \mid \text{if } y = \underline{0} \text{ then } \bar{b}(y), \\ B(y) &\stackrel{\text{def}}{=} E(y) \mid \text{if } y = \underline{1} \text{ then } \bar{b}(y). \end{aligned}$$

It is clear that $A(y) \simeq_{\text{Th}}^s B(y)$. The action $A(y) \xrightarrow{\bar{b}(y)}_{y=\underline{0}} E(y)$ is simulated by the single action

$$B(y) \xrightarrow{\bar{b}(y)}_{y=\underline{0} \vee y=\underline{1}} E(y) \mid \text{if } y = \underline{1} \text{ then } \bar{b}(y).$$

This example shows that the condition $\text{Th} \vdash \varphi_i \Rightarrow \psi_i$ in say (1) of Definition 4.5 should not be strengthened to $\text{Th} \vdash \varphi_i \Leftrightarrow \psi_i$.

The finiteness of partition is a genuine restriction. Let's see a counter example. Consider the following VPC -terms:

$$\begin{aligned} F(y) &\stackrel{\text{def}}{=} \mu X. a(x).(\bar{a}(s(x)) \mid \text{if } y=x \text{ then } \bar{b}(x) \text{ else } X), \\ C(y) &\stackrel{\text{def}}{=} \text{if } \underline{0} \leq y \text{ then } \bar{b}(y), \\ D(y) &\stackrel{\text{def}}{=} (a)(\bar{a}(\underline{0}) \mid F(y)). \end{aligned}$$

Typically D has the following action sequence:

$$D(y) \xrightarrow{\tau}_{y \neq \underline{0} \wedge \dots \wedge y \neq \underline{n-1}} \xrightarrow{\bar{b}(n)}_{y=\underline{n}} \dots$$

It is not difficult to see that $C(y) =_{\text{Th}} D(y)$. Intuitively the action $C(y) \xrightarrow{\bar{b}(y)}_{\underline{0} \leq y} \mathbf{0}$ is simulated by the *infinite* collection $\{D(y) \xrightarrow{\tau}_{y \neq \underline{0} \wedge \dots \wedge y \neq \underline{n-1}} \xrightarrow{\bar{b}(n)}_{y=\underline{n}} \dots\}_{n \in \mathbf{N}}$. There is no finite collection that can simulate $C(y) \xrightarrow{\bar{b}(y)}_{\underline{0} \leq y} \mathbf{0}$. So the symbolic bisimilarity \simeq_{Th}^s cannot coincide with the absolute equality $=_{\text{Th}}$ even if divergence is ignored.

The symbolic bisimilarity is however a correct approximation of the absolute equality.

THEOREM 4.8. *Let S, T be finite VPC_{Th} -terms. Then $S =_{\text{Th}} T$ if $S \simeq_{\text{Th}}^s T$.*

PROOF. Let \mathcal{R} be the following binary relation

$$\left\{ (S\rho, T\rho) \mid \begin{array}{l} S, T \text{ are finite, } S \simeq_{\text{Th}}^s T \text{ and} \\ \rho \text{ is an assignment such that } bv(S|T) \cap \text{dom}(\rho) = \emptyset \end{array} \right\}.$$

Suppose $S \simeq_{\text{Th}}^s T$ and $S\rho \xrightarrow{a(t)} P$ for some $t \in \mathbf{T}_{\Sigma}^0$ and some assignment ρ such that $bv(S|T) \cap \text{dom}(\rho) = \emptyset$. Let V be $fv(S|T)$. By Lemma 4.3, $S\rho \xrightarrow{a(x)}_{\varphi'} S'$ for some Th -theorem φ' and some x, S' such that $P \equiv S'\{t/x\}$. By Lemma 4.1,

there exists some φ, S_1 such that $\varphi' \equiv \varphi\rho$, $S' \equiv S_1\rho$ and $S \xrightarrow{\varphi} S_1$. By the definition of symbolic bisimulation there is a boolean Th-partition $\{\varphi_i\}_{i \in I}$ of φ on V , and for each $i \in I$ a boolean Th-partition $\{\phi_i^j\}_{j \in J_i}$ of \top on $\{x\}$, and moreover a collection $\{T \Longrightarrow_{\psi_{ij}} T_{ij} \xrightarrow{a(x)}_{\psi'_{ij}} T'_{ij}\}_{i \in I, j \in J_i}$ of actions such that $\text{Th} \vdash \varphi_i \Rightarrow \psi_{ij}\psi'_{ij}$, $\varphi_i S \simeq_{\text{Th}}^s \varphi_i T_{ij}$ and $\varphi_i \phi_i^j S_1 \simeq_{\text{Th}}^s \varphi_i \phi_i^j T'_{ij}$ for every $i \in I$ and every $j \in J_i$. Now $\{\varphi_i\rho\}_{i \in I}$ is a boolean Th-partition of $\varphi\rho$ on the empty set. So $\text{Th} \vdash \varphi'$ implies $\text{Th} \vdash \varphi_i\rho$ for some $i \in I$, which in turn implies $\text{Th} \vdash \psi_{ij}\psi'_{ij}\rho$ and $S\rho \mathcal{R} T_{ij}\rho$ for all $j \in J_i$. Since $\{\phi_i^j\}_{j \in J_i}$ is a boolean Th-partition of \top , there must be some $j \in J_i$ such that $\text{Th} \vdash \phi_i^j\{t/x\}$. Hence $S_1\rho[x \leftarrow t] \mathcal{R} T'_{ij}\rho[x \leftarrow t]$. At the meantime notice that $T\rho \Longrightarrow T_{ij}\rho \xrightarrow{a(t)} T'_{ij}\rho[x \leftarrow t]$ by Lemma 4.1 and Lemma 4.2. Conclude that \mathcal{R} is a bisimulation. The relation is also extensional by Proposition 4.6. It is equipollent since the external actions are explicitly bisimulated. The codivergence property is vacuously satisfied by the finite processes. \square

5. PROOF SYSTEM

When confined to the finite VPC_{Th} -processes, one expects that the equality can be mechanically checked. Often such an algorithm is based on a complete equational system. For most value-passing calculi defined in literature a self-contained decidable procedure for equivalence checking is out of the question since the logics/models are undecidable. For example, to check if $\varphi\bar{a}(t).S = \varphi\bar{a}(t').S$ holds, one has to check if $\varphi \Rightarrow t = t'$ is valid. An algorithm for checking the equivalence of finite processes has to make use of an oracle that answers every question on the validity of proposition. However for a value-passing calculus VPC_{Th} studied in this paper, equivalence checking algorithms do exist, thanks to the boolean completeness of Th.

In studying proof systems, a standard treatment is to remove the composition operator using the *expansion law* [Hennessy and Milner 1985; Milner 1989a]. This is done at the expense of introducing the unguarded choice operator. This operator destroys the congruence property of process equalities. As a consequence additional complication is necessary to produce an equational system. In this paper we provide an alternative approach that avoids the use of the unguarded choice terms, although we still need the mixed choice [Palamidessi 2003; Fu and Lu 2010]. In this section the notation $\sum_{i \in I} \varphi_i \lambda_i . S_i$ stands for a mixed choice.

Suppose $S \equiv \sum_{i \in I} \varphi_i \lambda_i . S_i$ and $T \equiv \sum_{j \in J} \psi_j \lambda_j . T_j$. The expansion law is formulated by the following equality:

$$\begin{aligned} S|T &= \sum_{i \in I} \varphi_i \lambda_i . (S_i|T) + \sum_{\substack{\lambda_i = a(x) \\ \lambda_j = \bar{a}(t_j)}} \varphi_i \psi_j \tau . (S_i\{t_j/x\}|T_j) \\ &\quad + \sum_{j \in J} \psi_j \lambda_j . (S|T_j) + \sum_{\substack{\lambda_j = b(y) \\ \lambda_i = \bar{b}(t_i)}} \varphi_i \psi_j \tau . (S_i|T_j\{t_i/y\}). \end{aligned}$$

The law converts two concurrent choice terms to a single choice term. An equational system AS_{Th} consists of the first order logic axioms defined in Fig. 1, the nonlogical axioms of Th, the equational axioms defined in Fig. 6 and the expansion law. In the

S1	$[\perp]\lambda.T + \sum$	$= \sum$
S2	$\varphi\lambda.T + \sum$	$= \psi\lambda.T + \sum, \text{ if } \text{Th} \vdash \varphi \Leftrightarrow \psi$
S3	$\varphi\bar{a}(t).T + \sum$	$= \varphi\bar{a}(t').T + \sum, \text{ if } \text{Th} \vdash \varphi \Rightarrow t = t'$
S4	$\varphi\lambda.\sum' + \sum$	$= \varphi\lambda.\varphi\sum' + \sum$
S5	$\varphi\lambda.T + \varphi'\lambda.T + \sum$	$= (\varphi \vee \varphi')\lambda.T + \sum$
S6	$\sum_{i \in I} \varphi n(x).T_i + \sum$	$= \sum_{i \in I} \varphi n(x).T_i + \varphi n(x).\sum_{i \in I} \phi_i \tau.T_i + \sum,$ if $\text{Th} \vdash \varphi \Leftrightarrow \bigvee_{i \in I} \phi_i$
I1	$[\perp]T$	$= \mathbf{0}$
I2	φT	$= \psi T, \text{ if } \text{Th} \vdash \varphi \Leftrightarrow \psi$
I3	$\varphi(\psi T)$	$= (\varphi\psi)T$
I4	$\phi \sum_{i \in I} \varphi \lambda_i.T_i$	$= \sum_{i \in I} \phi \varphi \lambda_i.T_i,$
I5	$\varphi(c)T$	$= (c)\varphi T$
L1	$(c)\mathbf{0}$	$= \mathbf{0}$
L2	$(c)(\varphi\lambda.T + \sum)$	$= (c)\sum, \text{ if } c \text{ is in } \lambda$
L3	$(c)\sum_{i \in I} \varphi_i \lambda_i.T_i$	$= \sum_{i \in I} \varphi_i \lambda_i.(c)T_i, \text{ if } c \text{ is not in } \sum_{i \in I} \varphi_i \lambda_i.-$
C	$\sum + \varphi\tau.(\sum' + \sum)$	$= \sum + \varphi\sum'$

Fig. 6. Axiom for Finite \mathbb{VPC}_{Th} -Term.

formulation of the laws we abbreviate for example $\varphi_0\lambda_0.T_0 + \varphi_1\lambda_1.T_1 + \dots + \varphi_n\lambda_n.T_n$ to $\varphi_0\lambda_0.T_0 + \sum$. We write $AS_{\text{Th}} \vdash S = T$ if the equality $S = T$ can be derived within AS_{Th} , and $S \stackrel{R}{=} T$ if R is the major law used to derive the equality $S = T$.

Most of the laws are variants of the axioms proposed in previous studies on the value-passing calculi [Ingólfssdóttir and Lin 2001] and the name-passing calculi [Parrow and Sangiorgi 1995]. The law S6 was proposed by Parrow and Sangiorgi [1995] as an axiom that tells apart the early equivalence and the late equivalence [Milner et al. 1992]. The action

$$a(x).S + a(x).T + a(x).\text{if } \varphi \text{ then } S \text{ else } T \xrightarrow{a(x)}_{\top} \text{if } \varphi \text{ then } S \text{ else } T$$

for instance can be simulated by

$$\{a(x).S + a(x).T \xrightarrow{a(x)}_{\top} S, a(x).S + a(x).T \xrightarrow{a(x)}_{\top} T\},$$

using the partition $\{\varphi, \neg\varphi\}$. The combination of the S-laws has powerful and interesting consequences, two of which are given by the following lemmas.

LEMMA 5.1. $AS_{\text{Th}} \vdash \sum + \psi\lambda.T = \sum + \varphi\lambda.T + \psi\lambda.T$ if $\text{Th} \vdash \varphi \Rightarrow \psi$.

PROOF. The equality is immediate from S2 and S5. \square

LEMMA 5.2. $AS_{\text{Th}} \vdash \sum + \varphi(\lambda.T)\{t/x\} = \sum + \varphi(\lambda.T)\{t'/x\}$ if $\text{Th} \vdash \varphi \Rightarrow t = t'$.

PROOF. The equality is a consequence of S2 and S3 of Fig. 6 and CG2 of Fig. 1. \square

The C-law is a variant of (17), the well-known B-law of van Glabbeek and Weijland [1989].

$$\lambda.(\tau.(S + T) + T) = \lambda.(S + T). \quad (17)$$

The B-law works for the unguarded choice. The C-law, which is stronger than the B-law, is formulated for the guarded choice. It should be pointed out that C-law implies the stronger forms of Milner's T1 and T2 [Milner 1989a; Fu and Yang 2003], as is stated in the next lemma.

LEMMA 5.3. *The following statements are valid.*

- (1) $AS_{\text{Th}} \vdash \tau.T = T.$
- (2) $AS_{\text{Th}} \vdash \sum + \varphi\tau. \sum = \sum.$

PROOF. We may convert a VPC_{Th} -term T to a choice term (see Lemma 5.6). So (1) can be turned into a special case of the C-law. \square

For the proof system AS_{Th} to be useful at all, all of its laws must be valid if the equality is interpreted as the symbolic bisimilarity. The verification of this soundness property of AS_{Th} is routine. It will not be carried out in this paper.

LEMMA 5.4. *If $AS_{\text{Th}} \vdash S = T$ then $S \simeq_{\text{Th}}^s T$.*

A nice property of a proof system is that it allows one to focus on terms in some special form. For the value-passing calculi the definition of normal form is straightforward.

Definition 5.5. A finite VPC_{Th} -term is a *normal form* if it only contains the summation operator.

By definition a normal form is a choice term $\sum_{i \in I} \varphi_i \lambda_i. T_i$ such that T_i is a normal form for every $i \in I$. Using I5 and L3 one can pull all the localization operations in a term to the very front. Then one may remove all the composition operations and the conditionals appearing in the term by applying the expansion law and the I-axioms. And finally one removes the localization operations using the L-axioms. What one gets is a term constructed solely by choice operations.

LEMMA 5.6. *For each finite VPC_{Th} -term T there is some normal form T' such that $AS_{\text{Th}} \vdash T = T'$.*

For the system AS_{Th} to be really useful, one would hope that the following Completeness Theorem holds.

THEOREM 5.7. *$S \simeq_{\text{Th}}^s T$ if and only if $AS_{\text{Th}} \vdash S = T$.*

The proof of Theorem 5.7 will be our main endeavor in Section 5.1.

5.1 Completeness Proof

Suppose $S \simeq_{\text{Th}}^s T$ and S, T are the normal forms $\sum_{i \in I} \varphi_i \lambda_i. S_i$ and $\sum_{j \in J} \psi_j \lambda_j. T_j$ respectively. We shall prove that the following properties hold for the normal forms:

- (S) $AS_{\text{Th}} \vdash T = T + \phi T'$ if $T \xrightarrow{\tau}_{\varphi} T'$, $\text{Th} \vdash \phi \Rightarrow \varphi$ and $\phi T \simeq_{\text{Th}}^s \phi T'$.
- (P) $AS_{\text{Th}} \vdash T = T + S = S$ if $T \simeq_{\text{Th}}^s S$.

This is done by induction on the maximal nested depth of the guarded choice operations. Let's write $\text{dep}(T)$ for the depth of T . Assume that (S) and (P) hold for all S, T with depths no more than $i - 1$. The inductive proof of (S) is given below:

—Suppose T is a normal form of depth i and $T \xrightarrow{\tau}_{\varphi_1} T_1 \xrightarrow{\tau}_{\varphi_2} \dots T_{n-1} \xrightarrow{\tau}_{\varphi_n} T_n$ such that $\varphi T_1 \simeq_{\text{Th}}^s \varphi T_n$, where $\varphi = \varphi_1 \dots \varphi_n$. Then $\varphi T_1 \simeq_{\text{Th}}^s \varphi T_2 \simeq_{\text{Th}}^s \dots \simeq_{\text{Th}}^s \varphi T_n$ by Lemma 4.7. By induction hypothesis on (P), $AS_{\text{Th}} \vdash \varphi T_1 = \varphi T_n$. By Lemma 5.1,

$$AS_{\text{Th}} \vdash T = T + \varphi_1 \tau. T_1 = T + \varphi \tau. T_1 = T + \varphi \tau. T_n. \quad (18)$$

Using the inductive proof of (P) given below, one can establish that

$$AS_{\text{Th}} \vdash \varphi T_n = \varphi(T_n + T). \quad (19)$$

It follows from (18), (19) and C-law that $AS_{\text{Th}} \vdash T = T + \varphi T_n$.

We now turn to the inductive proof of (P). Consider a summand $\varphi_i \lambda_i. S_i$ of S . The action $\sum_{i \in I} \varphi_i \lambda_i. S_i \xrightarrow{\lambda_i}_{\varphi_i} S_i$ must be simulated by the term $\sum_{j \in J} \psi_j \lambda_j. T_j$. There are three cases according to the shape of λ_i .

— $\lambda_i = \tau$. There is a boolean Th-partition $\{\varphi_i^k\}_{k \in K}$ of φ_i on $fv(S|T)$ and a collection $\{T \Rightarrow_{\psi_k'} T_k'\}_{k \in K}$ of actions such that for each $k \in K$ the followings are valid.

- (1) $\text{Th} \vdash \varphi_i^k \Rightarrow \psi_k'$.
- (2) $\varphi_i^k S \simeq_{\text{Th}}^s \varphi_i^k T_k'$.
- (3) One of the following properties holds:
 - (a) $\varphi_i^k S_i \simeq_{\text{Th}}^s \varphi_i^k T_k'$;
 - (b) $T_k' \xrightarrow{\tau}_{\psi_k'} T_k''$ for some ψ_k'', T_k'' such that $\text{Th} \vdash \varphi_i^k \Rightarrow \psi_k''$ and $\varphi_i^k S_i \simeq_{\text{Th}}^s \varphi_i^k T_k''$.

If (3a) is the case then $\varphi_i^k S_i \simeq_{\text{Th}}^s \varphi_i^k T_k' \simeq_{\text{Th}}^s \varphi_i^k S \simeq_{\text{Th}}^s \varphi_i^k T$. One may continue the inductive proof to establish that

$$AS_{\text{Th}} \vdash \varphi_i^k S_i = \varphi_i^k T. \quad (20)$$

If (3b) is the case then the induction hypothesis on (P) gives

$$AS_{\text{Th}} \vdash \varphi_i^k S_i = \varphi_i^k T_k''. \quad (21)$$

By the inductive proof of (S), one has

$$AS_{\text{Th}} \vdash T = T + \varphi_i^k T_k' = T + \varphi_i^k (T_k' + \psi_k'' \tau. T_k''). \quad (22)$$

Putting (20), (21) and (22) together, one gets that

$$AS_{\text{Th}} \vdash T + S = T + \sum_{k \in K'} \varphi_i^k \tau. T + \sum_{i' \in I \setminus \{i\}} \varphi_{i'} \lambda_{i'}. S_{i'}, \quad (23)$$

where K' is some subset of K .

— $\lambda_i = \bar{a}(t)$. In this case the validity of the following equality can be easily checked.

$$AS_{\text{Th}} \vdash T + S = T + \sum_{i' \in I \setminus \{i\}} \varphi_{i'} \lambda_{i'}. S_{i'}. \quad (24)$$

— $\lambda_i = a(x)$. By definition there are a boolean Th-partition $\{\varphi_i^k\}_{k \in K}$ of φ_i on $fv(S|T)$ and, for each $k \in K$ a boolean Th-partition $\{\phi_k^j\}_{j \in J_k}$ of \top on $\{x\}$, and moreover a collection $\{T \Rightarrow_{\psi_{kj}} T_{kj} \xrightarrow{a(x)}_{\psi_{kj}'} T_{kj}'\}_{k \in K, j \in J_k}$ such that

$$\text{Th} \vdash \varphi_i^k \Rightarrow \psi_{kj} \psi_{kj}'$$

for all $k \in K$ and all $j \in J_k$, and

$$\begin{aligned}\varphi_i^k S &\simeq_{\text{Th}}^s \varphi_i^k T_{kj}, \\ \varphi_i^k \phi_k^j S_i &\simeq_{\text{Th}}^s \varphi_i^k \phi_k^j T'_{kj}\end{aligned}$$

for every $k \in K$ and every $j \in J_k$. By induction hypothesis on (P),

$$AS_{\text{Th}} \vdash \varphi_i^k \phi_k^j S_i = \varphi_i^k \phi_k^j T'_{kj} \quad (25)$$

for every $j \in J$. The following rewriting makes use of the above equality and the inductive proof of (S).

$$\begin{aligned}AS_{\text{Th}} \vdash T &= T + \sum_{k \in K, j \in J_k} \psi_{kj} \psi'_{kj} a(x). T'_{kj} \\ &\stackrel{(*)}{=} T + \sum_{k \in K} \varphi_i^k \sum_{j \in J_k} a(x). T'_{kj} \\ &\stackrel{S6}{=} T + \sum_{k \in K} \varphi_i^k \left(\sum_{j \in J_k} a(x). T'_{kj} + a(x). \sum_{j \in J_k} \phi_k^j \tau. T'_{kj} \right) \\ &\stackrel{S4}{=} T + \sum_{k \in K} \varphi_i^k \left(\sum_{j \in J_k} a(x). T'_{kj} + a(x). \sum_{j \in J_k} \varphi_i^k \phi_k^j \tau. T'_{kj} \right) \\ &\stackrel{(25)}{=} T + \sum_{k \in K} \varphi_i^k \left(\sum_{j \in J_k} a(x). T'_{kj} + a(x). \sum_{j \in J_k} \varphi_i^k \phi_k^j \tau. S_i \right) \\ &\stackrel{(*)}{=} T + \sum_{k \in K} \varphi_i^k a(x). \sum_{j \in J_k} \phi_k^j \tau. S_i \\ &= T + \sum_{k \in K} \varphi_i^k a(x). \tau. S_i \\ &= T + \varphi_i a(x). S_i,\end{aligned}$$

where (*) is due to Lemma 5.1. We have essentially proved that (24) also holds when λ_i is an input prefix.

By carrying out the induction on the number of summands of S one may conclude with S5 that $AS_{\text{Th}} \vdash T + S = T + \phi \tau. T$ for some ϕ . It then follows from Lemma 5.3 that $AS_{\text{Th}} \vdash T + S = T$. Symmetrically $AS_{\text{Th}} \vdash S + T = S$. Hence $AS_{\text{Th}} \vdash S = T$.

6. TURING COMPLETENESS

The value-passing calculi are rudimentary process models. The question concerning their expressiveness has to be settled. Which value-passing calculi are for instance complete in the sense that all recursive functions [Rogers 1987] are definable? To answer the question we need to make it clear how the natural numbers are defined in a value-passing calculus. From the operational point of view, a natural number system is not just an infinite set of pairwise distinct closed Σ -terms, but also an effective way of generating these Σ -terms.

Definition 6.1. A *numeric system* for \mathbb{VPC}_{Th} consists of a countable subset $\{\widehat{0}, \widehat{1}, \widehat{2}, \dots, \widehat{n}, \dots\}$ of \mathbb{T}_{Σ}^0 and a \mathbb{VPC}_{Th} -term $S_d(x)$ that satisfy the followings:

- (1) The variable x is the only free variable appearing in $S_d(x)$.

(2) $\text{Th} \vdash \widehat{p} \neq \widehat{q}$ for all $p, q \in \mathbf{N}$ such that $p \neq q$.

(3) Every action sequence of $S_d(\widehat{n})$ is of the form $S_d(\widehat{n}) \Longrightarrow \xrightarrow{\widehat{d}(\widehat{n+1})} =_{\text{Th}} \mathbf{0}$.

It is clear from (3) of Definition 6.1 that every action sequence of $S_d(\widehat{n})$ is actually of the form $S_d(\widehat{n}) \rightarrow^* \xrightarrow{\widehat{d}(\widehat{n+1})} =_{\text{Th}} \mathbf{0}$ and $S_d(\widehat{n}) =_{\text{Th}} \widehat{d}(\widehat{n+1})$.

Using a numeric system, we may talk about functions in a value-passing calculus. The *predecessor function* \mathbf{p} for instance can be defined in such a calculus. Let's explain the idea in \mathbb{VPC} . Suppose we would like to have the \mathbb{VPC} -process $a(x).\bar{b}(\mathbf{p}(x))$. It can be defined as the following process

$$a(x).(c(\bar{c}(\mathbf{0}) \mid !c(y).if\ x = s(y)\ \text{then}\ \bar{b}(y)\ \text{else}\ \bar{c}(s(y))).$$

The process diverges when given the input $\mathbf{0}$. As is demonstrated by this example, each application of the predecessor function is implemented by an additional \mathbb{VPC} -term. In sequel we shall make use of the predecessor function without worrying about how a particular occurrence of the function is implemented.

An n -ary partial function $f(x_1, \dots, x_n)$ is *definable* in \mathbb{VPC}_{Th} with respect to the numeric system $\langle \{\widehat{0}, \widehat{1}, \widehat{2}, \dots, \widehat{n}, \dots\}, S_d(x) \rangle$ if, for all names a_1, \dots, a_n, b , there is a process $I(a_1, \dots, a_n, b)$ of the form

$$a_1(x_1) \dots a_n(x_n).T$$

such that the following properties hold:

—If $f(p_1, \dots, p_n)$ is defined, then all the action sequences of $T\{\widehat{p}_1/x_1, \dots, \widehat{p}_n/x_n\}$ are finite and are of the following form

$$T\{\widehat{p}_1/x_1, \dots, \widehat{p}_n/x_n\} \Longrightarrow \xrightarrow{\bar{b}(\widehat{p})} T' =_{\text{Th}} \mathbf{0}$$

where $p = f(p_1, \dots, p_n)$.

—If $f(p_1, \dots, p_n)$ is undefined, then $T\{\widehat{p}_1/x_1, \dots, \widehat{p}_n/x_n\}$ can only perform τ -action sequences and all its τ -action sequences are divergent.

We say that $f(x_1, \dots, x_n)$ is defined by $I(a_1, \dots, a_n, b)$ at a_1, \dots, a_n, b . A set of partial functions is definable with respect to a numeric system if every member of the set is definable with respect to the numeric system. A set of partial functions is definable in \mathbb{VPC}_{Th} if it is definable with respect to some numeric system of \mathbb{VPC}_{Th} .

If a function is definable in \mathbb{VPC}_{Th} with respect to $\langle \{\widehat{0}, \widehat{1}, \widehat{2}, \dots, \widehat{n}, \dots\}, S_d(x) \rangle$, we can design a procedure that, upon receiving the natural numbers p_1, \dots, p_n , traverses the derivation tree of $T\{\widehat{p}_1/x_1, \dots, \widehat{p}_n/x_n\}$. This is well defined since every \mathbb{VPC}_{Th} -term is finite branching. The procedure terminates with the result p if $T\{\widehat{p}_1/x_1, \dots, \widehat{p}_n/x_n\}$ export \widehat{p} at some b . It diverges otherwise. According to the Church-Turing Thesis, this procedure defines a recursive function. We conclude that all functions definable in a value-passing calculus are recursive.

The opposite question asks if all the recursive functions can be defined in a value-passing calculus. This amounts to asking if the value-passing calculus is Turing complete.

Definition 6.2. A value-passing calculus \mathbb{VPC}_{Th} is *Turing complete* if all the recursive functions are definable in \mathbb{VPC}_{Th} .

The next proposition provides a basic fact about the expressiveness of the value-passing calculi.

PROPOSITION 6.3. *A value-passing calculus \mathbb{VPC}_{Th} is Turing complete if and only if it has a numeric system.*

PROOF. The implication in one direction is clear. Now suppose \mathbb{VPC}_{Th} has a numeric system $\langle \{\widehat{0}, \widehat{1}, \widehat{2}, \dots, \widehat{n}, \dots\}, S_d(x) \rangle$. We are going to show that the recursive functions [Rogers 1987] are definable in \mathbb{VPC}_{Th} . The inductive definition is as follows:

—The *successor function* is interpreted at a, b as the process $Succ(a, b)$ of the form

$$Succ(a, b) = a(x).(d)(S_d(x) \mid d(z).\bar{b}(z)).$$

—The *constant function* is interpreted at a_1, \dots, a_n, b as the following process

$$Const(a_1, \dots, a_n, b) = a_1(x_1) \cdots a_n(x_n).\bar{b}(\widehat{n}).$$

—The interpretation $Pred(a, b)$ of the *predecessor function* is defined by the following process

$$a(x).(c)(\bar{c}(\widehat{0}) \mid !c(y).(d)(S_d(y) \mid d(z).if\ x = z\ then\ \bar{b}(y)\ else\ \bar{c}(z))).$$

—The *n-ary projection function* is interpreted at a_1, \dots, a_n, b as the process

$$Proj_n^i(a_1, \dots, a_n, b) = a_1(x_1) \cdots a_n(x_n).\bar{b}(x_i),$$

where $1 \leq i \leq n$.

—The *composition function* can be interpreted in a straightforward manner.

—Suppose $G(c_1, \dots, c_n, d, e, b)$ and $H(c_1, \dots, c_n, b)$ are the interpretations of two recursive functions. The interpretation $F(a_1, \dots, a_{n+1}, b)$ of the *recursion function* at $a_1, \dots, a_n, a_{n+1}, b$ is

$$a_1(x_1) \cdots a_{n+1}(x_{n+1}).(f)(\bar{f}(\widehat{0}, \widehat{1}, x_{n+1}, \widehat{0}) \mid (c_1 \dots c_n)(!\bar{c}_1(x_1) \mid \dots \mid !\bar{c}_n(x_n) \mid Rec)),$$

where Rec stands for the following process

$$\begin{aligned} & !f(y, y', z, v).if\ \widehat{0} = y = z\ then\ H(c_1, \dots, c_n, b) \\ & \quad else\ if\ \widehat{0} = y \wedge y' \leq z\ then\ (e)(H(c_1, \dots, c_n, e) \mid e(v).\bar{f}(\widehat{0}, \widehat{1}, z, v)) \\ & \quad \quad else\ if\ \widehat{0} < y \wedge y' = z\ then\ (d)(d')(G(c_1, \dots, c_n, d, d', b) \mid \bar{d}(y) \mid \bar{d}'(v)) \\ & \quad \quad \quad else\ if\ \widehat{0} < y \wedge y' < z\ then\ (d)(d')(e)(G(c_1, \dots, c_n, d, d', e) \mid \bar{d}(y) \mid \bar{d}'(v) \\ & \quad \quad \quad \quad \mid e(v).\bar{f}(S_d(y), S_d(y'), z, v)). \end{aligned}$$

—Suppose $G(a_1, \dots, a_{n+1}, b)$ is the interpretation of a recursive function. The *minimalization function* is interpreted at $a_1, \dots, a_n, a_{n+1}, b$ by the process

$$a_1(x_1) \cdots a_{n+1}(x_{n+1}).Mu,$$

where Mu is the following process

$$\begin{aligned} & (a_1 \dots a_{n+1})(!\bar{a}_1(x_1) \mid \dots \mid !\bar{a}_n(x_n) \mid \bar{f}(\widehat{0})) \\ & \quad \mid !f(z).(\bar{a}_{n+1}(z) \mid (c)(G(a_1, \dots, a_{n+1}, c) \mid c(v).if\ v = \widehat{0}\ then\ \bar{b}(z)\ else\ \bar{f}(S_d(z)))). \end{aligned}$$

We are done. \square

Proposition 6.3 adds weight to Definition 6.1 since a numeric system is intuitively a minimal requirement for a value-passing calculus to simulate all the recursive functions. The \mathbb{VPC}_{Th} -term $S_d(x)$ is nothing but an implementation of the successor function.

7. VPC AND RECURSION THEORY

The previous three sections have developed a rigorous theory for the value-passing calculi. When applied to a particular boolean complete first order theory, this general theory immediately generates an operational semantics and an observation theory for that calculus. We have studied only one boolean complete first order theory, the Peano Theory PA defined in Fig. 2. So in this section we take a closer look at the value-passing calculus defined on top of PA. We will write $=_{\mathbb{VPC}}$ for the absolute equality of \mathbb{VPC} and $\simeq_{\mathbb{VPC}}^s$ for the symbolic bisimilarity of \mathbb{VPC} .

According to our general setting, \mathbb{VPC} should have a number of virtues. Let's summarize its key features:

- \mathbb{VPC} is Turing complete. So it is among the good value-passing calculi.
- The validity of the boolean propositions is known to be decidable. This is the reason for us to see \mathbb{VPC} as a programming language. Our familiarity with the standard model \mathbf{N} helps confidence in programming with \mathbb{VPC} .
- The simplicity of our Peano theory offers a nice algebraic theory of \mathbb{VPC} . Formal comparison of \mathbb{VPC} against other well known process calculi is not only possible, but also instructive.

It is difficult to think of a value-passing calculus that is weaker than \mathbb{VPC} but is still expressive enough. We now elaborate on this point.

To start with observe that the binary relation $<$ is not absolutely necessary for \mathbb{VPC} . The following proposition explains why.

PROPOSITION 7.1. *For each \mathbb{VPC} -process P there is some \mathbb{VPC} -process P' such that $P =_{\mathbb{VPC}} P'$ and that P' contains no occurrence of the relation symbol $<$.*

PROOF. The basic idea is that the boolean value of a closed atomic Σ_{PA} -expression $t < t'$ can be calculated within \mathbb{VPC} . Given a \mathbb{VPC} -process P , we can translate it into an equal \mathbb{VPC} -process P' . The encoding is structural on composition, localization and replication operators. The interpretation of the guarded choice and the conditional are similar. We take a look at how the latter is translated. Consider the \mathbb{VPC} -term $S \stackrel{\text{def}}{=} \text{if } \varphi \text{ then } T$. The interpretation of S is given by the following term

$$(c)(\llbracket \varphi \rrbracket_c \mid c(z). \text{if } z = \underline{1} \text{ then } T')$$

where T' is the interpretation of T and $\llbracket \varphi \rrbracket_c$ is structurally defined as follows:

- If φ is $t = t'$, then $\llbracket \varphi \rrbracket_c$ is $\text{if } t = t' \text{ then } \bar{c}(\underline{1}) \text{ else } \bar{c}(\underline{0})$.
- If φ is $t < t'$, then $\llbracket \varphi \rrbracket_c$ is the following term

$$\bar{d}(t).\bar{e}(t') \mid !d(x).e(y). \text{if } y = \underline{0} \text{ then } \bar{c}(\underline{0}) \text{ else if } x = \underline{0} \text{ then } \bar{c}(\underline{1}) \text{ else } \bar{d}(\mathbf{p}(x)).\bar{e}(\mathbf{p}(y)).$$
- If φ is $\varphi' \wedge \varphi''$, then $\llbracket \varphi \rrbracket_c$ is

$$(de)(\llbracket \varphi' \rrbracket_d \mid \llbracket \varphi'' \rrbracket_e \mid d(y).e(z). \text{if } y = 1 \wedge z = 1 \text{ then } \bar{c}(\underline{1}) \text{ else } \bar{c}(\underline{0})).$$

—If φ is $\varphi' \vee \varphi''$, then $\llbracket \varphi \rrbracket_c$ is

$$(de)(\llbracket \varphi' \rrbracket_d \mid \llbracket \varphi'' \rrbracket_e \mid d(y).e(z).if\ y = 1 \vee z = 1\ then\ \bar{c}(1)\ else\ \bar{c}(0)).$$

The equality $P =_{\text{VPC}} P'$ holds in an obvious way. \square

7.1 Minimality of VPC

In this section we prove that VPC plays an authentic role in all the Turing complete value-passing calculi. Suppose $\langle \{\widehat{0}, \widehat{1}, \widehat{2}, \dots, \widehat{n}, \dots\}, S_d(x) \rangle$ is a numeric system of a Turing complete model VPC_{Th} . One could define a translation $\llbracket - \rrbracket_{\text{Th}}$ from VPC to VPC_{Th} using the ideas described in Fig. 7. The translation of the action labels $\llbracket \lambda \rrbracket_{\text{Th}}$ can be defined by

$$\llbracket \lambda \rrbracket_{\text{Th}} \stackrel{\text{def}}{=} \begin{cases} \tau, & \text{if } \lambda = \tau, \\ a(\widehat{n}), & \text{if } \lambda = a(\underline{n}), \\ \bar{a}(\widehat{n}), & \text{if } \lambda = \bar{a}(\underline{n}). \end{cases}$$

Three aspects of the encoding call for explanation.

- One is that we have identified the set of the term variables of VPC_{PA} with the term variables of VPC_{Th} .
- The second is that the operation $D_c(-)$ is defined as follows: for a term $t \equiv s^k(x)$ with $k > 0$, $D_c(t)$ is the following term

$$(c_0)(\bar{c}_0(x) \mid c_0(z_0).(c_1)(S_{c_1}(z_0) \mid c_1(z_1).(\dots c_{k-1}(z_{k-1}).S_c(z_{k-1}) \dots))).$$

The idea is that the term $s^k(x)$, for $k > 0$, is translated to an element of the numeric system $\langle \{\widehat{0}, \widehat{1}, \widehat{2}, \dots, \widehat{n}, \dots\}, S_d(x) \rangle$, achieved by counting the elements from x up to $\widehat{k+x}$ using $S_d(x)$.

- The third is that we have not given the encoding of the choice term $\sum_{i \in I} \varphi_i \lambda_i . T_i$. The reader can readily give it by himself/herself. It is simply a combination of the encodings for the prefix terms and the conditional terms. The translation of $\sum_{1 \leq i \leq k} \varphi_i \lambda_i . T_i$ takes the following form

$$(\tilde{c})(\prod_{1 \leq i \leq n_1} D_{c_i^1}(t_i^1) \mid \dots \mid \prod_{1 \leq i \leq n_k} D_{c_i^k}(t_i^k) \mid c_1^1(z_1^1) \dots c_{n_k}^k(z_{n_k}^k). \sum_{1 \leq i \leq k} \llbracket \varphi_i \lambda_i . T_i \rrbracket_{\text{Th}}),$$

where $\tilde{c} = c_1^1 \dots c_{n_k}^k$.

It is easy to see that the translation is sound and complete with respect to the operational semantics in the sense of the next proposition.

PROPOSITION 7.2. *Suppose P is a VPC-process that does not contain any occurrences of $<$. The following correspondences hold:*

- (i) *If $P \xrightarrow{\lambda} P'$ then $\llbracket P \rrbracket_{\text{Th}} \xrightarrow{\llbracket \lambda \rrbracket_{\text{Th}}} P_1 =_{\text{VPC}_{\text{Th}}} \llbracket P' \rrbracket_{\text{Th}}$ for some VPC_{Th} -process P_1 ;*
- (ii) *If $\llbracket P \rrbracket_{\text{Th}} \xrightarrow{\lambda} P_1$ then $P \xrightarrow{\lambda'} P'$ for some VPC-process P' and some λ' such that $P_1 =_{\text{VPC}_{\text{Th}}} \llbracket P' \rrbracket_{\text{Th}}$ and $\llbracket \lambda' \rrbracket_{\text{Th}} = \lambda$.*

It is possible to strengthen Proposition 7.2. In fact the composition of the relation

$$\{(P, \llbracket P \rrbracket_{\text{Th}}) \mid P \text{ is a VPC process}\}$$

$$\begin{aligned}
\llbracket \mathbf{0} \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} \mathbf{0}, \\
\llbracket \tau.T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} \tau.\llbracket T \rrbracket_{\text{Th}}, \\
\llbracket a(x).T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} a(x).\llbracket T \rrbracket_{\text{Th}}, \\
\llbracket \bar{a}(t).T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} \begin{cases} \bar{a}(\widehat{n}).\llbracket T \rrbracket_{\text{Th}}, & \text{if } t \equiv \underline{n}, \\ \bar{a}(x).\llbracket T \rrbracket_{\text{Th}}, & \text{if } t \equiv x, \\ (c)(D_c(t) \mid c(z).\bar{a}(z).\llbracket T \rrbracket_{\text{Th}}), & \text{if } t \equiv s^l(x) \text{ for some } l > 0; \end{cases} \\
\llbracket S \mid T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} \llbracket S \rrbracket_{\text{Th}} \mid \llbracket T \rrbracket_{\text{Th}}, \\
\llbracket (a)T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} (a)\llbracket T \rrbracket_{\text{Th}}, \\
\llbracket \text{if } \varphi \text{ then } T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} (c_1 \dots c_k) \left(\prod_{1 \leq i \leq k} D_{c_i}(t_i) \mid c_1(z_1) \dots c_k(z_k) \cdot \text{if } \varphi' \text{ then } \llbracket T \rrbracket_{\text{Th}} \right), \\
&\text{where } \varphi' \equiv \varphi'' \{ \widehat{n}_1/n_1, \dots, \widehat{n}_j/n_j \}, \varphi \equiv \varphi'' \{ t_1/z_1, \dots, t_k/z_k \}, \\
&\underline{n}_1, \dots, \underline{n}_j \text{ are the numerals in } \varphi, \text{ and } t_1, \dots, t_k \text{ are the terms} \\
&\text{in } \varphi \text{ that are of the form } s^l(x) \text{ for some } l > 0; \text{ we may regard} \\
&\text{if } \varphi' \text{ then } \llbracket T \rrbracket_{\text{Th}} \text{ as } \llbracket \text{if } \varphi \text{ then } T \rrbracket_{\text{Th}}; \\
\llbracket !a(x).T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} !a(x).\llbracket T \rrbracket_{\text{Th}}, \\
\llbracket !\bar{a}(t).T \rrbracket_{\text{Th}} &\stackrel{\text{def}}{=} \begin{cases} !\bar{a}(\widehat{n}).\llbracket T \rrbracket_{\text{Th}}, & \text{if } t \equiv \underline{n}, \\ !\bar{a}(x).\llbracket T \rrbracket_{\text{Th}}, & \text{if } t \equiv x, \\ (c)(D_c(t) \mid c(z)!\bar{a}(z).\llbracket T \rrbracket_{\text{Th}}), & \text{if } t \equiv s^l(x) \text{ for some } l > 0. \end{cases}
\end{aligned}$$

Fig. 7. Encoding of VPC into Turing Complete VPC_{Th} .

with $=_{\text{Th}}$ is a subbisimilarity. A subbisimilarity is a generalization of the absolute equality from a binary relation on one model to a binary relation from one calculus to another. See [Fu 2011b] for details.

It is a little step from Proposition 7.2 to the observational soundness and completeness.

THEOREM 7.3. *Suppose P, Q are VPC-processes that do not contain any occurrences of $<$. Then $P =_{\text{VPC}} Q$ if and only if $\llbracket P \rrbracket_{\text{Th}} =_{\text{VPC}_{\text{Th}}} \llbracket Q \rrbracket_{\text{Th}}$.*

PROOF. In view of Proposition 3.10, we only have to prove the theorem for the external bisimilarities. Let \mathcal{R} be the following relation

$$\left\{ \left((\tilde{c})(P_1 \mid \dots \mid P_k), (\tilde{c})(Q_1 \mid \dots \mid Q_k) \right) \left| \begin{array}{l} c \text{ is } c_1, \dots, c_j \text{ for some } j; P_i \text{ is neither} \\ \text{a composition nor a localization; and} \\ \llbracket P_i \rrbracket_{\text{Th}} \rightarrow^* Q_i, \text{ where } Q_i \text{ is obtained} \\ \text{from } \llbracket P_i \rrbracket_{\text{Th}} \text{ by resolving the } D_c \text{ 's.} \end{array} \right. \right\}.$$

Now \mathcal{R} is a VPC_{Th} -bisimulation up to strong bisimilarity [Milner 1989a]. Notice that every VPC-process is equal to a process of the form $(\tilde{c})(P_1 \mid \dots \mid P_k)$, where for each $i \in \{1, \dots, k\}$, the process P_i is neither a composition nor a localization. \square

So the translation $\llbracket _ \rrbracket_{\text{Th}}$ is correct for the VPC -processes that do not refer to the relation ‘ \langle ’. The restriction can be removed according to Proposition 7.1. We conclude that VPC is a submodel of every Turing complete VPC_{Th} . In other words it is the least expressive among all the Turing complete value-passing calculi.

8. CONCLUSION

The paper presents a treatment to the value-passing calculi at a more formal level than previous studies. A comparison to the related works is summarized as follows:

- The present approach emphasizes that a value-passing calculus should be a self-contained model of computation and interaction. The formal treatment of the logic of the model comes with the decidability requirement. The boolean completeness is singled out as the desirable property. It has taken some time to reach this level of formality. The study of Milner [1989a] was conducted at an *ad hoc* manner. The semantics of his value-passing CCS is defined by translating the model into the pure CCS with arbitrary choice operator. Hoare’s definition of his famous CSP [Hoare 1978; 1985], which is a value-passing calculus, is essentially algebraic. The operational semantics for CSP *a la* pure CCS is introduced in [Brookes 1983; Brookes et al. 1984; Roscoe 1997]. The labeled transition semantics for the value-passing CCS appears in [Hennessy and Ingólfssdóttir 1993a; 1993b]. A more serious attempt to study the operational semantics of the value-passing calculi is reported in the seminal paper by Hennessy and Lin [1995]. The significance of their work is that it points out the indispensable role the first order logic may play in the study of the semantics of the value-passing calculi. The present work can be seen as a further step that completes the picture outlined by Hennessy and Lin [1995].
- The observational equality of this paper is an application of the universal equality of Fu [2011b] to the value-passing calculi. This is an equality that differs from any equalities that have been proposed for the value-passing calculi. The algebraic theory of CSP has been extensively studied [Brookes et al. 1984; Roscoe 1997], with particular emphasis on the trace and failure semantics. In the literature on CSP, it is popular to see a set of algebraic laws as providing an axiomatic semantics. The algebraic theory of the value-passing calculi has been studied with motivation from the denotational semantics [Hennessy 1991; Hennessy and Ingólfssdóttir 1993a; 1993b]. The observational equivalence, the weak bisimilarity, and its symbolic characterization is systematically studied in [Hennessy and Lin 1995]. Proof systems for these equivalences have also been studied in [Hennessy 1991; Hennessy and Ingólfssdóttir 1993a; 1993b; Hennessy and Lin 1996]. These systems are parameterized over proof systems for the logics of data domains. The decidability of our equation system AS_{Th} compares favorably to these systems. The algorithmic aspect of the equivalence checking for the value-passing calculi is elaborated in [Lin 1993; 1996; 1998]. Our treatment to the value-passing calculi may cast new light on the equivalence checking algorithms for the value-passing processes.

A survey of the symbolic approach to the value-passing calculi and the related issues is given by Ingólfssdóttir and Lin [2001].

Our formalization of the value-passing calculi makes it possible to carry out a refined study on the expressiveness of these models. There have been early efforts that attack the expressiveness issue. See for example [Palamidessi 2003; Fu and Lu 2010]. However it is fair to say that none of the results obtained so far is conclusive. In this paper we have studied the absolute expressiveness of the value-passing calculi by characterizing the Turing complete value-passing calculi in terms of the numeric systems. We have shown that every value-passing calculus is a faithful extension of the recursion theory. We are not aware of any previous investigation that discusses the Turing completeness problem of the value-passing calculi. We have also studied the relative expressiveness of the value-passing calculi. It is shown in this paper that \mathbb{VPC} is the least expressive value-passing calculus. The minimality result sheds new light on the value-passing calculi studied by previous researchers, all of those models being informal variants of \mathbb{VPC} . It is worth remarking that \mathbb{VPC} is strictly less expressive than the π -calculus [Fu 2011b] and is strictly more expressive than the Interactive Machine Model of Fu [2011b].

We have emphasized the importance of confining our attention to the decidable fragment of the first order logic. The tradeoff is that \simeq_{Th}^s is much stronger than $=_{\text{Th}}$. A challenging task is to prove or disprove that \simeq_{Th}^s and $=_{\text{Th}}$ coincide on the finite control \mathbb{VPC}_{Th} -terms. But a more urgent problem to resolve is the following.

Problem 8.1. Does $\simeq_{\mathbb{VPC}}^s$ coincide with $=_{\mathbb{VPC}}$ on the finite \mathbb{VPC} -terms?

The symbolic bisimilarity studied in this paper is the simplest of its kind. It is too strong for the infinite state processes. Consider the \mathbb{VPC} processes

$$A_0 \stackrel{\text{def}}{=} a(x).if\ x\ \text{is even then } \bar{b}(x) + a(x).if\ x\ \text{is odd then } \bar{b}(x) \quad (26)$$

and

$$A_1 \stackrel{\text{def}}{=} a(x).\bar{b}(x) + a(x).if\ x\ \text{is even then } \bar{b}(x) + a(x).if\ x\ \text{is odd then } \bar{b}(x). \quad (27)$$

The term *if x is even then $\bar{b}(x)$* can be programmed as follows:

$$(c)(\bar{c}(\underline{0}) \mid !c(y).if\ x = y\ \text{then } \bar{b}(x)\ \text{else } \bar{c}(s(s(y)))).$$

The term *if x is odd then $\bar{b}(x)$* is defined similarly. It is clear that $A_0 =_{\mathbb{VPC}} A_1$. On the other hand $A_0 \not\approx_{\mathbb{VPC}}^s A_1$. The transition $A_1 \xrightarrow{a(x)}_{\top} \bar{b}(x)$ can be simulated by A_0 . There is however no boolean PA-partition on $\{x\}$ that witnesses the simulation. If we relax on the boolean restriction, then intuitively the set $\{\exists z.x = 2 * z, \exists z.x = 2 * z + 1\}$ forms a ‘partition’. In order to carry out the investigation along this line, the symbolic approach must be modified in a couple of ways.

- (1) Firstly the proper power of the Peano system should be retained. Specifically the addition operator ‘+’ and the multiplication operator ‘*’ are necessary to produce much more powerful specifications.
- (2) Secondly the symbolic bisimilarity should be defined by a family of relations indexed by the first order formulas [Hennessy and Lin 1995; Ingólfssdóttir and Lin 2001].

It is worth remarking that the introduction of the arithmetic operators does not change the grammar of \mathbb{VPC} . No \mathbb{VPC} -terms would contain any arithmetic oper-

ators. So the logic expressions of \mathbb{VPC} are still decidable. The extra expressive power is only exploited in equality reasoning. We may ask the following question.

Problem 8.2. What is the symbolic theory that exploits the richer Peano theory?

The equation system AS_{Th} provides an effective method to check the symbolic bisimilarity of two finite \mathbb{VPC}_{Th} -terms. A natural question to ask is how to extend AS_{Th} to a complete system for the finite control \mathbb{VPC}_{Th} -terms. Hennessy, Lin and Rathke have discussed the issue in [Hennessy and Lin 1997; Rathke 1997a; 1997b; Hennessy et al. 1997]. Their complete systems are more involved since they deal with parametric definitions, which are more complex than the fixpoint operator. It should be routine to adapt their approach and Milner's original approach [Milner 1989b] to \mathbb{VPC}_{Th} . Additional care should be taken to divergence [Lohrey et al. 2002; 2005; Fu 2011a]. The details are yet to be worked out.

Problem 8.3. How can we extend AS_{Th} to obtain a complete system for the finite control \mathbb{VPC}_{Th} -terms?

There are other aspects of the value-passing calculi that are worth investigating. One could for example take a look at the box equality introduced by Fu and Zhu [2011]. One could also take a look at the algorithm complexities for a number of decidability problems. The general methodology proposed in this paper has laid down a firm foundation for the solutions to these problems.

ACKNOWLEDGMENTS

This work has been supported by NSFC (60873034, 61033002). The author would like to thank the members of BASICS for their interest and feedbacks. In particular he would like to thank Yijia Chen for his instructive discussion on the proof of Theorem 2.2 and to Huan Long and Jianxin Xue for their careful proof-reading of the paper.

REFERENCES

- BROOKES, S. 1983. A model of communicating sequential processes. Ph.D. thesis, Oxford University.
- BROOKES, S., HOARE, C., AND ROSCOE, A. 1984. A theory of communicating sequential processes. *Journal of ACM* 31, 560–599.
- DE NICOLA, R., MANTANARI, U., AND VAANDRAGER, F. 1990. Back and forth bisimulations. In *Proc. CONCUR'90*. Lecture Notes in Computer Science, vol. 458. 152–165.
- FU, Y. 2011a. Axioms for computation. *Working paper, the paper is downloadable at <http://basics.sjtu.edu.cn/~yuxi/>.*
- FU, Y. 2011b. Theory of interaction. *Submitted for publication*.
- FU, Y. AND LU, H. 2010. On the expressiveness of interaction. *Theoretical Computer Science* 411, 1387–1451.
- FU, Y. AND YANG, Z. 2003. Tau laws for pi calculus. *Theoretical Computer Science* 308, 55–130.
- FU, Y. AND ZHU, H. 2011. The name-passing calculus. *Submitted for publication*.
- HENNESSY, M. 1991. A proof system for communicating processes with value-passing. *Journal of Formal Aspects of Computer Science* 3, 346–366.
- HENNESSY, M. AND INGÓLFSDÓTTIR, A. 1993a. Communicating processes with value-passing and assignment. *Journal of Formal Aspects of Computing* 5, 432–466.
- HENNESSY, M. AND INGÓLFSDÓTTIR, A. 1993b. A theory of communicating processes with value-passing. *Information and Computation* 107, 202–236.

- HENNESSY, M. AND LIN, H. 1995. Symbolic bisimulations. *Theoretical Computer Science* 138, 353–369.
- HENNESSY, M. AND LIN, H. 1996. Proof systems for message passing process algebras. *Formal Aspects of Computing* 8, 379–407.
- HENNESSY, M. AND LIN, H. 1997. Unique fixpoint induction for message-passing process calculi. In *Proc. Computing: Australian Theory Symposium (CAT'97)*. Vol. 8. 122–131.
- HENNESSY, M., LIN, H., AND RATHKE, J. 1997. Unique fixpoint induction for message-passing process calculi. *Science of Computer Programming* 41, 241–275.
- HENNESSY, M. AND MILNER, R. 1985. Algebraic laws for nondeterminism and concurrency. *Journal of ACM* 32, 137–161.
- HOARE, C. 1978. Communicating sequential processes. *Communications of ACM* 21, 666–677.
- HOARE, C. 1985. *Communicating Sequential Processes*. Prentice Hall.
- INGÓLFSDÓTTIR, A. AND LIN, H. 2001. A symbolic approach to value-passing processes. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. North-Holland, 427–478.
- LIN, H. 1993. A verification tool for value-passing processes. In *Proceedings of 13th International Symposium on Protocol Specification, Testing and Verification*. IFIP Transactions.
- LIN, H. 1996. Symbolic transition graphs with assignment. In *Proc. CONCUR '96*. Lecture Notes in Computer Science, vol. 1119. 50–65.
- LIN, H. 1998. “on-the-fly” instantiation of value-passing processes. In *Proc. FORTH/PSTV '98*.
- LOHREY, M., D'ARGENIO, P., AND HERMANN, H. 2002. Axiomatising divergence. In *Proc. ICALP 2002*. Lecture Notes in Computer Science, vol. 2380. Springer, 585–596.
- LOHREY, M., D'ARGENIO, P., AND HERMANN, H. 2005. Axiomatising divergence. *Information and Computation* 203, 115–144.
- MILNER, R. 1989a. *Communication and Concurrency*. Prentice Hall.
- MILNER, R. 1989b. A complete axiomatization system for observational congruence of finite state behaviours. *Information and Computation* 81, 227–247.
- MILNER, R., PARROW, J., AND WALKER, D. 1992. A calculus of mobile processes. *Information and Computation* 100, 1–40 (Part I), 41–77 (Part II).
- PALAMIDESI, C. 2003. Comparing the expressive power of the synchronous and the asynchronous π -calculus. *Mathematical Structures in Computer Science* 13, 685–719.
- PARK, D. 1981. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science* 104, 167–183.
- PARROW, J. AND SANGIORGI, D. 1995. Algebraic theories for name-passing calculi. *Information and Computation* 120, 174–197.
- PRIESE, L. 1978. On the concept of simulation in asynchronous, concurrent systems. *Progress in Cybernetics and Systems Research* 7, 85–92.
- RATHKE, J. 1997a. Symbolic techniques for value-passing calculi. Ph.D. thesis, University of Sussex.
- RATHKE, J. 1997b. Unique fixpoint induction for value-passing processes. In *Proc. LICS '97*. IEEE Press.
- ROGERS, H. 1987. *Theory of Recursive Functions and Effective Computability*. MIT Press.
- ROSCOE, A. 1997. *The Theory and Practice of Concurrency*. Prentice Hall.
- SANGIORGI, D. 1993. From π -calculus to higher order π -calculus—and back. In *Proc. TAPSOFT'93*. Lecture Notes in Computer Science, vol. 668. 151–166.
- THOMSEN, B. 1989. A calculus of higher order communicating systems. In *Proc. POPL'89*. 143–154.
- THOMSEN, B. 1993. Plain chocs — a second generation calculus for higher order processes. *Acta Informatica* 30, 1–59.
- THOMSEN, B. 1995. A theory of higher order communicating systems. *Information and Computation* 116, 38–57.
- VAN GLABBEK, R. AND WEIJLAND, W. 1989. Branching time and abstraction in bisimulation semantics. In *Information Processing'89*. North-Holland, 613–618.
- ACM Journal Name, Vol. V, No. N, November 2011.