The Value-Passing Calculus

Yuxi Fu

BASICS, Department of Computer Science, Shanghai Jiaotong University
MOE-MS Key Laboratory for Intelligent Computing and Intelligent Systems

Abstract. 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 complete first order theory with 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 Turing complete value-passing calculi.

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 [Mil89a], 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 [Fu12b], the basic theory of CCS does generalize to many process calculi. The value-passing CCS [Mil89a] 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, Parrow and Walker [MPW92], 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 stronger model. The process-passing calculi, or the higher order calculi [San93, Tho89, Tho93, Tho95], have typically processes as the contents of communications. This seemingly powerful communication mechanism turns out to be much less expressive than the

Z. Liu, J. Woodcock, and H. Zhu (Eds.): He Festschrift, LNCS 8051, pp. 166–195, 2013.

π-calculus [Fu12b]. 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.

The lightweight treatment of the oracle models/logics is an obstacle to both theoretical study and application. We mention three immediate consequences.

1. Deep theoretical investigations are inevitably hindered by the open-ended approach. For example it is not always possible to compare the expressiveness of a value-passing calculus to another concurrent model. 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 problems, 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 $M(x)$ defined by the following recursive equality.

$$M(x) = \mathit{if}\, \varphi(x)\ \mathit{then}\, \overline{a}(f(t))\ \mathit{else}\, M(x+1). \tag{1}$$

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 $\mathit{if}\, y = y\ \mathit{then}\, P\ \mathit{else}\, Q$ can be immediately put into action. On the other hand the behavior of the process expression $\mathit{if}\, y = z\ \mathit{then}\, P\ \mathit{else}\, Q$ 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. As the Incompleteness Theorem of Gödel [G$\ddot{3}$1] tells us, the set of the statements true in a model is far from being recursively enumerable. But an implemented system can only generate a recursively enumerable set of theorems. 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. Since the set of the theorems of a first order theory is typically creative [Cut80], 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 the Incompleteness Theorem, one looks for decidable theories in which a sentence is either provably true or provably false. In implementation one actually asks for more than decidability. It has been shown that validity checking for a decidable theory could be super exponential in complexity [FR74, Opp78]. So normally 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. If φ contains the free variables $x_1, \ldots, x_n$, then proving φ is equivalent to proving $\forall x_1 \ldots \forall x_n . \varphi$. There are situations, for example in equivalence checking algorithm, where formulas with free variables must be dealt with. This suggests to look for first order theories that are considerably weaker than say Peano Arithmetic.

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 $\varphi(0), \varphi(1), \varphi(2), \ldots$ 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 result 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 $\mathbf{0}$. In a basic model processes like (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 calculus is designed by taking the following 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 would be better not to define recursions by recursive definitions parameterized over value variables. 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. Section 3 studies the operational and the observational semantics of the value-passing calculi. Section 4 takes a look at symbolic approximation to the absolute equality. Section 5 provides a proof system for the finite terms. 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. Section 8 concludes with discussions on future research.

BA	$P\{P_1/X_1, \ldots, P_n/X_n\}$	P is a tautology
EQ1	$t = t$	
EQ2	$s = t \Rightarrow t = s$	
EQ3	$r = s \wedge s = t \Rightarrow r = t$	
CG1	$\bigwedge_{i=1}^{k} t_i = t'_i \Rightarrow \mathsf{f}(t_1, \ldots, t_k) = \mathsf{f}(t'_1, \ldots, t'_k)$	f is a k-ary function
CG2	$\bigwedge_{i=1}^{k} t_i = t'_i \Rightarrow \mathsf{r}(t_1, \ldots, t_k) \Rightarrow \mathsf{r}(t'_1, \ldots, t'_k)$	r is 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

2 Decidable Theory

Let $\mathbf{N}$ be the set of natural numbers. A *vocabulary* $\Sigma = (\mathsf{F}, \mathsf{R}, \mathsf{a})$ consists of two disjoint nonempty countable sets and one function: F is a finite set of *function symbols*; R is a finite set of *relation symbols*; and $\mathsf{a} : \mathsf{F} \cup \mathsf{R} \to \mathbf{N}$ is an *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 $\mathsf{F} \cup \mathsf{R}$ 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 *equality relation* $=$.

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

- $\mathsf{V}_\Sigma \subseteq \mathsf{T}_\Sigma$.
- If f is a k-ary function symbol and $t_1, \ldots, t_k$ are Σ-terms, then $\mathsf{f}(t_1, \ldots, t_k)$ is a Σ-term.

A Σ-term is *closed* if it does not contain any Σ-variable, it is *open* otherwise. The set of closed Σ-terms is denoted by T^0_Σ.

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

- The logical *false* $\bot$ is a Σ-expression.
- If r is a k-ary relation symbol and $t_1, \ldots, t_k$ are Σ-terms, then $\mathsf{r}(t_1, \ldots, t_k)$ 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 Σ-sentences is denoted by E^0_Σ. A *boolean* Σ-expression is a quantifier free Σ-expression. In sequel we shall freely use the derived logical connectives $\top, \neg, \wedge, \vee, \Leftrightarrow, \exists$.

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 instantiating all the propositional variables. The EQ-axioms are about the equivalence property, and

PA1	$\forall x.(\mathsf{s}(x) \neq 0)$
PA2	$\forall xy.(\mathsf{s}(x) = \mathsf{s}(y) \Rightarrow x = y)$
PA3	$\forall x.(x = 0 \vee \exists y.\mathsf{s}(y) = x)$
PA4	$\forall x.(x < \mathsf{s}(x))$
PA5	$\forall xy.(x < y \Rightarrow \mathsf{s}(x) \leq y)$
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 Theory PA

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 meta operation substitution is used.

Given a recursive set Γ of Σ-expressions, a *proof* of ψ from Γ is a finite sequence $(\phi_1, \ldots, \phi_n)$ 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;
- $\phi_i \in \Gamma$;
- There are two Σ-expressions φ and $\varphi \Rightarrow \phi_i$ in the proof $(\phi_1, \ldots, \phi_{i-1})$.

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 $\Gamma \vdash \bot$; 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 set Th of Σ-sentences, the elements of Th are called *nonlogical axioms*.

In the context of present paper the most useful first order theory is Presburger Arithmatic [Pre29]. This is what one gets if the multiplication operator '$\times$' is removed from the Peano Arithmetic. For the reasons explained in Section 1 we shall also leave out the addition operator '+'. The axioms of our theory 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 *numeral*

$$\underbrace{\mathsf{s}(\mathsf{s}(\ldots \mathsf{s}}_{i \text{ times}}(0)\ldots)).$$

Similarly we write $\mathsf{s}^i(x)$ for the open Σ_{PA}-term

$$\underbrace{\mathsf{s}(\mathsf{s}(\ldots \mathsf{s}}_{i \text{ times}}(x)\ldots)).$$

Presburger [Pre29] proved that remarkably his arithmetic, and consequently the theory PA defined in Fig. 2, is decidable. This is the foundation for the value-passing calculi studied in the rest of the paper.

Theorem 1. *PA is decidable.*

3 Value-Passing Calculus

According to our discussions in Section 1, we shall focus on the value-passing calculi defined in terms of decidable first order theories. Throughout this paper we assume that Th is a decidable first order theory of type Σ. The value-passing calculus defined on top of Th is denoted by $\mathbb{VPC}_{\mathsf{Th}}$. If Th is PA, the subscript in $\mathbb{VPC}_{\mathsf{Th}}$ 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 $\overline{\mathcal{N}}$ of conames is $\{\overline{a} \mid a \in \mathcal{N}\}$. A *substitution* is a partial map $\sigma : \mathsf{V}_\Sigma \rightharpoonup \mathsf{T}_\Sigma$ whose domain of definition is finite. An *assignment* is a partial map $\rho : \mathsf{V}_\Sigma \rightharpoonup \mathsf{T}^0_\Sigma$ whose domain of definition is cofinite. A substitution is often denoted explicitly by $\{t_1/x_1, \ldots, t_n/x_n\}$. The notations $\rho[x \leftarrow t]$ and $\sigma[x \leftarrow t]$ are understood in the standard interpretation.

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

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

where φ_i is a boolean Σ-expression, and I is a finite indexing set. The notation $\sum_{i\in\{1,\ldots,n\}} \varphi_i\lambda_i.T_i$ stands for either $\sum_{i\in I} \varphi_i a(x).T_i$ or $\sum_{i\in I} \varphi_i \overline{a}(t_i).T_i$. The prefix $a(x)$ is an input primitive that binds the Σ-variable (henceforth just variable) x, and the prefix $\overline{a}(t)$ is an output primitive. We write $fv(\_)$, respectively $bv(\_)$, for the function that returns the set of the free variables, respectively the bound variables; and let $v(\_)$ be $fv(\_) \cup bv(\_)$. A $\mathbb{VPC}_{\mathsf{Th}}$-term is *closed* if it does not contain any free variables. Otherwise it is called an *open* $\mathbb{VPC}_{\mathsf{Th}}$-term. A closed $\mathbb{VPC}_{\mathsf{Th}}$-term is also called a $\mathbb{VPC}_{\mathsf{Th}}$-*process*. We write $\mathcal{P}_{\mathbb{VPC}_{\mathsf{Th}}}$ for the set of the $\mathbb{VPC}_{\mathsf{Th}}$-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 $\prod_{1\le i\le n} T_i$ for the composition $T_1 \mid T_2 \mid \ldots \mid T_n$. The $\mathbb{VPC}_{\mathsf{Th}}$-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 $\varphi_1\lambda_1.T_1 + \ldots + \varphi_n\lambda_n.T_n$ for $\sum_{i\in\{1,\ldots,n\}} \varphi_i\lambda_i.T_i$. It should be remarked that in $\sum_{i\in I} \varphi_i\lambda_i.T_i$ the constructor is $\sum_{i\in I} \varphi_i\lambda_i.\_$. 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 $\mathbb{VPC}_{\mathsf{Th}}$-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 $(c)(E\{c(z).\mathbf{0}/X\} \mid !\overline{c}(r).E\{c(z).\mathbf{0}/X\})$, where c is fresh and r is a closed term. So the guarded fixpoint operator does not introduce extra expressive power [FL10]. A $\mathbb{VPC}_{\mathsf{Th}}$-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

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

Composition

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

Localization

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

Condition

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

Recursion

$$\frac{}{!a(x).T \xrightarrow{a(t)} T\{t/x\}\,|\,!a(x).T}\ t\in\mathsf{T}^0_\Sigma. \qquad \frac{}{!\overline{a}(t).T \xrightarrow{\overline{a}(t)} T\,|\,!\overline{a}(t).T}\ t\in\mathsf{T}^0_\Sigma.$$

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 $\mathbb{VPC}_{\mathsf{Th}}$-terms, only the behaviors of the $\mathbb{VPC}_{\mathsf{Th}}$-processes are completely characterized. If Th is the Peano Arithmetic PA, then under our semantics the $\mathbb{VPC}_{\mathsf{Th}}$-term *if* $\underline{0} \le x$ *then* $\overline{a}(\underline{0})$ can perform an action, but the obviously equivalent $\mathbb{VPC}_{\mathsf{Th}}$-term *if* $x = \underline{0}$ *then* $\overline{a}(\underline{0})$ *else* $\overline{a}(\underline{0})$ 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 [Mil89a], 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. Let A be $(a)(\overline{a}(\underline{0}) \mid \mu X.a(x).(\overline{a}(\mathsf{s}(x)) \mid (\tau.X + \overline{b}(x))))$. A typical action sequence of A is $A\underbrace{\xrightarrow{\tau}\xrightarrow{\tau}\ldots\xrightarrow{\tau}\xrightarrow{\tau}}_{2n+1 \text{ times}}\xrightarrow{\overline{b}(\underline{n})}\mathbf{0}$, 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 $(a)(\overline{a}(\underline{0}) \mid \mu X.a(x).(\overline{a}(\mathsf{s}(x)) \mid \mathit{if}\ \varphi\ \mathit{then}\ \overline{b}(x)\ \mathit{else}\ \tau.X))$. Notice that if no numerals satisfy φ then the process diverges.

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

3.2 Absolute Equality

A first attempt to define the bisimulations for $\mathbb{VPC}_{\mathsf{Th}}$ is to reiterate the definition from the theory of CCS for all $\mathbb{VPC}_{\mathsf{Th}}$-terms. This would require that $S \xrightarrow{\lambda} S'$ should be bisimulated by $T \overset{\widehat{\lambda}}{\Longrightarrow} T'$ whenever S is bisimilar to T. Consider however the $\mathbb{VPC}$-terms $\overline{a}(\underline{0}).S$ and $\mathit{if}\ x = \underline{0}\ \mathit{then}\ \overline{a}(\underline{0}).S\ \mathit{else}\ \overline{a}(\underline{0}).S$. Intuitively these two $\mathbb{VPC}$-terms are equivalent. But the action $\overline{a}(\underline{0}).S \xrightarrow{\overline{a}(\underline{0})} S$ cannot be simulated by any action of $\mathit{if}\ x = \underline{0}\ \mathit{then}\ \overline{a}(\underline{0}).S\ \mathit{else}\ \overline{a}(\underline{0}).S$. 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 $\mathbb{VPC}_{\mathsf{Th}}$ the model independent approach developed in [Fu12b]. The equality so obtained provides not only the intuition, but also a standard to compare against.

Process equivalences are observational. A process is observable if it may interact with another process.

Definition 1. *A process P is* observable*, notation $P{\Downarrow}$, if $\exists \lambda, P'.\ P \Longrightarrow \xrightarrow{\lambda} P'$.*

Now whatever an observational equivalence is, it must not identify an observable process with an unobservable process. Hence the next definition.

Definition 2. *A binary relation $\mathcal{R}$ is* equipollent *if $P{\Downarrow} \Leftrightarrow Q{\Downarrow}$ whenever $P\mathcal{R}Q$.*

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 $P \mid O$ and $Q \mid O$ 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. *A relation $\mathcal{R}$ is* extensional *if the following hold: (i) If $L\mathcal{R}M$ and $P\mathcal{R}Q$ then $(L \mid P)\ \mathcal{R}\ (M \mid Q)$. (ii) If $P\mathcal{R}Q$ then $(c)P\ \mathcal{R}\ (c)Q$ for all $c \in \mathcal{N}$.*

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 [Mil89a] and Park [Par81]. Following the idea of Fu [Fu12b], we actually will use a stronger version of the bisimulation introduced by van Glabbeek and Weijland [vGW89]. In the following definition, the notation $\mathcal{R}^{-1}$ stands for the inverse of $\mathcal{R}$.

Definition 4. *A binary relation is a* bisimulation *if the following hold:*

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

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 [Pri78].

Definition 5. *A relation* $\mathcal{R}$ *is* codivergent *if the following statements are valid:*

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

We have introduced four conditions for the equivalences on evolving processes, which are minimal from the point of view of interaction and computation. We turn these minimal conditions into defining properties of process equality.

Definition 6. *The* absolute equality $=_{Th}$ *is the largest reflexive, equipollent, extensional, codivergent bisimulation on* $\mathcal{P}_{\mathbb{VPC}_{Th}}$.

The well-definedness of Definition 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 $\mathbb{VPC}_{\mathsf{Th}}$. Before that, we state a useful lemma about computation, the Bisimulation Lemma [Fu12b]. The property stated in the lemma is called X-property by De Nicola, Montanari and Vaandrager [DNMV90].

Lemma 1. *If* $P \Longrightarrow P' =_{Th} Q$ *and* $Q \Longrightarrow Q' =_{Th} P$, *then* $P =_{Th} Q$.

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 $S \to T$, if $S \xrightarrow{\tau} T$ and $S =_{\mathsf{Th}} T$, and that S evolves to T in a *change-of-state* internal action, notation $S \xrightarrow{\iota} T$, if $S \xrightarrow{\tau} T$ and $S \neq_{\mathsf{Th}} T$. The reflexive and transitive closure of $\to$ will be denoted by $\to^*$.

The bisimulation property can now be defined in a more informative way. If $P =_{\mathsf{Th}} Q \xrightarrow{\tau} Q'$, then the simulation by P could be vacuous if $Q =_{\mathsf{Th}} Q'$; otherwise it must take the form $P \to^* P'' \xrightarrow{\iota} P'$ such that $P'' =_{\mathsf{Th}} Q$ and $P' =_{\mathsf{Th}} Q'$. Similarly if $P =_{\mathsf{Th}} Q \xrightarrow{\overline{a}(t)} Q'$, then the simulation of the output action must take the form $P \to^* P'' \xrightarrow{\overline{a}(t)} P'$ such that $P'' =_{\mathsf{Th}} Q$ and $P' =_{\mathsf{Th}} Q'$. 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 4, external bisimulations must explain how the external actions are bisimulated.

Definition 7. *A codivergent bisimulation $\mathcal{R}$ on $\mathcal{P}_{\mathbb{VPC}_{Th}}$ is a $\mathbb{VPC}_{Th}$-*bisimulation *if the following statements are valid for every $\lambda \neq \tau$.*

1. *If $Q\mathcal{R}^{-1}P \xrightarrow{\lambda} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\lambda} Q'\mathcal{R}^{-1}P'$ and $P\mathcal{R}Q''$ for some Q', Q''.*
2. *If $P\mathcal{R}Q \xrightarrow{\lambda} Q'$ then $P \Longrightarrow P'' \xrightarrow{\lambda} P'\mathcal{R}Q'$ and $P''\mathcal{R}Q$ for some P', P''.*

*The $\mathbb{VPC}_{Th}$-*bisimilarity *$\simeq_{Th}$ is the largest $\mathbb{VPC}_{Th}$-bisimulation.*

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

Lemma 2. *The external bisimilarity $\simeq_{Th}$ is extensional.*

The above lemma and the Bisimulation Lemma is the only thing we need to establish Proposition 1, which proves the correctness of Definition 7.

Proposition 1. *The relation $\simeq_{Th}$ coincides with the absolute equality $=_{Th}$.*

Proof. The inclusion $\simeq_{\mathsf{Th}} \subseteq =_{\mathsf{Th}}$ is immediate from Lemma 2. The proof of the reverse inclusion is standard using Bisimulation Lemma. Processes of the form $a(x).\mathit{if}\ x = \_\ \mathit{then}\ \overline{c}(\_)\ \mathit{else}\ \overline{d}(\_)$ and of the form $\overline{a}(\_) + \overline{a}(\_).\overline{c}(\_)$, 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 [FZ11]. □

Both the absolute equality and the external bisimilarity are relations on the processes. They can be extended to the $\mathbb{VPC}_{\mathsf{Th}}$-terms in the standard manner.

Definition 8. *$S \simeq_{Th} T$ if and only if $S\rho \simeq_{Th} T\rho$ for every assignment ρ whose domain of definition is disjoint from $bv(S \mid T)$.*

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

Proposition 2. *The absolute equality is equivalent and congruent.*

Action

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

Composition

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

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

$$\frac{}{!a(x).T \xrightarrow{a(x)}_{\top} T \,|\, !a(x).T} \qquad \frac{}{!\overline{a}(t).T \xrightarrow{\overline{a}(t)}_{\top} T \,|\, !\overline{a}(t).T}$$

Fig. 4. Symbolic Semantics

4 Symbolic Semantics

The absolute equality is not very convenient. Hennessy and Lin [HL95] 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 [HL95] 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. Symbolic bisimilarity is in our opinion a decidable approximation of the absolute equality. Our motivation for the symbolic bisimilarity is two folds: (i) the symbolic bisimilarity should be sound, meaning that it should be a subset of the absolute equality; (ii) ideally the symbolic bisimilarity is complete on the decidable subsets of the absolute equality. The latter is much more difficult to come by than the former.

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 $\mathbb{VPC}_{\mathsf{Th}}$-term *if* φ *then* $\overline{a}(\underline{0})$ *else* $\overline{b}(\underline{0})$ where the boolean Σ-expression φ contains free variables. The *symbolic semantics* [HL95, HL96] 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 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

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

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. The symbolic semantics is stable under substitution in the sense of the following lemma.

Lemma 3. *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 will abbreviate $\xrightarrow{\tau}_{\varphi_1} \ldots \xrightarrow{\tau}_{\varphi_n}$ to $\Longrightarrow_{\varphi_1 \ldots \varphi_n}$, and $\Longrightarrow_\varphi \xrightarrow{\lambda}_{\varphi'} \Longrightarrow_{\varphi''}$ to $\overset{\lambda}{\Longrightarrow}_{\varphi\varphi'\varphi''}$. We remark that $T \Longrightarrow_\top T$.

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

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

The process $L(y)$ may perform only one action $L(y) \xrightarrow{\overline{b}(\underline{0})}_{\underline{0} \leq y} \mathbf{0}$. The process $M(y)$ has an interesting action sequence $M(y) \Longrightarrow_\top \xrightarrow{\overline{b}(\underline{0})}_{y=\underline{n}} \mathbf{0}$. Similarly $N(y)$ has a similar action sequence $N(y) \Longrightarrow_\top \xrightarrow{\overline{b}(y)}_{y=\underline{n}} \mathbf{0}$. In the second and third examples the set of possible conditions is $\{y{=}\underline{0}, y{=}\underline{1}, y{=}\underline{2}, \ldots, y{=}\underline{n}, \ldots\}$. 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 $\overline{b}(\underline{0})$, 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 simple induction on derivation.

Lemma 4. *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 $S \xrightarrow{\overline{a}(t)}_\varphi T$ for some Th-theorem φ and some $t \in \mathsf{T}^0_\Sigma$ then $S \xrightarrow{\overline{a}(t)} T$.
(iii) If $S \xrightarrow{a(x)}_\varphi T$ for some Th-theorem φ then $S \xrightarrow{a(t)} T\{t/x\}$ for every $t \in \mathsf{T}^0_\Sigma$.

Lemma 5. *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 $S \xrightarrow{\overline{a}(t)} T$ *then* $S \xrightarrow{\overline{a}(t)}_{\varphi} T$ *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 $T \Longrightarrow_{\varphi'} T'$ for bisimilar S and T such that $\mathsf{Th} \vdash \varphi \Rightarrow \varphi'$? 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 $x \notin fv(\varphi)$. 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 $\mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$ 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 $\{T \overset{\widehat{\lambda_i}}{\Longrightarrow}_{\varphi_i} T'_i\}_{i \in I}$ such that $\mathsf{Th} \vdash \varphi\varphi_i \Rightarrow \lambda = \lambda_i$ 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 $\lambda \equiv \overline{a}(t)$ and $\lambda_i \equiv \overline{a}(t')$ for some name a; otherwise $\lambda = \lambda_i$ stands for $\bot$. The relation $\simeq^?$ is not very useful. Consider the $\mathbb{VPC}$-processes $R(y), S(y), T(y)$ defined as follows:

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

In the definition of $R(y)$ the term $\mathsf{p}(z)$ 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:

$$
R(\underline{n}) \xrightarrow{\tau} \top \xrightarrow{\overline{r}(\underline{n})} \top \overset{\tau}{\Longrightarrow}_{\underline{0} < \underline{n}} \xrightarrow{\overline{r}(\underline{n-1})} \top \overset{\tau}{\Longrightarrow}_{\underline{0} < \underline{n-1}} \cdots \overset{\tau}{\Longrightarrow}_{\underline{0} < \underline{1}} \xrightarrow{\overline{r}(\underline{1})} \top \cdots .
$$

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 $S(y) \xrightarrow{\tau}_{\underline{0} \leq y} R(y)$ can be

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

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

simulated by $T(y)$ as long as y is instantiated by a numeral. On the other hand it is easy to see that $S(y) \not\simeq^? T(y)$. The only way to simulate the action is by the collection $\{T(y) \stackrel{\tau}{\Longrightarrow}_{\underline{0}\le y\le \underline{n}}\ \mathit{if}\ y = \underline{n}\ \mathit{then}\ \tau.R(y)\ \mathit{else}\ \overline{e}(\underline{0})\}_{n\in\mathbf{N}}$. But notice that $\mathit{if}\ \underline{0} \le y \le \underline{n}\ \mathit{then}\ R(y)$ is *not* bisimilar to $\mathit{if}\ \underline{0} \le y \le \underline{n}\ \mathit{then}\ (\mathit{if}\ y = \underline{n}\ \mathit{then}\ \tau.R(y)\ \mathit{else}\ \overline{e}(\underline{0}))$. 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 $U(y) \simeq^? W(y)$. It follows that $U(y) \not\simeq^? W(y)$.

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 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.

Definition 9. *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* $fv(\bigvee_{i\in I}\varphi_i) \subseteq V$, $\mathsf{Th} \vdash \varphi_i \wedge \varphi_j \Rightarrow \bot$ *if* $i \neq j$, *and* $\mathsf{Th} \vdash \varphi \Leftrightarrow \bigvee_{i\in I}\varphi_i$.

Let $K(y) = (a)(\overline{a}(\underline{0}) \mid \mu X.a(x).(X \mid \mathit{if}\ x \le y\ \mathit{then}\ \overline{a}(\mathsf{s}(x))))$. It admits the following infinite sequence $K(y) \stackrel{\tau}{\longrightarrow}_{\top} \stackrel{\tau}{\longrightarrow}_{\underline{0}\le y} \stackrel{\tau}{\longrightarrow}_{\underline{1}\le y} \ldots \stackrel{\tau}{\longrightarrow}_{\underline{n}\le y} \ldots$. Although every finite subset of $\{\top, \underline{0} \le y, \underline{1} \le y, \ldots, \underline{n} \le y, \ldots\}$ is consistent, the infinite sequence 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 $\{\top, \underline{0} \le y, \underline{1} \le y, \ldots, \underline{n} \le y, \ldots\}$. From the point of view of the first order logic, no satisfiable Σ_{PA}-expression implies all the Σ_{PA}-expressions in $\{\top, \underline{0} \le y, \underline{1} \le y, \ldots, \underline{n} \le y, \ldots\}$. 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 of codivergence, take a look at $H(y) = (a)(\overline{a}(\underline{0}) \mid \mu X.a(x).(X \mid \mathit{if}\ y \le x\ \mathit{then}\ \overline{a}(\mathsf{s}(x))))$, which is a slight modification of the previous $\mathbb{VPC}$-term. This term also admit an infinite sequence

of τ-actions $H(y) \xrightarrow{\tau} \top \xrightarrow{\tau} y \leq \underline{0} \xrightarrow{\tau} y \leq \underline{1} \ldots \xrightarrow{\tau} y \leq \underline{n} \ldots$. The infinite sequence is not real for $H(\underline{1}), H(\underline{2}), H(\underline{3}), \ldots$. But it is real for $H(\underline{0})$. These examples suggest that it is difficult to give a nice symbolic description of the codivergence property at a general level. It should be remarked though that divergence is intrinsically an undecidable property. If our interest in symbolic semantics is confined to automatic verification, the codivergence condition has to be dropped.

Definition 10. *A symmetric relation $\mathcal{R}$ on $\mathcal{T}_{\mathbb{VPC}_{\mathsf{Th}}}$ is a* symbolic bisimulation *if the following hold whenever $S\mathcal{R}T$ and $V = fv(S \,|\, T)$:*

1. *If $S \xrightarrow{\tau}_{\varphi} S'$ and $\mathsf{Th} \cup \{\varphi\}$ 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 $i \in I$, $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i$ 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 $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i'$, and $\varphi_i S' \mathcal{R} \varphi_i T_i'$.
2. *If $S \xrightarrow{\overline{a}(t)}_{\varphi} S'$ and $\mathsf{Th} \cup \{\varphi\}$ is consistent, then there are a boolean Th-partition $\{\varphi_i\}_{i \in I}$ of φ on V and a collection $\{T \Longrightarrow_{\psi_i} T_i \xrightarrow{\overline{a}(t_i)}_{\psi_i'} T_i'\}_{i \in I}$ such that $\mathsf{Th} \vdash (\varphi_i \Rightarrow \psi_i \psi_i') \wedge (\varphi_i \Rightarrow t = t_i)$, and moreover $\varphi_i S \mathcal{R} \varphi_i T_i$ and $\varphi_i S' \mathcal{R} \varphi_i T_i'$ for every $i \in I$.*
3. *If $S \xrightarrow{a(x)}_{\varphi} S'$ and $\mathsf{Th} \cup \{\varphi\}$ is consistent, then there are a boolean Th-partition $\{\varphi_i\}_{i \in I}$ of φ on $V \cup \{x\}$ and a collection $\left\{T \Longrightarrow_{\psi_i} T_i \xrightarrow{a(x)}_{\psi_i'} T_i'\right\}_{i \in I}$ such that $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i \psi_i'$, $\varphi_i S \mathcal{R} \varphi_i T_i$ and $\varphi_i S' \mathcal{R} \varphi_i T_i'$ for all $i \in I$.*

The symbolic bisimilarity $\simeq^s_{\mathsf{Th}}$ *is the largest symbolic bisimulation.*

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

The main algebraic properties of $\simeq^s_{\mathsf{Th}}$ are summarized in the next proposition.

Proposition 3. *The following statements about $\simeq^s_{\mathsf{Th}}$ are valid.*

1. *The relation $\simeq^s_{\mathsf{Th}}$ is an equivalence.*
2. *The relation $\simeq^s_{\mathsf{Th}}$ is a congruence.*
3. *If $S \simeq^s_{\mathsf{Th}} T$ then $S\sigma \simeq^s_{\mathsf{Th}} T\sigma$ 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, $S \simeq^s_{\mathsf{Th}} T$ implies $\varphi S \simeq^s_{\mathsf{Th}} \varphi T$. (3) Using Lemma 3 it is easy to show that $\{(S\sigma, T\sigma) \mid S \simeq^s_{\mathsf{Th}} T\}$ is a symbolic bisimulation. We need to make use of a meta theoretical result asserting that $\mathsf{Th} \vdash \varphi\sigma$ for every substitution σ whenever $\mathsf{Th} \vdash \varphi$, which is an easy consequence of FO1 and FO2. □

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 [vGW89].

Lemma 6. *Suppose* $T \xrightarrow{\tau}_{\varphi_1} T_1 \xrightarrow{\tau}_{\varphi_2} T_2 \ldots \xrightarrow{\tau}_{\varphi_n} T_n$ *and* $\varphi \Rightarrow \varphi_1 \ldots \varphi_n$. *Then* $\varphi T \simeq^s_{\mathsf{Th}} \varphi T_n$ *implies* $\varphi T_1 \simeq^s_{\mathsf{Th}} \varphi T_2 \simeq^s_{\mathsf{Th}} \ldots \simeq^s_{\mathsf{Th}} \varphi T_n$.

Now let's take a look at some examples. Consider the following $\mathbb{VPC}$-terms:

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

It is clear that $A(y) \simeq^s_{\mathsf{Th}} B(y)$. The action $A(y) \xrightarrow{\overline{b}(y)}_{y=\underline{0}} E(y)$ is simulated by the single action $B(y) \xrightarrow{\overline{b}(y)}_{y=\underline{0}\vee y=\underline{1}} E(y) \,|\, \mathit{if}\ y = \underline{1}\ \mathit{then}\ \overline{b}(y)$. This example shows that the condition $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i$ in say (1) of Definition 10 should not be strengthened to $\mathsf{Th} \vdash \varphi_i \Leftrightarrow \psi_i$.

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

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

Typically D has the action sequence $D(y) \stackrel{\tau}{\Longrightarrow}_{y\neq\underline{0}\wedge\ldots\wedge y\neq\underline{n-1}} \xrightarrow{\overline{b}(\underline{n})}_{y=\underline{n}}$. It is not difficult to see that $C(y) =_{\mathsf{Th}} D(y)$. The action $C(y) \xrightarrow{\overline{b}(y)}_{\underline{0}\leq y} \mathbf{0}$ is simulated by the *infinite* collection $\{D(y) \stackrel{\tau}{\Longrightarrow}_{y\neq\underline{0}\wedge\ldots\wedge y\neq\underline{n-1}} \xrightarrow{\overline{b}(\underline{n})}_{y=\underline{n}}\}_{n\in\mathbf{N}}$. There is no finite collection that can simulate $C(y) \xrightarrow{\overline{b}(y)}_{\underline{0}\leq y} \mathbf{0}$. So the symbolic bisimilarity $\simeq^s_{\mathsf{Th}}$ cannot coincide with the absolute equality $=_{\mathsf{Th}}$ even if divergence is ignored. It is however a correct approximation of the absolute equality.

Theorem 2. *Let* S, T *be finite* $\mathbb{VPC}_{\mathsf{Th}}$*-terms. Then* $S =_{\mathsf{Th}} T$ *if* $S \simeq^s_{\mathsf{Th}} T$.

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

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

Suppose $S \simeq^s_{\mathsf{Th}} T$ and $S\rho \xrightarrow{a(t)} P$ for some $t \in \mathsf{T}^0_\Sigma$ and some assignment ρ such that $bv(S \,|\, T) \cap dom(\rho) = \emptyset$. Let V be $fv(S \,|\, T)$. By Lemma 5, $S\rho \xrightarrow{a(x)}_{\varphi'} S'$ for some Th-theorem φ' and some x, S' such that $P \equiv S'\{t/x\}$. By Lemma 3, there exist some φ, S_1 such that $\varphi' \equiv \varphi\rho$, $S' \equiv S_1\rho$ and $S \xrightarrow{a(x)}_{\varphi} S_1$. By the definition of symbolic bisimulation there are a boolean Th-partition $\{\varphi_i\}_{i\in I}$ of φ on $V \cup \{x\}$ and a collection $\left\{ T \Longrightarrow_{\psi_i} T_i \xrightarrow{a(x)}_{\psi'_i} T'_i \right\}_{i\in I}$ such that $\mathsf{Th} \vdash \varphi_i \Rightarrow \psi_i\psi'_i$,

$\varphi_i S \mathcal{R} \varphi_i T_i$ and $\varphi_i S' \mathcal{R} \varphi_i T_i'$ for all $i \in I$. Now $\mathsf{Th} \vdash \varphi_i \rho[x \leftarrow t]$ for some $i \in I$, which in turn implies $\mathsf{Th} \vdash \psi_i \psi_i' \rho[x \leftarrow t]$, $S\rho \mathcal{R} T_i \rho$ and $S'\rho[x \leftarrow t] \mathcal{R} T_i'\rho[x \leftarrow t]$. At the meantime notice that $T\rho \Longrightarrow T_i\rho \xrightarrow{a(t)} T_i'\rho[x \leftarrow t]$ by Lemma 3 and Lemma 4. Conclude that $\mathcal{R}$ is a bisimulation. The relation is also extensional by Proposition 3. It is equipollent since the external actions are bisimulated. □

5 Proof System

When confined to the finite $\mathbb{VPC}_{\mathsf{Th}}$-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 $\varphi\overline{a}(t).S = \varphi\overline{a}(t').S$ 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 a logical formula. For a value-passing calculus $\mathbb{VPC}_{\mathsf{Th}}$ studied in this paper, equivalence checking algorithms do exist.

In studying proof systems, a standard treatment is to remove the composition operator using *expansion law* [HM85, Mil89a]. This is done at the expense of introducing the unguarded choice operator '+' whose semantics is defined by

$$\frac{S \xrightarrow{\lambda}_{\varphi} S'}{S + T \xrightarrow{\lambda}_{\varphi} S'} \qquad \frac{T \xrightarrow{\lambda}_{\varphi} T'}{S + T \xrightarrow{\lambda}_{\varphi} T'}$$

The operator destroys the congruence property of process equalities. As a consequence additional complication is introduced to produce a congruence. We avoid the complication by not using the congruence relation. In this section the notation $\sum_{i \in I} \varphi_i \lambda_i.S_i$ stands for a mixed choice [Pal03, FL10].

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:

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

Our 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. We write $AS_{\mathsf{Th}} \vdash S = T$ if the equality $S = T$ can be derived within AS_{Th}, and $S \overset{R}{=} T$ 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 [IL01] and the name-passing calculi [PS95]. The law $S6$ is a generalization of the following law proposed by Parrow and Sangiorgi [PS95]:

$$a(x).S + a(x).T = a(x).S + a(x).T + a(x).(\varphi S + \neg\varphi T). \tag{2}$$

$S1$	$T + \mathbf{0} = T$	
$S2$	$S + T = T + S$	
$S3$	$R + (S + T) = (R + S) + T$	
$S4$	$T + T = T$	
$S5$	$\varphi T + \varphi' T = (\varphi \vee \varphi')T$	
$S6$	$\sum_{i\in I} \phi_i a(x).(\neg\varphi_i T_i + \varphi_i \tau.T) = \varphi a(x).T + \sum_{i\in I} \phi_i a(x).(\neg\varphi_i T_i + \varphi_i \tau.T)$	if $\mathsf{Th} \vdash \varphi_i \Rightarrow \phi_i$ for all $i \in I$ and $\mathsf{Th} \vdash \varphi \Leftrightarrow \bigvee_{i\in I} \varphi_i$.
$C1$	$[\bot]T = \mathbf{0}$	
$C2$	$[\top]T = T$	
$C3$	$\varphi\overline{a}(t).T = \varphi\overline{a}(t').T$,	if $\mathsf{Th} \vdash \varphi \Rightarrow t = t'$
$C4$	$\varphi\lambda.T = \varphi\lambda.\varphi T$	
$C5$	$\phi(T + T') = \phi T + \phi T'$,	
$C6$	$\varphi T = \psi T$,	if $\mathsf{Th} \vdash \varphi \Leftrightarrow \psi$
$C7$	$\varphi(\psi T) = (\varphi\psi)T$	
$L1$	$(c)\mathbf{0} = \mathbf{0}$	
$L2$	$(c)\lambda.T = \mathbf{0}$,	if c is in λ
$L3$	$(c)\lambda.T = \lambda.(c)T$,	if c is not in λ
$L4$	$(c)(T + T') = (c)T + (c)T'$	
$L5$	$(c)\varphi T = \varphi(c)T$	
B	$\lambda.(S + T) = \lambda.(\tau.(S + T) + T)$	

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

As pointed out in [PS95], this is the law that tells apart the early equivalence and the late equivalence [MPW92]. The combination of the S-laws has powerful consequences, two of which are given by the following lemmas.

Lemma 7. *$AS_{Th} \vdash \psi\lambda.T = \varphi\lambda.T + \psi\lambda.T$ if $\mathsf{Th} \vdash \varphi \Rightarrow \psi$.*

Lemma 8. *$AS_{Th} \vdash \varphi(\lambda.T)\{t/x\} = \varphi(\lambda.T)\{t'/x\}$ if $\mathsf{Th} \vdash \varphi \Rightarrow t = t'$.*

The B-law is due to van Glabbeek and Weijland [vGW89]. It implies Milner's first tau law $\lambda.\tau.T = \lambda.T$ and a weaker form $\tau.(T + \varphi\tau.T) = \tau.T$ of Milner's second tau law [Mil89a].

The verification of soundness property of AS_{Th} is routine.

Lemma 9. *If $AS_{Th} \vdash S = T$ then $S \simeq^s_{Th} T$.*

A nice property of a proof system is that it allows one to focus on terms in some special form.

Definition 11. *A finite $\mathbb{VPC}_{Th}$-term is a* normal form *if it is either $\mathbf{0}$ or of the form $\sum_{i\in I} \varphi_i\lambda_i.T_i$ such that T_i is a normal form for every $i \in I$.*

Using $L4$ and $L5$ one can pull all the localization operations in a term to the very front. Then one may remove all the composition operations by applying the expansion law. And finally one removes the localization operations using the L-axioms. What one gets is a normal form term. Hence the following lemma.

Lemma 10. *For each finite $\mathbb{VPC}_{\mathsf{Th}}$-term T there is some normal form T' such that $AS_{\mathsf{Th}} \vdash T = T'$.*

In the rest of the section the following Completeness Theorem is proved.

Theorem 3. *$S \simeq^s_{\mathsf{Th}} T$ if and only if $AS_{\mathsf{Th}} \vdash \tau.S = \tau.T$.*

Proof. A sequence $T \stackrel{\tau}{\Longrightarrow}_{\varphi} T'$ is *maximal* with regards to ϕ if (i) $\mathsf{Th} \vdash \phi \Rightarrow \varphi$, (ii) $\phi T \simeq^s_{\mathsf{Th}} \phi T''$, and (iii) there does not exist any T'' such that $T' \stackrel{\tau}{\longrightarrow}_{\varphi'} T''$, $\mathsf{Th} \vdash \phi \Rightarrow \varphi'$ and $\phi T \simeq^s_{\mathsf{Th}} \phi T''$. Intuitively $T \stackrel{\tau}{\Longrightarrow}_{\varphi} T'$ is maximal if T' cannot evolve to another state that stays equal to T under the condition ϕ.

We confine our attention to the normal forms. Suppose $S \simeq^s_{\mathsf{Th}} T$ 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) If $\mathsf{Th} \vdash \phi \Rightarrow \varphi$, $\phi T \simeq^s_{\mathsf{Th}} \phi T'$ and $T \stackrel{\tau}{\Longrightarrow}_{\varphi} T'$ is maximal with regards to ϕ, then $AS_{\mathsf{Th}} \vdash \tau.T = \tau.(T + \phi T')$.
(P) If $\phi T \simeq^s_{\mathsf{Th}} \phi S$, then $AS_{\mathsf{Th}} \vdash \phi\tau.T = \phi\tau.(T + S) = \phi\tau.S$.

Let's write $dep(T)$ for the maximal nested depth of prefixing operation of T. Our proof strategy will be as follows:

1. Prove (S) assuming that (S) and (P) hold for terms with depths less than i.
2. Prove (P) assuming that (P) holds for terms whose depths are less than i and that (S) holds for terms whose depths are less than or equal to i.

The inductive proof of (S) is given as follows: Suppose T is a normal form of depth i and $T \stackrel{\tau}{\longrightarrow}_{\varphi_1} T_1 \stackrel{\tau}{\longrightarrow}_{\varphi_2} \ldots T_{n-1} \stackrel{\tau}{\longrightarrow}_{\varphi_n} T_n$ is maximal with regards to φ with $\varphi T_1 \simeq^s_{\mathsf{Th}} \varphi T_n$ and $\varphi = \varphi_1 \ldots \varphi_n$. Then $\varphi T_1 \simeq^s_{\mathsf{Th}} \varphi T_2 \simeq^s_{\mathsf{Th}} \ldots \simeq^s_{\mathsf{Th}} \varphi T_n$ by Lemma 6. By the induction hypothesis on (P), $AS_{\mathsf{Th}} \vdash \varphi\tau.T_1 = \varphi\tau.T_n$. By Lemma 7, $AS_{\mathsf{Th}} \vdash \tau.T = \tau.(T + \varphi_1\tau.T_1) = \tau.(T + \varphi\tau.T_1) = \tau.(T + \varphi\tau.T_n)$. Since $T \stackrel{\tau}{\longrightarrow}_{\varphi_1} T_1 \stackrel{\tau}{\longrightarrow}_{\varphi_2} \ldots T_{n-1} \stackrel{\tau}{\longrightarrow}_{\varphi_n} T_n$ is maximal, the action of each summand $\psi_j\lambda_j.T_j$ of T can be simulated by a set of summands of T_n using induction hypothesis, and consequently $\psi_j\lambda_j.T_j$ can be assimilated by T_n. We conclude that $AS_{\mathsf{Th}} \vdash \varphi\tau.T_n = \varphi\tau.(T_n + T)$. Consequently

$$\begin{aligned}
AS_{\mathsf{Th}} \vdash \tau.T &= \tau.(T + \varphi\tau.(T_n + T)) \\
&\stackrel{S5}{=} \varphi\tau.(T + \varphi\tau.(T_n + T)) + \neg\varphi\tau.(T + \varphi\tau.(T_n + T)) \\
&\stackrel{C \text{ laws}}{=} \varphi\tau.(T + \tau.(T_n + T)) + \neg\varphi\tau.T \\
&\stackrel{B}{=} \varphi\tau.(T + T_n) + \neg\varphi\tau.(T + \varphi T_n) \\
&\stackrel{S5}{=} \varphi\tau.(T + \varphi T_n) + \neg\varphi\tau.(T + \varphi T_n) \\
&\stackrel{S5}{=} \tau.(T + \varphi T_n).
\end{aligned}$$

We turn to the inductive proof of (P). By the C-laws we only have to prove the special case when $\phi = \top$. 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. We only consider the case $\lambda_i = a(x)$. By definition there are a boolean Th-partition $\{\varphi_i^k\}_{k\in K}$ of φ_i on $fv(S\,|\,T)\cup\{x\}$ and a collection $\{T \Longrightarrow_{\psi_k} T_k \xrightarrow{a(x)}_{\psi'_k} T'_k\}_{k\in K}$ such that $\mathsf{Th}\vdash \varphi_i^k \Rightarrow \psi_k\psi'_k$ for all $k\in K$, and $\varphi_i^k S \simeq^s_{\mathsf{Th}} \varphi_i^k T_k$, $\varphi_i^k S_i \simeq^s_{\mathsf{Th}} \varphi_i^k T'_k$ for every $k\in K$. We assume that for each $k\in K$, $T\Longrightarrow_{\psi_k} T_k$ is maximal with regards to φ_i^k. By induction on (P),

$$AS_{\mathsf{Th}} \vdash \varphi_i^k\tau.S = \varphi_i^k\tau.T_k, \tag{3}$$

$$AS_{\mathsf{Th}} \vdash \varphi_i^k\tau.S_i = \varphi_i^k\tau.T'_k \tag{4}$$

for every $k\in K$. Since $\{T+S \Longrightarrow_{\psi_k} T_k \xrightarrow{a(x)}_{\psi'_k} T'_k\}_{k\in K}$, we could have the following rewriting that makes use of the inductive hypothesis on (S).

$$\begin{aligned}
\tau.(T+S) &\overset{I,H.}{=} \tau.\left(T+S+\sum_{k\in K}\psi_k T_k\right)\\
&= \tau.\left(T+S+\sum_{k\in K}\psi_k(T_k+\psi'_k a(x).T'_k)\right)\\
&\overset{C5}{=} \tau.\left(T+S+\sum_{k\in K}\psi_k T_k+\sum_{k\in K}\psi_k\psi'_k a(x).T'_k\right)\\
&\overset{B}{=} \tau.\left(T+S+\sum_{k\in K}\psi_k T_k+\sum_{k\in K}\psi_k\psi'_k a(x).\tau.T'_k\right)\\
&\overset{S5}{=} \tau.\left(T+S+\sum_{k\in K}\psi_k T_k+\sum_{k\in K}\psi_k\psi'_k a(x).(\neg\varphi_i^k\tau.T'_k+\varphi_i^k\tau.T'_k)\right)\\
&\overset{(4)}{=} \tau.\left(T+S+\sum_{k\in K}\psi_k T_k+\sum_{k\in K}\psi_k\psi'_k a(x).(\neg\varphi_i^k\tau.T'_k+\varphi_i^k\tau.S_i)\right)\\
&\overset{S6}{=} \tau.\left(T+\sum_{i'\in I\setminus\{i\}}\varphi_{i'}\lambda_{i'}.S_{i'}+\sum_{k\in K}\psi_k T_k+\sum_{k\in K}\psi_k\psi'_k a(x).(\neg\varphi_i^k\tau.T'_k+\varphi_i^k\tau.S_i)\right)\\
&= \tau.\left(T+\sum_{i'\in I\setminus\{i\}}\varphi_{i'}\lambda_{i'}.S_{i'}+\sum_{k\in K}\psi_k T_k\right)\\
&\overset{I,H.}{=} \tau.\left(T+\sum_{i'\in I\setminus\{i\}}\varphi_{i'}\lambda_{i'}.S_{i'}\right).
\end{aligned}$$

It follows from induction that $\tau.(T+S) = \tau.\left(T+\sum_{i\in I'}\phi_i\tau.T\right)$ for some $I'\subseteq I$ and soem $\{\phi_i\}_{i\in I'}$. Let $\phi = \bigvee_{i\in I'}\phi_i$. Then

$$\begin{aligned}
\tau.(T+S) &\overset{S5}{=} \tau.(T+\phi\tau.T)\\
&\overset{S5,C2}{=} \phi\tau.(T+\phi\tau.T)+\neg\phi\tau.(T+\phi\tau.T)\\
&\overset{C\ \text{laws},B}{=} \phi\tau.T+\neg\phi\tau.T\\
&\overset{S5,C2}{=} \tau.T.
\end{aligned}$$

Symmetrically $AS_{\mathsf{Th}}\vdash\tau.(S+T)=\tau.S$. We are done. □

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 [Rog87] 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 12. *A* numeric system *for* $\mathbb{VPC}_{\mathsf{Th}}$ *consists of a countable subset* $\{\widehat{0}, \widehat{1}, \widehat{2}, \ldots, \widehat{n}, \ldots\}$ *of* $\mathcal{T}^0_\Sigma$ *and a* $\mathbb{VPC}_{\mathsf{Th}}$*-term* $S_d(x)$ *that satisfy the followings:*

1. *The variable* x *is the only free variable appearing in* $S_d(x)$.
2. $\mathsf{Th} \vdash \widehat{p} \neq \widehat{q}$ *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* $S_d(\widehat{n}) \Longrightarrow \stackrel{\overline{d}(\widehat{n+1})}{\longrightarrow} =_{\mathsf{Th}} \mathbf{0}$.

It is clear from (3) of Definition 12 that every action sequence of $S_d(\widehat{n})$ is actually of the form $S_d(\widehat{n}) \rightarrow^* \stackrel{\overline{d}(\widehat{n+1})}{\longrightarrow} =_{\mathsf{Th}} \mathbf{0}$ and $S_d(\widehat{n}) =_{\mathsf{Th}} \overline{d}(\widehat{n+1})$.

Using a numeric system, we may talk about functions in a value-passing calculus. The *predecessor function* p for instance can be defined in such a calculus. Suppose we would like to have the $\mathbb{VPC}$-process $a(x).\overline{b}(\mathsf{p}(x))$. It can be defined by $a(x).(c)(\overline{c}(\underline{0}) \mid !c(y).\mathit{if}\ x = \mathsf{s}(y)\ \mathit{then}\ \overline{b}(y)\ \mathit{else}\ \overline{c}(\mathsf{s}(y)))$. The process diverges when given the input $\underline{0}$. 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 $f(x_1, \ldots, x_n)$ is *definable* in $\mathbb{VPC}_{\mathsf{Th}}$ with respect to the numeric system $\langle\{\widehat{0}, \widehat{1}, \widehat{2}, \ldots, \widehat{n}, \ldots\}, S_d(x)\rangle$ if, for all names $a_1, \ldots, a_n, b$, there is a process $I(a_1, \ldots, a_n, b)$ of the form $a_1(x_1)....a_n(x_n).T$ such that (i) if $f(p_1, \ldots, p_n)$ is defined, then all the action sequences of $T\{\widehat{p_1}/x_1, \ldots, \widehat{p_n}/x_n\}$ are finite and are of the following form $T\{\widehat{p_1}/x_1, \ldots, \widehat{p_n}/x_n\} \Longrightarrow \stackrel{\overline{b}(\widehat{p})}{\longrightarrow} T' =_{\mathsf{Th}} \mathbf{0}$, where $p = f(p_1, \ldots, p_n)$; and (ii) if $f(p_1, \ldots, p_n)$ is undefined, then $T\{\widehat{p_1}/x_1, \ldots, \widehat{p_n}/x_n\}$ can only perform τ-action sequences and all its τ-action sequences are divergent. We say that $f(x_1, \ldots, x_n)$ is defined by $I(a_1, \ldots, a_n, b)$ at $a_1, \ldots, a_n, b$. 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 $\mathbb{VPC}_{\mathsf{Th}}$ if it is definable with respect to some numeric system of $\mathbb{VPC}_{\mathsf{Th}}$.

If a function is definable in $\mathbb{VPC}_{\mathsf{Th}}$ with respect to $\langle\{\widehat{0}, \widehat{1}, \widehat{2}, \ldots, \widehat{n}, \ldots\}, S_d(x)\rangle$, we can design a procedure that, upon receiving the natural numbers $p_1, \ldots, p_n$, traverses the derivation tree of $T\{\widehat{p_1}/x_1, \ldots, \widehat{p_n}/x_n\}$. This is well defined since every $\mathbb{VPC}_{\mathsf{Th}}$-term is finite branching. The procedure terminates with the result p if $T\{\widehat{p_1}/x_1, \ldots, \widehat{p_n}/x_n\}$ 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 13. *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 4. *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 $\mathbb{VPC}_{\mathsf{Th}}$ has a numeric system $\langle\{\widehat{0},\widehat{1},\widehat{2},\ldots,\widehat{n},\ldots\}, S_d(x)\rangle$. We show that the recursive functions [Rog87] are definable in $\mathbb{VPC}_{\mathsf{Th}}$. We consider only two cases.

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

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

  where *Rec* stands for the following process

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

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

  $$a_1(x_1).\cdots.a_{n+1}(x_{n+1}).(a_1\ldots a_{n+1})(!\overline{a_1}(x_1) \mid \cdots \mid !\overline{a_n}(x_n) \mid \overline{f}(\widehat{0}) \mid !f(z).Mu),$$

  where Mu is the following process

  $$\overline{a_{n+1}}(z) \mid (c)(G(a_1,\cdots,a_{n+1},c) \mid c(v).\mathit{if}\ v=\widehat{0}\ \mathit{then}\ \overline{b}(z)\ \mathit{else}\ \overline{f}(S_d(z))).$$

We are done. □

Proposition 4 adds weight to Definition 12 since a numeric system is intuitively a minimal requirement for a value-passing calculus to simulate all the recursive functions. The $\mathbb{VPC}_{\mathsf{Th}}$-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 decidable first order theory, this

general theory immediately generates an operational semantics and an observation theory for that calculus. We have studied only one decidable first order theory, the 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 $\simeq^{s}_{\mathbb{VPC}}$ 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 5. *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 $S \stackrel{\text{def}}{=} \mathit{if}\ \varphi\ \mathit{then}\ T$. The interpretation of S is given by the following term

$$(c)([\![\varphi]\!]_c \mid c(z).\mathit{if}\ z = \underline{1}\ \mathit{then}\ T')$$

where T' is the interpretation of T and $[\![\varphi]\!]_c$ is structurally defined as follows:

- If φ is $t = t'$, then $[\![\varphi]\!]_c$ is $\mathit{if}\ t = t'\ \mathit{then}\ \overline{c}(\underline{1})\ \mathit{else}\ \overline{c}(\underline{0})$.
- If φ is $t < t'$, then $[\![\varphi]\!]_c$ is the following term

$$\overline{d}(t).\overline{e}(t') \mid !d(x).e(y).\mathit{if}\ y = \underline{0}\ \mathit{then}\ \overline{c}(\underline{0})\ \mathit{else}\ \mathit{if}\ x = \underline{0}\ \mathit{then}\ \overline{c}(\underline{1})\ \mathit{else}\ \overline{d}(\mathsf{p}(x)).\overline{e}(\mathsf{p}(y)).$$

- If φ is $\varphi' \wedge \varphi''$, then $[\![\varphi]\!]_c$ is

$$(de)([\![\varphi']\!]_d \mid [\![\varphi'']\!]_e \mid d(y).e(z).\mathit{if}\ y = 1 \wedge z = 1\ \mathit{then}\ \overline{c}(\underline{1})\ \mathit{else}\ \overline{c}(\underline{0})).$$

- If φ is $\varphi' \vee \varphi''$, then $[\![\varphi]\!]_c$ is

$$(de)([\![\varphi']\!]_d \mid [\![\varphi'']\!]_e \mid d(y).e(z).\mathit{if}\ y = 1 \vee z = 1\ \mathit{then}\ \overline{c}(\underline{1})\ \mathit{else}\ \overline{c}(\underline{0})).$$

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

7.1 Minimality of VPC

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

$$[\![\lambda]\!]_{\mathsf{Th}} \stackrel{\text{def}}{=} \begin{cases} \tau, & \text{if } \lambda=\tau,\\ a(\widehat{n}), & \text{if } \lambda=a(\underline{n}),\\ \overline{a}(\widehat{n}), & \text{if } \lambda=\overline{a}(\underline{n}).\end{cases}$$

Three aspects of the encoding call for explanation.

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

  $$(c_0)(\overline{c_0}(x)\,|\,c_0(z_0).(c_1)(S_{c_1}(z_0)\,|\,c_1(z_1).(\ldots c_{k-1}(z_{k-1}).S_c(z_{k-1})\ldots))).$$

  The idea is that the term $\mathsf{s}^k(x)$, for $k>0$, is translated to an element of the numeric system $\langle\{\widehat{0},\widehat{1},\widehat{2},\ldots,\widehat{n},\ldots\},S_d(x)\rangle$, 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 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

  $$(\widetilde{c})(\prod_{1\leq i\leq n_1} D_{c_i^1}(t_i^1)\,|\,\ldots\,|\,\prod_{1\leq i\leq n_k} D_{c_i^k}(t_i^k)\,|\,c_1^1(z_1^1)\ldots c_{n^k}^k(z_{n^k}^k).\sum_{1\leq i\leq k}[\![\varphi_i\lambda_i.T_i]\!]_{\mathsf{Th}}),$$

  where $\widetilde{c}=c_1^1\ldots c_{n^k}^k$.

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 6. *Suppose P is a $\mathbb{VPC}$-process that does not contain any occurrences of $<$. The following correspondences hold:*
(i) If $P\stackrel{\lambda}{\longrightarrow}P'$ then $[\![P]\!]_{\mathsf{Th}}\rightarrow^*\stackrel{[\![\lambda]\!]_{\mathsf{Th}}}{\longrightarrow}P_1=_{\mathbb{VPC}_{\mathsf{Th}}}[\![P']\!]_{\mathsf{Th}}$ for some $\mathbb{VPC}_{\mathsf{Th}}$-process P_1;
(ii) If $[\![P]\!]_{\mathsf{Th}}\stackrel{\lambda}{\longrightarrow}P_1$ then $P\stackrel{\lambda'}{\longrightarrow}P'$ for some $\mathbb{VPC}$-process P' and some λ' such that $P_1=_{\mathbb{VPC}_{\mathsf{Th}}}[\![P']\!]_{\mathsf{Th}}$ and $[\![\lambda']\!]_{\mathsf{Th}}=\lambda$.

It is possible to strengthen Proposition 6. In fact the composition of the relation

$$\{(P,[\![P]\!]_{\mathsf{Th}})\mid P \text{ is a } \mathbb{VPC} \text{ process}\}$$

with $=_{\mathsf{Th}}$ 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 [Fu12b] for details.

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

Fig. 7. Encoding of $\mathbb{VPC}$ into Turing Complete $\mathbb{VPC}_{\mathsf{Th}}$

Theorem 4. *Suppose P, Q are $\mathbb{VPC}$-processes that do not contain any occurrences of $<$. Then $P =_{\mathbb{VPC}} Q$ if and only if $[\![P]\!]_{\mathsf{Th}} =_{\mathbb{VPC}_{\mathsf{Th}}} [\![Q]\!]_{\mathsf{Th}}$.*

Proof. In view of Proposition 1, we only have to prove the theorem for the external bisimilarities. Let $\mathcal{R}$ be the following relation

$$
\left\{ ((\widetilde{c})(P_1 \mid \ldots \mid P_k), (\widetilde{c})(Q_1 \mid \ldots \mid Q_k)) \;\middle|\; \begin{array}{l} c \text{ is } c_1,\ldots,c_j \text{ for some } j;\ P_i \text{ is neither}\\ \text{a composition nor a localization; and}\\ [\![P_i]\!]_{\mathsf{Th}} \to^* Q_i, \text{ where } Q_i \text{ is obtained}\\ \text{from } [\![P_i]\!]_{\mathsf{Th}} \text{ by resolving the } D_c\text{'s.} \end{array} \right\}.
$$

Now $\mathcal{R}$ is a $\mathbb{VPC}_{\mathsf{Th}}$-bisimulation up to strong bisimilarity [Mil89a]. Notice that every $\mathbb{VPC}$-process is equal to a process of the form $(\widetilde{c})(P_1 \mid \ldots \mid P_k)$, where for each $i \in \{1,\ldots,k\}$, the process P_i is neither a composition nor a localization. □

So the translation $[\![\_]\!]_{\mathsf{Th}}$ is correct for the $\mathbb{VPC}$-processes that do not refer to the relation '$<$'. The restriction can be removed according to Proposition 5. We conclude that $\mathbb{VPC}$ is a submodel of every Turing complete $\mathbb{VPC}_{\mathsf{Th}}$. In other words it is the least expressive among all Turing complete value-passing calculi.

8 Conclusion

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. It has taken some time to reach this level of formality. The study of Milner [Mil89a] 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 [Hoa78, Hoa85], which is a value-passing calculus, is essentially algebraic. The operational semantics for CSP *a la* pure CCS is introduced in [Bro83, BHR84, Ros97]. The labeled transition semantics for the value-passing CCS appears in [HI93a, HI93b]. A more serious attempt to study the operational semantics of the value-passing calculi is reported in the seminal paper by Hennessy and Lin [HL95]. 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 [HL95].

The observational equality of this paper is an application of the universal equality of Fu [Fu12b] 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 [BHR84, Ros97], 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 [Hen91, HI93a, HI93b]. The observational equivalence, the weak bisimilarity, and its symbolic characterization is systematically studied in [HL95]. Proof systems for these equivalences have also been studied in [Hen91, HI93a, HI93b, HL96]. 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 [Lin93, Lin96, Lin98]. 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 is given by Ingolfsdottir and Lin [IL01].

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 [Pal03, FL10]. 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 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 [Fu12b] and is strictly more expressive than the Interactive Machine Model [Fu12b].

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

Problem 1. Does $\simeq^s_{\mathbb{VPC}}$ 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 for example the process $A_0 \stackrel{\text{def}}{=} a(x).\mathit{if}\ x$ is even $\mathit{then}\ \overline{b}(x) + a(x).\mathit{if}\ x$ is odd $\mathit{then}\ \overline{b}(x)$ and the process $A_1 \stackrel{\text{def}}{=} a(x).\overline{b}(x) + a(x).\mathit{if}\, x$ is even $\mathit{then}\, \overline{b}(x) + a(x).\mathit{if}\, x$ is odd $\mathit{then}\, \overline{b}(x)$. An implementation of $\mathit{if}\, x$ is even $\mathit{then}\, \overline{b}(x)$ is $(c)(\overline{c}(\underline{0}) \mid !c(y).\mathit{if}\, x = y\ \mathit{then}\, \overline{b}(x)\ \mathit{else}\, \overline{c}(\mathsf{s}(\mathsf{s}(y))))$. The term $\mathit{if}\, x$ is odd $\mathit{then}\, \overline{b}(x)$ is defined similarly. It is clear that $A_0 =_{\mathbb{VPC}} A_1$. On the other hand $A_0 \not\simeq^s_{\mathbb{VPC}} A_1$. The transition $A_1 \stackrel{a(x)}{\longrightarrow}_\top \overline{b}(x)$ 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 investigation along this line, the symbolic approach must be modified in a couple of ways. 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. Secondly the symbolic bisimilarity should be defined by a family of relations indexed by the first order formulas [HL95, IL01]. The introduction of the arithmetic operators does not change the grammar of $\mathbb{VPC}$. No $\mathbb{VPC}$-terms would contain any arithmetic operators. So the logic expressions of $\mathbb{VPC}$ are still decidable. The extra expressive power is only exploited in verification. We may ask the following question.

Problem 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 $\mathbb{VPC}_{\mathsf{Th}}$-terms. A natural question to ask is how to extend AS_{Th} to a complete system for the finite control $\mathbb{VPC}_{\mathsf{Th}}$-terms. Hennessy, Lin and Rathke have discussed the issue in [HL97, Rat97a, Rat97b, HLR97]. It should be routine to adapt their approach and Milner's original approach [Mil89b] to $\mathbb{VPC}_{\mathsf{Th}}$. Additional care should be taken to divergence [LDH02, LDH05, Fu12a]. The details are yet to be worked out.

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 [FZ11]. 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.

Acknowledgement. The author would like to thank the members of BASICS, especially to Yijia Chen, Huan Long and Jianxin Xue, for their interest and feedbacks. The support from NSFC (60873034, 61033002) and STCSM (11XD1402800) is gratefully acknowledged.

References

[BHR84] Brookes, S., Hoare, C., Roscoe, A.: A theory of communicating sequential processes. Journal of ACM 31, 560–599 (1984)

[Bro83] Brookes, S.: A Model of Communicating Sequential Processes. PhD thesis, Oxford University (1983)

[Cut80] Cutland, N.: Computability: An Introduction to Recursive Function Theory. Cambridge University Press (1980)

[DNMV90] De Nicola, R., Mantanari, U., Vaandrager, F.: Back and forth bisimulations. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 152–165. Springer, Heidelberg (1990)

[FL10] Fu, Y., Lu, H.: On the expressiveness of interaction. Theoretical Computer Science 411, 1387–1451 (2010)

[FR74] Fischer, M., Rabin, M.: Super-exponential complexity of presburger arithmetic. In: Karp, R. (ed.) Complexity of Computation, pp. 27–41. American Mathematical Society (1974)

[Fu12a] Fu, Y.: Nondeterministic structure of computation (2012)

[Fu12b] Fu, Y.: Theory of interaction (2012)

[FZ11] Fu, Y., Zhu, H.: The name-passing calculus (2011)

[Gö31] Gödel, K.: Über formal unentscheidbare sätze der principia mathematica und verwandter systeme. Monatshefte für Mathematik und verwandter Systeme I 38, 173–198 (1931)

[Hen91] Hennessy, M.: A proof system for communicating processes with value-passing. Journal of Formal Aspects of Computer Science 3, 346–366 (1991)

[HI93a] Hennessy, M., Ingólfsdóttir, A.: Communicating processes with value-passing and assignment. Journal of Formal Aspects of Computing 5, 432–466 (1993)

[HI93b] Hennessy, M., Ingólfsdóttir, A.: A theory of communicating processes with value-passing. Information and Computation 107, 202–236 (1993)

[HL95] Hennessy, M., Lin, H.: Symbolic bisimulations. Theoretical Computer Science 138, 353–369 (1995)

[HL96] Hennessy, M., Lin, H.: Proof systems for message passing process algebras. Formal Aspects of Computing 8, 379–407 (1996)

[HL97] Hennessy, M., Lin, H.: Unique fixpoint induction for message-passing process calculi. In: Proc. Computing: Australian Theory Symposium (CAT 1997), vol. 8, pp. 122–131 (1997)

[HLR97] Hennessy, M., Lin, H., Rathke, J.: Unique fixpoint induction for message-passing process calculi. Science of Computer Programming 41, 241–275 (1997)

[HM85] Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. Journal of ACM 32, 137–161 (1985)

[Hoa78] Hoare, C.: Communicating sequential processes. Communications of ACM 21, 666–677 (1978)

[Hoa85] Hoare, C.: Communicating Sequential Processes. Prentice Hall (1985)

[IL01] Ingólfsdóttir, A., Lin, H.: A symbolic approach to value-passing processes. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebra, pp. 427–478. North-Holland (2001)

[LDH02] Lohrey, M., D'Argenio, P.R., Hermanns, H.: Axiomatising divergence. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 585–596. Springer, Heidelberg (2002)

[LDH05] Lohrey, M., D'Argenio, P., Hermanns, H.: Axiomatising divergence. Information and Computatio 203, 115–144 (2005)

[Lin93] Lin, H.: A verification tool for value-passing processes. In: Proceedings of 13th International Symposium on Protocol Specification, Testing and Verification, IFIP Transactions (1993)

[Lin96] Lin, H.: Symbolic transition graphs with assignment. In: Sassone, V., Montanari, U. (eds.) CONCUR 1996. LNCS, vol. 1119, pp. 50–65. Springer, Heidelberg (1996)

[Lin98] Lin, H.: "On-the-fly" instantiation of value-passing processes. In: Proc. FORTH/PSTV 1998 (1998)

[Mil89a] Milner, R.: Communication and Concurrency. Prentice Hall (1989)

[Mil89b] Milner, R.: A complete axiomatization system for observational congruence of finite state behaviours. Information and Computation 81, 227–247 (1989)

[MPW92] Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes. Information and Computation 100, 1–40 (Part I), 41–77 (Part II) (1992)

[Opp78] Oppen, D.: A $2^{2^{2^{pn}}}$ upper bound on the complexity of presburger arithmetic. Journal of Computer and System Sciences 16, 323–332 (1978)

[Pal03] Palamidessi, C.: Comparing the expressive power of the synchronous and the asynchronous π-calculus. Mathematical Structures in Computer Science 13, 685–719 (2003)

[Par81] Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)

[Pre29] Presburger, M.: Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. In: Warsaw Mathematics Congress, vol. 395, pp. 92–101 (1929)

[Pri78] Priese, L.: On the concept of simulation in asynchronous, concurrent systems. Progress in Cybernatics and Systems Research 7, 85–92 (1978)

[PS95] Parrow, J., Sangiorgi, D.: Algebraic theories for name-passing calculi. Information and Computation 120, 174–197 (1995)

[Rat97a] Rathke, J.: Symbolic Techniques for Value-Passing Calculi. PhD thesis, University of Sussex (1997)

[Rat97b] Rathke, J.: Unique fixpoint induction for value-passing processes. In: Proc. LICS 1997. IEEE Press (1997)

[Rog87] Rogers, H.: Theory of Recursive Functions and Effective Computability. MIT Press (1987)

[Ros97] Roscoe, A.: The Theory and Practice of Concurrency. Prentice Hall (1997)

[San93] Sangiorgi, D.: From π-calculus to higher order π-calculus–and back. In: Gaudel, M.-C., Jouannaud, J.-P. (eds.)TAPSOFT 1993. LNCS, vol. 668, pp. 151–166. Springer, Heidelberg (1993)

[Tho89] Thomsen, B.: A calculus of higher order communicating systems. In: Proc. POPL 1989, pp. 143–154 (1989)

[Tho93] Thomsen, B.: Plain chocs — a second generation calculus for higher order processes. Acta Informatica 30, 1–59 (1993)

[Tho95] Thomsen, B.: A theory of higher order communicating systems. Information and Computation 116, 38–57 (1995)

[vGW89] van Glabbeek, R., Weijland, W.: Branching time and abstraction in bisimulation semantics. In: Information Processing 1989, pp. 613–618. North-Holland (1989)