A Complete Symbolic Bisimulation for Full Applied Pi Calculus

Jia Liu[1,2,⋆] and Huimin Lin[1]

[1] State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
[2] Graduate University, Chinese Academy of Sciences
{jliu,lhm}@ios.ac.cn

Abstract. Symbolic characterizations of bisimilarities for the applied pi-calculus proposed so far are sound but incomplete, even restricted to the finite fragment of the calculus. In this paper we present a novel approach to symbolic semantics for the applied pi-calculus, leading to a notion of symbolic bisimulation which is both sound and complete with respect to the standard concrete bisimulation. Moreover, our approach accommodates recursions hence works for the full calculus.

1 Introduction

The applied pi-calculus [2] can be regarded as a generalization of the spi-calculus [3] in that it allows user-provided primitives for cryptographic operations. The calculus inherits the constructs for communication, concurrency and scope extrusion from the original pi-calculus [21]. It has a special mechanism for outputting compound messages which entails an auxiliary substitution construct of the form $\{M/x\}$, known as an "active substitution" that behaves like a floating "let" and serves to capture the partial environmental knowledge.

As in the standard operational semantics for the pi-calculus and value-passing CCS, in the applied pi-calculus an input may give rise to an infinite number of branches, which hinders the efforts to develop automated tools for the calculus. The standard approach is to develop symbolic theory which enables more amenable and more efficient automatic verification. *Symbolic semantics* have been developed smoothly for the value-passing CCS and many of its derivatives[15,9,18], in which all possible values offered by the environment are replaced by a *single* "symbolic variable". However it turns out that defining a symbolic semantics for the applied pi-calculus is unexpectedly technically difficult [13]. Despite various efforts from the community, up till now no complete symbolic semantics has been established for the applied pi-calculus, even when restricted to the finite fragment of the calculus, *i.e.* the fragment without recursions.

⋆ This work is supported by the National Natural Science Foundation of China (Grants No.60721061 and No.60833001) and the National High Technology Development Program of China (Grant No.2007AA01Z147).

J. van Leeuwen et al. (Eds.): SOFSEM 2010, LNCS 5901, pp. 552–563, 2010.

Symbolic bisimulations for value-passing CCS [15] and the pure pi-calculus [18,9] are indexed by boolean conditions to constrain the sets of values which symbolic variables may take. To achieve completeness, when comparing two processes for bisimilarity, it is necessary to "partition" the indexing condition into some sub-conditions in such a way that the two processes can simulate each other's transitions under each sub-condition. In the applied pi-calculus a symbolic bisimulation will be indexed by *constraints* each of which encompass a similar condition as well as *deducibilities*. The concept of deducibilities is by now widely used in modeling security protocols to represent all the messages that can be computed by intruders using a given set of messages [20,12,4,13]. As such deducibilities can not, and should not, be partitioned. In the previous work on symbolic bisimulation for the applied pi-calculus, not only conditions and deducibilities are entangled together in constraints, but also constraints and processes are mixed in transitions and bisimulations. This makes it impossible to partition indexing conditions, causing incompleteness.

To harness the difficulty we need to separate constraints from processes and conditions from deducibilities. To this end we extend the language for boolean expressions with two novel operators, $\sigma \blacktriangleright \Phi$, read "$\sigma$ guards Φ", and $\mathrm{H}n.\,\Phi$, read "hide n in Φ". In $\sigma \blacktriangleright \Phi$, the substitution σ represents (partial) environmental knowledge for the variables occurring in Φ. In $\mathrm{H}n.\,\Phi$ the name n is made local in Φ. With these operators we are able to make constraints completely independent of processes and to clearly separate conditions from deducibilities. As a result our notion of a symbolic bisimulation will be of the form $A \approx^{(\mathcal{D},\Phi)} B$, where $\mathcal{D}$ is a special set of deducibilities and Φ a condition formula, only the later is subject to partitioning when A and B are compared for bisimilarity. Besides, recursions were outside the scope of [13,14], due to difficulties arising from infinite name binders. Having hiding in formulas helps to keep track of the scope of names in constraints, which enables us to handle recursions smoothly with an "on-the-fly" approach.

The main contributions of the paper are twofold: (1) establishing the first sound and complete theory of symbolic bisimulation for the applied pi-calculus, and (2) our result covers recursions and is the first symbolic theory for the full applied pi-calculus.

Related work. Our work is inspired by [13,14] and the notion of "intermediate processes" used in this paper is taken from there. An ad hoc symbolic method relying on a notion of unification is proposed in [8] for the analysis of security protocols in a calculus akin to the applied pi-calculus. In [17,5] two variants of the applied pi-calculus, called the extended pi-calculi and psi-calculi respectively, are proposed aiming at being more amenable to technical treatments. Automated tools for security protocols in the context of the applied pi calculus are reported in [6,7]. A sound symbolic semantics for spi-calculus is introduced in [11]. Recently this semantics is revised to achieve completeness in [10], but their techniques are not applicable to the applied pi-calculus.

Due to space limitation proofs have been omitted from this extended abstract. They can be found in the full version of this paper [19].

2 Applied Pi-Calculus

2.1 Syntax

We assume two disjoint, infinite sets $\mathcal{N}$ and $\mathcal{V}$ of names and variables, respectively. A signature is a finite set of function symbols, such as f, h, each having a non-negative arity. Terms are defined by the grammar:

$$\begin{array}{lll} M, N ::= a, b, c, \cdots, k, \cdots, m, n, \cdots, s & & \text{name} \\ \quad\quad x, y, z & & \text{variable} \\ \quad\quad f(M_1, \cdots, M_\ell) & & \text{function application} \end{array}$$

We write $names(M)$ and $vars(M)$ for the sets of names and resp. variables in M. Let $atoms(M) = vars(M) \cup names(M)$. A *ground term* is a term containing no variable.

We rely on a sort system including a universal base sort and a channel sort. The sort system splits $\mathcal{N}$ and $\mathcal{V}$ respectively into channel names $\mathcal{N}_{ch}$ and base names $\mathcal{N}_b$, channel variables $\mathcal{V}_{ch}$ and base variables $\mathcal{V}_b$. Function symbols take arguments and produce results of the base sort only. We will use a, b, c as channel names, s, k as base names, and m, n as names of any sort. We use meta variables u, v, w to range over both names and variables. We abbreviate tuples $u_1, \cdots, u_\ell$ and $M_1, \cdots, M_\ell$ to $\widetilde{u}$ and $\widetilde{M}$ respectively.

Plain processes are constructed using the standard operators 0 (nil), | (parallel composition), νn (name restriction), if-then-else (conditional), $u(x)$ (input), $\overline{u}\langle N\rangle$ (output), and recursion. Extended processes are created by extending plain processes with active substitutions that float and apply to any process coming into contact with it. The grammar for plain processes and extended processes are given below:

$$\begin{array}{lll} P_r, Q_r, R_r ::= & \text{plain processes} \\ \quad\quad 0 \\ \quad\quad P_r \mid Q_r \\ \quad\quad \nu n.P_r \\ \quad\quad \text{if } M = N \text{ then } P_r \text{ then } Q_r \\ \quad\quad u(x).P_r \\ \quad\quad \overline{u}\langle N\rangle.P_r \\ \quad\quad K\langle\widetilde{M}\rangle \end{array} \qquad \begin{array}{lll} A_r, B_r, C_r ::= & \text{extended processes} \\ \quad\quad P_r \\ \quad\quad A_r \mid B_r \\ \quad\quad \nu n.A_r \\ \quad\quad \nu x.A_r \\ \quad\quad \{M/x\} \end{array}$$

As in the pi-calculus, $u(x)$, νn and νx are binding, which lead to the usual notions of bound and free names and variables. We shall use $fn(A_r)$, $bn(A_r)$, $fv(A_r)$, and $bv(A_r)$ to denote the sets of free names and bound names (resp. variables), of A_r. Let $fnv(A_r) = fn(A_r) \cup fv(A_r)$ and $bnv(A_r) = bn(A_r) \cup bv(A_r)$. We shall identify α-convertible processes. Capture of bound names and bound variables is avoided by implicit α-conversion.

To define recursive processes we use process constants, ranged over by K, each of which is associated with a declaration of the form $K(\widetilde{x}) \triangleq P_r$, where $fv(P_r) \subseteq \{\widetilde{x}\}$ and $fn(P_r) = \emptyset$.

Substitutions are sort-respecting partial mappings of finite domains. Substitutions of terms for variables, ranged over by σ, θ, are always required to be cycle-free. The domain and range of σ are denoted by $dom(\sigma)$ and $range(\sigma)$, respectively. We say σ is *idempotent* if $dom(\sigma) \cap vars(range(\sigma)) = \emptyset$. We write $vars(\sigma)$ and $names(\sigma)$ for the sets of variables and names occurring in σ, respectively. Also let $atoms(\sigma) = vars(\sigma) \cup names(\sigma)$. Let Z be an expression which may be a term, a process, or a substitution. The application of σ to Z is written in postfix notation, $Z\sigma$. Let $\{M_1/x_1, \cdots, M_n/x_n\}\sigma = \{M_1\sigma/x_1, \cdots, M_n\sigma/x_n\}$. The composition of substitutions is denoted $\sigma_1 \circ \sigma_2$, and $Z(\sigma_1 \circ \sigma_2) = (Z\sigma_2)\sigma_1$. σ^* is the result of composing with σ repeatedly until obtaining an idempotent substitution. We use $\sigma_1 \cup \sigma_2$ to denote the union when their domains are disjoint.

Due to technical reasons active substitutions are required to be defined on the base sort only, but this does not impose any restrictions on applications. In an extended process, active substitutions must be cycle-free and there is at most one substitution for each variable and exactly one when the variable is restricted. We say A_r is *closed* if every variable is either bound or defined by an active substitution. A *frame* is an extended process built up from 0 and active substitutions by parallel composition and restriction. The domain of frame ϕ, denoted by $dom(\phi)$, is the set of variables x for which ϕ contains a substitution $\{M/x\}$ not under νx. The frame of an extended process A_r, denoted by $\phi(A_r)$, is obtained by replacing every plain process embedded in A_r with 0. Note that $dom(A_r)$ is the same as $dom(\phi(A_r))$.

2.2 Semantics

A *context* is an extended process with a hole. An *evaluation context* is a context in which the hole is not under an input, an output or a conditional. A *term context* is a term with holes. Terms are equipped with an equational theory $=_E$ that is an equivalence relation closed under substitutions of terms for variables, one-to-one renamings and term contexts. We write $M \neq_E N$ for the negation of $M =_E N$. Structural equivalence $\equiv$ is the smallest equivalence relation on extended processes closed by evaluation contexts, α-conversion and such that:

$$A_r \equiv A_r \mid 0$$
$$A_r \mid B_r \equiv B_r \mid A_r$$
$$A_r \mid (B_r \mid C_r) \equiv (A_r \mid B_r) \mid C_r$$
$$\nu x.\{M/x\} \equiv 0 \qquad \nu n.0 \equiv 0$$
$$\{M/x\} \equiv \{N/x\} \text{ when } M =_E N \qquad \nu u.\nu v.A_r \equiv \nu v.\nu u.A_r$$
$$\{M/x\} \mid A_r \equiv \{M/x\} \mid A_r\{M/x\} \qquad A_r \mid \nu u.B_r \equiv \nu u.(A_r \mid B_r) \text{ when } u \notin fnv(A_r)$$

The labeled operational semantics is given in Fig. 1. To handle recursions, we take advantage of an internal transition to unfold them.[1] We denote by $\Longrightarrow$ the

[1] We have chosen to use recursions with $K\langle\widetilde{M}\rangle \xrightarrow{\tau} P_r\{\widetilde{M}/\widetilde{x}\}$ to avoid the technical difficulties arising from the structural equivalence rule $K\langle\widetilde{M}\rangle \equiv P_r\{\widetilde{M}/\widetilde{x}\}$ or $!P \equiv P \mid !P$. Our results will carry over if replication is used with this rule removed and replaced by $!P \xrightarrow{\tau} P \mid !P$ in the operational semantics.

$$\text{Then if } M = M \text{ then } P_r \text{ else } Q_r \xrightarrow{\tau} P_r \qquad \text{Else if } M = N \text{ then } P_r \text{ else } Q_r \xrightarrow{\tau} Q_r \quad \text{if } M, N \text{ are ground terms and } M \neq_E N$$

$$\text{Comm } \overline{a}\langle M\rangle.P_r \mid a(x).Q_r \xrightarrow{\tau} P_r \mid Q_r\{M/x\} \qquad \text{Rec } K\langle\widetilde{M}\rangle \xrightarrow{\tau} P_r\{\widetilde{M}/\widetilde{x}\} \text{ where } K(\widetilde{x}) \triangleq P_r$$

$$\text{In } \; a(x).P_r \xrightarrow{a(M)} P_r\{M/x\} \qquad \text{Out-atom } \; \overline{a}\langle u\rangle.P_r \xrightarrow{\overline{a}\langle u\rangle} P_r \text{ where } u \in \mathcal{N}_{ch} \cup \mathcal{V}_b$$

$$\text{Open-atom } \frac{A_r \xrightarrow{\overline{a}\langle u\rangle} B_r \quad a \neq u}{\nu u.A_r \xrightarrow{\nu u.\overline{a}\langle u\rangle} B_r} \qquad \text{Scope } \frac{A_r \xrightarrow{\alpha} B_r \quad u \text{ does not occur in } \alpha}{\nu u.A_r \xrightarrow{\alpha} \nu u.B_r}$$

$$\text{Par } \frac{A_r \xrightarrow{\alpha} A'_r \quad bnv(\alpha) \cap fnv(B_r) = \emptyset}{A_r \mid B_r \xrightarrow{\alpha} A'_r \mid B_r} \qquad \text{Struct } \frac{A_r \equiv C_r \xrightarrow{\alpha} C'_r \equiv B_r}{A_r \xrightarrow{\alpha} B_r}$$

Fig. 1. The Operational Semantics

reflexive and transitive closure of $\xrightarrow{\tau}$, and write $\stackrel{\alpha}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$, and $\stackrel{\widehat{\alpha}}{\Longrightarrow}$ for $\stackrel{\alpha}{\Longrightarrow}$ if α is not τ and $\Longrightarrow$ otherwise.

We say two terms M and N are *equal in the frame* ϕ, written $(M = N)\phi$, iff $\phi \equiv \nu\widetilde{n}.\sigma$, $M\sigma =_E N\sigma$ and $\{\widetilde{n}\} \cap names(M, N) = \emptyset$ for some names $\widetilde{n}$ and substitution σ. The notion of *static equivalence* is introduced to ensure that two processes expose the same information to the environment. Static equivalence is decidable for a large class of equational theories [1].

Definition 1. *Two closed frames ϕ_1 and ϕ_2 are statically equivalent, written $\phi_1 \sim \phi_2$, if $dom(\phi_1) = dom(\phi_2)$, and for all terms M and N such that $vars(M, N) \subseteq dom(\phi_1)$ it holds $(M = N)\phi_1$ iff $(M = N)\phi_2$. Two closed extended processes A_r and B_r are statically equivalent, written $A_r \sim B_r$, when their frames are.*

Definition 2. Labeled bisimilarity ($\approx$) *is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A_r \mathcal{R} B_r$ implies:*

1. *$A_r \sim B_r$*
2. *if $A_r \xrightarrow{\alpha} A'_r$ and $fv(\alpha) \subseteq dom(A_r)$ and $bn(\alpha) \cap fn(B_r) = \emptyset$, then $B_r \stackrel{\widehat{\alpha}}{\Longrightarrow} B'_r$ and $A'_r \mathcal{R} B'_r$ for some B'_r.*

2.3 Intermediate Representation

The *intermediate representation* was introduced in [13] as a means to circumvent the difficulties caused by structural equivalence for developing symbolic semantics. In the sequel we shall abbreviate "intermediate" to "inter.". The grammar of inter. processes is given below:

$$\begin{array}{rcl} P, Q, R := & \text{inter. plain processes} \\ & 0 \\ & P \mid Q \\ & \text{if } M = N \text{ then } P \text{ else } Q \\ & u(x).P \\ & \overline{u}\langle M\rangle.P \\ & K\langle\widetilde{M}\rangle \end{array} \qquad \begin{array}{rcl} F, G, H := & \text{inter. framed processes} \\ & P \\ & \{M/x\} \\ & F \mid G \\ A, B, C := & \text{inter. extended processes} \\ & F \\ & \nu n.A \end{array}$$

$$\Gamma(0) = 0 \qquad \Gamma(u(x).P_r) = \nu\tilde{n}.u(x).P \quad \text{where } \Gamma(P_r) = \nu\tilde{n}.P$$

$$\Gamma(K\langle\widetilde{M}\rangle) = K\langle\widetilde{M}\rangle \qquad \Gamma(\overline{u}\langle N\rangle.P_r) = \nu\tilde{n}.\overline{u}\langle N\rangle.P \quad \text{where } \Gamma(P_r) = \nu\tilde{n}.P$$

$$\Gamma(\text{if } M = N \text{ then } P_r \text{ else } Q_r) = \nu\tilde{n}.\nu\tilde{m}.\text{ if } M = N \text{ then } P \text{ else } Q \quad \text{where } \Gamma(P_r) = \nu\tilde{n}.P,\ \Gamma(Q_r) = \nu\tilde{m}.Q$$

$$\Gamma(A_r \mid B_r) = \nu\tilde{n}.\nu\tilde{m}.(F \mid G)\varphi(F \mid G)^* \quad \text{where } \Gamma(A_r) = \nu\tilde{n}.F, \Gamma(B_r) = \nu\tilde{m}.G$$

$$\Gamma(\{M/x\}) = \{M/x\} \qquad \Gamma(\nu n.A_r) = \nu n.\Gamma(A_r) \qquad \Gamma(\nu x.A_r) = \Gamma(A_r)_{\backslash x}$$

where $\Gamma(A_r)_{\backslash x}$ is obtained by replacing $\{M/x\}$ in $\Gamma(A_r)$ with 0

Fig. 2. Transformation

Inter. processes are required to be *applied*, that is, each variable in $dom(A)$ occurs only once in A. Thus, for example, $\overline{a}\langle f(k)\rangle \mid \{f(k)/x\}$ is applied while $\overline{a}\langle x\rangle \mid \{f(k)/x\}$ is not. For an inter. framed process F, we write $\varphi(F)$ for the substitution obtained by taking the union of the active substitutions in F. As an example, $\varphi(\overline{a}\langle f(k)\rangle \mid \{k/x\} \mid \{h(k)/y\}) = \{k/x, h(k)/y\}$. An *inter. evaluation context* is an inter. extended process with a hole not under an input, an output or a conditional. We shall also identify α-convertible inter. processes.

The function Γ in Fig. 2 turns an extended processes into an inter. extended process("$\downarrow$" in [13]) where we assume bound names are pairwise-distinct and different from free names. It transforms an extended process into an inter. extended process by pulling all name binders to the top level, applying active substitutions and eliminating variable restrictions. For example, $\Gamma(\nu x.(\overline{a}\langle f(x)\rangle.\nu n.\overline{a}\langle n\rangle \mid \nu k.\{h(k)/x\})) = \nu n.\nu k.(\overline{a}\langle f(h(k))\rangle.\overline{a}\langle n\rangle \mid 0)$. For a recursion $K\langle\widetilde{M}\rangle$ it is feasible to work "on-the-fly" and keep $K\langle\widetilde{M}\rangle$ invariant under Γ. It can be shown that Γ preserves α-conversion.

The inter. processes are a selected subset of the original processes. It turns out that it is sufficient to build symbolic semantics on top of inter. processes only. Thus this representation behaves like some kind of "normal form" and indeed facilitates the development of a symbolic framework.

3 Constraints

Symbolic bisimulations for value-passing CCS [15] and original pi-calculus [18,9] are indexed by boolean conditions under which transitions can be fired, and to achieve completeness it is essential to partition the indexing boolean conditions when comparing processes for bisimilarity. In the applied pi-calculus a symbolic bisimulation will be indexed by constraints, each of which encompass a similar condition as well as *deducibilities*. The concept of deducibilities is by now widely used in modeling security protocols to represent all the messages that can be computed by intruders using a given set of messages [20,12,4,13]. In the previous work on symbolic bisimulation for the applied pi-calculus, not only conditions and deducibilities are entangled together in constraints, but constraints and processes are mixed in transitions and bisimulations. This makes it impossible to partition conditions, thus causing incompleteness.

In our framework constraints are separated from processes. Furthermore, a constraint is split into a "trail" of deducibilities and a formula, only the latter is subject to partitioning when two processes are compared for bisimilarity.

A *deducibility* takes the form $x : U$ where $x \in \mathcal{V}$, U is a finite subset of $\mathcal{N}_{ch} \cup \mathcal{V}_b$, and $x \notin U$. Let $names(U) = U \cap \mathcal{N}$ and $vars(U) = U \cap \mathcal{V}$. We say that a set of deducibilities is a *trail* if it can be ordered as $x_1 : U_1, \cdots, x_\ell : U_\ell$ satisfying the following conditions:

1. $x_1, \cdots, x_\ell$ are pairwise-distinct and do not appear in any U_j $(1 \leq j \leq \ell)$;
2. for each $1 \leq i < \ell$, $names(U_i) \supseteq names(U_{i+1})$ and $vars(U_i) \subseteq vars(U_{i+1})$.

We shall use $\mathcal{D}, \mathcal{E}, \mathcal{F}$ to range over trails.

Let $\mathcal{E} = \{ x_i : U_i \mid 1 \leq i \leq \ell \}$ be a trail. We write $dom(\mathcal{E})$ for the set $\{x_1, \cdots, x_\ell\}$ and $atoms(\mathcal{E})$ for the set $dom(\mathcal{E}) \cup \bigcup_{i=1}^{\ell} U_i$. We shall often abuse the notation by writing $fnv(\mathcal{E})$ for $atoms(\mathcal{E})$. A substitution θ *respects* $\mathcal{E}$ if

1. $dom(\theta) = dom(\mathcal{E})$, and
2. for any $1 \leq i \leq \ell$, $vars(x_i\theta) \subseteq U_i$ and $names(x_i\,\theta) \cap U_i = \emptyset$.

Given an inter. extended process $A = \nu\widetilde{n}.F$, we say $\mathcal{E}$ is *compatible with* A if

1. $dom(\mathcal{E}) \cap dom(A) = \emptyset$,
2. $vars(\bigcup_{i=1}^{\ell} U_i) \subseteq dom(A)$, $fv(A) \subseteq dom(A) \cup dom(\mathcal{E})$, and
3. for any $x_i : U_i$ and $y \in vars(U_i)$, $x_i \notin vars(y\varphi(F))$.

Our notion of a deducibility relies on a set of names and variables rather than a set of terms [20,4] or a frame [13]. By dropping the details, a collection of processes can share a common trail. In the above definition of trail, the sets U_i are in increasing order on variables while decreasing on names. It records the variables which can be used and names which can not be used by x_i. Intuitively $vars(U_i)$ can be understood as a snapshot of environmental knowledge at the time when x_i is input. Since the knowledge never decreases, the order of deducibilities on variables reflects a coarse-grained order of inputs recorded by timing information in [11,10].

Example 1. Let $A = \overline{x}_3\langle k \rangle \mid c(x) \mid \{f(x_1)/y\} \mid \{g(x_1, x_2)/z\}$, $\mathcal{D} = \{x_1 : \emptyset, x_2 : \{y\}, x_3 : \{y, z\}\}$ and $\mathcal{E} = \{x_3 : \{c\}, x_1 : \emptyset, x_2 : \{y\}\}$. Clearly both $\mathcal{D}$ and $\mathcal{E}$ are trails and compatible with A. Let $\theta = \{k/x_1,\ f(y)/x_2,\ c/x_3\}$. Then θ respects $\mathcal{D}$ but not $\mathcal{E}$, because $\mathcal{E}$ forbids x_3 to access c.

Formulas are specified by the following grammar:

$$\begin{array}{lcl} \Phi,\ \Psi & ::= & \mathrm{H}n.\Phi \mid \Phi \wedge \Phi \mid \sigma \blacktriangleright \Phi_{atom} \mid \Phi_{atom} \\ \Phi_{atom} & ::= & true \mid M = N \mid M \neq N \end{array}$$

where σ is idempotent. We identify $\sigma \blacktriangleright \Phi_{atom}$ with Φ_{atom} when $dom(\sigma) = \emptyset$. We write $fn(\Phi)$ and $fv(\Phi)$ for the sets of free names and free variables of Φ, respectively. In particular,

$$\begin{array}{ll} fn(\mathrm{H}n.\Phi) \triangleq fn(\Phi) \setminus \{n\} & fv(\mathrm{H}n.\Phi) \triangleq fv(\Phi) \\ fn(\sigma \blacktriangleright \Phi) \triangleq names(\sigma) \cup fn(\Phi) & fv(\sigma \blacktriangleright \Phi) \triangleq vars(\sigma) \cup fv(\Phi) \end{array}$$

In $\mathrm{H}n.\Phi$ the name n is binding, and we shall identify α-convertible formulas.

The satisfiability relation $\models$ between idempotent substitutions θ and formulas Φ are defined by induction on the structure of Φ:

$$\begin{array}{ll}
\theta \models \mathit{true} & \\
\theta \models M = N & \mathit{if}\ \ M\theta =_E N\theta \\
\theta \models M \neq N & \mathit{if}\ \ M\theta \neq_E N\theta \\
\theta \models \sigma \blacktriangleright \Phi & \mathit{if}\ \ \theta\sigma \text{ is cycle-free and } (\theta\sigma)^* \models \Phi \\
\theta \models \Phi \wedge \Psi & \mathit{if}\ \ \theta \models \Phi \text{ and } \theta \models \Psi \\
\theta \models \mathrm{H}n.\Phi & \mathit{if}\ \ n \notin \mathit{names}(\theta) \text{ and } \theta \models \Phi
\end{array}$$

In $\sigma \blacktriangleright \Phi$, σ represents the environmental knowledge accumulated so far to define some variables occurring in Φ, hence Φ should be evaluated with the application of σ. Having substitutions embedded in formulas echoes the fact that active substitutions are part of the syntax for the extended processes in applied pi-calculus. $\mathrm{H}n.\,\Phi$ hides n in Φ. In the original pi-calculus, a restricted name can never appear in any constraint since this private information cannot be accessed. In contrast, the applied pi-calculus provides a mechanism to indirectly access a datum which may contain restricted names. For example, by the rules in Fig. 1, we have $\nu k.\ (a(x).\ \text{if } x = k \text{ then } \overline{c} \text{ else } 0 \mid \{k/y\}) \xrightarrow{a(y)} \nu k.\ (\ \text{if } y = k \text{ then } \overline{c} \text{ else } 0 \mid \{k/y\}) \equiv \nu k.\ (\ \text{if } k = k \text{ then } \overline{c} \text{ else } 0 \mid \{k/y\}) \xrightarrow{\tau} \nu k.\ (\overline{c} \mid \{k/y\})$. To reflect this we equip every equality test on terms with its own "private" environmental knowledge, which relies on the introduction of hidden names in formulas. This makes it possible for formulas to "stand alone", without depending on processes, which significantly simplifies technical developments. In particular, α-conversion, which was forbidden in the symbolic semantics of [13,14], is allowed in our framework. Furthermore, introducing hidden names enables us to handle recursions smoothly, which is beyond the scope of any previous approach.

We call a pair $(\mathcal{D}, \Phi)$ a *constraint*, where $\mathcal{D}$ is a trail and Φ a formula. We denote by $\Omega(\mathcal{D}, \Phi)$ the set of substitutions that respect $\mathcal{D}$ and satisfy Φ. Note that "θ respects $\mathcal{D}$" if and only if $\theta \in \Omega(\mathcal{D}, \mathit{true})$, and $\Omega(\mathcal{D}, \Phi) \subseteq \Omega(\mathcal{D}, \mathit{true})$ for any Φ. We also observe that $\Omega(\mathcal{D}, \Phi \wedge \Psi) = \Omega(\mathcal{D}, \Phi) \cap \Omega(\mathcal{D}, \Psi)$. For a collection of formulas Σ, we use $\Omega(\mathcal{D}, \bigvee \Sigma)$ to refer to the set $\bigcup_{\Psi \in \Sigma} \Omega(\mathcal{D}, \Psi)$.

Definition 3. *A collection of formulas Σ is a* partition of Φ under $\mathcal{D}$ *if* $\Omega(\mathcal{D}, \Phi) \subseteq \Omega(\mathcal{D}, \bigvee \Sigma)$.

Example 2. Let $\Phi = \mathrm{H}m.\mathrm{H}s.(\{\mathit{enc}(m,s)/y\} \blacktriangleright (\mathit{dec}(x,s) = m))$ where *enc* and *dec* are encryption and decryption functions, respectively, with the equation $\mathit{dec}(\mathit{enc}(x,y),y) =_E x$. Clearly $\{\mathit{enc}(m,s)/x\} \models \mathit{dec}(x,s) = m$. From $\{y/x\}\{\mathit{enc}(m,s)/y\} = \{\mathit{enc}(m,s)/x\}$, we have $\{y/x\} \models \Phi$. Let $\mathcal{D} = \{x : \{y\}\}$. Then $\Omega(\mathcal{D}, \Phi) = \Omega(\mathcal{D}, x = y) = \{\, \{M/x\} \mid \mathit{vars}(M) \subseteq \{y\}, M =_E y \,\}$.

4 Symbolic Semantics

Symbolic structural equivalence $\equiv_s$ is the smallest equivalence relation on inter. extended processes which is closed by application of inter. evaluation contexts and α-conversion and satisfies:

assume $u, v \in \mathcal{N}_{ch} \cup \mathcal{V}_{ch}$

$$\text{Then}_s \quad \text{if } M = N \text{ then } P \text{ else } Q \xrightarrow{M=N,\,\tau} P$$

$$\text{Else}_s \quad \text{if } M = N \text{ then } P \text{ else } Q \xrightarrow{M\neq N,\,\tau} Q$$

$$\text{Comm}_s \quad \overline{u}\langle M\rangle.P \mid v(x).Q \xrightarrow{u=v,\,\tau} P \mid Q\{M/x\}$$

$$\text{In}_s \quad u(x).P \xrightarrow{true,\ u(x)} P \qquad \text{Outch}_s \quad \overline{u}\langle v\rangle.P \xrightarrow{true,\,\overline{u}\langle v\rangle} P$$

$$\text{Outt}_s \quad \overline{u}\langle M\rangle.P \xrightarrow[x\in\mathcal{V}_b,\ x\notin fv(\overline{u}\langle M\rangle.P)]{true,\,\nu x.\overline{u}\langle x\rangle} P \mid \{M/x\} \qquad \text{Rec}_s \quad K\langle\widetilde{M}\rangle \xrightarrow{true,\,\tau} \nu\widetilde{m}.P\{\widetilde{M}/\widetilde{x}\} \quad K(\tilde{x}) \triangleq P_r,\ \Gamma(P_r) = \nu\widetilde{m}.P,\ \{\widetilde{m}\}\cap fn(\widetilde{M}) = \emptyset$$

$$\text{Par}_s \quad \frac{A \xrightarrow{\Psi,\,\alpha} \nu\widetilde{n}.F \quad bv(\alpha)\cap fv(B)=\{\widetilde{n}\}\cap fn(B)=\emptyset}{A \mid B \xrightarrow{\Phi,\,\alpha} \nu\widetilde{n}.(F \mid B)} \quad \Phi \vDash \begin{cases} \Psi & \text{if } fv(\Psi) = \emptyset \\ (\sigma \cup \varphi(B)) \blacktriangleright \Psi_1 & \text{else if } \Psi = \sigma \blacktriangleright \Psi_1 \text{ and } dom(B)\cap dom(\sigma)=\emptyset \end{cases}$$

$$\text{Scope}_s \quad \frac{A \xrightarrow{\Psi,\,\alpha} A' \quad n\notin names(\alpha)}{\nu n.A \xrightarrow{\Phi,\,\alpha} \nu n.A'} \quad \Phi = \begin{cases} \Psi & \text{if } n \notin fn(\Psi) \\ \mathrm{H}n.\Psi & \text{otherwise} \end{cases}$$

$$\text{Opench}_s \quad \frac{A \xrightarrow{\Phi,\,\overline{u}\langle c\rangle} A' \quad u\neq c,\ c\in\mathcal{N}_{ch}}{\nu c.A \xrightarrow{\Phi,\,\nu c.\overline{u}\langle c\rangle} A'} \qquad \text{Struct}_s \quad \frac{A \equiv_s B \xrightarrow{\Phi,\,\alpha} B' \equiv_s A'}{A \xrightarrow{\Phi,\,\alpha} A'}$$

Fig. 3. Symbolic Operational Semantics

$$A \equiv_s A \mid 0 \qquad A \mid B \equiv_s B \mid A \qquad A \mid (B \mid C) \equiv_s (A \mid B) \mid C$$

A *symbolic action* is either an internal action τ, an input action $u(x)$ with $u \in \mathcal{N}_{ch} \cup \mathcal{V}_{ch}$ and $x \in \mathcal{V}$, an output action $\overline{u}\langle v\rangle$ with $u, v \in \mathcal{N}_{ch} \cup \mathcal{V}_{ch}$, or an bound output action $\nu w.\overline{u}\langle w\rangle$ with $u \in \mathcal{N}_{ch} \cup \mathcal{V}_{ch}$ and $w \in \mathcal{N}_{ch} \cup \mathcal{V}_b$.

Symbolic transition relations $\xrightarrow{\Phi,\,\alpha}$, where Φ is a formula and α a symbolic action, are defined by the rules in Fig. 3.

In Scope_s, we hide the restricted name n in Ψ so that it will not be exposed to the environment. In Par_s, when A is put in parallel with B, some of the variables recorded in the formula Ψ will be subject to the active substitutions in B, therefore we extract these substitutions from B (using operator φ) and add them to Ψ so that the formula Φ contains necessary environmental knowledge.

Example 3. By Then_s, Par_s and Scope_s, we have $\nu k.(\text{ if } x = k \text{ then } P \text{ else } Q \mid \{h(k)/y\}) \xrightarrow{\Phi,\,\tau} \nu k.(P \mid \{h(k)/y\})$ with $\Phi = \mathrm{H}k.(\{h(k)/y\} \blacktriangleright (x = k))$.

When unfolding a recursion $K\langle\widetilde{M}\rangle$, we transform P_r to $\Gamma(P_r)$ first. This causes a problem that the result of evolution of an inter. framed process may contain bound names. To obtain an inter. extended process, in Par_s rule the parallel component is placed inside the restricted names before juxtaposing it.

Example 4. Let $K(x) \triangleq P_r$ with $P_r = x(z).\nu n.\overline{z}\langle h(n)\rangle.K\langle x\rangle$. Clearly $\Gamma(P_r) = \nu n.\, x(z).\overline{z}\langle h(n)\rangle.K\langle x\rangle$. By Rec_s and Par_s, $K\langle a\rangle \xrightarrow{true,\,\tau} \nu n.\, a(z).\overline{z}\langle h(n)\rangle.K\langle a\rangle$ and $K\langle a\rangle \mid \overline{a}\langle c\rangle \xrightarrow{true,\,\tau} \nu n.\,(a(z).\overline{z}\langle h(n)\rangle.K\langle a\rangle \mid \overline{a}\langle c\rangle)$.

Suppose a trail $\mathcal{D} = \{ x_i : U_i \mid 1 \le i \le \ell \}$ is compatible with A and $A \xrightarrow{\Phi,\alpha} A'$. Then we define

$$\mathcal{X}(\alpha, dom(A), \mathcal{D}) \triangleq \begin{cases} \mathcal{D} \cup \{x : dom(A)\} & \alpha \text{ is } u(x) \\ \{ x_i : U_i \cup \{c\} \mid 1 \le i \le \ell \} & \alpha \text{ is } \nu c.\, \overline{u}\langle c \rangle \\ \mathcal{D} & \text{otherwise} \end{cases}$$

and we can show that $\mathcal{X}(\alpha, dom(A), \mathcal{D})$ is also a trail compatible with A'. Intuitively, function $\mathcal{X}$ updates the current trail so that the resulting trail will be compatible with the residual process after a transition. It records the current environmental knowledge on inputs and prevents the prior input variables from using the fresh names generated by bound outputs.

Example 5. Starting with an empty trail, we have

$$\begin{array}{lll} & \nu k.\nu c.\ (a(x).\overline{b}\langle c\rangle \mid \{h(k)/z\}) & \emptyset \\ \xrightarrow{true,a(x)} & \nu k.\nu c.\ (\overline{b}\langle c\rangle \mid \{h(k)/z\}) & \{x : \{z\}\} \\ \xrightarrow{true,\nu c.\overline{b}\langle c\rangle} & \nu k.\ \{h(k)/z\} & \{x : \{c, z\}\} \end{array}$$

The updated trails are shown on the right. When inputting x, the knowledge represented by z is available to x. The private channel name c are opened after the input of x, hence x cannot access c.

To compare the knowledge of two inter. processes, we define *symbolic static equivalence* similar to [13,4].

Definition 4. *Let $(\mathcal{D}, \Phi)$ be a constraint, $A = \nu\widetilde{n}_1.F_1$ and $B = \nu\widetilde{n}_2.F_2$ inter. extended processes. A and B are* symbolically statically equivalent *w.r.t. $(\mathcal{D}, \Phi)$, written $A \sim^{(\mathcal{D},\Phi)} B$ if*

1. *$\mathcal{D}$ is compatible with A and B,*
2. *$dom(A) = dom(B)$,*
3. *for some fresh variables $x, y \in \mathcal{V}_b$, then $\Omega(\mathcal{E}, \Phi \wedge \Phi_1) = \Omega(\mathcal{E}, \Phi \wedge \Phi_2)$ where $\mathcal{E} = \mathcal{D} \cup \{x : dom(A),\ y : dom(A)\}$ and $\Phi_i = \mathrm{H}\widetilde{n}_i.\, \varphi(F_i) \blacktriangleright (x = y)$, $i = 1, 2$.*

Intuitively, $x = y$ can be understood as representing a set of concrete equality tests on terms like $M = N$. The above definition symbolically characterizes the idea that such equality tests will be satisfied simultaneously by the frames of the concrete processes corresponding to A and B.

The symbolic weak transition relations $\overset{\Phi,\gamma}{\Longrightarrow}$ (γ is α or ϵ) are defined thus:

$$A \overset{true,\epsilon}{\Longrightarrow} A \qquad A \xrightarrow{\Phi,\alpha} B \text{ implies } A \overset{\Phi,\alpha}{\Longrightarrow} B$$

$$A \xrightarrow{\Phi,\tau} \overset{\Psi,\gamma}{\Longrightarrow} B \text{ implies } A \overset{\Phi\wedge\Psi,\gamma}{\Longrightarrow} B \qquad A \overset{\Phi,\gamma}{\Longrightarrow} \xrightarrow{\Psi,\tau} B \text{ implies } A \overset{\Phi\wedge\Psi,\gamma}{\Longrightarrow} B$$

We write $\overset{\Phi,\widehat{\alpha}}{\Longrightarrow}$ to mean $\overset{\Phi,\alpha}{\Longrightarrow}$ if α is not τ and $\overset{\Phi,\epsilon}{\Longrightarrow}$ otherwise. Let

$$[\alpha = \beta] \triangleq \begin{cases} u = v & \alpha = u(x), \beta = v(x) \\ & \quad \text{or } \alpha = \nu w.\overline{u}\langle w\rangle, \beta = \nu w.\overline{v}\langle w\rangle \\ (u = v) \wedge (w = w') & \alpha = \overline{u}\langle w\rangle,\ \beta = \overline{v}\langle w'\rangle \\ true & \alpha = \beta = \tau \end{cases}$$

We now proceed to define the notion of a symbolic bisimulation. In the definition below, we assume $x \in \mathcal{V}_{ch}$, $y \in \mathcal{V}_b$ and $c \in \mathcal{N}_{ch}$.

Definition 5. *A constraint indexed family of symmetric relations* $\mathcal{S} = \{S^{(\mathcal{D},\Phi)}\}$ *between inter. extended processes is a* symbolic bisimulation *if for any* $(A, B) \in S^{(\mathcal{D},\Phi)}$,

1. $A \sim^{(\mathcal{D},\Phi)} B$
2. *for some* $\Delta = \{x, y, c\}$ *and* $\Delta \cap fnv(A, B, \mathcal{D}, \Phi) = \emptyset$, *if* $A \xrightarrow{\Phi_1,\, \alpha} A'$ *and* $bnv(\alpha) \subset \Delta$, *let* $\mathcal{F} = \mathcal{X}(\alpha, dom(A), \mathcal{D})$, *then there is a partition* Σ *of* $\Phi \wedge \Phi_1$ *under* $\mathcal{F}$, *such that for any* $\Psi \in \Sigma$ *there are* Φ_2, β, B' *satisfying*
   (a) $B \overset{\Phi_2,\, \widehat{\beta}}{\Longrightarrow} B'$
   (b) $\Omega(\mathcal{F}, \Psi) \subseteq \Omega(\mathcal{F},\ [\alpha = \beta] \wedge \Phi_2)$
   (c) $(A', B') \in S^{(\mathcal{F},\Psi)}$.

We write $A \approx^{(\mathcal{D},\Phi)} B$ if there is a symbolic bisimulation $\mathcal{S}$ such that $(A, B) \in S^{(\mathcal{D},\Phi)}$ and $S^{(\mathcal{D},\Phi)} \in \mathcal{S}$.

Now we state the main result of this paper:

Theorem 1. *Let* A_r *and* B_r *be two closed extended processes. Then* $A_r \approx B_r$ *iff* $\Gamma(A_r) \approx^{(\emptyset,\, true)} \Gamma(B_r)$.

Example 6. Let $A = \nu c.\, a(x).\, \overline{x}\langle c\rangle$ and $B = \nu c.\, a(x).$ if $x = a$ then $\overline{a}\langle c\rangle$ else $\overline{x}\langle c\rangle$. Clearly $A \approx B$. Let $B_1 = \nu c.$ if $x = a$ then $\overline{a}\langle c\rangle$ else $\overline{x}\langle c\rangle$. The following sets constitute a symbolic bisimulation (the symmetric pairs are omitted):

$$
\begin{array}{ll}
S^{(\emptyset,\, true)} & = \{(A,\ B)\} \\
S^{(\{x:\emptyset\},\, x=a)} & = \{(\nu c.\, \overline{x}\langle c\rangle,\ \nu c.\, \overline{a}\langle c\rangle)\} \\
S^{(\{x:\{c\}\},\, x=a)} & = \{(0,\ 0)\}
\end{array}
\qquad
\begin{array}{ll}
S^{(\{x:\emptyset\},\, true)} & = \{(\nu c.\, \overline{x}\langle c\rangle,\ B_1)\} \\
S^{(\{x:\emptyset\},\, x\neq a)} & = \{(\nu c.\, \overline{x}\langle c\rangle,\ \nu c.\, \overline{x}\langle c\rangle)\} \\
S^{(\{x:\{c\}\},\, x\neq a)} & = \{(0,\ 0)\}.
\end{array}
$$

Hence $A \approx^{(\emptyset,\, true)} B$.

This is a typical example which shows partitions of indexing conditions are necessary for symbolic bisimulations to be complete. Simple as they are, A and B are *not* symbolically equivalent according to any previous symbolic theory for the applied pi-calculus [13,14].

Observational equivalence has been shown to coincide with the labeled bisimilarity in [2]. Hence by transitivity, our symbolic bisimulation is fully abstract w.r.t observational equivalence.

5 Conclusion

We have presented a symbolic framework for the applied pi calculus in which a sound and complete notion of symbolic bisimulation is devised. This is achieved by a careful separation of condition formulas from deducibilities, and constraints from processes. Moreover, our framework accommodates recursions hence our result works for the full applied pi calculus. To the best of our knowledge, this is the first symbolic theory for the applied pi calculus which is both sound and complete and which allows recursion.

As future work we would like to formulate a symbolic style proof system for the applied pi calculus along the lines of [16,9,18].

References

1. Abadi, M., Cortier, V.: Deciding Knowledge in Security Protocols under Equational Theories. Theor. Comput. Sci. 367(1-2), 2–32 (2006)
2. Abadi, M., Fournet, C.: Mobile Values, New Names, and Secure Communication. In: POPL, pp. 104–115 (2001)
3. Abadi, M., Gordon, A.D.: A Calculus for Cryptographic Protocols: the spi Calculus. In: CCS 1997: Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 36–47. ACM, New York (1997)
4. Baudet, M.: Deciding Security of Protocols against Off-Line Quessing Attacks. In: CCS 2005: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 16–25. ACM, New York (2005)
5. Bengtson, J., Johansson, M., Parrow, J., Victor, B.: Psi-Calculi: Mobile Processes, Nominal Data, and Logic. In: LICS 2009: Proceedings of the 2009 24th Annual IEEE Symposium on Logic In Computer Science, pp. 39–48 (2009)
6. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: CSFW 2001: Proceedings of the 14th IEEE Workshop on Computer Security Foundations, p. 82 (2001)
7. Blanchet, B., Abadi, M., Fournet, C.: Automated Verification of Selected Equivalences for Security Protocols. In: LICS, pp. 331–340 (2005)
8. Boreale, M., Buscemi, M.: A Method for Symbolic Analysis of Security Protocols. Theor. Comput. Sci. 338(1-3), 393–425 (2005)
9. Boreale, M., De Nicola, R.: A Symbolic Semantics for the Pi-Calculus. Inf. Comput. 126(1), 34–52 (1996)
10. Borgström, J.: A Complete Symbolic Bisimilarity for an Extended spi Calculus. Electron. Notes Theor. Comput. Sci. 242(3) (2009)
11. Borgström, J., Briais, S., Nestmann, U.: Symbolic Bisimulation in the spi Calculus. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004. LNCS, vol. 3170, pp. 161–176. Springer, Heidelberg (2004)
12. Comon-Lundh, H., Shmatikov, V.: Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive or. In: LICS 2003: Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science, p. 271 (2003)
13. Delaune, S., Kremer, S., Ryan, M.D.: Symbolic Bisimulation for the Applied Pi Calculus. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 133–145. Springer, Heidelberg (2007)
14. Delaune, S., Kremer, S., Ryan, M.D.: Symbolic Bisimulation for the Applied pi Calculus. Journal of Computer Security (to appear, 2009)
15. Hennessy, M., Lin, H.: Symbolic Bisimulations. Theor. Comput. Sci. 138(2), 353–389 (1995)
16. Hennessy, M., Lin, H.: Proof Systems for Message-Passing Process Algebras. Formal Asp. Comput. 8(4), 379–407 (1996)
17. Johansson, M., Parrow, J., Victor, B., Bengtson, J.: Extended Pi-Calculi. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 87–98. Springer, Heidelberg (2008)
18. Lin, H.: Complete Inference Systems for Weak Bisimulation Equivalences in the Pi-Calculus. Inf. Comput. 180(1), 1–29 (2003)
19. Liu, J., Lin, H.: A Complete Symbolic Bisimulation for Full Applied pi Calculus (full version) (2009), `http://lcs.ios.ac.cn/~jliu`
20. Millen, J., Shmatikov, V.: Constraint Solving for Bounded-Process Cryptographic Protocol Analysis. In: CCS 2001: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 166–175. ACM, New York (2001)
21. Milner, R.: Communicating and Mobile Systems: the pi Calculus. Cambridge University Press, Cambridge (1999)