Symbolic Bisimulation for the Applied Pi Calculus [*]

Stéphanie Delaune[1,2,3], Steve Kremer[2], and Mark Ryan[3]

[1] LORIA, CNRS & INRIA, France
[2] LSV, ENS Cachan & CNRS & INRIA, France
[3] School of Computer Science, University of Birmingham, UK

Abstract. We propose a symbolic semantics for the finite applied pi calculus, which is a variant of the pi calculus with extensions for modelling cryptographic protocols. By treating inputs symbolically, our semantics avoids potentially infinite branching of execution trees due to inputs from the environment. Correctness is maintained by associating with each process a set of constraints on terms. We define a sound symbolic labelled bisimulation relation. This is an important step towards automation of observational equivalence for the finite applied pi calculus, e.g. for verification of anonymity or strong secrecy properties.

1 Introduction

The *applied pi calculus* [2] is a derivative of the pi calculus that is specialised for modelling cryptographic protocols. Participants in a protocol are modelled as processes, and the communication between them is modelled by means of channels, names and message passing. The main difference with the pi calculus is that the applied pi calculus allows one to manipulate *complex data*, instead of just names. These data are generated by a term algebra and equality is treated modulo an *equational theory*. For instance the equation $dec(enc(x, y), y) = x$ models the fact that encryption and decryption with the same key cancel out in the style of the Dolev-Yao model. Such complex data requires the use of a special kind of processes called *active substitutions*. As an example consider the following process and reduction step.

$$\nu a, k.\mathsf{out}(c, enc(a, k)).P \xrightarrow{\nu x.out(c,x)} \nu a, k.(P \mid \{^{enc(a,k)}/_x\}).$$

The process outputs a secret name a which has been encrypted with the secret key k on a public channel c. The active substitution $\{^{enc(a,k)}/_x\}$ gives the environment the ability to access the term $enc(a, k)$ via the fresh variable x without revealing a or k. The applied pi calculus also generalizes the *spi calculus* [3] which only allows a fixed set of built-in primitives (symmetric and public-key encryption), while the applied pi calculus allows one to define a variety of primitives by means of an equational theory.

One of the difficulties in automating the proof of properties of systems in the applied pi calculus is the infinite number of possible behaviours of the attacker, even in the case that the protocol process itself is finite. When the process requests an input

from the environment, the attacker can give any term which can be constructed from the terms it has learned so far in the protocol, and therefore the execution tree of the process is potentially infinite-branching. To address this problem, researchers have proposed *symbolic abstractions* of processes, in which terms input from the environment are represented as symbolic variables, together with some constraints. These constraints describe the knowledge of the attacker (and therefore, the range of possible values of the symbolic variable) at the time the input was performed.

Reachability properties can be verified by deciding satisfiability of constraint systems resulting from symbolic executions of process algebras (e.g. [16, 4]). Similarly, *off-line guessing attacks* coded as *static equivalence* between process states [5] can be decided using such symbolic executions, but this requires one to check the equivalence of constraint systems, rather than satisfiability. Decision procedures for both satisfiability [11] and equivalence [5] of constraint systems exist for significant families of equational theories. *Observational equivalence properties*, which can be characterized as a bisimulation, express the inability of the attacker to distinguish between two processes no matter how it interacts with them. These properties are useful for modelling anonymity and privacy properties (e.g. [12]), as well as strong secrecy. Symbolic methods have also been used for bisimulation in process algebras [14, 9]. In particular, Borgström *et al.* [10] define a sound symbolic bisimulation for the spi calculus.

In this paper we propose a symbolic semantics for the applied pi calculus together with a sound symbolic bisimulation. To show that a symbolic bisimulation implies the concrete one, we generally need to prove that the symbolic semantics is both sound and complete. The semantics of the applied pi calculus is not well suited for defining such a symbolic semantics. In particular, we argue in Section 2 that defining a symbolic structural equivalence which is both sound and complete seems impossible. The absence of sound and complete symbolic structural equivalence significantly complicates the proof of our main result. We therefore split it into two parts. We define a more restricted semantics which will provide an *intermediate* representation of applied pi calculus processes. These intermediate processes are a selected (but sufficient) subset of the original processes. One may think of them as being processes in some kind of normal form. We equip these intermediate processes with a labelled bisimulation that coincides with the original one. Then we present a symbolic semantics which is both sound and complete with respect to the intermediate one and give a sound symbolic bisimulation. To keep track of the constraints on symbolic variables we associate a separate constraint system to each symbolic process. Keeping these constraint systems separate allows us to have a clean separation between the bisimulation and the constraint solving part. In particular we can directly build on existing work [5] and obtain a decision procedure for our symbolic bisimulation for a significant family of equational theories whenever the constraint system does not contain disequalities. This corresponds to the fragment of the applied pi calculus without else branches in the conditional. For this fragment, one may also notice that our symbolic semantics can be used to verify reachability properties using the constraint solving techniques from [11]. Another side-effect of the separation between the processes and the constraint system is that we forbid α-conversion on symbolic processes as we lose the scope of names in the constraint system, but allow explicit renaming when necessary (using *naming environments*). We believe that the

simplicity of our intermediate calculus (especially the structural equivalence) and the absence of α-conversion is appealing in view of an implementation. Finally, one may note that as in [10, 8], our technique for deciding bisimulation is incomplete (see Section 5.1). However, we argue that our technique works for many interesting cases. The intermediate semantics and proofs are omitted, but can be found in [13].

2 The Applied Pi Calculus

The applied pi calculus [2] is a language for describing processes and their interactions. We only consider the *finite* applied pi calculus which does not have process replication. Details about syntax and semantics of the original applied pi calculus may be found in [2]. We briefly recall them for the convenience of the reader.

Terms are defined as names, variables, and function symbols applied to other terms (of base type). We denote by $\mathcal{N}$ (resp. $\mathcal{X}$) the set of names (resp. variables) and distinguish the set $\mathcal{N}_{ch}$ (resp. $\mathcal{X}_{ch}$) of channel names (resp. variables) and the set $\mathcal{N}_b$ (resp. $\mathcal{X}_b$) of names (resp. variables) of base type. We define the equations which hold on terms as an equational theory E. We denote $=_{\mathsf{E}}$ the equivalence relation induced by E. A typical example of an equational theory is $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$.

Plain processes (P, Q, R) are built up in a similar way to processes in the pi calculus, except that messages can contain terms (rather than just names). *Extended processes* (A, B, C) add *active substitutions* $\{^M/_x\}$, and restriction on variables. An evaluation context $C[\_]$ is an extended process with a hole instead of an extended process.

As usual, names and variables have scopes, which are delimited by restrictions and by inputs. We write $fv(A)$, $bv(A)$, $fn(A)$ and $bn(A)$ for the sets of free and bound variables (resp. names). In an extended process, there is at most one substitution for each variable, and exactly one when the variable is restricted. An extended process is *closed* if all its variables are either bound or defined by an active substitution. Active substitutions allow us to map an extended process A to its *frame* $\phi(A)$ by replacing every plain process in A with 0. The domain of a frame φ, denoted by $\mathrm{dom}(\varphi)$, is the set of variables for which φ contains a substitution $\{^M/_x\}$ not under νx.

Throughout the paper we always suppose that substitutions are cycle-free. Given substitutions σ_1 and σ_2 with $\mathrm{dom}(\sigma_1) \cap \mathrm{dom}(\sigma_2) = \emptyset$, we write $\sigma_1 \cup \sigma_2$ to denote the substitution whose domain is $\mathrm{dom}(\sigma_1) \cup \mathrm{dom}(\sigma_2)$ and that is equal to σ_1 on $\mathrm{dom}(\sigma_1)$ and to σ_2 on $\mathrm{dom}(\sigma_2)$. We write $\sigma_1\sigma_2$ for the substitution σ whose domain is $\mathrm{dom}(\sigma_1)$ and such that $x\sigma = (x\sigma_1)\sigma_2$. We define $\mathrm{img}(\sigma)$ to be $\{x\sigma \mid x \in \mathrm{dom}(\sigma)\}$. We write $\sigma^\star$ to emphasize that we iterate the substitution until obtaining idempotence.

Semantics. Structural equivalence, noted $\equiv$, is the smallest equivalence relation on extended processes that is closed under α-conversion on names and variables, application of evaluation contexts, and some other standard rules such as associativity and commutativity of the parallel operator and commutativity of the bindings. In addition the following three rules are related to active substitutions and equational theories:

$$\nu x.\{^M/_x\} \equiv 0, \ \{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}, \ \text{and} \ \{^M/_x\} \equiv \{^N/_x\} \ \text{if} \ M =_{\mathsf{E}} N$$

As mentioned in the introduction, it seems difficult to define symbolic structural equivalence ($\equiv_s$) which is sound and complete in the following (informal) sense:

- *Soundness*: $P_s \equiv_s Q_s$ implies for any valid instantiation σ, $P_s\sigma \equiv Q_s\sigma$;
- *Completeness*: $P_s\sigma \equiv Q$ implies there exists Q_s such that $P_s \equiv_s Q_s$ and $Q_s\sigma = Q$.

To see this, consider the process $P = \text{in}(c, x).\text{in}(c, y).\text{out}(c, f(x)).\text{out}(c, g(y))$ which can be reduced to $P' = \text{out}(c, f(M_1)).\text{out}(c, g(M_2))$ where M_1 and M_2 are two arbitrary terms provided by the environment. In the case that $f(M_1) =_\mathsf{E} g(M_2)$, we have $P' \equiv \nu z.(\text{out}(c, z).\text{out}(c, z) \mid \{f(M_1)/z\})$, but this structural equivalence does not hold whenever $f(M_1) \neq_\mathsf{E} g(M_2)$. The aim of our symbolic semantics is to avoid instantiating the variables x and y: the process P would reduce to $P'_s = \text{out}(c, f(x)).\text{out}(c, g(y))$. In this case we need to keep auxiliary information that allows us to infer that x and y may take arbitrary values. The process P'_s represents the two cases in which x and y are equal or distinct. Hence, the question of whether the symbolic structural equivalence $P'_s \equiv_s \nu z.(\text{out}(c, z).\text{out}(c, z) \mid \{f(x)/z\})$ is valid cannot be decided, as it depends on the concrete values of x and y. Therefore, our notion of symbolic structural equivalence is sound but not complete in the sense above (we will give a weaker completeness result). This seems to be an inherent problem and it propagates to internal and labelled reduction, since they are closed under structural equivalence. In this example, the *control flow* is not affected by whether $f(x) =_\mathsf{E} g(y)$. When control flow is affected by conditions on input variables, we maintain those conditions as a set of constraints.

Internal reduction $\rightarrow$ is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts such that

COMM	$\text{out}(a, M).P \mid \text{in}(a, x).Q \rightarrow P \mid Q\{M/x\}$	
THEN	if $M = N$ then P else $Q \rightarrow P$	where $M =_\mathsf{E} N$
ELSE	if $M = N$ then P else $Q \rightarrow Q$	where M, N are ground and $M \neq_\mathsf{E} N$

Note that the presentation of the internal reduction slightly differs from the one given in [2], but it is easily shown to be equivalent.

The operational semantics is extended by a *labelled* operational semantics enabling us to reason about processes that interact with their environment. Below, a and c are channel names whereas x is a variable of base type.

IN $\quad$ $\text{in}(a, x).P \xrightarrow{in(a,M)} P\{M/x\}$

SCOPE $\quad \dfrac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$

OUT-CH $\quad$ $\text{out}(a, c).P \xrightarrow{out(a,c)} P$

OPEN-CH $\quad \dfrac{A \xrightarrow{out(a,c)} A' \quad c \neq a}{\nu c.A \xrightarrow{\nu c.out(a,c)} A'}$

PAR $\quad \dfrac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = \emptyset \quad bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$

OUT-T $\quad$ $\text{out}(a, M).P \xrightarrow{\nu x.out(a,x)} P \mid \{M/x\}$
$\qquad\qquad\qquad\qquad x \notin fv(P) \cup fv(M)$

STRUCT $\quad \dfrac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}$

Our rules differ slightly from those described in [2], although we prove in [13] that labelled bisimulation in our system coincides with labelled bisimulation in [2].

Equivalences. In [2], it is shown that observational equivalence coincides with labelled bisimilarity. This relation is like the usual definition of bisimilarity, except that at each step one additionally requires that the processes are statically equivalent.

Definition 1 (static equivalence ($\sim$)). *Two closed frames φ_1, φ_2 are statically equivalent if $\varphi_1 \equiv \nu\tilde{n}.\sigma_1$ and $\varphi_2 \equiv \nu\tilde{n}.\sigma_2$ for some names $\tilde{n}$ and substitutions σ_1, σ_2 s.t.*

(i) $\mathrm{dom}(\varphi_1) = \mathrm{dom}(\varphi_2)$,
(ii) $\forall M, N$ *such that* $(fn(M) \cup fn(N)) \cap \tilde{n} = \emptyset$, $M\sigma_1 =_\mathsf{E} N\sigma_1 \Leftrightarrow M\sigma_2 =_\mathsf{E} N\sigma_2$.

Example 1. Let $\varphi_0 = \nu k.\sigma_0$ and $\varphi_1 = \nu k.\sigma_1$ where $\sigma_0 = \{\mathsf{enc}(s_0, k)/x_1, k/x_2\}$, $\sigma_1 = \{\mathsf{enc}(s_1, k)/x_1, k/x_2\}$ and s_0, s_1 and k are names. Let E be the theory defined by the axiom $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$. We have $\mathsf{dec}(x_1, x_2)\sigma_0 =_\mathsf{E} s_0$ whereas $\mathsf{dec}(x_1, x_2)\sigma_1 \neq_\mathsf{E} s_0$, thus $\varphi_0 \not\sim \varphi_1$.

Definition 2 (labelled bisimilarity ($\approx$)). Labelled bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that $A \mathcal{R} B$ implies*

1. $\phi(A) \sim \phi(B)$,
2. *if $A \to A'$, then $B \to^* B'$ and $A' \mathcal{R} B'$ for some B',*
3. *if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq \mathrm{dom}(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathcal{R} B'$ for some B'.*

3 Constraint Systems

The idea of symbolic semantics is to avoid infinite branching due to inputs from the environment. This is achieved by inputting a variable rather than one of infinitely many possible terms, and maintaining *constraints* on what value the variable may take.

Definition 3 (constraint system). *A* constraint system $\mathcal{C}$ *is a set of constraints where every constraint is either*

– *a* deducibility constraint *of the form $\varphi \Vdash x$ where φ is a frame and x a variable, or*
– *a constraint of the form $M = N$, $M \neq N$ or $\mathsf{gd}(M)$ where M, N are terms.*

The constraint $\varphi \Vdash x$ is useful for specifying the information φ held by the environment when it supplies an input x. The constraint $\mathsf{gd}(M)$ means that M is ground. We denote by $names(\mathcal{C})$ (resp. $vars(\mathcal{C})$) for the names (resp. variables) of $\mathcal{C}$. We define $cv(\mathcal{C}) = \{x \mid \varphi \Vdash x \in \mathcal{C}\}$ to be the *constraint variables* of $\mathcal{C}$, and assume that those constraint variables do not appear in the domain of any frame in $\mathcal{C}$. The constraint systems that we consider arise while executing symbolic processes. We therefore restrict ourselves to *well-formed* constraint systems, capturing the fact that the knowledge of the environment always increases along the execution: we allow it to use more names and variables (less restrictions) or give it access to more terms (larger substitution). More formally, $\phi_1 \stackrel{def}{=} \nu\tilde{u}_1.\sigma_1 \preceq \nu\tilde{u}_2.\sigma_2 \stackrel{def}{=} \phi_2$ if $\tilde{u}_1 \supseteq \tilde{u}_2$, and $\mathrm{dom}(\sigma_1) \subseteq \mathrm{dom}(\sigma_2)$ and $\forall y \in \mathrm{dom}(\sigma_1). y\sigma_1 = y\sigma_2$.

Definition 4 (well-formed constraint system). *A constraint system $\mathcal{C}$ is* well-formed *if its deducibility constraints can be written $\phi_1 \Vdash x_1, \ldots, \phi_\ell \Vdash x_\ell$ such that $\phi_1 \preceq \phi_2 \preceq \ldots \preceq \phi_n$ and $\forall i. 1 \leq i \leq \ell, \forall x \in vars(\mathrm{img}(\phi_i)) \cap cv(\mathcal{C}), \exists j < i. x = x_j$.*

The second condition corresponds to the way in which variables are bound: each time a symbolic message M (which may contain variables) is put in the frame the variables in $vars(M)$ have to have been previously instantiated. Hence, those variables have to appear on the right of a smaller deducibility constraint. Given a constraint system $\mathcal{C}$ we write $\mathcal{D}ed(\mathcal{C}) = \{\phi_1 \Vdash x_1, \ldots, \phi_\ell \Vdash x_\ell\}$. Two well-formed constraint systems $\mathcal{C}$ and $\mathcal{C}'$ with $\mathcal{D}ed(\mathcal{C}) = \{\phi_1 \Vdash x_1, \ldots, \phi_\ell \Vdash x_\ell\}$ and $\mathcal{D}ed(\mathcal{C}') = \{\phi'_1 \Vdash x'_1, \ldots, \phi'_\ell \Vdash x'_\ell\}$ have *same basis* if $x_i = x'_i$ and $\mathrm{dom}(\phi_i) = \mathrm{dom}(\phi'_i)$ for $1 \leq i \leq \ell$.

Definition 5 (E-solution). *Let $\mathcal{C}$ be a well-formed constraint system such that $\mathcal{D}ed(\mathcal{C}) = \{\phi_1 \Vdash x_1, \ldots, \phi_\ell \Vdash x_\ell\}$ where each $\phi_i = \nu\tilde{u}_i.\sigma_i$ for some $\tilde{u}_i$ and some substitution σ_i. An E-solution of $\mathcal{C}$ is a substitution θ whose domain is $cv(\mathcal{C})$ and such that*

- $vars(x_i\theta) \cap cv(\mathcal{C}) = \emptyset$ *and* $vars(x_i\theta) \cap (\mathrm{dom}(\phi_\ell) \smallsetminus \mathrm{dom}(\phi_i)) = \emptyset$;
- $names(x_i\theta) \cap \tilde{u}_i = \emptyset$ *and* $vars(x_i\theta) \cap \tilde{u}_i = \emptyset$;
- *for "$M = N$" $\in \mathcal{C}$ (resp. "$M \neq N$" $\in \mathcal{C}$), we have $M(\theta\sigma_\ell)^\star =_E N(\theta\sigma_\ell)^\star$ (resp. $M(\theta\sigma_\ell)^\star \neq_E N(\theta\sigma_\ell)^\star$);*
- *for "$\mathsf{gd}(M)$" $\in \mathcal{C}$, we have that the term $M(\theta\sigma_\ell)^\star$ is ground.*

We denote by $Sol_E(\mathcal{C})$ the set of E-solutions of $\mathcal{C}$. An E-solution θ of $\mathcal{C}$ is closed *if $vars(x_i\theta) \subseteq \mathrm{dom}(\phi_i)$ for any $i \in \{1, \ldots, \ell\}$.*

Example 2. Let $\mathcal{C} = \{\nu k.\nu s.\{^{\mathsf{enc}(s,k)}/_{y_1}, {}^{k}/_{y_2}\} \Vdash x', \mathsf{gd}(c), x' = s\}$. Let E be the equational theory $\mathsf{dec}(\mathsf{enc}(x, y), y) = x$ and $\theta = \{^{\mathsf{dec}(y_1, y_2)}/_{x'}\}$. We have that θ is a closed E-solution of $\mathcal{C}$. Note that $\theta' = \{^{\mathsf{dec}(y_1, k)}/_{x'}\}$ is not an E-solution of $\mathcal{C}$.

4   Symbolic Applied Pi Calculus

Intermediate extended processes (denoted A, B, C) are given by the grammar below. They may be seen as an extended process in normal form.

$P, Q, R :=$	inter. plain process		$F, G, H :=$	P	inter. framed process
0				$\{^{M}/_x\}$	
$P \mid Q$				$F \mid G$	
if $M = N$ then P else Q					
$\mathsf{in}(u, x).P$			$A, B, C :=$	F	inter. extended processes
$\mathsf{out}(u, N).P$				$\nu n.A$	

A symbolic process is an intermediate extended process together with a *constraint system*. We require intermediate extended processes to be

- *name and variable distinct* (*nv-distinct*): $bn(A) \cap fn(A) = bv(A) \cap fv(A) = \emptyset$ and any name and variable is bound at most once; and
- *applied*, meaning that each variable in $\mathrm{dom}(A)$ occurs only once in A.

Intuitively, in an applied process all active substitutions have been applied. For instance the extended process $\mathsf{out}(c, x) \mid \{^{M}/_x\}$ is not applied, as x occurs twice. A symbolic process is made up of two parts: a process and a constraint system. The nv-distinctness condition allows us to link the names and the variables in the constraint systems to those used in the process. We denote by $\psi(A)$ the substitution obtained when taking the active substitutions $\{^{M}/_x\}$ in A. We now define the $\downarrow$ operator which transforms an nv-distinct process into an intermediate process.

Definition 6 ($A\downarrow$). *Given an nv-distinct extended process A, the intermediate extended process $A\downarrow$ is defined inductively as follows.*

$$0\downarrow = 0 \qquad\qquad in(u,x).P\downarrow = \nu\tilde{n}.in(u,x).P' \qquad (\nu n.A)\downarrow = \nu n.(A\downarrow)$$
$$\{^M/_x\}\downarrow = \{^M/_x\} \qquad out(u,N).P\downarrow = \nu\tilde{n}.out(u,N).P' \qquad (\nu x.A)\downarrow = \tilde{A}$$
$$if\ M = N\ then\ P\ else\ Q\downarrow = \nu\tilde{n}.\nu\tilde{m}.if\ M = N then\ P'\ else\ Q'$$
$$(A \mid B)\downarrow = \nu\tilde{n}.\nu\tilde{m}.(A' \mid B')(\psi(A') \cup \psi(B'))^\star$$

where $P\downarrow = \nu\tilde{n}.P'$, $Q\downarrow = \nu\tilde{m}.Q'$, $A\downarrow = \nu\tilde{n}.A'$, $B\downarrow = \nu\tilde{m}.B'$, and $\tilde{A}$ is $A\downarrow$ but with the unique occurrence of $\{^M/_x\}$ replaced by 0.

For example, let $A = \nu x.(\text{in}(c,y).\nu b.\text{out}(a,x) \mid \{^{f(b)}/_x\})$. Then $A\downarrow = \nu b.\text{in}(c,y).\text{out}(a,f(b))$. Note that the processes A and $A\downarrow$ are bisimilar but not structurally equivalent. As expected, an *intermediate context* is an intermediate extended process with a hole instead of an intermediate extended process. An *intermediate evaluation context* is an intermediate context whose hole is not under a conditional, an input or an output. We also define what it means to apply an evaluation context on a constraint system. This is needed because we define the semantics in a compositional way.

Definition 7 (constraint system $C[\mathcal{C}]$). *Let $C = \nu\tilde{n}.(\_ \mid D)$ be an intermediate evaluation context and e be a constraint. We have that*

- $C[e] = e\psi(D)$ *when e is a constraint of the form $M = N$, $M \neq N$ or $\mathsf{gd}(M)$;*
- $C[\nu\tilde{v}.\sigma \Vdash x] = \nu\tilde{n}.\nu\tilde{v}.(\sigma \cup \psi(D)) \Vdash x$ *otherwise.*

Given a constraint system $\mathcal{C}$, we have that $C[\mathcal{C}] = \{C[e] \mid e \in \mathcal{C}\}$.

As we do not allow α-conversion we explicitly run intermediate extended processes in a *naming environment* $\mathsf{N} : \mathcal{N} \cup \mathcal{X} \rightarrow \{\mathsf{n}, \mathsf{f}, \mathsf{b}, \mathsf{c}\}$. Intuitively, $\mathsf{N}(u) = \mathsf{f}$ if the name or variable u occurs *free* in A, and $\mathsf{N}(u) = \mathsf{b}$ if u has been *bound* and will not be used again. $\mathsf{N}(u) = \mathsf{n}$ means u is *new* and has not been used before, either as free or bound. $\mathsf{N}(x) = \mathsf{c}$ means that the variable x is a *constraint* variable (i.e. an input from the environment subject to constraints in $\mathcal{C}$). This discipline helps us avoid name and variable conflicts. If $\mathsf{N}(u) = t$ then the naming environment $\mathsf{N}' = \mathsf{N}[u \mapsto t']$ is defined to be the same as N except that $\mathsf{N}'(u) = t'$; and $\mathsf{N}[U \mapsto t']$ is defined as $\mathsf{N}[u_1 \mapsto t', \ldots, u_n \mapsto t']$ if $U = \{u_1, \ldots, u_n\}$. If U is a set of names and variables then $\mathsf{N}(U) = \{\mathsf{N}(u) \mid u \in U\}$, and we write $\mathsf{N}(U) = t$ if $\mathsf{N}(U) \subseteq \{t\}$. A naming environment N is compatible with an intermediate extended process A and a constraint system $\mathcal{C}$ if

- $\mathsf{N}(fn(A)) = \mathsf{f}$ $-\ \mathsf{N}(bn(A) \cup bv(A)) = \mathsf{b}$ $-\ \mathsf{N}(names(\mathcal{C})) \subseteq \{\mathsf{f}, \mathsf{b}\}$
- $\mathsf{N}(fv(A)) \subseteq \{\mathsf{f}, \mathsf{c}\}$ $-\ \mathsf{N}(x) = \mathsf{c}$ iff $x \in cv(\mathcal{C})$ $-\ \mathsf{N}(vars(\mathcal{C})) \subseteq \{\mathsf{f}, \mathsf{c}, \mathsf{b}\}$

Definition 8 (Symbolic process). *A* symbolic process *is a triple $(A\ ;\ \mathcal{C}\ ;\ \mathsf{N}_s)$ where A is an intermediate extended process, $\mathcal{C}$ a constraint system and N_s a naming environment compatible with A and $\mathcal{C}$. The symbolic process $(A\ ;\ \mathcal{C}\ ;\ \mathsf{N})$ is* well-formed *if $\mathcal{C}$ is well-formed and if $\phi(A) \succeq max\{\phi \mid \phi \Vdash x \in \mathcal{C}\}$ when $\mathcal{D}ed(\mathcal{C}) \neq \emptyset$.*

Given a well-formed symbolic process $(A\ ;\ \mathcal{C}\ ;\ \mathsf{N})$ we define by $Sol_{\mathsf{E}}(\mathcal{C}\ ;\ \mathsf{N})$ the set of solutions of $\mathcal{C}$ which are compatible with N, i.e.

$$Sol_{\mathsf{E}}(\mathcal{C}, \mathsf{N}) = \{\theta \mid \theta \in Sol_{\mathsf{E}}(\mathcal{C}), \mathsf{N}(names(\text{img}(\theta)) \cup vars(\text{img}(\theta))) = \mathsf{f}\}.$$

Example 3. Let $A = \text{out}(c, x)$, $\mathcal{C} = \{\nu a.\nu b.\{^b/_y\} \Vdash x, \ x \neq c\}$ and N be a naming environment compatible with A and $\mathcal{C}$ such that $\mathsf{N}(d) = \mathsf{f}$. Let $\theta_1 = \{^d/_x\}$, $\theta_2 = \{^y/_x\}$. We have that $\theta_1, \theta_2 \in Sol_{\mathsf{E}}(\mathcal{C}, \mathsf{N})$. Hence $\text{out}(c, d)$ (resp. $\text{out}(c, b)$) is the concrete process obtained by the solution θ_1 (resp. θ_2). However, note that $\text{out}(c, a)$ is not a concretization of $(A \; ; \; \mathcal{C} \; ; \; \mathsf{N})$.

4.1 Symbolic Semantics

Symbolic structural equivalence ($\equiv_s$) is the smallest equivalence relation on well-formed symbolic processes such that:

$\text{PAR-0}_s \qquad\qquad (A \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \equiv_s (A \mid 0 \; ; \; \mathcal{C} \; ; \; \mathsf{N})$
$\text{PAR-A}_s \qquad (A \mid (B \mid D) \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \equiv_s ((A \mid B) \mid D \; ; \; \mathcal{C} \; ; \; \mathsf{N})$
$\text{PAR-C}_s \qquad\quad (A \mid B \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \equiv_s (B \mid A \; ; \; \mathcal{C} \; ; \; \mathsf{N})$
$\text{NEW-C}_s \qquad (\nu n.\nu m.A \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \equiv_s (\nu m.\nu n.A \; ; \; \mathcal{C} \; ; \; \mathsf{N})$

$$\frac{(A \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \equiv_s (B \; ; \; \mathcal{C}_B \; ; \; \mathsf{N})}{(C[A] \; ; \; C[\mathcal{C}_A] \; ; \; \mathsf{N}') \equiv_s (C[B] \; ; \; C[\mathcal{C}_B] \; ; \; \mathsf{N}')} \quad \begin{array}{l} \text{where } \mathsf{N}' = \mathsf{N}[S \mapsto \mathsf{b}] \text{ for some set} \\ \text{of names } S \text{ such that } \mathsf{N}(S) = \mathsf{f} \end{array}$$

Symbolic internal reduction $\rightarrow_s$ is the smallest relation on well-formed symbolic processes closed under $\equiv_s$, application of intermediate evaluation context and such that:

$\text{COMM}_s \quad (\text{out}(u, M).P \mid \text{in}(v, x).Q \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \rightarrow_s$
$$(P \mid Q\{^M/_x\} \; ; \; \mathcal{C} \cup \{u = v \; , \; \mathsf{gd}(u) \; , \; \mathsf{gd}(v)\} \; ; \; \mathsf{N})$$
$$\text{where } u, v \in \mathcal{N}_{ch} \cup (cv(\mathcal{C}) \cap \mathcal{X}_{ch}).$$

$\text{THEN}_s \qquad (\text{if } M = N \text{ then } P \text{ else } Q \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \rightarrow_s (P \; ; \; \mathcal{C} \cup \{M = N\} \; ; \; \mathsf{N})$

$\text{ELSE}_s \quad (\text{if } M = N \text{ then } P \text{ else } Q \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \rightarrow_s (Q \; ; \; \mathcal{C} \cup \{M \neq N \; ; \; \mathsf{gd}(M) \; ; \; \mathsf{gd}(N)\} \; ; \; \mathsf{N})$

Symbolic labelled reduction is the smallest relation closed under symbolic structural equivalence ($\equiv_s$) and such that

$\text{IN}_s \qquad (\text{in}(u, x).P \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \xrightarrow{in(u,y)}_s (P\{^y/_x\} \; ; \; \mathcal{C} \cup \{0 \Vdash y, \mathsf{gd}(u)\} \; ; \; \mathsf{N}[y \mapsto \mathsf{c}])$
$$\text{where } u \in \mathcal{N}_{ch} \cup (\mathcal{X}_{ch} \cap cv(\mathcal{C})), \ \mathsf{N}(y) = \mathsf{n}.$$

$\text{OUT-CH}_s \quad (\text{out}(u, v).P \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \xrightarrow{out(u,v)}_s (P \; ; \; \mathcal{C} \cup \{\mathsf{gd}(u), \mathsf{gd}(v)\} \; ; \; \mathsf{N})$
$$\text{where } u, v \in \mathcal{N}_{ch} \cup (\mathcal{X}_{ch} \cap cv(\mathcal{C})).$$

OUT-T_s

$\qquad (\text{out}(u, M).P \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \xrightarrow{\nu x.out(u,x)}_s (P \mid \{^M/_x\} \; ; \; \nu x.\mathcal{C} \cup \{\mathsf{gd}(u)\} \; ; \; \mathsf{N}[x \mapsto \mathsf{f}])$
$$\text{where } x \in \mathcal{X}_b, \ \mathsf{N}(x) = \mathsf{n}.$$

$$\text{OPEN-CH}_s \quad \frac{(A \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \xrightarrow{out(u,c)}_s (A' \; ; \; \mathcal{C}' \; ; \; \mathsf{N}') \qquad u \neq c, \ d \in \mathcal{N}_{ch}, \ \mathsf{N}(d) = \mathsf{n}}{(\nu c.A \; ; \; \nu c.\mathcal{C} \; ; \; \mathsf{N}[c \mapsto \mathsf{b}]) \xrightarrow{\nu d.out(u,d)}_s (A'\{^d/_c\} \; ; \; \nu d.(\mathcal{C}'\{^d/_c\}) \; ; \; \mathsf{N}'[c \mapsto \mathsf{b}, d \mapsto \mathsf{f}])}$$

$$\text{SCOPE}_s \qquad \frac{(A \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \xrightarrow{\alpha}_s (A' \; ; \; \mathcal{C}' \; ; \; \mathsf{N}') \qquad n \text{ does not occur in } \alpha}{(\nu n.A \; ; \; \nu n.\mathcal{C} \; ; \; \mathsf{N}[n \mapsto \mathsf{b}]) \xrightarrow{\alpha}_s (\nu n.A' \; ; \; \nu n.\mathcal{C}' \; ; \; \mathsf{N}[n \mapsto \mathsf{b}])}$$

$$\text{PAR}_s \qquad \frac{(A \; ; \; \mathcal{C} \; ; \; \mathsf{N}) \xrightarrow{\alpha}_s (A' \; ; \; \mathcal{C}' \; ; \; \mathsf{N}')}{(A \mid B \; ; \; \mathcal{C} \mid \psi(B) \; ; \; \mathsf{N}) \xrightarrow{\alpha}_s (A' \mid B \; ; \; \mathcal{C} \mid \psi(B) \; ; \; \mathsf{N}')}$$

We may note that the rules IN_s and OPEN-CH_s require "on-the-fly renaming". This will be needed in the bisimulation because we require both the left- and right-hand processes to use the same label without allowing α-conversion. When a transition is executed under a context (by the rules SCOPE_s and PAR_s) the constraint system must also be put in the context (according to Definition 7). In particular, these rules allow to add restrictions and active substitutions to the constraint $0 \Vdash y$ inserted by the rule IN_s.

Example 4. To illustrate our symbolic semantics, consider the process $(A \; ; \; \emptyset \; ; \; \mathsf{N})$ where $A = \nu k.\nu s.(\text{in}(c,x).\text{if } x = s \text{ then out}(c,ok) \mid \{^{\text{enc}(s,k)}/_{y_1}\} \mid \{^{k}/_{y_2}\})$ and N is a naming environment compatible with A. Let x' be a variable such that $\mathsf{N}(x') = \mathsf{n}$.

$$(A \; ; \; \emptyset \; ; \; \mathsf{N}) \xrightarrow{\;in(c,x')\;}_s (A' \; ; \; \{\nu k.\nu s.\{^{\text{enc}(s,k)}/_{y_1}, ^{k}/_{y_2}\} \Vdash x' \, , \, \text{gd}(c)\} \; ; \; \mathsf{N}[x' \mapsto \mathsf{c}])$$

$$\longrightarrow_s (\nu k.\nu s.(\text{out}(c,ok) \mid \{^{\text{enc}(s,k)}/_{y_1}\} \mid \{^{k}/_{y_2}\}) \; ; \; \mathcal{C} \; ; \; \mathsf{N}[x' \mapsto \mathsf{c}])$$

where $A' = \nu k.\nu s.(\text{if } x' = s \text{ then out}(c,ok) \mid \{^{\text{enc}(s,k)}/_{y_1}\} \mid \{^{k}/_{y_2}\})$ and $\mathcal{C}$ is the system $\{\nu k.\nu s.\{^{\text{enc}(s,k)}/_{y_1}, ^{k}/_{y_2}\} \Vdash x' \, , \, \text{gd}(c) \, , \, x' = s\}$. Let $\theta = \{^{\text{dec}(y_1,y_2)}/_{x'}\}$. We have $\theta \in Sol_{\mathsf{E}}(\mathcal{C} \; ; \; \mathsf{N}[x' \mapsto \mathsf{c}])$ (see Example 2).

4.2 Symbolic Equivalences

We define symbolic static equivalence using a similar encoding as [5]. The tests used to distinguish two frames in the definition of static equivalence are encoded by means of two additional deduction constraints on fresh variables x, y and by the equation $x = y$.

Definition 9 (symbolic static equivalence ($\sim_s$)). *Two closed well-formed symbolic processes are* statically equivalent, *written* $(A_s \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \sim_s (B_s \; ; \; \mathcal{C}_B \; ; \; \mathsf{N})$ *if for some variables x, y such that $\mathsf{N}(\{x,y\}) = \mathsf{n}$, the constraint systems $\mathcal{C}'_A, \mathcal{C}'_B$ have the same basis and $Sol_{\mathsf{E}}(\mathcal{C}'_A \; ; \; \mathsf{N}[x,y \mapsto \mathsf{c}]) = Sol_{\mathsf{E}}(\mathcal{C}'_B \; ; \; \mathsf{N}[x,y \mapsto \mathsf{c}])$ where*

- *$\mathcal{C}'_A = \mathcal{C}_A \cup \{\phi(A_s) \Vdash x \, , \, \phi(A_s) \Vdash y \, , \, x = y\}$, and*
- *$\mathcal{C}'_B = \mathcal{C}_B \cup \{\phi(B_s) \Vdash x \, , \, \phi(B_s) \Vdash y \, , \, x = y\}$.*

Proposition 1 (soundness of $\sim_s$). *Consider two closed and well-formed symbolic processes such that $(A_s \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \sim_s (B_s \; ; \; \mathcal{C}_B \; ; \; \mathsf{N})$. We have that:*

1. *$Sol_{\mathsf{E}}(\mathcal{C}_A \; ; \; \mathsf{N}) = Sol_{\mathsf{E}}(\mathcal{C}_B \; ; \; \mathsf{N})$, and*
2. *for all closed $\theta \in Sol_{\mathsf{E}}(\mathcal{C}_A \; ; \; \mathsf{N})$ we have $\phi(A_s(\theta\sigma_A)^\star) \sim \phi(B_s(\theta\sigma_B)^\star)$, where σ_A (resp. σ_B) is the substitution corresponding to the maximal frame of $\mathcal{C}_A$ (resp. $\mathcal{C}_B$).*

Definition 10 (Symbolic labelled bisimilarity ($\approx_s$)). Symbolic labelled bisimilarity *is the largest symmetric relation $\mathcal{R}$ on closed well-formed symbolic processes with same naming environment, such that $(A_s \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \mathcal{R} (B_s \; ; \; \mathcal{C}_B \; ; \; \mathsf{N})$ implies*

1. *$(A_s \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \sim_s (B_s \; ; \; \mathcal{C}_B \; ; \; \mathsf{N})$*
2. *if $(A_s \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \to_s (A'_s \; ; \; \mathcal{C}'_A \; ; \; \mathsf{N})$ with $Sol_{\mathsf{E}}(\mathcal{C}'_A \; ; \; \mathsf{N}) \neq \emptyset$, then there exists $(B'_s \; ; \; \mathcal{C}'_B \; ; \; \mathsf{N})$ such that $(B_s \; ; \; \mathcal{C}_B \; ; \; \mathsf{N}) \to_s^\star (B'_s \; ; \; \mathcal{C}'_B \; ; \; \mathsf{N})$ and $(A'_s \; ; \; \mathcal{C}'_A \; ; \; \mathsf{N}) \mathcal{R} (B'_s \; ; \; \mathcal{C}'_B \; ; \; \mathsf{N})$;*

3. if $(A_s \; ; \; \mathcal{C}_A \; ; \; \mathsf{N}) \xrightarrow{\alpha_s}_s (A'_s \; ; \; \mathcal{C}'_A \; ; \; \mathsf{N}')$ with $Sol_\mathsf{E}(\mathcal{C}'_A \; ; \; \mathsf{N}') \neq \emptyset$, then there exists $(B'_s \; ; \; \mathcal{C}'_B \; ; \; \mathsf{N}')$ such that $(B_s \; ; \; \mathcal{C}_B \; ; \; \mathsf{N}) \rightarrow^*_s \xrightarrow{\alpha_s}_s \rightarrow^*_s (B'_s \; ; \; \mathcal{C}'_B \; ; \; \mathsf{N}')$, and $(A'_s \; ; \; \mathcal{C}'_A \; ; \; \mathsf{N}') \; \mathcal{R} \; (B'_s \; ; \; \mathcal{C}'_B \; ; \; \mathsf{N}')$.

Baudet [6] presents a (co-NP) decision procedure to check $\sim_s$ (condition 1) for constraint systems without disequality constraints and subterm convergent[4] equational theories. This includes among others the well-known Dolev-Yao theory used to model symmetric (resp. asymmetric) encryption with composed keys, signatures and pairing. Building on this existing work, we obtain a procedure to decide our symbolic bisimulation for the fragment of the finite applied pi calculus without else branches in the conditional.

Theorem 1 (Main result). *Let A and B be two closed, nv-distinct extended processes and N be a naming environment compatible with $A{\downarrow}$, $B{\downarrow}$. We have that*

$$(A{\downarrow} \; ; \; \emptyset \; ; \; \mathsf{N}) \approx_s (B{\downarrow} \; ; \; \emptyset \; ; \; \mathsf{N}) \; \text{implies} \; A \approx B.$$

Note that limiting the theorem to nv-distinct processes is not a real restriction. If we want to prove that $A \approx B$, we can construct by α-conversion two nv-distinct processes A', B' such that $A' \equiv A$ and $B' \equiv B$. Showing $A' \approx B'$ implies that $A \approx B$, since $\approx$ is closed under structural equivalence.

Theorem 1 is proved by using our intermediate semantics. We define labelled bisimilarity on intermediate extended processes, and show it to coincide with labelled bisimilarity in applied pi. Soundness and completeness of the symbolic semantics is shown with respect to the intermediate semantics. This allows to obtain soundness of the symbolic bisimulation. All the details are given in [13].

5 Discussion, Related and Future Work

5.1 Sources of Incompleteness

Our techniques suffer from the same sources of incompleteness as the ones described for the spi calculus in [10]. In a symbolic bisimulation the instantiation of input variables is postponed until the point at which they are actually used, leading to a finer relation. We illustrate this point on an example, similar to one given in [10].

Example 5. Consider the two following processes:

$$P_1 = \nu c_1.\mathsf{in}(c_2, x).(\mathsf{out}(c_1, b) \mid \mathsf{in}(c_1, y) \mid \text{if } x = a \text{ then } \mathsf{in}(c_1, z).\mathsf{out}(c_2, a))$$
$$Q_1 = \nu c_1.\mathsf{in}(c_2, x).(\mathsf{out}(c_1, b) \mid \mathsf{in}(c_1, y) \mid \mathsf{in}(c_1, z).\text{if } x = a \text{ then } \mathsf{out}(c_2, a))$$

We have that $P_1 \approx Q_1$ whereas $(P_1 \; ; \; \emptyset \; ; \; \mathsf{N}) \not\approx_s (Q_1 \; ; \; \emptyset \; ; \; \mathsf{N})$ for any compatible naming environment N. Depending on the value of the input, i.e. if x is equal to a or not, P_1 and Q_1 know if the test $x = a$ will succeed or not. However, on the symbolic side, the instantiation of x is postponed until the moment where x is really used, i.e. until the moment of the test itself, when it is too late to choose the right branch.

---

[4] An equational theory induced by a finite set of equations $M = N$ where N is a subterm of M and such that the associated rewriting system is convergent.

5.2 Related Work

A pioneering work has been done by Henessy and Lin [14] for value-passing CCS. However, the result which is most closely related to ours is by Borgström *et al.* [10]: they define a symbolic bisimulation for the spi calculus with the same sources of incompleteness as we have. However, our treatment of general equational theories is non trivial as illustrated by the problems implied for structural equivalence.

For many important equational theories, static equivalence has been shown to be decidable in [1]. More interestingly, some works have also been done to automate observational equivalence. The ProVerif tool [7] automates observational equivalence checking for the applied pi calculus (with process replication), but since the problem is undecidable the technique it uses is necessarily incomplete. The tool aims at proving a finer equivalence relation and relies on easily matching up the execution paths of the two processes [8]. In his thesis, Baudet [6] presents a decision procedure for a similar equivalence, called diff-equivalence, in a simplified process calculus. Examples where this equivalence relation is too fine occur when proving the observational equivalence required to show vote-privacy [15, 12]. Although our symbolic bisimulation is not complete, we are able to conclude on examples where ProVerif fails. For instance, ProVerif is unable to prove that the processes $\mathsf{out}(c, a) \mid \mathsf{out}(c, b)$ and $\mathsf{out}(c, b) \mid \mathsf{out}(c, a)$ are bisimilar whereas of course we are able to deal with such examples. A more interesting example, for which our symbolic semantics plays an important role is as follows.

Example 6. Consider the following two processes
$$P = \nu c_1.(\mathsf{in}(c_2, x).\mathsf{out}(c_1, x).\mathsf{out}(c_2, a) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, y))$$
$$Q = \nu c_1.(\mathsf{in}(c_2, x).\mathsf{out}(c_1, x).\mathsf{out}(c_2, x) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, a))$$

These two processes are labelled bisimilar and our symbolic labelled bisimilation is complete enough to prove this. In particular, let $P' = \nu c_1.(\mathsf{out}(c_1, x').\mathsf{out}(c_2, a) \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, y))$ and $Q' = \nu c_1.(\mathsf{out}(c_1, x').\mathsf{out}(c_2, x') \mid \mathsf{in}(c_1, y).\mathsf{out}(c_2, a))$. The relation $\mathcal{R}$, that witnesses the symbolic bisimulation, includes

$$(P ; \emptyset ; \mathsf{N}) \, \mathcal{R} \, (Q ; \emptyset ; \mathsf{N})$$

$$(P' ; \{\nu c_1.0 \Vdash x' , \mathsf{gd}(c_2)\} ; \mathsf{N}') \, \mathcal{R} \, (Q' ; \{\nu c_1.0 \Vdash x' , \mathsf{gd}(c_2)\} ; \mathsf{N}')$$

$$\begin{array}{cc} (\nu c_1.(\mathsf{out}(c_2, a) \mid \mathsf{out}(c_2, x'))) ; & (\nu c_1.(\mathsf{out}(c_2, x') \mid \mathsf{out}(c_2, a)) ; \\ \{\nu c_1.0 \Vdash x' , \mathsf{gd}(c_2) , \mathsf{gd}(c_1)\} ; \mathsf{N}') & \mathcal{R} \quad \{\nu c_1.0 \Vdash x' , \mathsf{gd}(c_2) , \mathsf{gd}(c_1)\} ; \mathsf{N}') \end{array}$$

The technique used in ProVerif will generally fail in the case where the two processes take different branches at some point. This is the case in Example 6: after a synchronisation (modelled by a communication on the private channel c_1) between the two parallel components of process P (resp. Q), the output action of the left component of P matches the output action of the right component of Q. This example is actually inspired by the problems we encountered when we tried to verify privacy in electronic voting protocols using ProVerif. In order to establish privacy of an electronic voting protocol (according to the definition given in [15]), we need a bisimulation relation, as the one described in this paper, which is coarse enough to allow processes to differ on their structure. We think that our symbolic bisimulation is complete enough to deal with many other interesting cases since other privacy and anonymity properties are facing the same difficulty.

5.3 Future Work

The obvious next step is to study the equivalence of solutions for constraint systems under different equational theories. Promising results have already been shown in [5] for a significant class of equational theories but for constraint systems that do not have disequalities. These results readily apply for deciding our symbolic bisimulation on the fragment without else branches in conditionals. We plan to implement an automated tool for checking observational equivalence. In particular we aim at automating proofs arising in case studies of electronic voting protocols which currently rely on hand proofs [12].

References

1. M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 387(1-2):2–32, 2006.
2. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages*, pages 104–115, 2001.
3. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th Conference on Computer and Communications Security*, pages 36–47. ACM, 1997.
4. R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290:695–740, 2002.
5. M. Baudet. Deciding security of protocols against off-line guessing attacks. In *Proc. 12th Conference on Computer and Communications Security*, pages 16–25. ACM Press, 2005.
6. M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, LSV, ENS Cachan, France, Jan. 2007.
7. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *Proc. 14th Computer Security Foundations Workshop*, pages 82–96. IEEE Comp. Soc. Press, 2001.
8. B. Blanchet, M. Abadi, and C. Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Proc. 20th Symposium on Logic in Computer Science*, pages 331–340. IEEE Comp. Soc. Press, 2005.
9. M. Boreale and R. D. Nicola. A symbolic semantics for the pi-calculus. *Information and Computation*, 126(1):34–52, 1996.
10. J. Borgström, S. Briais, and U. Nestmann. Symbolic bisimulation in the spi calculus. In *Proc. 15th Int. Conference on Concurrency Theory*, volume 3170 of *LNCS*. Springer, 2004.
11. S. Delaune and F. Jacquemard. A decision procedure for the verification of security protocols with explicit destructors. In *Proc. 11th ACM Conference on Computer and Communications Security (CCS'04)*, pages 278–287. ACM Press, 2004.
12. S. Delaune, S. Kremer, and M. D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proc. 19th Computer Security Foundations Workshop*, pages 28–39. IEEE Comp. Soc. Press, 2006.
13. S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. Research Report LSV-07-14, LSV, ENS Cachan, France, Apr. 2007. 47 pages.
14. M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.
15. S. Kremer and M. D. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *Proc. 14th European Symposium on Programming*, volume 3444 of *LNCS*, pages 186–200. Springer, 2005.
16. J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th Conference on Computer and Communications Security*, pages 166–175, 2001.