Symbolic bisimulation for the applied pi calculus (extended abstract) ⋆

Stéphanie Delaune [a], Steve Kremer [b], Mark Ryan [c]

[a] *LORIA, CNRS & INRIA, Nancy, France*

[b] *LSV, CNRS & ENS de Cachan & INRIA, France*

[c] *School of Computer Science, University of Birmingham, UK*

Abstract

Recently, we have proposed in [10] a symbolic semantics together with a sound symbolic labelled bisimulation relation for the finite applied pi calculus. By treating inputs symbolically, our semantics avoids potentially infinite branching of execution trees due to inputs from the environment. This work 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. We present some of the difficulties we have encountered in the design of the symbolic semantics.

1 Introduction

The *applied pi calculus* [1] 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. These messages are generated by a term algebra and equality is treated modulo an *equational theory*. For instance the equation $\mathsf{dec}(\mathsf{enc}(x,y),y) = x$ models the fact that encryption and decryption with the same key cancel out. *Active substitutions* model the availability of data to the environment.

As an example consider the following reduction step:

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

⋆ This work has been partly supported by the RNTL project POSÉ, EPSRC projects EP/E029833, *Verifying Properties in Electronic Voting Protocols* and EP/E040829/1, *Verifying anonymity and privacy properties of security protocols* and the ARTIST2 Network of Excellence. We thank Magnus Johansson and Björn Victor for interesting discussions.

The process outputs on the channel c a secret name s encrypted with the key k. The active substitution $\{^{\mathsf{enc}(s,k)}/_{x}\}$ gives the environment the ability to access the term $\mathsf{enc}(s,k)$ via the fresh variable x without revealing either s, or k. The applied pi calculus generalizes the *spi calculus* [2] which only allows a fixed set of primitives built-in (symmetric and public-key encryption), while the applied pi calculus allows one to define these and other less usual primitives by means of an equational theory.

One of the difficulties in automating proofs in such a 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 variables) at the time the input was performed.

Reachability properties and also existence of off-line guessing attacks can be verified by solving the constraint systems arising from symbolic executions (e.g. [3,4]). *Observational equivalence properties* express the inability of the attacker to distinguish between two processes no matter how it interacts with them. These properties have been found useful for modelling anonymity and privacy properties (*e.g.* [9]), as well as other requirements [5,2]. Symbolic methods have already been used in the case of observational equivalence or bisimulation properties in classical process algebras (e.g. [11,6]). In particular, Borgström *et al.* [7] have defined a sound symbolic bisimulation for the spi calculus.

To show that a symbolic bisimulation implies the concrete one, we generally need to prove that the symbolic semantics is both sound and complete. Defining a symbolic semantics for the applied pi calculus has shown to be surprisingly difficult technically. In this paper, rather than describing our symbolic semantics we present several difficulties that we encountered and motivate some of our design choices. A complete description of the semantics and a sound symbolic bisimulation are available in [10].

2 Towards a symbolic semantics

In this section, we demonstrate some of the difficulties in defining a symbolic semantics for the applied pi calculus that is both sound and complete. Details about syntax and semantics of the original applied pi calculus are not given here. We will just recall some notions when we need them to explain the difficulties we have encountered.

2.1 Structural equivalence

The semantics of the applied-pi calculus is defined by structural rules defining three relations: structural equivalence ($\equiv$), internal ($\rightarrow$) and labelled ($\xrightarrow{\alpha}$) reduction. The last two relations are closed under structural equivalence. Hence, the natural first step seems to define a 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 $\exists\, Q_s$ such that $P_s \equiv_s Q_s$ and $Q_s\sigma = Q$.

However, it seems difficult to achieve this. Before explaining the difficulty, we introduce structural equivalence more formally. *Structural equivalence* is the smallest equivalence relation $\equiv$ on extended processes that is closed under α-conversion on names and variables, application of evaluation contexts (an extended process with a hole), and some other standard rules such as associativity and 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.

$$\nu x.\{^M/_x\} \equiv 0 \qquad \{^M/_x\} \mid A \equiv \{^M/_x\} \mid A\{^M/_x\}$$

$$\{^M/_x\} \;\equiv\; \{^N/_x\} \quad \text{if } M =_{\mathsf{E}} N$$

Consider the process $P = \mathrm{in}(c,x).\mathrm{in}(c,y).\mathrm{out}(c,f(x)).\mathrm{out}(c,g(y))$ which can be reduced to $P' = \mathrm{out}(c,f(M_1)).\mathrm{out}(c,g(M_2))$ where M_1 and M_2 are two arbitrary terms provided by the environment. When $f(M_1) =_{\mathsf{E}} g(M_2)$, *i.e.* $f(M_1)$ and $g(M_2)$ are equal modulo the equational theory, we have that $P' \equiv \nu z.(\mathrm{out}(c,z).\mathrm{out}(c,z) \mid \{^{f(M_1)}/_z\})$, but this structural equivalence does not hold whenever $f(M_1) \neq_{\mathsf{E}} g(M_2)$.

The symbolic process $P'_s = \mathrm{out}(c,f(x)).\mathrm{out}(c,g(y))$ has to represent the different cases where $f(x)$ and $g(y)$ are equal or not. Hence, the question of whether the structural equivalence $P'_s \equiv_s \nu z.(\mathrm{out}(c,z).\mathrm{out}(c,z) \mid \{^{f(x)}/_z\})$ is valid cannot be decided, as it depends on the concrete values of x and y. Therefore, symbolic structural equivalence cannot be both sound and complete. This seems to be an inherent problem and it propagates to internal and labelled reduction, since they are closed under structural equivalence.

2.2 Intermediate semantics

The absence of sound and complete symbolic structural equivalence, mentioned above, significantly complicates the proof of our main result given in [10]. 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. They only have name restriction (no variable restriction) and all restrictions have to be in front of the process. They

have to be *name and variable distinct* meaning that we have to use different names (resp. variables) to represent free and bound names (resp. variables), and also that any name (resp. variable) is at most bound once. Moreover, we require an intermediate process to be *applied* meaning that each variable in the domain of the process occurs only once. For instance, the process $A\downarrow = \nu b.\mathrm{in}(c, y).\mathrm{out}(a, f(b))$ is the intermediate process associated to the process $A = \nu x.(\mathrm{in}(c, y).\nu b.\mathrm{out}(a, x) \mid \{^{f(b)}/_{x}\})$.

Then, we equip these intermediate processes with a labelled bisimulation that coincides with the original one. The intermediate semantics differs from the semantics of the original applied pi calculus. In particular, we do not consider the three rules, given in Section 2.1, related to active substitution for structural equivalence and we do not substitute equals for equals in structural equivalence, but only in a controlled way in certain reduction rules. Thus, we avoid the problem mentioned in the previous section.

Finally we present a symbolic semantics which is both sound and complete with respect to the intermediate one and give a sound symbolic bisimulation. Another way to tackle this problem may be to define a symbolic structural equivalence which is not complete and to prove directly completeness of the symbolic reduction and labeled transition without having completeness for structural equivalence, but we have not been able to carry this through to completion and therefore we did not take that approach.

2.3 Separate constraint systems and explicit renaming

To keep track of the constraints on symbolic variables we have chosen to 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 [4] and decide 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 [8]. 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. Thus, we have to deal with explicit renaming when necessary. This adds some additional complexity, but the benefit of keeping the constraints separate seems to be appealing in view of an implementation.

3 Conclusion and future work

Designing a symbolic semantics for the applied pi calculus has revealed to be much more difficult technically than expected. In this paper we have

highlighted some difficulties we encountered and motivated some of our design choices. The result of these design choices is given in [10]. The obvious next step is to study solving constraint systems in the presence of disequalities. This would enable us to decide our bisimulation even in the case of else branches.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages*, pages 104–115. ACM Press, 2001.

[2] 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.

[3] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290:695–740, 2002.

[4] M. Baudet. *Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*. Thèse de doctorat, Laboratoire Spécification et Vérification, ENS Cachan, France, 2007.

[5] 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.

[6] M. Boreale and R. D. Nicola. A symbolic semantics for the pi-calculus. *Information and Computation*, 126(1):34–52, 1996.

[7] 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*, pages 161–176. Springer, 2004.

[8] 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.

[9] 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.

[10] S. Delaune, S. Kremer, and M. D. Ryan. Symbolic bisimulation for the applied pi calculus. Research Report LSV-07-14, Laboratoire Spécification et Vérification, ENS Cachan, France, Apr. 2007. 47 pages.

[11] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.