A Complete Symbolic Bisimilarity for an Extended Spi Calculus

Johannes Borgström

A Complete Symbolic Bisimilarity for an Extended Spi Calculus

Johannes Borgström

[SecCo 08]
The Bottom Line

1. **Constructor-Destructor expression languages** permit a smooth extension of classical spi.

2. **Environment decompositions** yield a complete symbolic bisimilarity.

3. **Infinite decompositions** may be needed unlike in the pi calculus.
A Complete Symbolic Bisimilarity for an Extended Spi Calculus

Johannes Borgström

Overview

1 Language
   Constructor-Destructor Languages

2 Correctness
   Symbolic Bisimulation

3 Practicality
   Infinite Branching
A Complete Symbolic Bisimilarity for an Extended Spi Calculus

Johannes Borgström

Introduction
Language Constructor-Destructor Languages
Correctness Symbolic Bisimulation
Practicality Infinite Branching
Conclusion

Spi Calculus

\[
P, Q ::= 0 \mid F(x).P \mid \overline{F} \langle G \rangle .P \mid ! F(x).P \mid \phi P \\
    \mid P + P \mid P \parallel P \mid (\nu a) P
\]

\[
\phi, \psi ::= tt \mid [G = F] \mid [F : N] \mid \phi \land \phi \mid \neg \phi
\]

\[
F, G ::= a \mid x \mid E_G(F) \mid D_G(F)
\]

Parameter: Expression language $F, G$ with evaluation $e$. Example: $e$ induced by $D_y(E_y(x)) \rightarrow x$.

\[
\begin{align*}
(\text{INP}) & \quad e(G) = a \quad G(x).P \xrightarrow{a(x)} P \\
(\text{OUT}) & \quad e(G) = a \quad e(F) = M \quad \overline{G} \langle F \rangle .P \xrightarrow{\overline{a}(M)} P \\
(\text{COM-L}) & \quad P \xrightarrow{a(x)} P' \quad Q \xrightarrow{(\nu \tilde{b}) \overline{a}(M)} Q' \\
& \quad P \parallel Q \xrightarrow{\tau} (\nu \tilde{b}) \left( P' \left\{ M \over x \right\} \parallel Q' \right) \quad \text{if} \quad \{ \tilde{b} \} \cap \text{fn}(P) = \emptyset
\end{align*}
\]
Expression Languages

- Symmetric key encryption is insufficient.
- $\mathcal{E}$ term algebra over names modulo congruence $\equiv$.
- Often: Congruence derived from rewrite rules $D_y(E_y(x)) \rightarrow x$.
- Knowledge environment $\sigma : V \rightarrow \mathcal{E}$ knows $F$ if there is $G$ with $n(G) \cap n(\sigma) = \emptyset$ and $F \equiv G_\sigma$.

- Two environments $\sigma, \rho$ are statically equivalent if for all $F, G$ with $n(F, G) \cap n(\sigma, \rho) = \emptyset$ we have $F_\sigma \equiv G_\sigma$ iff $F_\rho \equiv G_\rho$.

**Theorem (EXPRESS’06)**

*There exist $\mathcal{E}, \equiv$ where knowledge is decidable but static equivalence is not.*
Constructors & Destructors

- Split function symbols into constructors and destructors.
- Messages $M \in \mathcal{M}$ only contain constructors.
- One rule per destructor: $g(f(\tilde{M}), \tilde{N}) \rightarrow M_j$
- Evaluation: $e(F) := F \downarrow$ whenever $G \downarrow \in \mathcal{M}$ for all subterms $G$ of $F$.
- Competitors
  - Subterm-convergent: $G \prec F$ in every rule $F \rightarrow G$.
  - Data-term: Every rule is of the form $g(\tilde{M}) \rightarrow x$.

**Result:** Smooth extension of our earlier work; Closed form for static equivalence. (See my PhD thesis)
A Complete Symbolic Bisimilarity for an Extended Spi Calculus

Johannes Borgström

Introduction

Language
Constructor-Destructor Languages
Correctness
Symbolic Bisimulation
Practicality
Infinite Branching
Conclusion

Synthesis, Analysis and Irreducibles

| SYN-K | SYN-f |
|-------|-------|
| \( M \in \kappa \) | \( \{ \tilde{M} \} \subseteq S(\kappa) \) |
| \( M \in S(\kappa) \) | \( f(\tilde{M}) \in S(\kappa) \) |

| ANA-K | ANA-SK | ANA-S |
|-------|--------|-------|
| \( M \in \kappa \) | \( M \in A(\kappa) \) | \( \{ \tilde{M} \} \subseteq SA(\kappa) \) |
| \( M \in A(\kappa) \) | \( M \in SA(\kappa) \) | \( f(\tilde{M}) \in SA(\kappa) \) |

| ANA-g |
|-------|
| \( f(\tilde{M}) \in A(\kappa) \) | \( \{ \tilde{N} \} \subseteq SA(\kappa) \) | if \( g(f(\tilde{M}), \tilde{N}) \rightarrow^H M \) |
| \( M \in A(\kappa) \) |

\( \mathcal{I}(\kappa) := A(\kappa) \setminus S^+(A(\kappa)) \) where
\( S^+(\kappa) := \{ f(\tilde{M}) \mid \{ \tilde{M} \} \subseteq S(\kappa) \} \).
Correctness Properties

- Based on the concept of indistinguishability (written $\equiv$). The same attacks should yield the same behavior.

- Secrecy: $\forall M, N \quad P \{ M/x \} = P \{ N/x \}$

- Authenticity: $\forall M \quad P \{ M/x \} = P_{spec} \{ M/x \}$

- The (Dolev-Yao) attacker can
  - Intercept messages on public channels
  - Inject messages into public channels
  - Perform cryptographic operations, including key generation

The attacker corresponds to an arbitrary process.
A Complete Symbolic Bisimilarity for an Extended Spi Calculus

Johannes Borgström

Introduction

Language

Constructor-Destructor Languages

Correctness

Symbolic Bisimulation

Practicality

Infinite Branching

Conclusion

Strong Barbed Equivalence

Definition

- \( P \downarrow_b (P \text{ exhibits barb } b) \) if \( \exists x, P' \text{ s.t. } P \xrightarrow{b(x)} P' \).

- Barbed bisimilarity \( \simeq \) is the greatest \( \xrightarrow{\tau} \)-bisimulation such that if \( P \simeq Q \) and \( b \in \mathcal{N} \) then \( P \downarrow_b \) iff \( Q \downarrow_b \).

- Two processes \( P \) and \( Q \) are barbed equivalent, written \( P \simeq Q \), if for all processes \( R \), \( (P \mid R) \simeq (Q \mid R) \).

- \( R \) above models the intruder, performing tests \( \downarrow_b \).

- Our target equivalence.

- Three issues for mechanization.
The Observer

Problem: Quantification over observer contexts is hard [AG99]

Idea: Consider the LTS, keep track of adversary knowledge and message correspondence with environments. [AG98, BDP02]

- Environments are usually not needed.
- $e \vdash P \sim Q$: $P$ and $Q$ are bisimilar under $e$.
- $e$ is the “knowledge of the attacker”.

- Our choice: Hedges $h \subset \mathcal{E} \times \mathcal{E}$
- Synthesis, Analysis, Irreducibles defined \textit{mutatis mutandis}. 
Branching on Input

**Problem:** Infinite branching on process input.

**Idea:** Symbolic semantics, input a variable and successively add constraints.

- Symbolic semantics exist for pi and value-passing CCS. Environments are needed in both cases.
- This work: A general symbolic semantics for spi, and a corresponding bisimilarity.
Constraint Systems

Problem: Infinitely many potential solutions to constraints.

Idea: Find a finite set of representative solutions.

- Always possible in the pi calculus [Bor95].
- In NP for subterm-convergent message languages and positive guards [Bau07].
- Not treated here.
Symbolic Operational Semantics

- The Spi calculus only permits communication of *proper messages* on *channels that are names*.
- Action prefixes introduce constraints.

\[
F(x).P \xrightarrow{e(F)(x)} P \\
[F : N]
\]

\[
\overline{F}\langle G \rangle.P \xrightarrow{e(F)\langle e(G) \rangle} P \\
[G : M] \land [F : N]
\]

- Abstract evaluation prevents unneeded scope extrusion.
  \[\text{if } e(F) \in N \text{ and } e(G) \in M\]
  \[g(f(\tilde{x}), \tilde{y}) \rightarrow_a x_i \text{ where } g(f(\tilde{M}), \tilde{N}) \rightarrow M_i; \quad e_a(F) = F \downarrow_a.\]

**Lemma**

*For all* \(F, \sigma, M, n, \) *if* \(e(F\sigma) = M\) *then* \(e(e_a(F)\sigma) = M.\)
Symbolic Operational Semantics

- How to handle constraints with restricted names?
- We add (top-level) restriction to the transition constraints.

\[
P \xrightarrow{\mu} P' \quad \frac{(\nu \tilde{c}) \phi}{(\nu a) P \xrightarrow{\mu} (\nu a) P'} \quad \text{if } n(\mu, \tilde{c}) \not\ni a \in n(\phi)
\]

- Input variables may be used in channel expressions.
- Solution: Communication also introduces constraints.

\[
P \xrightarrow{F(x)} P' \quad Q \xrightarrow{(\nu \tilde{b}) F' \langle G \rangle} Q' \\
\frac{(\nu \tilde{c}) \phi}{(\nu \tilde{d}) \psi} \\
P \mid Q \xrightarrow{\tau} (\nu \tilde{b}) (P' \{ G / x \} \mid Q') \quad \text{if } \ldots
\]

Lemma

\[P \xrightarrow{\mu} P' \iff \exists \phi, \tilde{c} \text{ such that } P \xrightarrow{(\nu \tilde{c}) \phi} P' \text{ and } [\phi].\]
Symbolic Environments

- Environments se track knowledge and constraints.
- se := (th, tv, ((νC) φ, (νD) ψ)), where
  - th : E × E → N and tv : V → N give the knowledge available to instantiate input variables;
  - φ and ψ are accumulated transition constraints; C, D are fresh names.
- σ, ρ : dom(tv) → M instantiate se if they
  - are generatable from the knowledge; and
  - satisfy the accumulated constraints.

We then get an instance C^B_{σ,ρ}(th) of se.

- se is consistent if all instances of se are consistent and φ and ψ are simultaneously satisfied.

- \{se_i\}_{i ∈ I} is a decomposition of se if every instantiation of se is instantiation of some se_i and all se_i are consistent.
Symbolic Bisimulation

- An environment-sensitive relation
  \[ se \vdash P \sim_s Q. \]

- When to simulate a transition \( P \xrightarrow{\mu_s \phi'} P' \):
  - Detectable – additional knowledge constraint.
  - Possible – satisfiability of constraints.

- Find a consistent decomposition of the resulting constraint \( \phi \wedge \phi' \).

- For each environment in the decomposition, find a matching simulating transition
  \( Q \xrightarrow{\mu'_s \psi'} Q' \) such that \( \psi' \) is true in all instantiations.
Symbolic Bisimulation

• Assume that \((th, tw, (\nu C \phi, (\nu D) \psi)) \vdash P \equiv Q\) and
  \[ P \xrightarrow{(\nu \tilde{c}) F(x)} P' \] with \(\{\tilde{c}, \tilde{c}'\} \cap n_1(se) = \emptyset\) and \(x \notin \text{dom}(tv)\).

• If there are \(\sigma, \rho, B, y\) with \(se \vdash \sigma \leftrightarrow_B \rho\),
  \[ \{\tilde{c}\} \cup \text{fn}(P, Q)) \cap B = \emptyset, \llbracket \phi' \sigma \rrbracket \]
  \[ \text{e}(F\sigma) \in \pi_1(C_{\sigma, \rho}(th)) \text{ and } y \notin (\text{dom}(tv) \cup \{x\}) \]
  we let \(se := (th, tv', ((\nu C \cup \{\tilde{c}, \tilde{c}'\}) \phi \land \phi' \land [y = F], (\nu D) \psi))\)
  where \(tv' = tv \cup \{x \mapsto t+1, y \mapsto t+1\}\).

• then there is a consistent decomposition \(\{se_i\}_{i \in I}\) of \(se\) such that for each \(i \in I\),

• there are \(\tilde{e}, \tilde{e}'\), \(\psi', Q', F'\) with \(Q \xrightarrow{(\nu \tilde{e}) F'(x)} Q'\),
  \(\{\tilde{e}, \tilde{e}'\} \cap (n_2(se) \cup B) = \emptyset\),
  \(se'_i \vdash tt \leftrightarrow \psi' \land [y = F']\) and \(se'_i \vdash P' \equiv Q'\) where
  \(se'_i = (th, tv', ((\nu C \cup \{\tilde{c}, \tilde{c}'\}) \phi_i, (\nu D \cup \{\tilde{e}, \tilde{e}'\}) \psi_i))\).
Soundness

Theorem

If $se \vdash P \sim_s Q$ then $h \vdash P\sigma \sim_h Q\rho$ for any instantiation $(\sigma, \rho)$ with instance $h$.

- Any transition of an instantiation that must be simulated has a corresponding transition that must be simulated symbolically.
- Any symbolically simulated transition can always be concretely simulated (given a detected concrete transition) preserving the “instantiation” relationship.

\[
se \vdash P, Q \xrightarrow{\phi} se' \vdash P', Q'
\]

\[
h \vdash P\sigma, Q\rho \xrightarrow{\mu?} h' \vdash P'\sigma', Q'\rho'
\]
Completeness

Theorem

If \( h \vdash P\sigma \sim_h Q\rho \) for all instantiations \((\sigma, \rho)\) of \( se \), then \( se \vdash P \sim_s Q \).

- For every transition of \( P \), decompose into all corresponding instantiations and simulate according to \( \sim_h \).
Finding a Decomposition

• Can one always compute a decomposition?
  • NO: using replication, we can simulate Turing machines.

• Is there always a finite decomposition?
  • NO (dependent on the logic at the level of environments).
Example

\[ \Sigma = (\{s, p\}, \{s \mapsto 1, p \mapsto 1\}) \] with the rule \( p(s(x)) \rightarrow x \).

- \( P \) either diverges or behaves as \( P' \), where
  - \( P' \) decrements its input to 0 (a name) and signals success.

- \( Q \) behaves as either \( Q_1 \) or \( Q_2 \):
  - \( Q_1 \) diverges if the input is even, otherwise behaves as \( P' \).
  - \( Q_2 \) diverges if the input is odd, otherwise behaves as \( P' \).

\[
P = a(x).\Omega + a(x). (\nu c) (P'(x) | !c(y). P'(y))
\]

\[
P'(x) = \bar{x} \langle a \rangle + \bar{c} \langle p(x) \rangle
\]

\[
\Omega = (\nu c) (\bar{c} \langle c \rangle | !c(z). \bar{c} \langle c \rangle)
\]

\[
Q = (\nu c) ((a(x).Q_1(x) + a(x).Q_2(x)) | !c(y). Q_2(y))
\]

\[
Q_1(x) = [x : \mathcal{N}] \Omega + \bar{c} \langle p(x) \rangle
\]

\[
Q_2(x) = \bar{x} \langle a \rangle + (\nu d) (\bar{d} \langle p(x) \rangle | d(z). Q_1(z))
\]
Leftovers - Constraints

- Possibility and detectability
  - Solved for negation-free formulas in a large class of msg languages [BB05].

- Symbolic consistency
  - Solved for negation-free formulas and subterm-convergent msg languages. [Bau07]
  - Would Constructor-Destructor languages help with negation?

- Environment decompositions (needed for completeness)
  - Finding a finite decomposition may be impossible/undecidable.
  - Possible to defer the choice of simulating transition?
The Bottom Line

1. Constructor-Destructor expression languages permit a smooth extension of classical spi.

2. Environment decompositions yield a complete symbolic bisimilarity.

3. Infinite decompositions may be needed unlike in the pi calculus.

Questions?
Bibliography

Martín Abadi and Andrew D. Gordon.  
A bisimulation method for cryptographic protocols.  
*Nordic Journal of Computing*, 5(4):267–303, 1998.

Martín Abadi and Andrew D. Gordon.  
A calculus for cryptographic protocols: The Spi calculus.  
*Journal of Information and Computation*, 148(1):1–70, 1999.

Mathieu Baudet.  
*Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*.  
PhD thesis, École Normale Supérieure de Cachan, 2007.

Michele Boreale and Maria Grazia Buscemi.  
A method for symbolic analysis of security protocols.  
*Theoretical Computer Science*, 338(1-3):393–425, 2005.

Michele Boreale, Rocco De Nicola, and Rosario Pugliese.  
Proof techniques for cryptographic processes.  
*SIAM Journal on Computing*, 31(3):947–986, 2002.

Mathieu Baudet.  
*Sécurité des protocoles cryptographiques : aspects logiques et calculatoires*.  
PhD thesis, École Normale Supérieure de Cachan, 2007.

Michele Boreale and Maria Grazia Buscemi.  
A method for symbolic analysis of security protocols.  
*Theoretical Computer Science*, 338(1-3):393–425, 2005.

Michele Boreale, Rocco De Nicola, and Rosario Pugliese.  
Proof techniques for cryptographic processes.  
*SIAM Journal on Computing*, 31(3):947–986, 2002.