From a Concurrency Course to Automatic Verification of Process Equivalences

Bruno Blanchet

INRIA, École Normale Supérieure, CNRS
Bruno.Blanchet@ens.fr

February 2011

1995-1996: DEA Sémantique, Preuves et Programmation

Concurrence 2

Communication par canal

et

Pi-calcul

Jean-Jacques.Levy@inria.fr
INRIA – Rocquencourt

tel: +33-1-39-63-56-89

http://pauillac.inria.fr/∼levy

1

Plan

1. Mémoire partagée
2. Réseaux de Petri
3. CSP, CCS, Meije, ACP
4. Pi-calcul (définitions)
5. Pi-calcul (exemples)
6. Pi-calcul polyadique
7. Pi-calcul asynchrone

2

1995-1996: DEA Sémantique, Preuves et Programmation

Concurrence 3

Pi-calcul

Pict

Pi-calcul d'ordre supérieur

Jean-Jacques.Levy@inria.fr
INRIA – Rocquencourt

tel: +33-1-39-63-56-89
http://pauillac.inria.fr/~levy

Plan

1. Codage de l'arithmétique
2. Codage λ-calcul
3. Codage des structures de données (listes, arbres)
4. Typage des canaux
5. Pi-calcul asynchrone
6. Machines, Expériences, Equivalences

1995-1996: DEA Sémantique, Preuves et Programmation

Concurrence 4

Bisimulations

Pict

Pi-calcul d'ordre supérieur

Jean-Jacques.Levy@inria.fr
INRIA – Rocquencourt

tel: +33-1-39-63-56-89
http://pauillac.inria.fr/~levy

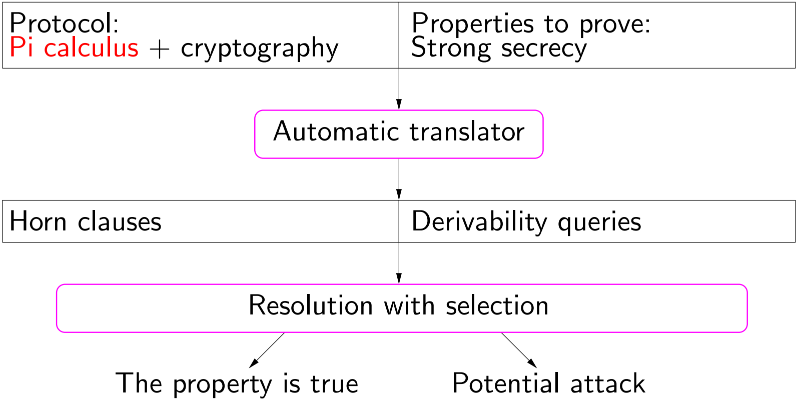
1

Plan

1. Machines, Expériences, Equivalences
2. Pi-calcul avec abstractions
3. Pi-calcul d'ordre supérieur

2

2001-2002: ProVerif: automatic security protocol verifier



(also owes much to Martín Abadi)

2005: Automatic verification of observational equivalence (joint work with Martín Abadi and Cédric Fournet)

- ProVerif initially verified only properties on behaviors (traces) of protocols (secrecy of keys, correspondences).
- Many important properties can be formalized as process equivalences, not as properties on behaviors:
  - secrecy of a boolean x in $P(x)$: $P(\mathrm{true}) \approx P(\mathrm{false})$
  - the process P implements an ideal specification Q: $P \approx Q$

  Equivalences are usually proved by difficult, long manual proofs.

  Much research on this topic, using in particular bisimulation techniques (e.g., Boreale et al).

Equivalences as properties of behaviors (1)

Goal: extend tools designed for proving properties of behaviors (here ProVerif) to the proof of process equivalences.

- We focus on equivalences between processes that differ only by the terms they contain, e.g., $P(\mathrm{true}) \approx P(\mathrm{false})$.

  Many interesting equivalences fall into this category.

- Biprocesses represent pairs of processes that differ only by the terms they contain.

  $P(\mathrm{true})$ and $P(\mathrm{false})$ are variants of a biprocess $P(\mathsf{diff}[\mathrm{true}, \mathrm{false}])$.

  The variants give a different interpretation to $\mathsf{diff}[\mathrm{true}, \mathrm{false}]$, true for the first variant, false for the second one.

Equivalences as properties of behaviors (2)

- We introduce a new operational semantics for biprocesses:

  A biprocess reduces when both variants reduce in the same way and after reduction, they still differ only by terms (so can be written using diff).

- We establish $P(\mathrm{true}) \approx P(\mathrm{false})$ by reasoning on behaviors of $P(\mathsf{diff}[\mathrm{true}, \mathrm{false}])$:

  If, for all reachable configurations, both variants reduce in the same way, then we have equivalence.

(extends to cryptography an idea by Pottier and Simonet)

The process calculus

Extension of the pi-calculus with function symbols for cryptographic primitives.

$M, N ::=$	terms
$\quad x, y, z$	variable
$\quad a, b, c, k, s$	name
$\quad f(M_1, \ldots, M_n)$	constructor application
$D ::=$	term evaluations
$\quad M$	term
$\quad \mathrm{eval}\ h(D_1, \ldots, D_n)$	function evaluation
$P, Q, R ::=$	processes
$\quad M(x).P$	input
$\quad \overline{M}\langle N\rangle.P$	output
$\quad \mathit{let}\ x = D\ \mathit{in}\ P\ \mathit{else}\ Q$	term evaluation
$\quad \mathbf{0} \quad P \mid Q \quad !P \quad (\nu a)P$	

Representation of cryptographic primitives

Two possible representations:

- When success/failure is visible: destructors with rewrite rules
  - constructor *sencrypt*
  - destructor $sdecrypt(sencrypt(x, y), y) \to x$

  The *else* clause of the term evaluation is executed when no rewrite rule of some destructor applies.
- When success/failure is not visible: equations
  - $sdecrypt(sencrypt(x, y), y) = x$
  - $sencrypt(sdecrypt(x, y), y) = x$

Semantics

$D \Downarrow M$ when the term evaluation D evaluates to M.
Uses rewrite rules of destructors and equations.

$\equiv$ transforms processes so that reduction rules can be applied.

Main reduction rules:

$\overline{N}\langle M\rangle.Q \mid N'(x).P \ \rightarrow\ Q \mid P\{M/x\}$ (Red I/O)
if $\Sigma \vdash N = N'$

$let\ x = D\ in\ P\ else\ Q \rightarrow P\{M/x\}$ (Red Fun 1)
if $D \Downarrow M$

$let\ x = D\ in\ P\ else\ Q \rightarrow Q$ (Red Fun 2)
if there is no M such that $D \Downarrow M$

Observational equivalences and biprocesses

Two processes P and Q are observationally equivalent ($P \approx Q$) when the adversary cannot distinguish them.

A biprocess P is a process with diff.
$\mathsf{fst}(P)$ = the process obtained by replacing $\mathsf{diff}[M, M']$ with M.
$\mathsf{snd}(P)$ = the process obtained by replacing $\mathsf{diff}[M, M']$ with M'.

P satisfies observational equivalence when $\mathsf{fst}(P) \approx \mathsf{snd}(P)$.

Semantics of biprocesses

A biprocess reduces when both variants of the process reduce in the same way.

$$\overline{N}\langle M\rangle.Q \mid N'(x).P \;\rightarrow\; Q \mid P\{M/x\} \qquad \text{(Red I/O)}$$

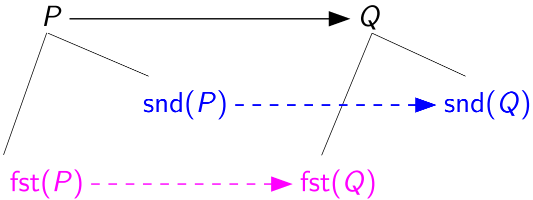
if $\Sigma \vdash \mathsf{fst}(N) = \mathsf{fst}(N')$ and $\Sigma \vdash \mathsf{snd}(N) = \mathsf{snd}(N')$

$$\mathit{let}\ x = D\ \mathit{in}\ P\ \mathit{else}\ Q \rightarrow P\{\mathsf{diff}[M_1, M_2]/x\} \qquad \text{(Red Fun 1)}$$

if $\mathsf{fst}(D) \Downarrow M_1$ and $\mathsf{snd}(D) \Downarrow M_2$

$$\mathit{let}\ x = D\ \mathit{in}\ P\ \mathit{else}\ Q \rightarrow Q \qquad \text{(Red Fun 2)}$$

if there is no M_1 such that $\mathsf{fst}(D) \Downarrow M_1$ and
there is no M_2 such that $\mathsf{snd}(D) \Downarrow M_2$



Proof of observational equivalence using biprocesses

Let P_0 be a closed biprocess.

> *If for all configurations P reachable from P_0 (in the presence of an adversary), both variants of P reduce in the same way, then P_0 satisfies observational equivalence.*

Proof of observational equivalence using biprocesses

Let P_0 be a closed biprocess.

> *If for all configurations P reachable from P_0 (in the presence of an adversary), both variants of P reduce in the same way, then P_0 satisfies observational equivalence.*

An adversary is represented by a plain evaluation context (evaluation context without diff), so:

> *If, for all plain evaluation contexts C and reductions $C[P_0] \rightarrow^* P$,*
> *both variants of P reduce in the same way,*
> *then P_0 satisfies observational equivalence.*

Result

Let P_0 be a closed biprocess.

Suppose that, for all plain evaluation contexts C and reductions $C[P_0] \rightarrow^* P$,

1. the (Red I/O) rules apply in the same way on both variants.

   if $P \equiv C'[\overline{N}\langle M\rangle.Q \mid N'(x).R]$, then
   $\Sigma \vdash \mathsf{fst}(N) = \mathsf{fst}(N')$ if and only if $\Sigma \vdash \mathsf{snd}(N) = \mathsf{snd}(N')$.

2. the (Red Fun) rules apply in the same way on both variants.

   if $P \equiv C'[\mathit{let}\ x = D\ \mathit{in}\ Q\ \mathit{else}\ R]$, then
   there exists M_1 such that $\mathsf{fst}(D) \Downarrow M_1$
   if and only if
   there exists M_2 such that $\mathsf{snd}(D) \Downarrow M_2$.

Then P_0 satisfies observational equivalence.

Conclusion

- Thanks Jean-Jacques for all that you taught me!
    - pi-calculus $\Rightarrow$ influence on the design of ProVerif
    - equivalences $\Rightarrow$ automatic proof of observational equivalences
      Application, e.g., to the proof of resistance to dictionary attacks
- Implementation and papers at
  `http://www.proverif.ens.fr/`