UNIVERSITAS STUDII PADUANI · UNIVERSITY OF PADOVA

Department of Pure and Applied Mathematics

Expressive power of definite clauses for verifying authenticity

Gilberto Filé   Roberto Vigo

March 12^{th}, 2009

Outline

Outline

Aims

Our work aims at clarifying the expressive power of the approach to protocol verification by means of definite clauses. This power relies on

- information embedded into clauses
- verification method

And nodal points are

- conditions for soundness wrt secrecy
- conditions for soundness wrt authenticity
- way of tuning precision
- possibility of achieving completeness (modulo non–termination)

Pole star: Bruno's framework for verification in the formal model

- Translation from protocols into clauses changed in time...
- ...and accordingly the power of the analysis increased!
- But whether a link exists is not explicitly proved

Results sketch

In nuce

We investigate the relationship between traces and proofs constructed with definite clauses

The main outcomes

- a translation from protocols into definite clauses
- a verification method

on which the analysis is both sound and complete (modulo non–termination)

These results help us in understanding Bruno's choices

- why clauses changed in time
- how unbounded session handling is achieved
- why Bruno's approach is not complete

Restrictions of our approach

Assumptions on protocols

- we assume that protocols communicate only through a public channel

Limitations of the approach

- we don't have an automatic method for constructing proofs (yet?)
- but we assume that proofs are given

Outline

Applied π–calculus – Syntax

c	channel
$M, N ::=$	terms
$x, y, z \mid a, b \mid f(M_1, \ldots, M_n)$	var, name, constr
$P, Q ::=$	processes
$\overline{c}\langle N \rangle^k.P \mid c(x)^k.P \mid 0 \mid$	out, in, nil
$(P \mid Q) \mid !^k P \mid$	par, replication
$(\nu a)^k P \mid begin(M)^k.P \mid$	restriction, begin
$end(M)^k.P \mid$	end event
$let^k\ x = g(M_1, \ldots, M_n)\ in\ P\ else\ Q \mid$	destr
$if^k\ M = N\ then\ P\ else\ Q$	conditional

Observations

- Each action is labelled by a unique integer k,
- Events are used to check authenticity as a correspondence

A simple example

(non–injective) Authenticity by means of an outgoing test (CP handshake)

$$\begin{array}{lcl} \text{Alice} \longrightarrow \text{Bob} & : & \{a\}_{bobKey} \\ \text{Bob} \longrightarrow \text{Alice} & : & a \end{array}$$

$$Auth2 \equiv (Alice) \mid (Bob)$$

where

$$\begin{aligned} Alice \equiv\, & !^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6. \\ & c(y)^7.if^8\ y = a\ then\ end(partner, a)^9 \end{aligned}$$

$$\begin{aligned} Bob \equiv\, & !^2(\nu\, bobKey)^{10}.\overline{c}\langle host(bobKey)\rangle^{11}.c(Xa)^{12}.let^{13}\ b \\ & = decrypt(Xa, bobkey)\ in\ begin(host(bobKey), b)^{14}.\overline{c}\langle b\rangle^{15} \end{aligned}$$

Semantics pills

$$Alice \equiv !^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6.$$
$$c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

$$(Alice, []) \rightarrow$$

Observation

Input and output are non–blocking since we assume protocols running in presence of a Dolev–Yao adversary

Semantics pills

$$Alice \equiv !^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6.$$
$$c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

$$(Alice, []) \rightarrow ((c(partner))^3 \dots, [sid_1^1]), (Alice, []) \rightarrow^*$$

Observation

Input and output are non–blocking since we assume protocols running in presence of a Dolev–Yao adversary

Semantics pills

$$Alice \equiv !^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6.$$
$$c(y)^7.if^8\ y = a\ then\ end(partner,a)^9$$

$$(Alice,[]) \rightarrow ((c(partner))^3 \ldots, [sid_1^1]), (Alice,[]) \rightarrow^*$$
$$(((\nu a)^4 \ldots, [sid_1^1]), (Alice,[]) \rightarrow$$

Observation

Input and output are non–blocking since we assume protocols running in presence of a Dolev–Yao adversary

Semantics pills

$$Alice \equiv !^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6.$$
$$c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

$$\begin{aligned}(Alice, []) \rightarrow & ((c(partner))^3 \ldots, [sid_1^1]), (Alice, []) \rightarrow^* \\ & (((\nu a)^4 \ldots, [sid_1^1]), (Alice, []) \rightarrow \\ & (\{a \mapsto a[sid_1^1]\} let^5 \ldots, [sid_1^1]), (Alice, [])\end{aligned}$$

Observation

Input and output are non–blocking since we assume protocols running in presence of a Dolev–Yao adversary

Semantics: terminology

Moral

An action A^k is uniquely identified by its label k and the sequence L of *sid*'s coming from replications preceding A^k. We call

- (k, L) an action instance,
- session leading to (k, L), the sequence of action instances on the path to k

In the previous example

- $(1, []), (3, [sid_1^1]), (4, [sid_1^1])$ is the $SL(5, [sid_1^1])$

Outline

From π to definite clauses

A protocol P is represented by a set $[\![P]\!]_{lab}$ of definite clauses of the form

$$att(p_1)[k_1, L_1] \wedge \ldots att(p_n)[k_n, L_n] \Rightarrow att(p)[k, L]$$

where

- an output in the head and preceding inputs in the body
- there are clauses also for the adversary

Main differences with Bruno's translations

1. each atom is equipped with a pair $[k, L]$
2. bound name representation: $!^1 \ldots !^n \ldots (\nu a)^k$ leads to $a[sid_1, \ldots, sid_n]$

$[\![Alice]\!]_{lab}$

$$!^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a, k)\rangle^6.$$
$$c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

Translation example

$[\![Alice]\!]_{lab}$

$$!^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6$$
$$.c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

Translation example

$[\![Alice]\!]_{lab}$

$$!^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6$$
$$.c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

- clause of the form $att(partner) \Rightarrow att(encrypt(a,k))$

Translation example

$[\![Alice]\!]_{lab}$

$$!^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6$$
$$.c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

- clause of the form $att(partner) \Rightarrow att(encrypt(a,k))$
- *getkey* computes $partner \mapsto host(x), k \mapsto x$

Translation example

$[\![Alice]\!]_{lab}$

$$!^1 c(partner)^3.(\nu a)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6$$
$$.c(y)^7.if^8\ y = a\ then\ end(partner, a)^9$$

- clause of the form $att(partner) \Rightarrow att(encrypt(a,k))$
- $getkey$ computes $partner \mapsto host(x)$, $k \mapsto x$
- then
  1. $att(host(x))[3, sid_1] \Rightarrow att(encrypt(a[sid_1], x))[6, sid_1]$

Translation example

$[\![!^2 Bob]\!]_{lab}$

$$!^2(\nu bobKey)^{10}.\overline{c}\langle host(bobKey)\rangle^{11}.c(Xa)^{12}.let^{13}\ b$$
$$= decrypt(Xa, bobkey)\ in\ begin(host(bobKey), b)^{14}.\overline{c}\langle b\rangle^{15}$$

- no input/begin preceding the current output
- then
  2. $\Rightarrow host(bobKey[sid_2])[11, sid_2]$
- and so on

Verification in action

Properties are verified by means of backward chaining

- each clause is renamed apart when selected for unification, *sid*'s included
  - sid_r is renamed as $sid_r^1, sid_r^2, \ldots$
- proofs have the form of trees

Assume we want know whether in *Auth*2 Alice is sure to talk with Bob.

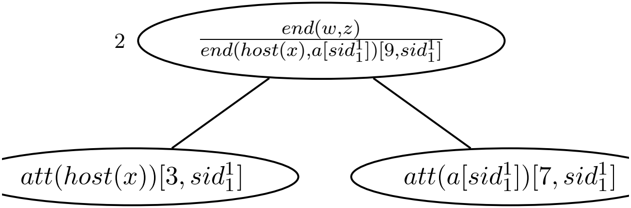
- Property: $end(w, z) \Rightarrow begin(w, z)$

Proof tree for *Auth2*

$$end(w, z)$$

The root node contains (the head of) the query

Proof tree for *Auth2* [cont.]



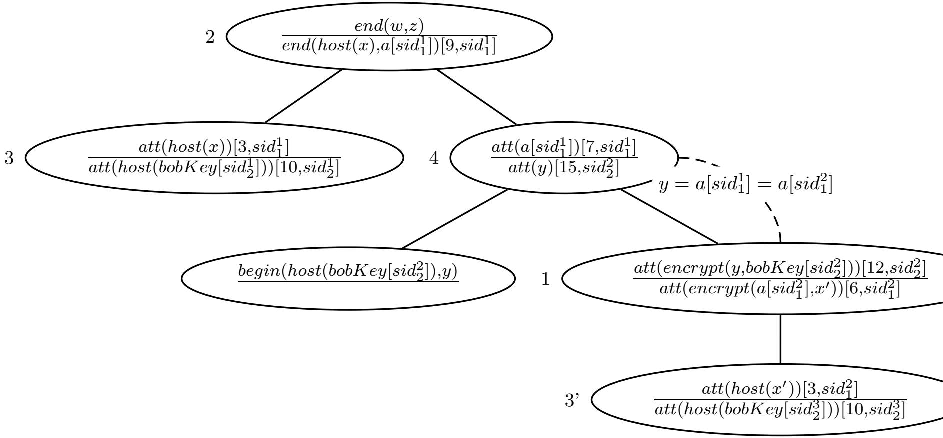
Clause 2 is selected for unification:

$$att(host(x)) \wedge att(a[sid_1^1]) \Rightarrow end(host(x), a[sid_1^1])$$

we get equations

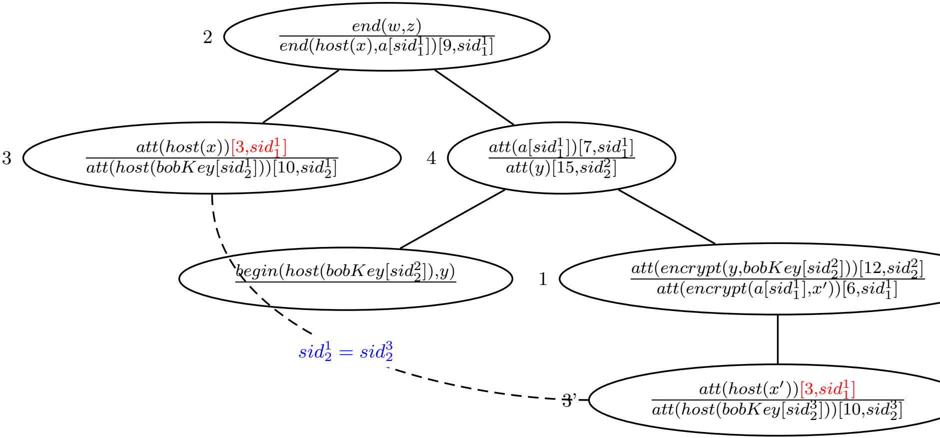
$$w = host(x), z = a[sid_1^1]$$

Proof tree for *Auth2* [cont.]



...and so on. $Eq(T)$ denotes the set of equations related to nodes of a tree T

Proof tree for *Auth2* [cont.]



Labels $[k, L]$ give extra information: same action instances are unified! Iterate until a fix point is reached.

Solvable proof trees

A proof tree T is solvable if

- the set $Eq(T)$ is solvable,
- the unifications induced by labels correctly extend a mgu of $Eq(T)$,
- the final mgu is correct wrt destructor applications

When resolution ends correctly and a mgu θ is computed, θ is applied to T and the resulting tree is said to be fully instantiated.

Verification method

If the analysis on proof trees is sound wrt trace then

- P supports secrecy of M if there is no solvable proof trees built with $[\![P]\!]_{lab}$ whose root is $att(M)$
- P supports (non–injective) authenticity if each solvable proof tree built with $[\![P]\!]_{lab}$ whose root is $end(M)$ contains an atom $begin(M)$

We now give an intuition about the correctness of this approach.

Outline

Soundness of analysis with proof trees

The correctness of the approach is based on two results

1. a strong relationship between traces and clauses
2. a strong relationship between terms in traces and in trees

Lemma

Given a trace leading to an output

$$c(x_1)\dots c(x_2)\dots \overline{c}\langle x_0\rangle$$

assume that t_j is the actual value used in the trace for x_j.
Then there exists a clause

$$C \equiv att(s_1) \wedge att(s_2) \Rightarrow att(s_0)$$

such that s_i unifies with t_i.

Hence,

- if we could simulate with proof trees the output of t_1, t_2
- gluing C on top we could simulate the whole trace!
- Note: the same let choices of the trace must be taken when translating

Soundness of analysis with proof trees [cont'ed]

Complication:

- Terms in a tree may differ from terms in a corresponding trace

When different terms in a tree correspond to a unique term in the trace, either

- the resolution process unifies them
- or they remain distinct

but this is sufficient for soundness:

1. proof trees contain equal terms only when the corresponding trace uses equal terms
2. distinct terms in the trace remain distinct also in the tree
3. thus if no trace shows an attack then all the more so trees

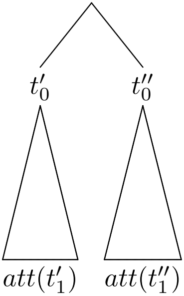
Completeness of analysis with proof trees

- Given a solvable proof tree we must show that a corresponding trace exists
- The proof does build such a trace

Intuitively

After iterating on labels, no more unification is needed for the tree to correspond to a trace.

Trees and traces, soundness and completeness



$$\Longleftarrow \ldots \underset{\substack{\uparrow \\ t_1}}{c(x_1)} \ldots \underset{\substack{\uparrow \\ t_0}}{\overline{c}\langle x_0 \rangle} \ldots \ \text{[Soundness]}$$

$$\Longrightarrow \ldots \underset{\substack{\uparrow \\ t'_1}}{c(x_1)} \ldots \underset{\substack{\uparrow \\ t'_0}}{\overline{c}\langle x_0 \rangle} \ldots \mid \ldots \underset{\substack{\uparrow \\ t''_1}}{c(x_1)} \ldots \underset{\substack{\uparrow \\ t''_0}}{\overline{c}\langle x_0 \rangle} \ldots \ \text{[Completeness]}$$

Glancing at Bruno's formalisations

Various translations have been proposed in time, that differs in bound names representations

1. $(\nu a)^k \mapsto a[p_1, \ldots, p_m]$ with p_i previous inputted terms
   - introduced with, and proved sound wrt, secrecy
2. $(\nu a)^k \mapsto a[p_1, \ldots, p_m, sid_1, \ldots, sid_n]$
   - introduced with, and proved sound wrt, authenticity

Proof trees help in clarifying that

- *sid*'s are not mandatory to be sound wrt secrecy
- *sid*'s are mandatory to be sound wrt authenticity
- adding input terms and destructor choices help in avoiding some false negative, but not to attain completeness
  - in fact bound names can contain only information about previous actions!

Authenticity needs session variables

Consider

$$Auth2' \equiv (Alice') \mid (Bob)$$

where

$$\begin{aligned} Alice' \equiv & (\nu a)^3 !^1 c(partner)^4.let^5\ k = getkey(partner)\ in\ \overline{c}\langle encrypt(a,k)\rangle^6. \\ & c(y)^7.if^8\ y = a\ then\ end(partner, y)^9 \end{aligned}$$

The protocol is flawed!

- After a run is completed the attacker knows *bobKey*[] and *a*[]
- No way to observe that *begin* and *end* events do not match, since *partner* and *y* are represented by bound names containing no parameter
- But is simple, in fact, to fool *Alice* by simply replaying them

A flawed protocol is passed as correct ⇒ the method is not sound

Completeness needs labels

Consider the protocol $P \equiv !^1 A \mid !^2 B \mid !^3 C$, where

$$A = (\nu a)^4.\overline{c}\langle a\rangle^5.c(b)^6.if^7\ b = P(a)\ then\ \overline{c}\langle K1(a)\rangle^8\ else\ \overline{c}\langle K2(a)\rangle^9$$
$$B = c(x)^{10}.\overline{c}\langle P(x)\rangle^{11}.c(y1)^{12}.c(y2)^{13}.if^{14}\ y1 = K1(x)$$
$$then\ if^{15}\ y2 = K2(x)\ then\ \overline{c}\langle OK\rangle^{16}$$
$$C = c(w)^{17}.\overline{c}\langle S(w)\rangle^{18}$$

Clauses have no control on input 6, since no bound name carry information about it (its session)

- In the proof tree input 6 is used two times, but the read values are not unified
- Thus we find that $K1(a[])$ and $K2(a[])$ can be read in the same session
- And so that OK is outputted...but this is not the case!

A correct protocol is passed as flawed $\Rightarrow$ the method is not complete

Outline

Future work

A line for future work include

- studying for a feasible implementation of our strategy
- considering injective–authenticity in the light of proof trees
- studying termination: what protocols have a finite number of proof trees for a given property
  - = tagged protocols? (on which ProVerif terminates)
- extending the established relationship between definite clauses and types up to consider authenticity in this light