Advanced Solver Technology

Helmut Seidl

TUM + DTU

2006

Part 2

Beyond Recognizability:

One-Variable and Flat Clauses

Outline of Part 2:

- Protocols with single blind copying
- One-variable Clauses
- Flat Clauses
- One-variable or Flat Clauses
- ... and Beyond :-)

1. Protocols with Single Blind Copying

Rules for exchanging messages:

$$\text{Alice} \xrightarrow{\{\mathsf{Alice}, Na\}_{\mathsf{pub(Bob)}}} \text{Bob}$$

$$\text{Alice} \xleftarrow{\{Na, Nb\}_{\mathsf{pub(Alice)}}} \text{Bob}$$

$$\text{Alice} \xrightarrow{\{Nb\}_{\mathsf{pub(Bob)}}} \text{Bob}$$

Properties to be verified: secrecy, authenticity, ...

The Dolev-Yao Model:

- Messages are terms:

	representation
$\{m\}_k$	$\mathsf{encrypt}(m, k)$
$\langle m_1, m_2 \rangle$	$\mathtt{pair}(m_1, m_2)$

$\Longrightarrow$ Distinct terms represent distinct messages :-)

$\Longrightarrow$ perfect cryptography. Thus, e.g.,

$\{m\}_k = \{m'\}_{k'}$ iff $m = m'$ and $k = k'$

- Intruder has full control over the network:

  All messages are sent to the intruder and received from the intruder.

Single blind Copying:

The Needham-Schroeder public key example:

$$
\begin{array}{ll}
1. & A \longrightarrow B : \{A, N_a\}_{K_b} \\
2. & B \longrightarrow A : \{N_a, N_b\}_{K_a} \\
3. & A \longrightarrow B : \{N_b\}_{K_b}
\end{array}
$$

Abstraction:

- Unbounded number of sessions !!
- Nonces may be non-fresh ??

Idea:

Model intruder's knowledge with Horn clauses ...

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$ $\quad \mathsf{known}(\{a, n_a\}_{k_b}) \Leftarrow$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$ $\quad \mathsf{known}(\{X, n_b\}_{k_a}) \Leftarrow \mathsf{known}(\{a, X\}_{k_b})$
3. $A \longrightarrow B : \{N_b\}_{K_b}$ $\quad \mathsf{known}(\{X\}_{k_b}) \Leftarrow \mathsf{known}(\{n_a, X\}_{k_a})$

Secrecy of N_b : $\quad \Leftarrow \mathsf{known}(n_b).$

$\Longrightarrow$ One-variable clauses :-)

Discussion:

- We abstracted all nonces to finitely many.
- Less severe (still safe) abstractions are possible ...

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$ $\quad \ldots$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$ $\quad \mathsf{known}(\{X, n_b(X)\}_{k_a}) \Leftarrow \mathsf{known}(\{a, X\}_{k_b})$
3. $A \longrightarrow B : \{N_b\}_{K_b}$ $\quad \ldots$

The generated nonce is a function of the received nonce :-)

Blanchet 2001

This is still one-variable !!!

Other Abilities of the Intruder:

$$\begin{array}{lcl}
\mathsf{known}(\mathsf{encrypt}(X,Y)) & \Leftarrow & \mathsf{known}(X) \wedge \mathsf{known}(Y) \\
 & & // \quad \text{Intruder can encrypt messages} \\
\mathsf{known}(\langle X,Y\rangle) & \Leftarrow & \mathsf{known}(X) \wedge \mathsf{known}(Y) \\
 & & // \quad \text{Intruder can form pairs} \\
\mathsf{known}(X) & \Leftarrow & \mathsf{known}(\mathsf{encrypt}(X,Y)) \wedge \mathsf{known}(Y) \\
 & & // \quad \text{Intruder can decrypt messages} \\
\mathsf{known}(X) & \Leftarrow & \mathsf{known}(\langle X,Y\rangle) \\
\mathsf{known}(Y) & \Leftarrow & \mathsf{known}(\langle X,Y\rangle) \\
 & & // \quad \text{Intruder can unpair messages}
\end{array}$$

$\Longrightarrow$ Flat clauses !!!

2. One-variable Clauses

- Every predicate is monadic.
- Every clause contains at most one variable :-)

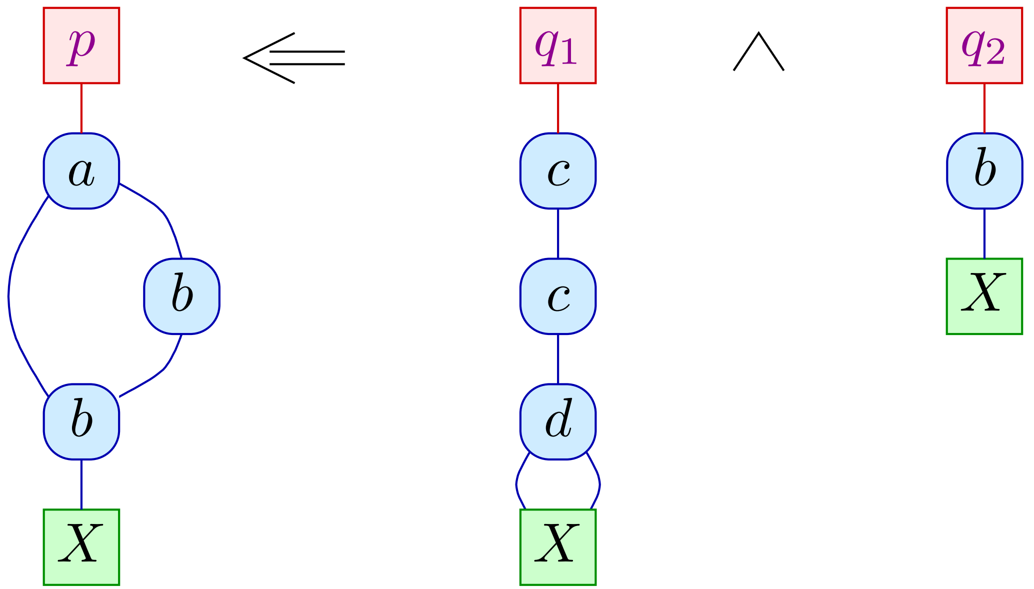
Example:

$$
\begin{aligned}
p(a(b(X), b(b(X)))) \quad \Leftarrow \quad & q_1(c(c(d(X, X)))) \wedge \\
& q_2(b(X))
\end{aligned}
$$

Observation:

Terms containing exactly one variable behave like strings ...

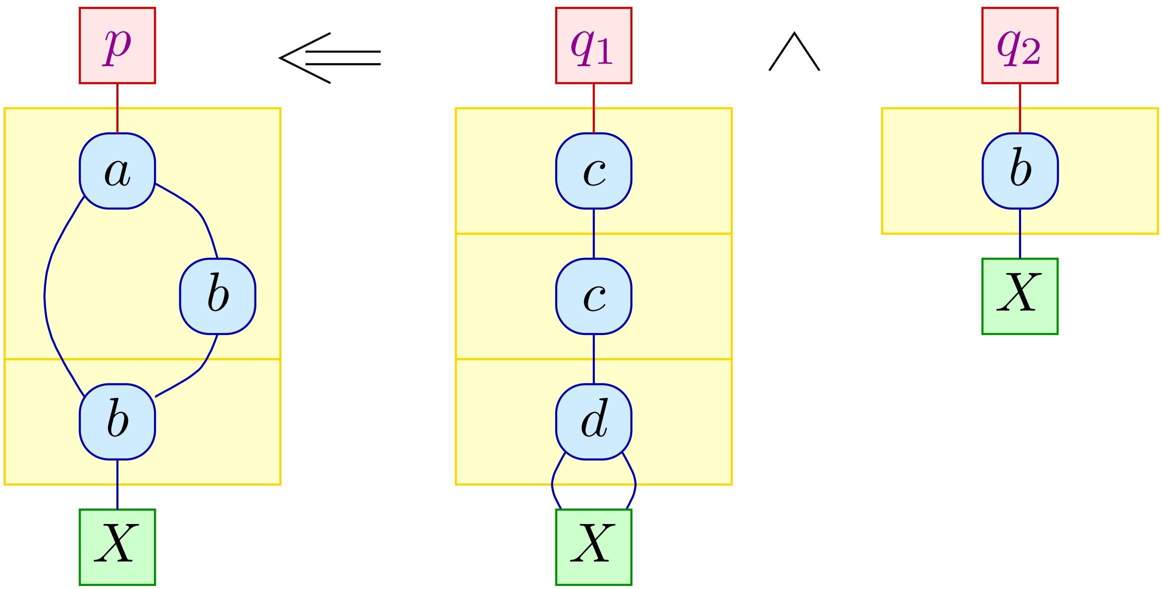
$$p(a(b(X), b(b(X)))) \;\Leftarrow\; q_1(c(c(d(X, X)))) \;\wedge\; q_2(b(X))$$



Observation:

Terms containing exactly one variable behave like strings ...

$$p(a(b(X), b(b(X)))) \;\Leftarrow\; q_1(c(c(d(X,X)))) \;\wedge\; q_2(b(X))$$



Fact:

- The set of all non-ground one-variable trees forms a free monoid !
- Each such tree can be uniquely factored into primitive trees ...

$$a(b(X), b(b(X))) \quad = \quad a(X, b(X)) \bullet b(X)$$

Fact:

- The set of all non-ground one-variable trees forms a free monoid !
- Each such tree can be uniquely factored into primitive trees ...

$$a(b(X), b(b(X))) \quad = \quad a(X, b(X)) \bullet b(X)$$

- Unification of two primitive terms maps variable to variable or both variables to ground subterms ...

$$\begin{array}{lcl} \mathsf{unify}(d(X, a(X)), d(Y, a(Y))) & = & \{X \mapsto Y\} \\ \mathsf{unify}(d(X, a(b)), d(Y, Y)) & = & \{X \mapsto a(b), Y \mapsto a(b)\} \end{array}$$

Idea:

Goal: Automata clauses:

$$
\begin{array}{lcl}
p(t) & \Leftarrow & q_1(X) \wedge \ldots \wedge q_k(X) \\
 & & \qquad // \quad t \text{ primitive} \\
p(t) & \Leftarrow & \qquad // \quad t \text{ ground} \\
p(X) & \Leftarrow &
\end{array}
$$

Idea:

Goal: Automata clauses:

$$
\begin{array}{lll}
p(t) & \Leftarrow & q_1(X) \wedge \ldots \wedge q_k(X) \\
 & & \qquad // \quad t \ \text{primitive} \\
p(t) & \Leftarrow & \qquad // \quad t \ \text{ground} \\
p(X) & \Leftarrow &
\end{array}
$$

Technique:

- Introduce auxiliary predicates to factor complex heads ...
- normalize :-)

(0) Factorization of heads:

$$p(a(\boxed{b(X)}, b(\boxed{b(X)}))) \Leftarrow q_1(c(c(d(X, X)))) \wedge q_2(b(X))$$

is simulated by:

$$\begin{array}{lcl} p(a(Z, b(Z))) & \Leftarrow & p_1(Z) \\ p_1(\boxed{b(X)}) & \Leftarrow & q_1(c(c(d(X, X)))) \wedge q_2(b(X)) \end{array}$$

(1) Resolution Step:

$$
\begin{array}{lcl}
p_1(b(X)) & \Leftarrow & \boxed{q_1(c(\boxed{c(d(X,X))}))} \;\wedge\; q_2(b(X)) \\
\boxed{q_1(c(X))} & \Leftarrow & r(X) \;\wedge\; s(X)
\end{array}
$$

resolves into:

$$
p_1(b(X)) \;\Leftarrow\; r(\boxed{c(d(X,X))}) \;\wedge\; s(\boxed{c(d(X,X))}) \;\wedge\; q_2(b(X))
$$

- The terms in the new body get smaller :-)

(2) Resolution Step:

$$
\begin{array}{lcl}
h(X) & \Leftarrow & \boxed{s_1(d(X,X))} \wedge s_2(X) \\
\boxed{s_1(d(\boxed{b(a)},X))} & \Leftarrow & p(X)
\end{array}
$$

resolves into:

$$
h(\boxed{b(a)}) \Leftarrow p(\boxed{b(a)}) \wedge s_2(\boxed{b(a)})
$$

- The terms in the new body get ground :-)

(3) Resolution Step:

$$
\begin{array}{lcl}
h(X) & \Leftarrow & \boxed{s_1(d(X, X))} \;\wedge\; s_2(X) \\
\boxed{s_1(d(\boxed{a}, \boxed{a}))} & \Leftarrow &
\end{array}
$$

resolves into:

$$
h(\boxed{a}) \quad \Leftarrow \quad s_2(\boxed{a})
$$

- The terms in the new body get ground :-)

Theorem

- Normalization of one-variable clauses is DEXPTIME-complete.
- Satisfiability of one-variable queries is is DEXPTIME-complete.

3. Flat Clauses

- Every predicate is monadic.
- Every literal contains at most one constructor ...

Example:

$$
\begin{aligned}
p(a(X,X,Y)) \quad \Leftarrow \quad & q(Z) \wedge \\
& p_1(f(Z,Y,Y)) \wedge \\
& p_2(g(X,Y,Y))
\end{aligned}
$$

Observation:

- These clauses generalize the Horn fragments of:

  ⇒ the Bernays-Schoenfinkel Class;

  ⇒ the Monadic Class.

Observation:

- These clauses generalize the Horn fragments of:

  ⇒ the Bernays-Schoenfinkel Class;

  ⇒ the Monadic Class.

- They are related to finite tree automata with equality constraints:

$$p(f(Z_1, \ldots, Z_k)) \quad \Leftarrow \quad p_1(X_1) \ \wedge \ \ldots \ \wedge \ p_n(X_n)$$

// the Z_i not necessarily distinct

// the X_j among the Z_i

Theorem:

- Normalization of flat clauses is DEXPTIME-complete.
- Satisfiability of flat queries is DEXPTIME-complete.

Idea:

Just normalize the set of clauses ...

One Typical Case:

$$
\begin{array}{lcl}
h(a(X,Y)) & \Leftarrow & \boxed{p(f(X,Y,Z))} \wedge q(g(X,X)) \\
\boxed{p(f(X,X,Z))} & \Leftarrow & s_1(X) \wedge s_2(Z)
\end{array}
$$

resolves into:

$$h(a(X,X)) \Leftarrow s_1(X) \wedge s_2(Z) \wedge q(g(X,X))$$

- The resolved complex goal disappears :-)
- Variables may be instantiated with variables :-))

4. One-variable or Flat Clauses

Idea:

- Allow automata clauses which are either one-variable or flat :-)
- Ensure that resolution between flat and one-variable clauses are flat or one-variable — whenever one is an automata clause ...

Observation:

Unification of a flat term $f(Z_1, \ldots, Z_k)$ with a one-variable term t ...

- maps all variables to ground terms or
- maps X to X and all variables Z_i to one-variable subterms ...

$$\begin{aligned} \mathsf{unify}(f(Z,Z,Y)), f(a(b,X), a(X,b), c(X)) \quad &= \\ &\{X \mapsto b, Y \mapsto c(b), Z \mapsto a(b,b)\} \\ \mathsf{unify}(f(Z,Z,Y), f(a(b,X), a(b,X), c(X)) \quad &= \\ &\{X \mapsto X, Y \mapsto c(X), Z \mapsto a(b,X)\} \end{aligned}$$

Example:

$$
\begin{array}{rcl}
p(X) & \Leftarrow & \boxed{q(a(X, \boxed{b(X)}))} \wedge p'(X) \\
\boxed{q(a(X,Y))} & \Leftarrow & q_1(X) \wedge q_2(Y)
\end{array}
$$

resolves into:

$$
p(X) \Leftarrow q_1(X) \wedge q_2(\boxed{b(X)}) \wedge p'(X)
$$

- If the automata clause is flat, the result is one-variable :-)

Example (cont.):

$$
\begin{array}{lcl}
p(X) & \Leftarrow & \boxed{q(a(X,Y))} \wedge h(f(Y,Z)) \\
\boxed{q(a(X,\boxed{b(X)}))} & \Leftarrow & q_1(X) \wedge q_2(X)
\end{array}
$$

resolves into:

$$
p(X) \Leftarrow q_1(X) \wedge q_2(X) \wedge \boxed{h(f(\boxed{b(X)}, Z))}
$$

- If the automata clause is one-variable, the result need neither be one-variable nor flat :-(

Restriction:

- Every literal in the body containing a constructor also contains all variables of the clause :-)

Thus, e.g.,

$$
\begin{array}{rcl}
p(X) & \Leftarrow & \boxed{q(a(X,Y))} \ \wedge \ h(f(X,X,Y)) \\
\boxed{q(a(X, \boxed{b(X)}))} & \Leftarrow & q_1(X) \ \wedge \ q_2(X)
\end{array}
$$

resolves into:

$$
p(X) \ \Leftarrow \ q_1(X) \ \wedge \ q_2(X) \ \wedge \ h(f(X,X,\boxed{b(X)}))
$$

- If the automata clause is one-variable, the result then is also one-variable :-)

Warning:

- The new one-variable heads need not be primitive ...

$$\begin{array}{lcl} p(f(Y)) & \Leftarrow & \boxed{q(a(X,Y))} \wedge h(X) \\ \boxed{q(a(X, \boxed{b(X)}))} & \Leftarrow & q_1(X) \wedge q_2(X) \end{array}$$

resolves into:

$$p(f(\boxed{b(X)})) \quad \Leftarrow \quad q_1(X) \wedge q_2(X) \wedge h(X)$$

- ... but they consist at most of a constructor applied to a proper subterm of a primitive term :-)
- ... which we will factor again :-))

Theorem: Verma, S. 2005

- Normalization of restricted flat or one-variable clauses is DEXPTIME-complete.
- Satisfiability of flat or one-variable queries is DEXPTIME-complete.

Corollary:

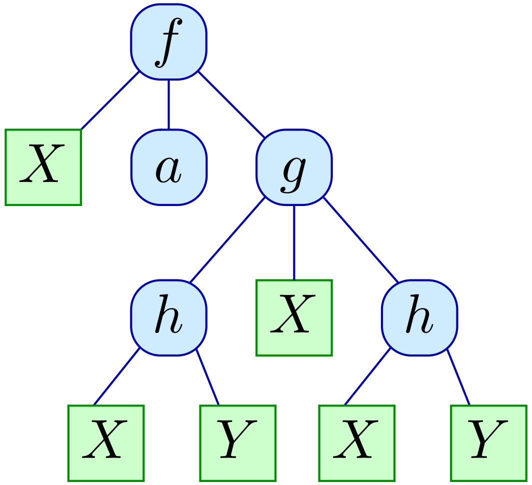
Secrecy for protocols with single blind copying is in DEXPTIME.

5. ... and Beyond?

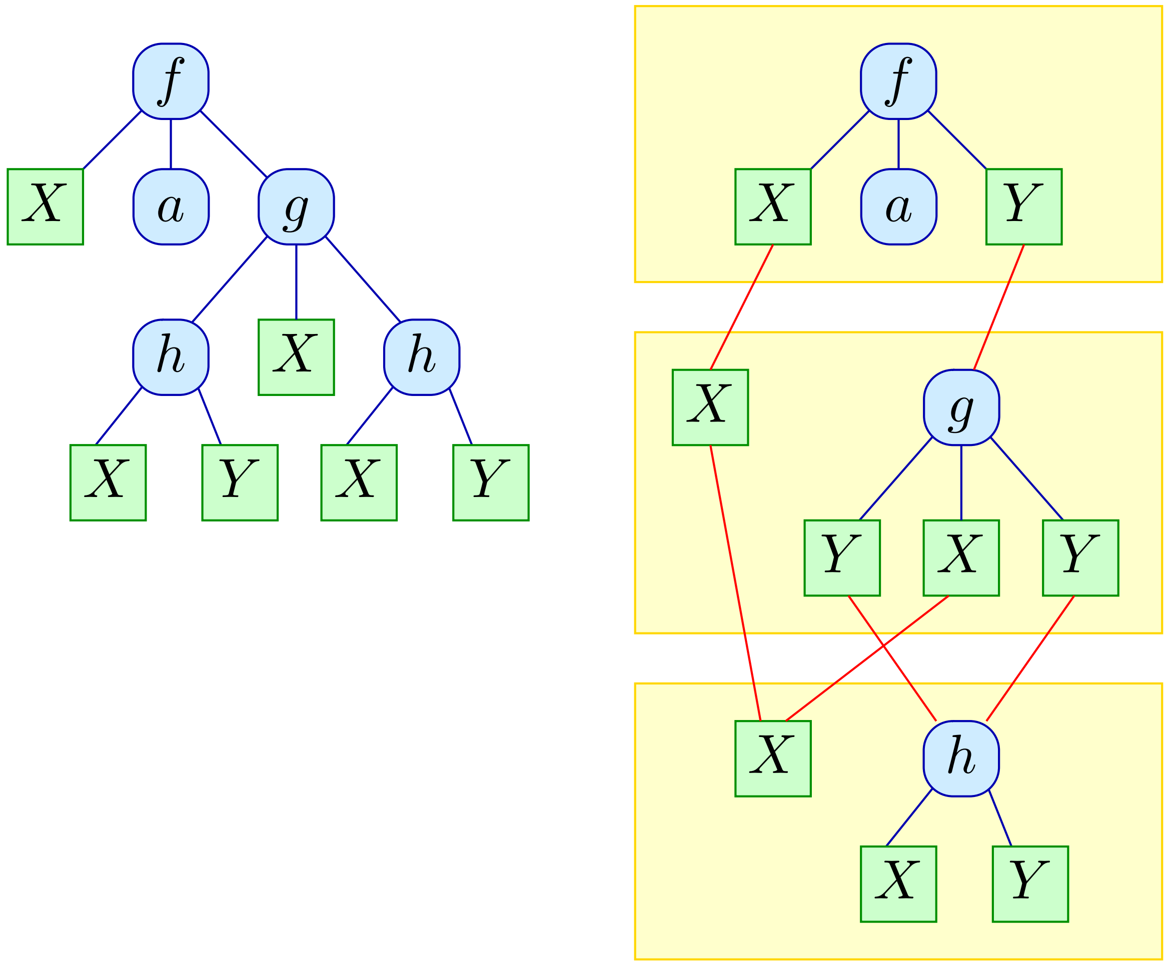
Discussion:

- Single blind copying is weak :-(
- One-variable and restricted flat Horn clauses are contained in the clausal format of the initially extended Skolem class $\mathcal{S}^+$:-)
- In the class $\mathcal{S}^+$, every functional non-ground term contains all variables ...

An Example Tree:

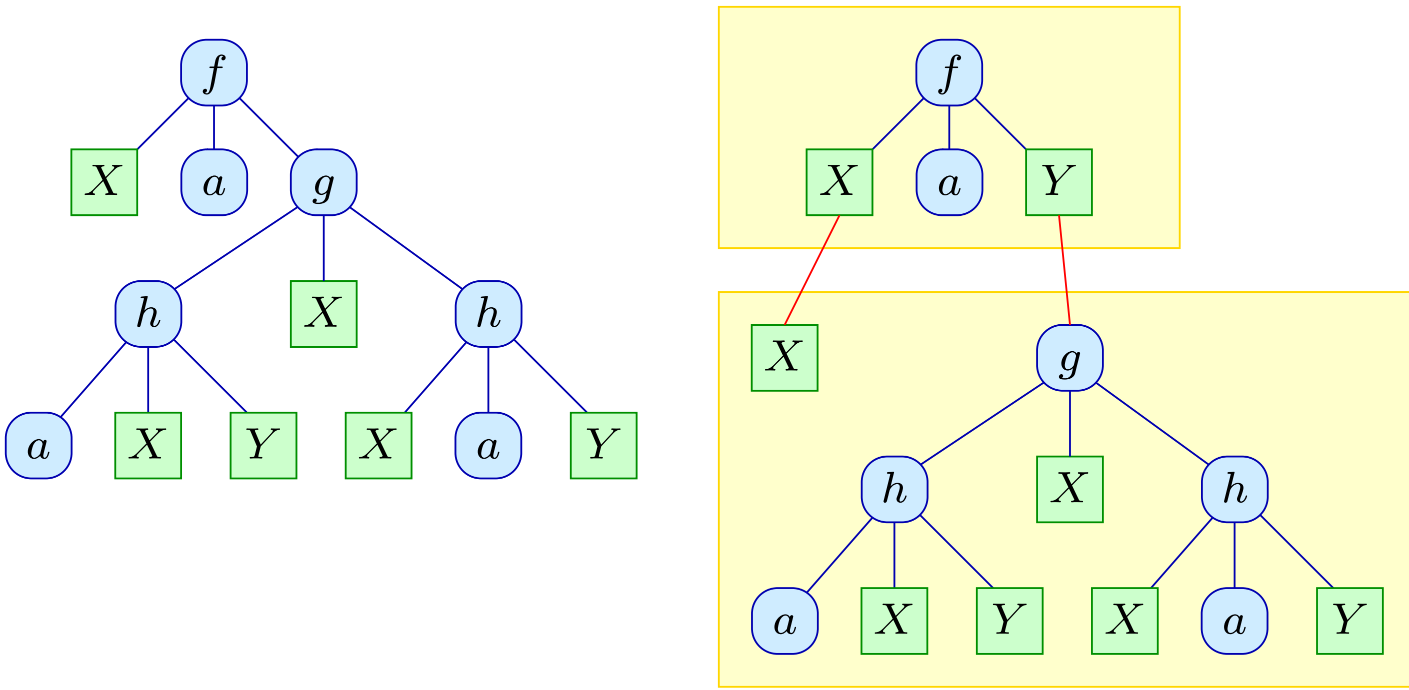


... which again can be factored:


f
X
a
g
h
X
h
X
Y
X
Y
f
X
a
Y
X
g
Y
X
Y
X
h
X
Y

Warning:

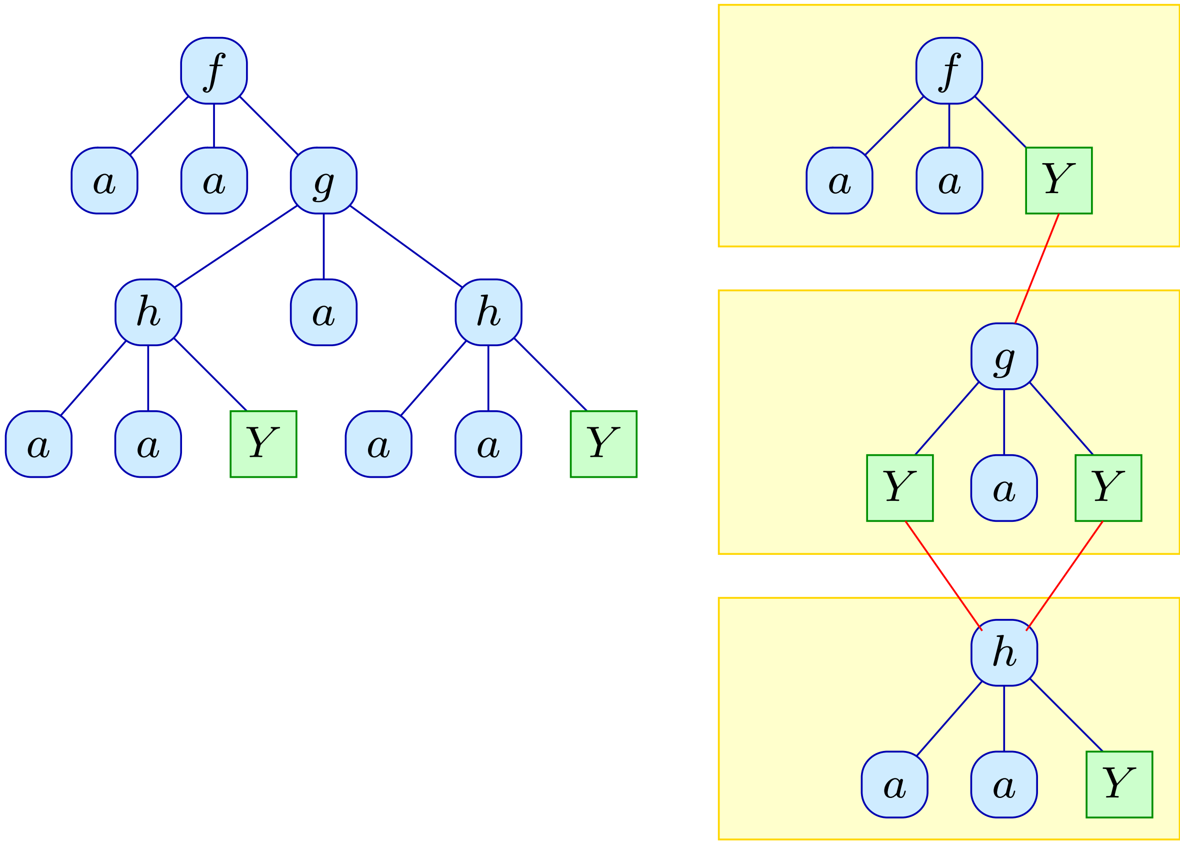
Instantiation may introduce further factorization ...



Instantiating X with a yields ...

Warning:

Instantiation may introduce further factorization ...



Observation:

Unification of extended regular terms over disjoint sets of variables can be performed in two phases ...

$$t_1 = f(a, X_1, Y_1) \qquad t_2 = f(Y_2, X_2, g(X_2, Y_2))$$

Phase 1: In each term some of the variables are equated or replaced by ground terms:

$$Y_2 \mapsto a$$

$$t_1' = f(a, X_1, Y_1) \qquad t_2' = f(a, X_2, g(X_2, a))$$

Phase 2: Variables of one term are mapped to extended regular suffices of the other ...

$$X_1 \mapsto X_2 \qquad Y_1 \mapsto g(X_2, a)$$

Idea:

- Proceed similar to the one-variable or flat case :-)
- Use factorization of heads of automata clauses — but
- Do this factorization only when needed !
- For the factorization, automata clauses may additionally contain literals like $p(X, Y, Z)$ ...

$$\boxed{\begin{array}{lll} p(f(X,Y,Z),Y,g(b)) & \Leftarrow & p(X,Y,Z) \\ p(a(X,Y),f(X,Y,X),Y) & \Leftarrow & q(X,Y) \\ q(X,Y) & \Leftarrow & q_1(X) \wedge q_2(Y) \end{array}}$$

We obtain:

Verma, Seidl 2005

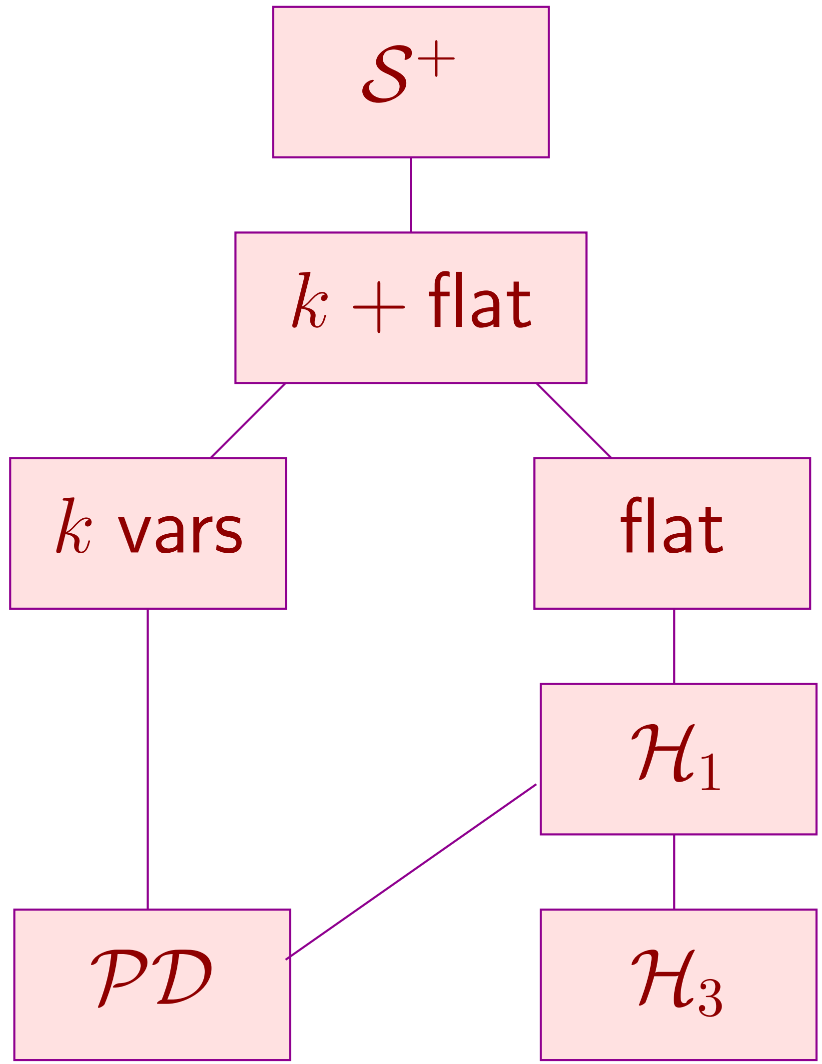
Theorem:

- For restricted flat and k-variable Horn clauses (k fix), normalization can be performed in DEXPTIME.
- For arbitrary $\mathcal{S}^+$ Horn clauses, normalization can be performed in $\mathrm{DEXP}^2\mathrm{TIME}$.

Corollary:

Secrecy for protocols with joint copying is decidable :-)

Summary:


$\mathcal{S}^+$
$k + \mathsf{flat}$
k vars
flat
$\mathcal{H}_1$
$\mathcal{PD}$
$\mathcal{H}_3$