

Advanced Solver Technology

Helmut Seidl

TUM + DTU

2006

Part 2

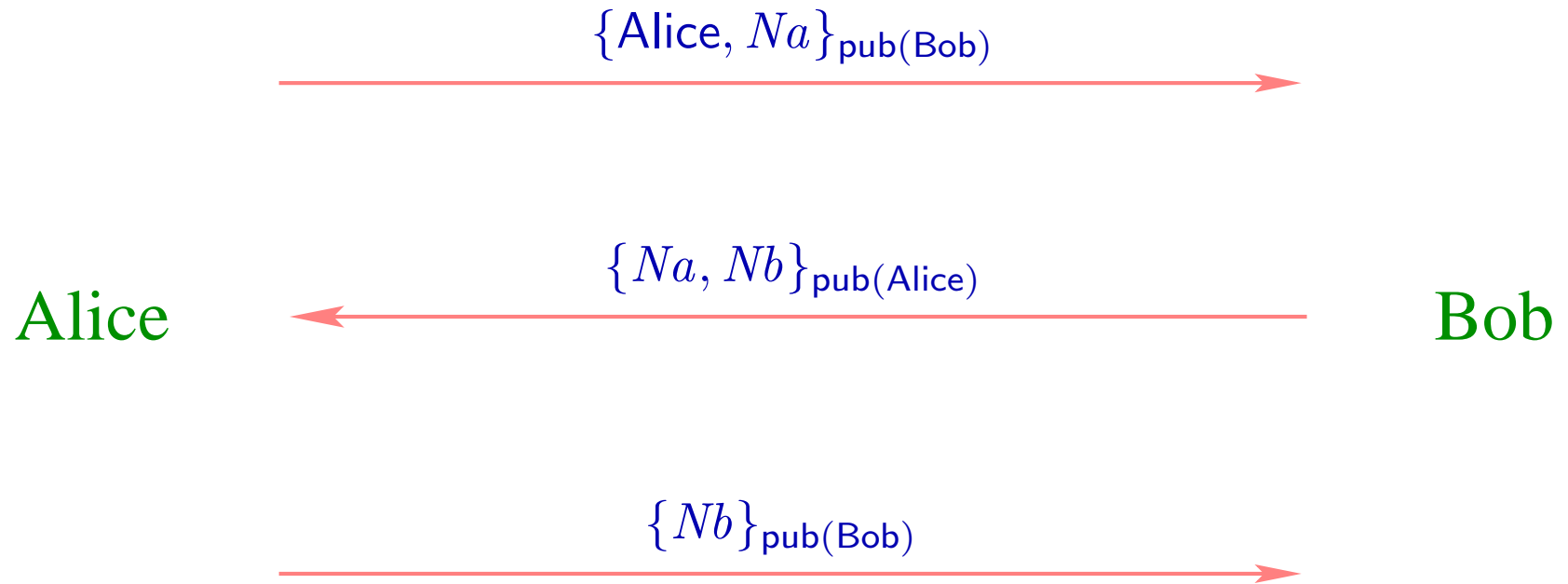
Beyond Recognizability: One-Variable and Flat Clauses

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:



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

The Dolev-Yao Model:

- Messages are terms:

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

\implies Distinct terms represent distinct messages :-)

\implies perfect cryptography. Thus, e.g.,

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

- 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:

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$
3. $A \longrightarrow B : \{N_b\}_{K_b}$

Abstraction:

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

Idea:

Model intruder's knowledge with Horn clauses ...

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

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

\implies One-variable clauses :-)

Discussion:

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

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

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

Blanchet 2001

This is still one-variable !!!

Other Abilities of the Intruder:

$\text{known}(\text{encrypt}(X, Y)) \Leftarrow \text{known}(X) \wedge \text{known}(Y)$
// Intruder can encrypt messages

$\text{known}(\langle X, Y \rangle) \Leftarrow \text{known}(X) \wedge \text{known}(Y)$
// Intruder can form pairs

$\text{known}(X) \Leftarrow \text{known}(\text{encrypt}(X, Y)) \wedge \text{known}(Y)$
// Intruder can decrypt messages

$\text{known}(X) \Leftarrow \text{known}(\langle X, Y \rangle)$

$\text{known}(Y) \Leftarrow \text{known}(\langle X, Y \rangle)$

// Intruder can unpair messages

\implies Flat clauses !!!

2. One-variable Clauses

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

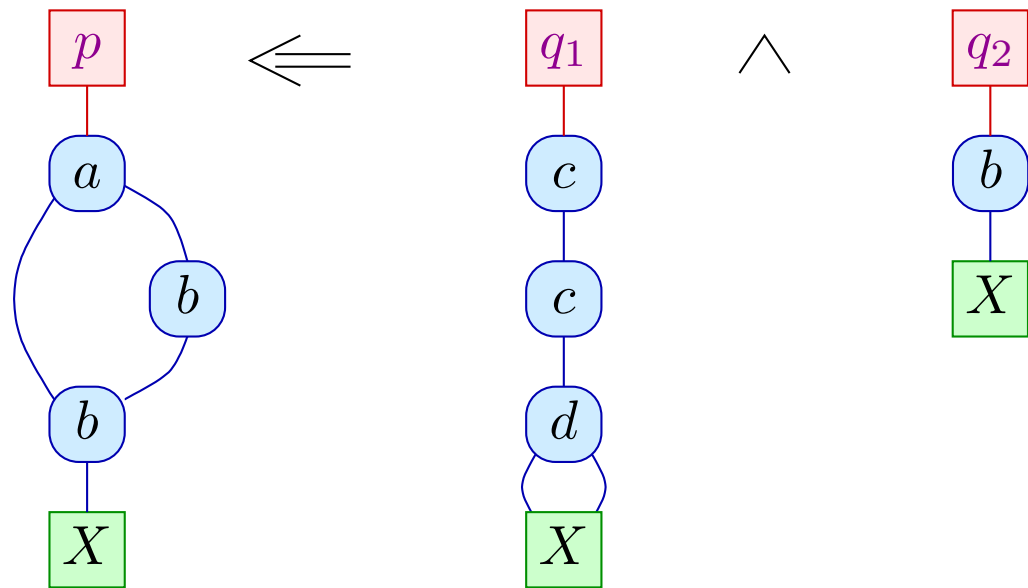
Example:

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

Observation:

Terms containing exactly one variable behave like strings ...

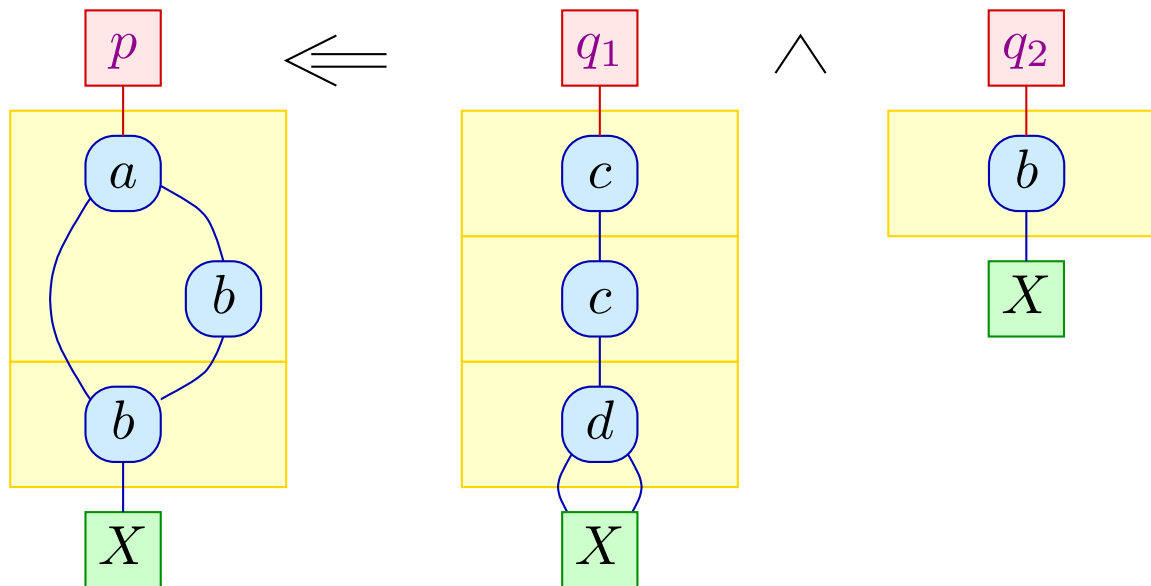
$$p(a(b(X), b(b(X)))) \Leftarrow q_1(c(c(d(X, X)))) \wedge q_2(b(X))$$



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 ...**

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

Fact:

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

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

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

$$\text{unify}(d(X, a(X)), d(Y, a(Y))) = \{X \mapsto Y\}$$

$$\text{unify}(d(X, a(b)), d(Y, Y)) = \{X \mapsto a(b), Y \mapsto a(b)\}$$

Idea:

Goal: Automata clauses:

$$p(t) \quad \Leftarrow \quad q_1(X) \wedge \dots \wedge q_k(X)$$

// t primitive

$$p(t) \quad \Leftarrow \quad // \quad t \text{ ground}$$

$$p(X) \quad \Leftarrow$$

Idea:

Goal: Automata clauses:

$$p(t) \iff q_1(X) \wedge \dots \wedge q_k(X)$$

// t primitive

$$p(t) \iff // \quad t \text{ ground}$$

$$p(X) \iff$$

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:

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

(1) Resolution Step:

$$p_1(b(X)) \Leftarrow q_1(c(c(d(X, X)))) \wedge q_2(b(X))$$
$$q_1(c(X)) \Leftarrow r(X) \wedge s(X)$$

resolves into:

$$p_1(b(X)) \Leftarrow r(c(d(X, X))) \wedge s(c(d(X, X))) \wedge q_2(b(X))$$

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

(2) Resolution Step:

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

resolves into:

$$h(b(a)) \Leftarrow p(b(a)) \wedge s_2(b(a))$$

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

(3) Resolution Step:

$$\begin{aligned} h(X) &\Leftarrow s_1(d(X, X)) \wedge s_2(X) \\ s_1(d(a, a)) &\Leftarrow \end{aligned}$$

resolves into:

$$h(a) \Leftarrow s_2(a)$$

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

Theorem

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

3. Flat Clauses

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

Example:

$$p(a(X, X, Y)) \Leftarrow q(Z) \wedge p_1(f(Z, Y, Y)) \wedge p_2(g(X, Y, Y))$$

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:

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

// 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:

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

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 $f(Z_1, \dots, Z_k)$ with a one-variable term $t \dots$

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

$$\text{unify}(f(Z, Z, Y), f(a(b, X), a(X, b), c(X))) = \\ \{X \mapsto b, Y \mapsto c(b), Z \mapsto a(b, b)\}$$

$$\text{unify}(f(Z, Z, Y), f(a(b, X), a(b, X), c(X))) = \\ \{X \mapsto X, Y \mapsto c(X), Z \mapsto a(b, X)\}$$

Example:

$$\begin{aligned} p(X) &\Leftrightarrow q(a(X, b(X))) \wedge p'(X) \\ q(a(X, Y)) &\Leftrightarrow q_1(X) \wedge q_2(Y) \end{aligned}$$

resolves into:

$$p(X) \Leftrightarrow q_1(X) \wedge q_2(b(X)) \wedge p'(X)$$

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

Example (cont.):

$$\begin{aligned} p(X) &\Leftrightarrow q(a(X, Y)) \wedge h(f(Y, Z)) \\ q(a(X, b(X))) &\Leftrightarrow q_1(X) \wedge q_2(X) \end{aligned}$$

resolves into:

$$p(X) \Leftrightarrow q_1(X) \wedge q_2(X) \wedge h(f(b(X), Z))$$

- 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.,

$$\begin{aligned} p(X) &\Leftrightarrow q(a(X, Y)) \wedge h(f(X, X, Y)) \\ q(a(X, b(X))) &\Leftrightarrow q_1(X) \wedge q_2(X) \end{aligned}$$

resolves into:

$$p(X) \Leftrightarrow q_1(X) \wedge q_2(X) \wedge h(f(X, X, b(X)))$$

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

Warning:

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

$$\begin{aligned} p(f(Y)) &\Leftrightarrow q(a(X, Y)) \wedge h(X) \\ q(a(X, b(X))) &\Leftrightarrow q_1(X) \wedge q_2(X) \end{aligned}$$

resolves into:

$$p(f(b(X))) \Leftrightarrow q_1(X) \wedge q_2(X) \wedge h(X)$$

- ... 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:

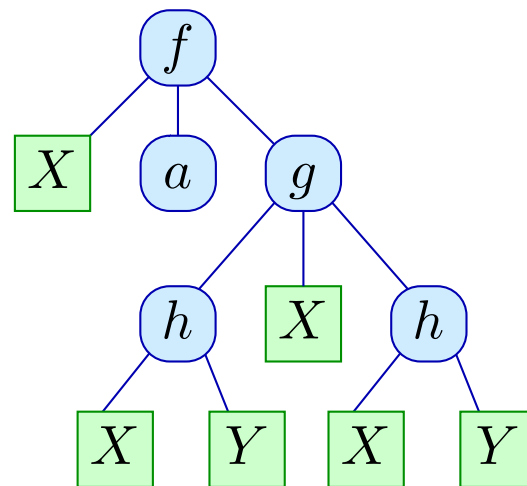
Secrecy for protocols with single blind copying is in **DEXPTIME**.

5. ... and Beyond?

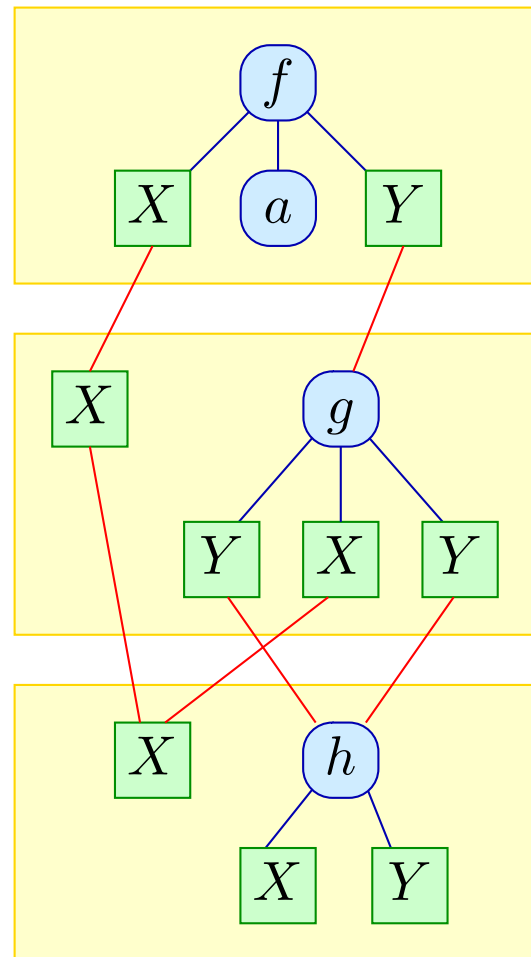
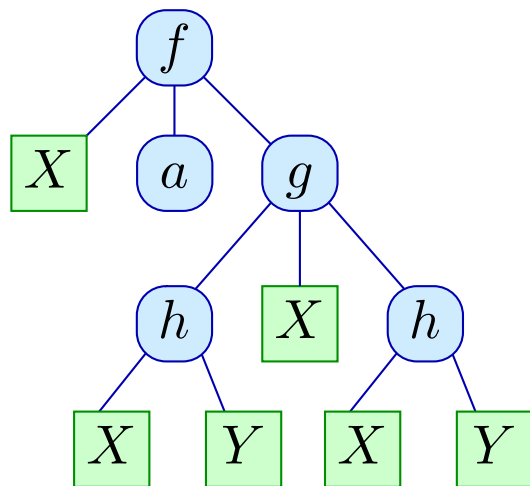
Discussion:

- 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:

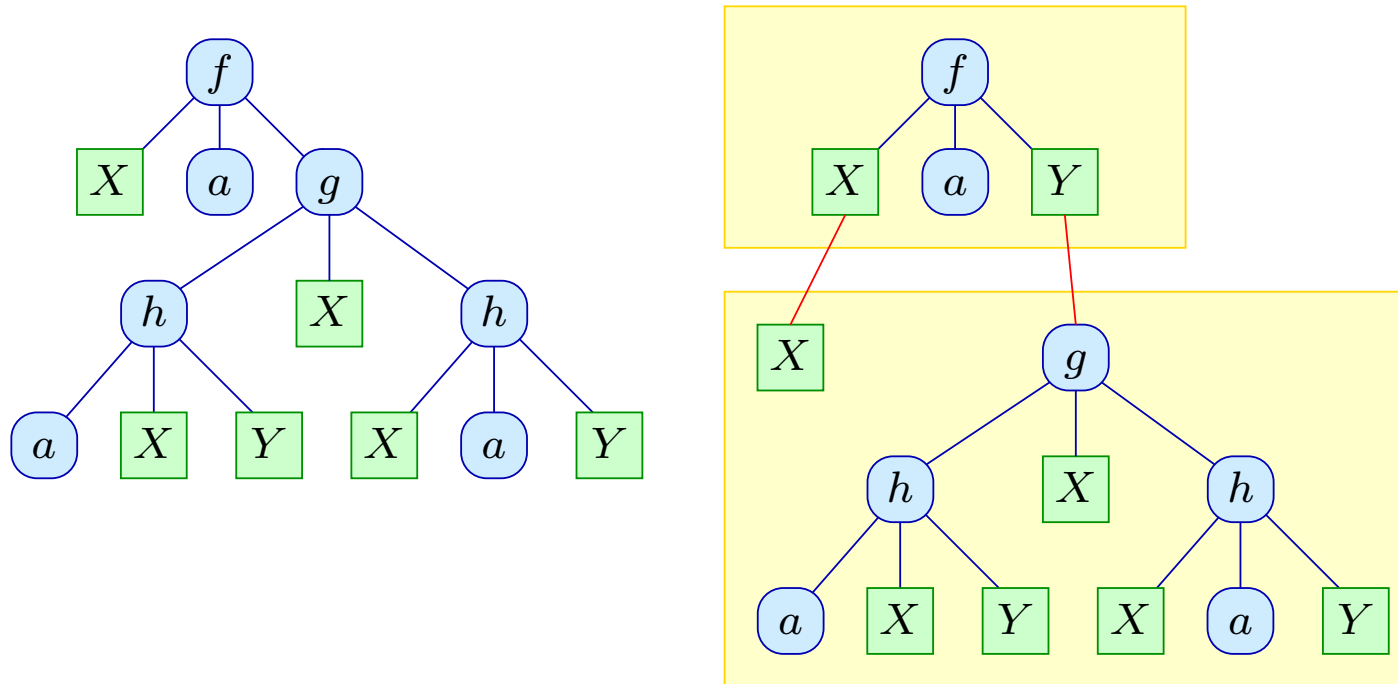


... which again can be factored:



Warning:

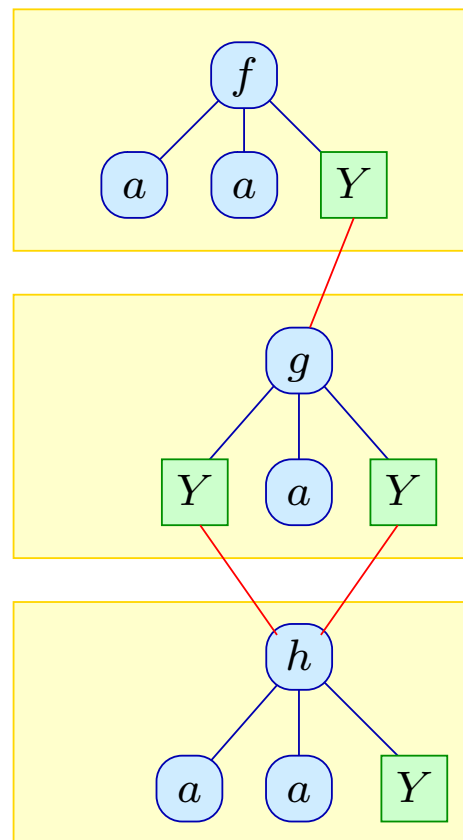
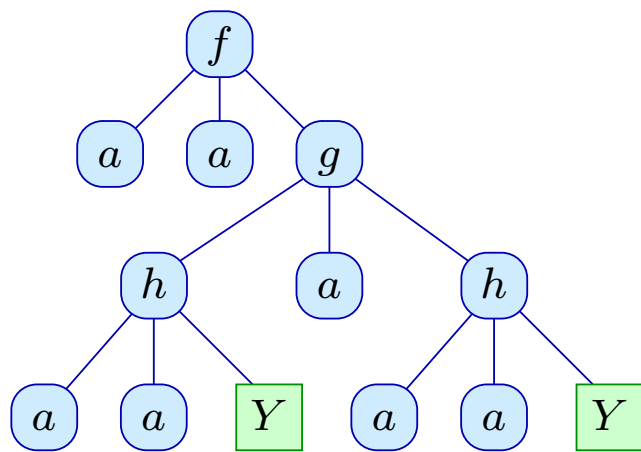
Instantiation may introduce further factorization ...



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 ...

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

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

$$Y_2 \mapsto a$$

$$t'_1 = f(a, X_1, Y_1) \quad t'_2 = f(a, X_2, g(X_2, a))$$

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

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

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 $p(X, Y, Z) \dots$

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

We obtain:

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 **DEXP²TIME**.

Corollary:

Secrecy for protocols with **joint copying** is decidable :-)

Summary:

