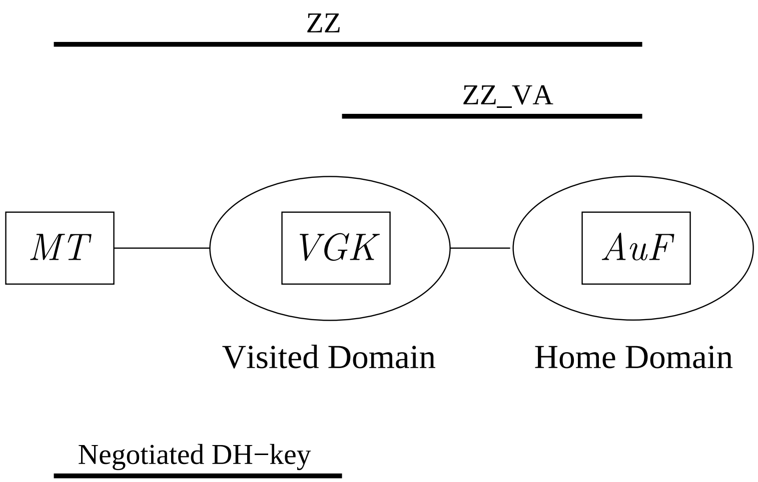
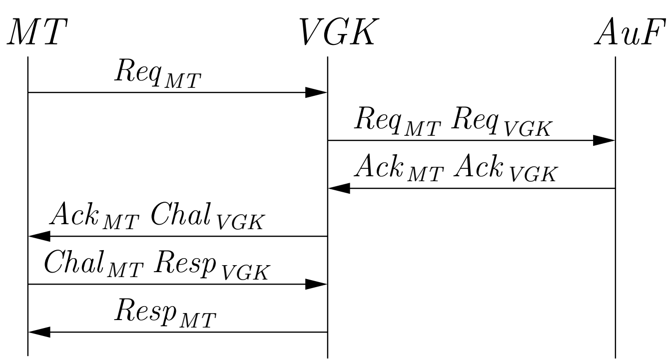
Methods for Automated Protocol Analysis

Sebastian Mödersheim

03.04.2005

Example: H.530 Protocol






```
1. MT  -> VGK : MT,VGK,NIL,CH1,{G}DHX,
                F(ZZ,MT,VGK,NIL,CH1,{G}DHX)
2. VGK -> AuF : MT,VGK,NIL,CH1,{G}DHX,
                F(ZZ,MT,VGK,NIL,CH1,{G}DHX),
                VGK,{G}DHX XOR {G}DHY,
                F(ZZ_VA,MT,VGK,NIL,CH1,{G}DHX,
                  F(ZZ,MT,VGK,NIL,CH1,{G}DHX),
                  VGK,{G}DHX XOR {G}DHY)
3. AuF -> VGK : VGK,MT,F(ZZ,VGK),
                F(ZZ,{G}DHX XOR {G}DHY),
                F(ZZ_VA,VGK,MT,F(ZZ,VGK),
                  F(ZZ,{G}DHX XOR {G}DHY))
4. VGK -> MT  : VGK,MT,CH1,CH2,{G}DHY,
                F(ZZ,{G}DHX XOR {G}DHY),
                F(ZZ,VGK),
                F({{G}DHX}DHY,VGK,MT,CH1,CH2,{G}DHY,
                  F(ZZ,{G}DHX XOR {G}DHY),F(ZZ,VGK))
5. MT -> VGK  : MT,VGK,CH2,CH3,
                F({{G}DHX}DHY,MT,VGK,CH2,CH3)
6. VGK -> MT  : VGK,MT,CH3,CH4,
                F({{G}DHX}DHY,VGK,MT,CH3,CH4)
```

Automated Analysis of Security Protocols

- Several sources of infinity in protocol analysis:
  - Unbounded message depth.
  - Unbounded number of agents.
  - Unbounded number of sessions or protocol steps.
- Possible approaches:
  - Falsification identifies attack traces but does not guarantee correctness.
  - Verification proves correctness but is difficult to automate.

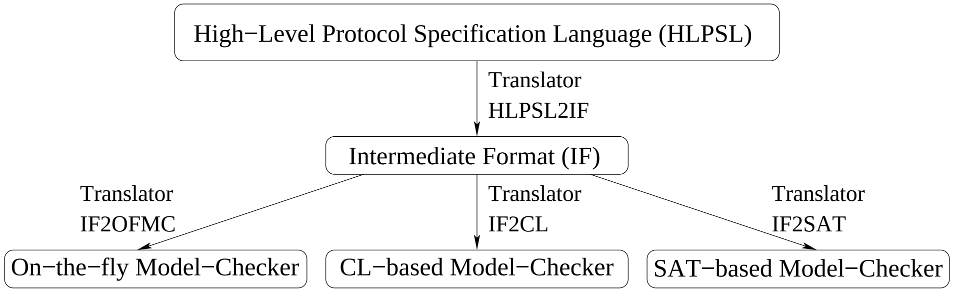
Automated Analysis of Security Protocols

- Several sources of infinity in protocol analysis:
  - Unbounded message depth.
  - Unbounded number of agents.
  - Unbounded number of sessions or protocol steps.
- Possible approaches:
  - Falsification identifies attack traces but does not guarantee correctness.
  - Verification proves correctness but is difficult to automate.
- Today's focus: falsification and verification for a bounded number of sessions.
- Still challenging model-checking problem due to state explosions:
  - The prolific Dolev-Yao intruder model
  - Concurrency: number of parallel sessions executed by honest agents.

Automated Analysis of Security Protocols

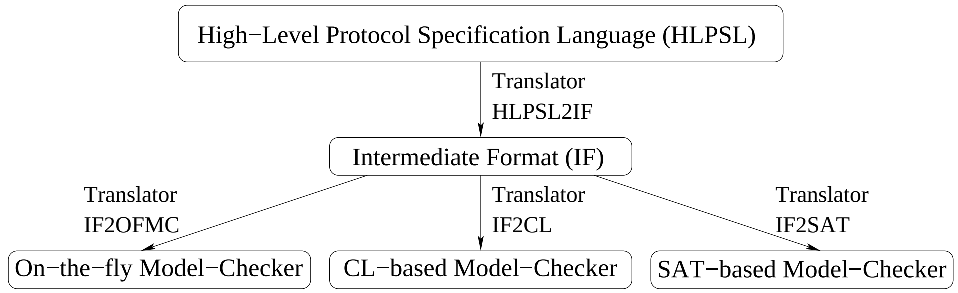
- Several sources of infinity in protocol analysis:
  - Unbounded message depth.
  - Unbounded number of agents.
  - Unbounded number of sessions or protocol steps.
- Possible approaches:
  - Falsification identifies attack traces but does not guarantee correctness.
  - Verification proves correctness but is difficult to automate.
- Today's focus: falsification and verification for a bounded number of sessions.
- Still challenging model-checking problem due to state explosions:
  - The prolific Dolev-Yao intruder model
  - Concurrency: number of parallel sessions executed by honest agents.
- Methods to reduce the search-space without excluding or introducing attacks.

Overview

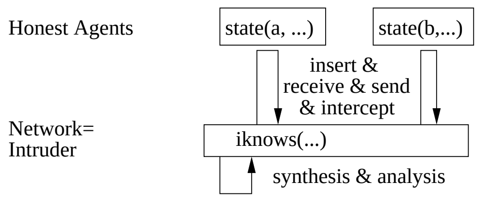


- Introduction: IF

Overview

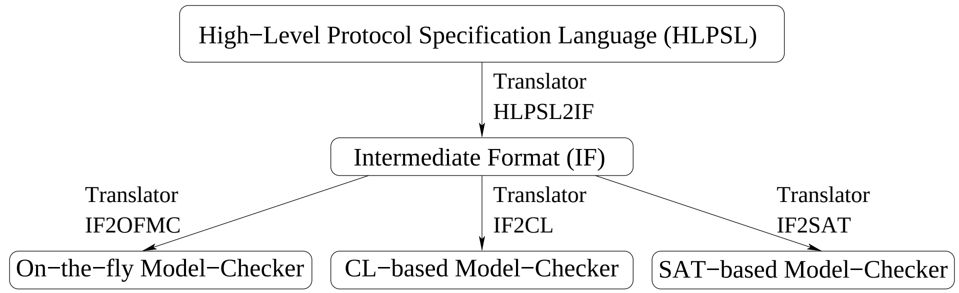


- Introduction: IF

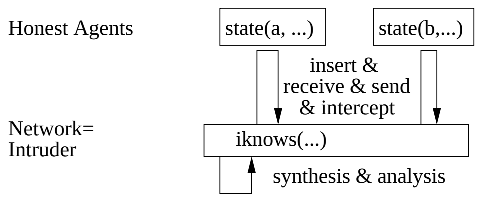


- Compressions

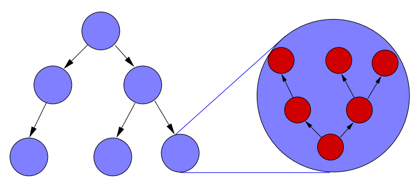
Overview



- Introduction: IF

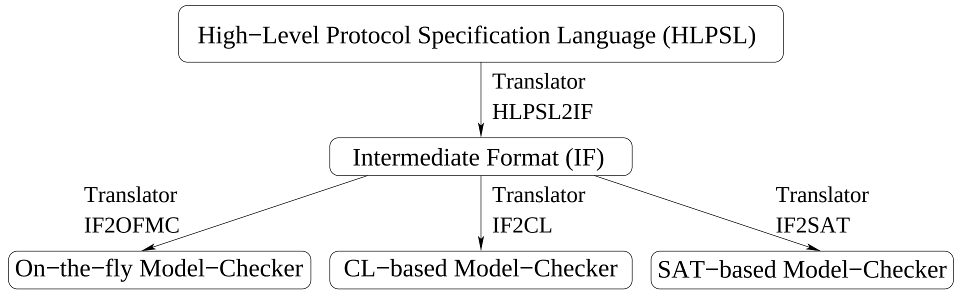


- Compressions

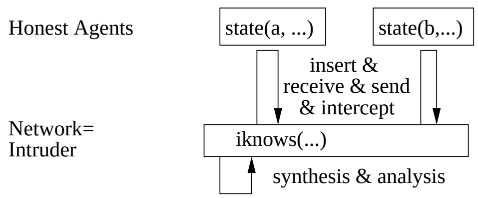


- The Lazy Intruder

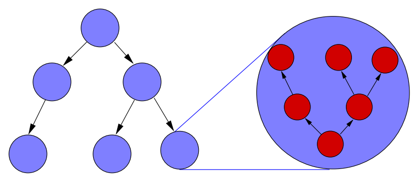
Overview



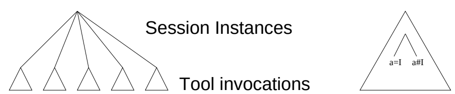
- Introduction: IF



- Compressions

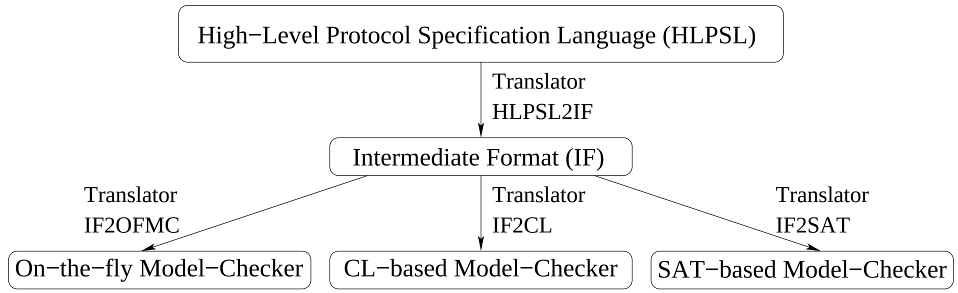


- The Lazy Intruder

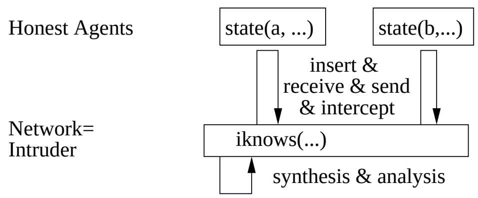


- Symbolic Sessions

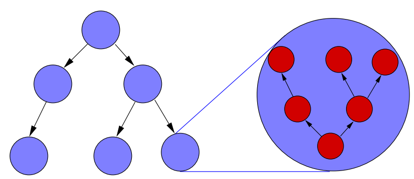
Overview



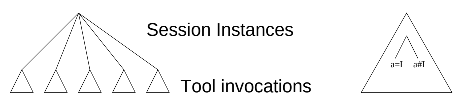
- Introduction: IF



- Compressions

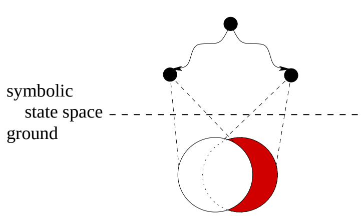


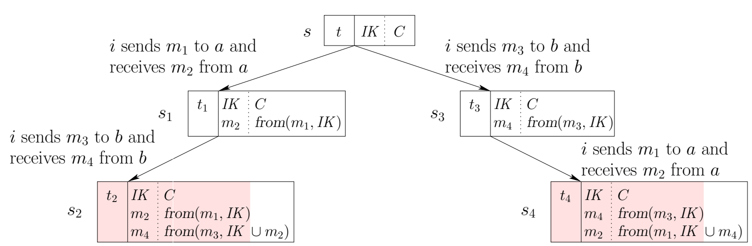
- The Lazy Intruder



- Symbolic Sessions

- Constraint Differentiation





IF: Protocol Model

- Protocol modeled as an transition system.
  - States: local states of honest agents and current knowledge of the intruder.
  - Transitions: actions of the honest agents and the intruder.

- The Dolev-Yao intruder:
  - Controls the entire network.
  - Perfect cryptography.
  - Unbounded composition of messages.

- Security properties: attack predicate on states.

- `Prelude.if` file: protocol-independent declarations (operator symbols, algebraic properties, intruder model)

IF: Messages

- Messages are represented by terms:
  - Countable set of constants, a, b, c, ...
  - Countable set of variables, A, B, C, ...
  - Function symbols for (cryptographic) operators:

$\{m\}_k$	asymmetric encryption	`crypt/2`
$\{\|m\|\}_k$	symmetric encryption	`scrypt/2`
$<m_1, m_2>$	concatenation	`pair/2`
m^{-1}	the inverse of a public/private key	`inv/1`

    also: hash functions, key tables, exponentiation, xor,. . .

- Here: Free algebra assumption:

$$f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m) \text{ iff } f = g \wedge n = m \wedge t_1 = s_1 \wedge \ldots \wedge t_n = s_m$$

- In general not valid, e.g.

$$\begin{aligned} {m^{-1}}^{-1} &= m \\ \ll m_1, m_2>, m_3> &= <m_1, <m_2, m_3\gg \end{aligned}$$

IF: Facts & States

- A fact is one of the following:

msg(m)	there is a (not yet delivered) message m on the network
state(m)	the local state of an agent, described by the term m
i_knows(m)	the intruder has learned message m

- A state is a set of facts, separated by *dots*. E.g. initial state for NSPK, for one session between a and b:

$$\mathsf{state}(\mathsf{roleA}, 0, \mathsf{a}, \mathsf{b}, \mathsf{session1}, \mathsf{pk}(\mathsf{a}), \mathsf{pk}(\mathsf{a})^{-1}, \mathsf{pk}(\mathsf{b})).$$
$$\mathsf{state}(\mathsf{roleB}, 0, \mathsf{b}, \mathsf{a}, \mathsf{session1}, \mathsf{pk}(\mathsf{b}), \mathsf{pk}(\mathsf{b})^{-1}, \mathsf{pk}(\mathsf{a})).$$

IF: Facts & States

- A fact is one of the following:

msg(m)	there is a (not yet delivered) message m on the network
state(m)	the local state of an agent, described by the term m
i_knows(m)	the intruder has learned message m

- A state is a set of facts, separated by *dots*. E.g. initial state for NSPK, for one session between a and b:

$$\mathsf{state}(\mathsf{roleA}, 0, \mathsf{a}, \mathsf{b}, \mathsf{session1}, \mathsf{pk}).$$
$$\mathsf{state}(\mathsf{roleB}, 0, \mathsf{b}, \mathsf{a}, \mathsf{session1}, \mathsf{pk}).$$
$$\mathsf{i\_knows}(\mathsf{i}).\mathsf{i\_knows}(\mathsf{a}).\mathsf{i\_knows}(\mathsf{b}).\mathsf{i\_knows}(\mathsf{pk}).\mathsf{i\_knows}(\mathsf{pk}(\mathsf{i})^{-1})$$

IF: Facts & States

- A fact is one of the following:

msg(m)	there is a (not yet delivered) message m on the network
state(m)	the local state of an agent, described by the term m
i_knows(m)	the intruder has learned message m

- A state is a set of facts, separated by *dots*. E.g. initial state for NSPK, for one session between a and b:

$$\mathsf{state}(\mathsf{roleA}, 0, \mathsf{a}, \mathsf{b}, \mathsf{session1}, \mathsf{pk}).$$
$$\mathsf{state}(\mathsf{roleB}, 0, \mathsf{b}, \mathsf{a}, \mathsf{session1}, \mathsf{pk}).$$
$$\mathsf{i\_knows}(\mathsf{i}).\mathsf{i\_knows}(\mathsf{a}).\mathsf{i\_knows}(\mathsf{b}).\mathsf{i\_knows}(\mathsf{pk}).\mathsf{i\_knows}(\mathsf{pk}(\mathsf{i})^{-1})$$

- The *dot* is associative, commutative, and idempotent:
  $f_1.(f_2.f_3) = (f_1.f_2).f_3 \quad f_1.f_2 = f_2.f_1 \quad f.f = f$

IF: Rules

- Example: NSPK/Role Alice:

$$
\begin{array}{rll}
(step\ 1) & \mathsf{state(roleA, 0, A, B, SID, K)} & \\
\exists \mathsf{NA} \Rightarrow & \mathsf{state(roleA, 1, A, B, SID, K, NA).} & \mathsf{msg(\{NA, A\}_{K(B)})} \\[1ex]
(step\ 2) & \mathsf{state(roleA, 1, A, B, SID, K, NA).} & \mathsf{msg(\{NA, NB\}_{K(A)})} \\
\Rightarrow & \mathsf{state(roleA, 2, A, B, SID, K, NA, NB)} & \\[1ex]
(step\ 3) & \mathsf{state(roleA, 2, A, B, SID, K, NA, NB)} & \\
\Rightarrow & \mathsf{state(roleA, 3, A, B, SID, K, NA, NB).} & \mathsf{msg(\{NB\}_{K(B)})}
\end{array}
$$

- Asynchronous Communication: sending and receiving messages are atomic events.

IF: Dolev-Yao intruder

$$(intercept) \qquad \mathsf{msg}(\mathsf{M}) \quad \Rightarrow \quad \mathsf{i\_knows}(\mathsf{M})$$

IF: Dolev-Yao intruder

$$\begin{array}{rrcl} (intercept) & \mathsf{msg}(\mathsf{M}) & \Rightarrow & \mathsf{i\_knows}(\mathsf{M}) \\ (insert) & \mathsf{i\_knows}(\mathsf{M}) & \Rightarrow & \mathsf{msg}(\mathsf{M}).\mathsf{i\_knows}(\mathsf{M}) \end{array}$$

IF: Dolev-Yao intruder

$$
\begin{array}{rrcl}
(intercept) & \mathsf{msg(M)} & \Rightarrow & \mathsf{i\_knows(M)} \\
(insert) & \mathsf{i\_knows(M)} & \Rightarrow & \mathsf{msg(M).i\_knows(M)} \\
(synthesis) & \mathsf{i\_knows(M_1).i\_knows(M_2)} & \Rightarrow & \mathsf{i\_knows(<M_1, M_2>)} \\
 & \mathsf{i\_knows(M_1).i\_knows(M_2)} & \Rightarrow & \mathsf{i\_knows(\{M_2\}_{M_1})} \\
 & \mathsf{i\_knows(M_1).i\_knows(M_2)} & \Rightarrow & \mathsf{i\_knows(\{|M_2|\}_{M_1})}
\end{array}
$$

IF: Dolev-Yao intruder

$$
\begin{array}{rrcl}
(\mathit{intercept}) & \mathsf{msg(M)} & \Rightarrow & \mathsf{i\_knows(M)} \\
(\mathit{insert}) & \mathsf{i\_knows(M)} & \Rightarrow & \mathsf{msg(M).i\_knows(M)} \\
(\mathit{synthesis}) & \mathsf{i\_knows(M_1).i\_knows(M_2)} & \Rightarrow & \mathsf{i\_knows(<M_1, M_2>)} \\
 & \mathsf{i\_knows(M_1).i\_knows(M_2)} & \Rightarrow & \mathsf{i\_knows(\{M_2\}_{M_1})} \\
 & \mathsf{i\_knows(M_1).i\_knows(M_2)} & \Rightarrow & \mathsf{i\_knows(\{\!|M_2|\!\}_{M_1})} \\
(\mathit{analysis}) & \mathsf{i\_knows(<M_1, M_2>)} & \Rightarrow & \mathsf{i\_knows(M_1).i\_knows(M_2)} \\
 & \mathsf{i\_knows(\{M_2\}_{M_1}).i\_knows({M_1}^{-1})} & \Rightarrow & \mathsf{i\_knows(M_2)} \\
 & \mathsf{i\_knows(\{M_2\}_{{M_1}^{-1}}).i\_knows(M_1)} & \Rightarrow & \mathsf{i\_knows(M_2)} \\
 & \mathsf{i\_knows(\{\!|M_2|\!\}_{M_1}).i\_knows(M_1)} & \Rightarrow & \mathsf{i\_knows(M_2)}
\end{array}
$$

$$
\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{\!|m_2|\!\}_{m_1} \in \mathcal{DY}(M)} \, G_{\text{scrypt}} \,,
$$

$$
\frac{\{\!|m_2|\!\}_{m_1} \in \mathcal{DY}(M) \quad m_1 \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)} \, A_{\text{scrypt}} \,,
$$

IF: As Search Tree

IF	Search Tree
state	node
initial state	root node
transition relation	descendants of a node
attack state	attack node

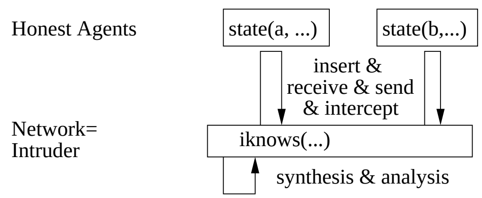
- States are always ground terms.
- The search tree is, in general, infinitely deep.
- Also the tree might have infinite branching.

⇒ Semi-decision procedure for insecurity.

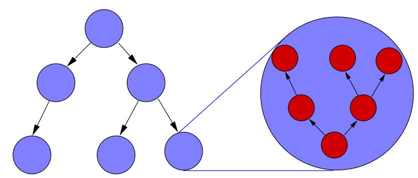
⇒ Use lazy data-structures, e.g. in Haskell: formulate infinite tree; heuristics and attack search as tree-transformers.

Overview

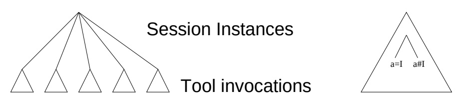
- Introduction: IF



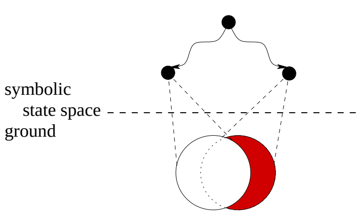
- Compressions

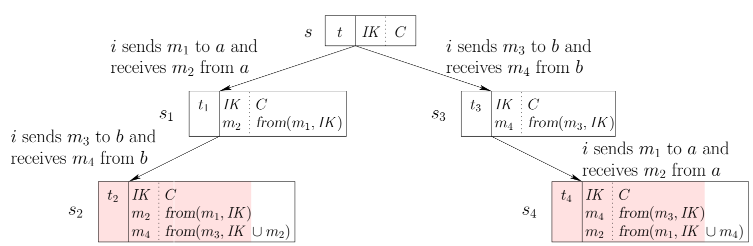


- The Lazy Intruder



- Symbolic Sessions
- Constraint Differentiation





Compression: Immediate Reaction

- Idea [Denker, Millen, Grau, and Kuester Filipe]: combine rules for receiving messages and for sending the reply, e.g. NSPK/Role Alice:

$$
\boxed{
\begin{array}{rll}
(\mathit{step}\ 2) & \mathsf{state}(\mathsf{roleA}, 1, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}). & \mathsf{msg}(\{\mathsf{NA}, \mathsf{NB}\}_{\mathsf{K(A)}}) \\
\Rightarrow & \mathsf{state}(\mathsf{roleA}, 2, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}, \mathsf{NB}) & \\
 & & \\
(\mathit{step}\ 3) & \mathsf{state}(\mathsf{roleA}, 2, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}, \mathsf{NB}) & \\
\Rightarrow & \mathsf{state}(\mathsf{roleA}, 3, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}, \mathsf{NB}). & \mathsf{msg}(\{\mathsf{NB}\}_{\mathsf{K(B)}})
\end{array}
}
$$

$$\downarrow$$

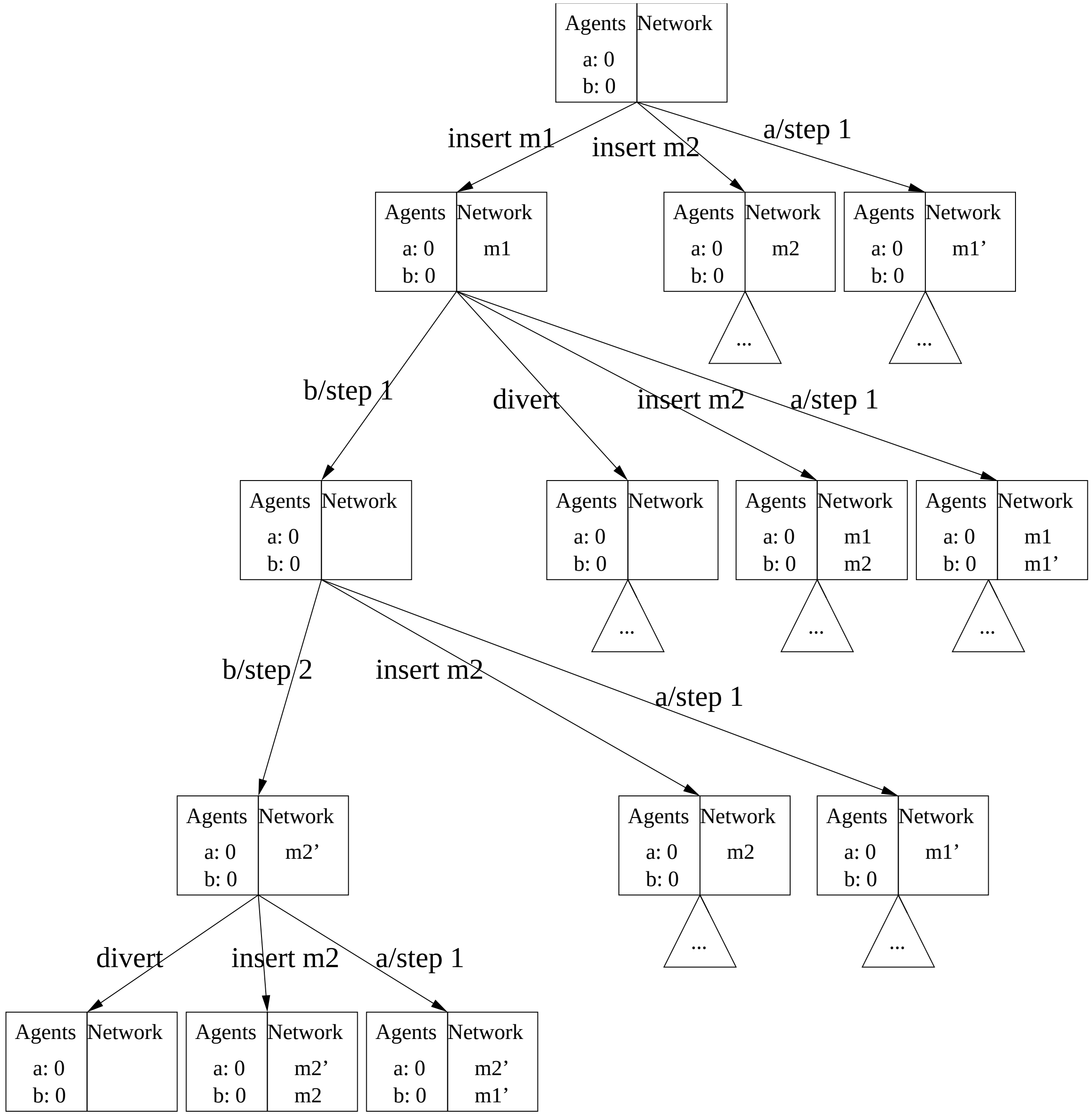
$$
\boxed{
\begin{array}{rll}
(\mathit{step}\ 2/3) & \mathsf{state}(\mathsf{roleA}, 1, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}). & \mathsf{msg}(\{\mathsf{NA}, \mathsf{NB}\}_{\mathsf{K(A)}}) \\
\Rightarrow & \mathsf{state}(\mathsf{roleA}, 3, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}, \mathsf{NB}). & \mathsf{msg}(\{\mathsf{NB}\}_{\mathsf{K(B)}})
\end{array}
}
$$

- Correct: composed rule can be simulated using the uncomposed rules.

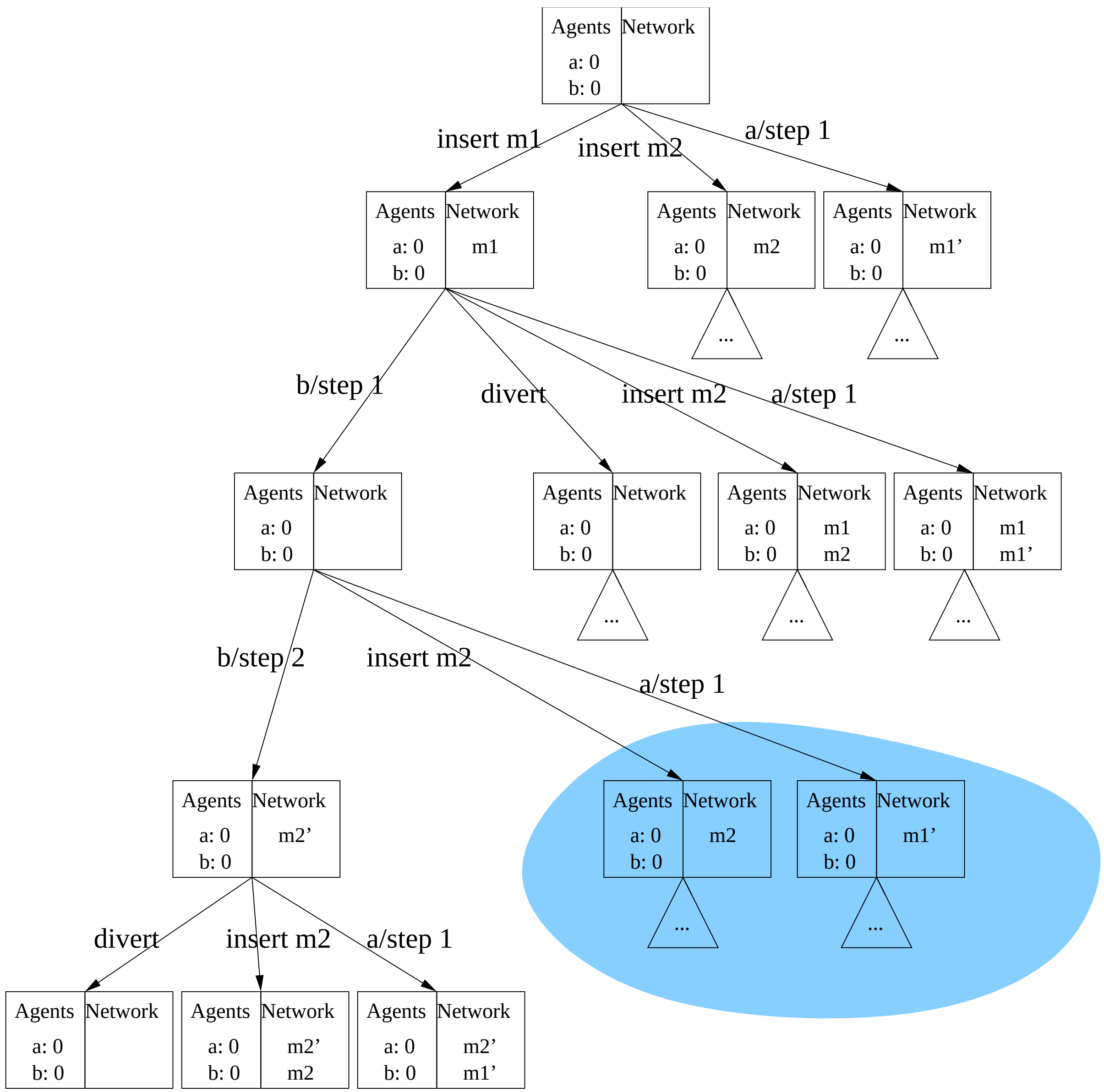
- Complete: (for standard security properties) this does not exclude any attacks.

- The compression drastically reduces the search space.

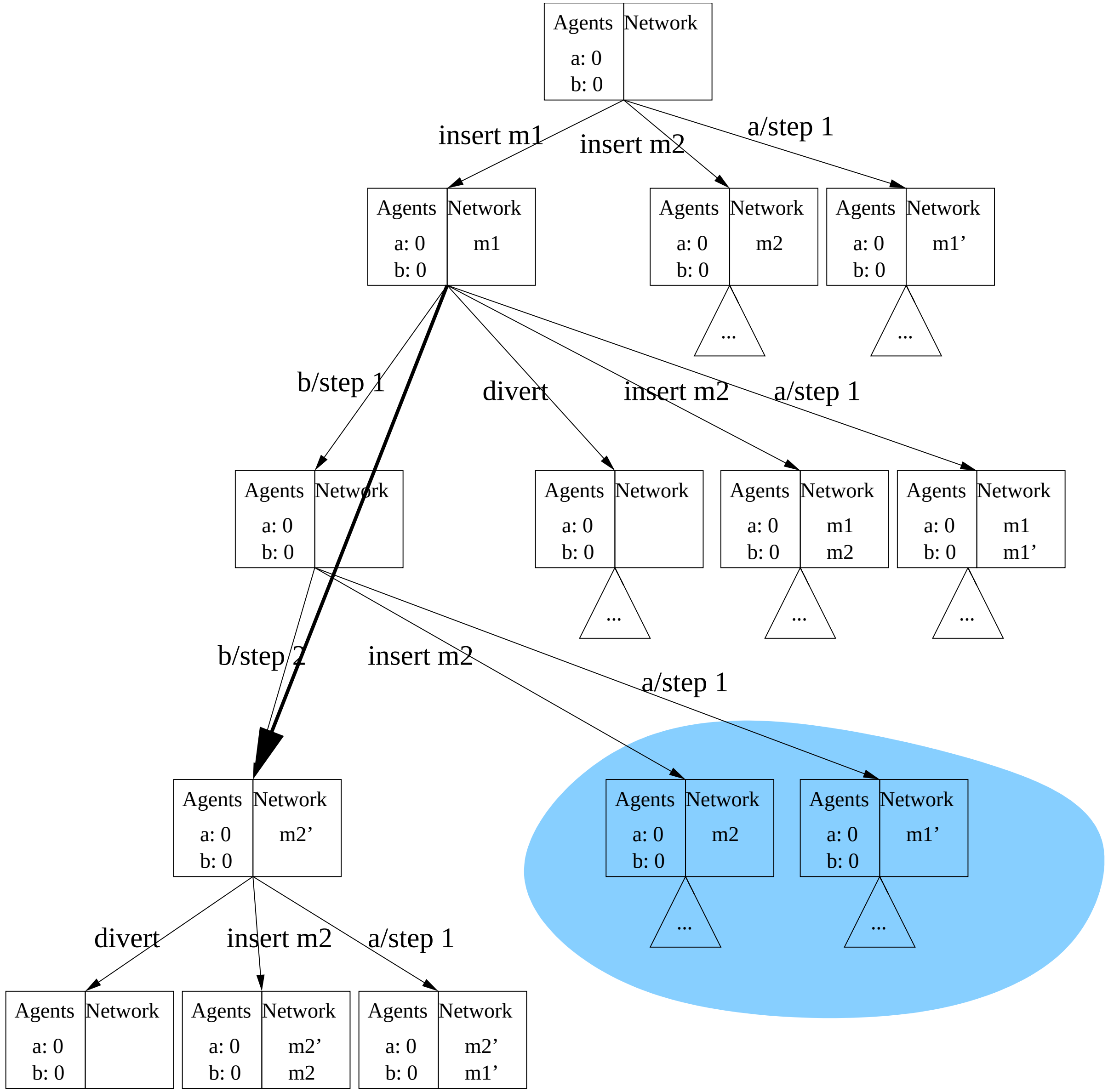
Immediate Reaction: Effects



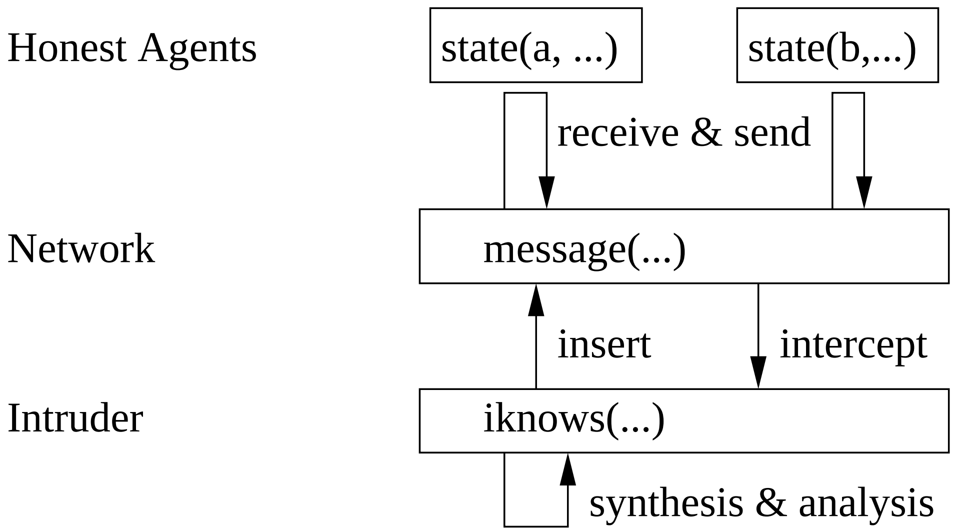
Immediate Reaction: Effects



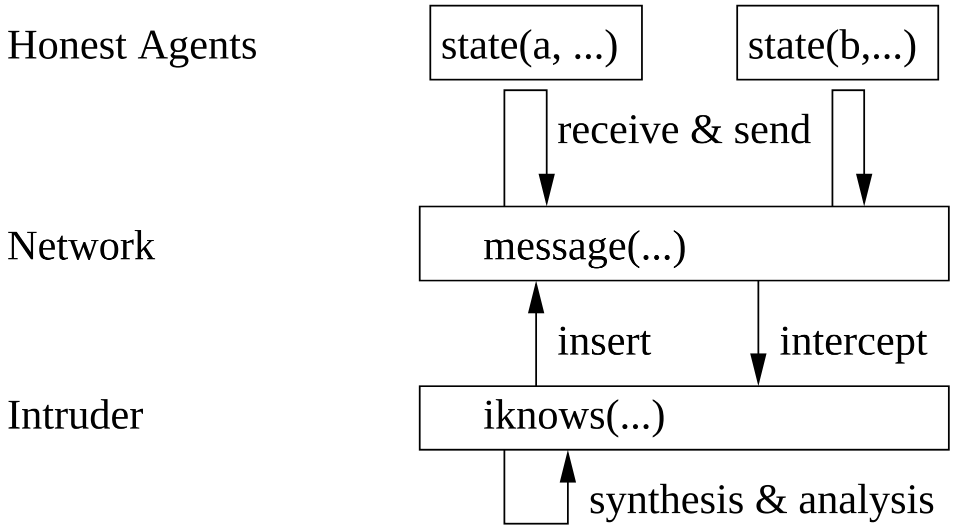
Immediate Reaction: Effects



Compressing Further

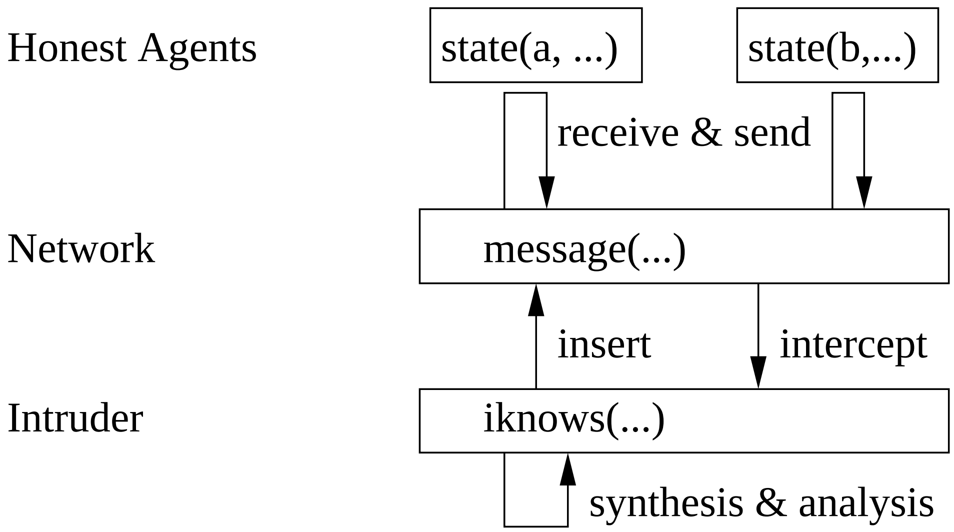


Compressing Further

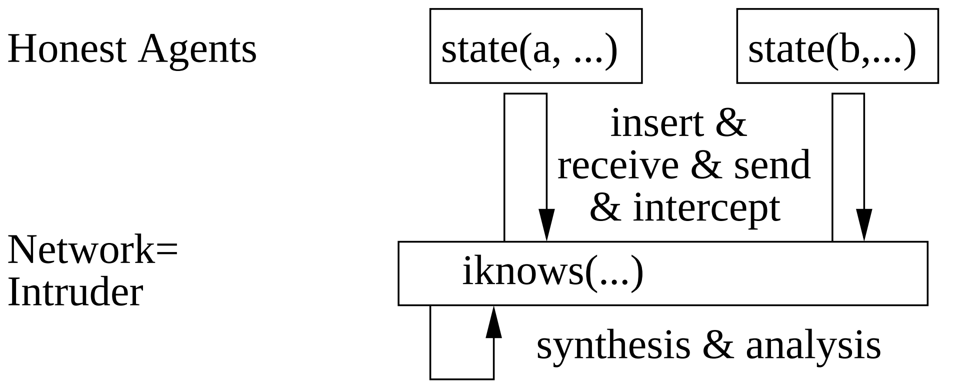


Idea: the intruder *is* the network.

Compressing Further



Idea: the intruder *is* the network.



Step-Compression

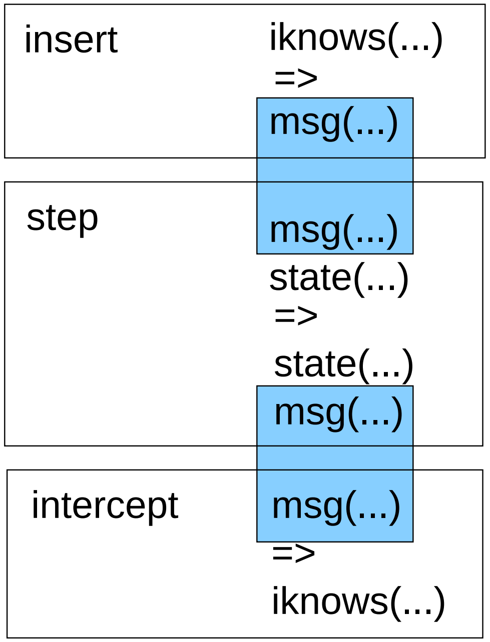
- Idea: since we do not distinguish intruder and network
  - every message that an honest agent sends to the network is automatically diverted; and
  - every message that an honest agent reveices from the network was earlier inserted by the intruder.

- $\Rightarrow$ Compression of 3 rules:

insert	iknows(...) => msg(...)
step	msg(...) state(...) => state(...) msg(...)
intercept	msg(...) => iknows(...)

Step-Compression

- Idea: since we do not distinguish intruder and network
  - every message that an honest agent sends to the network is automatically diverted; and
  - every message that an honest agent reveices from the network was earlier inserted by the intruder.
- ⇒ Compression of 3 rules:



Step-Compression

- Idea: since we do not distinguish intruder and network
  - every message that an honest agent sends to the network is automatically diverted; and
  - every message that an honest agent reveices from the network was earlier inserted by the intruder.
- $\Rightarrow$ Compression of 3 rules:

insert	state(...)
	iknows(...)
step	=>
	state(...)
intercept	iknows(...)

- Correctness & completeness similar as for the immediate reaction.

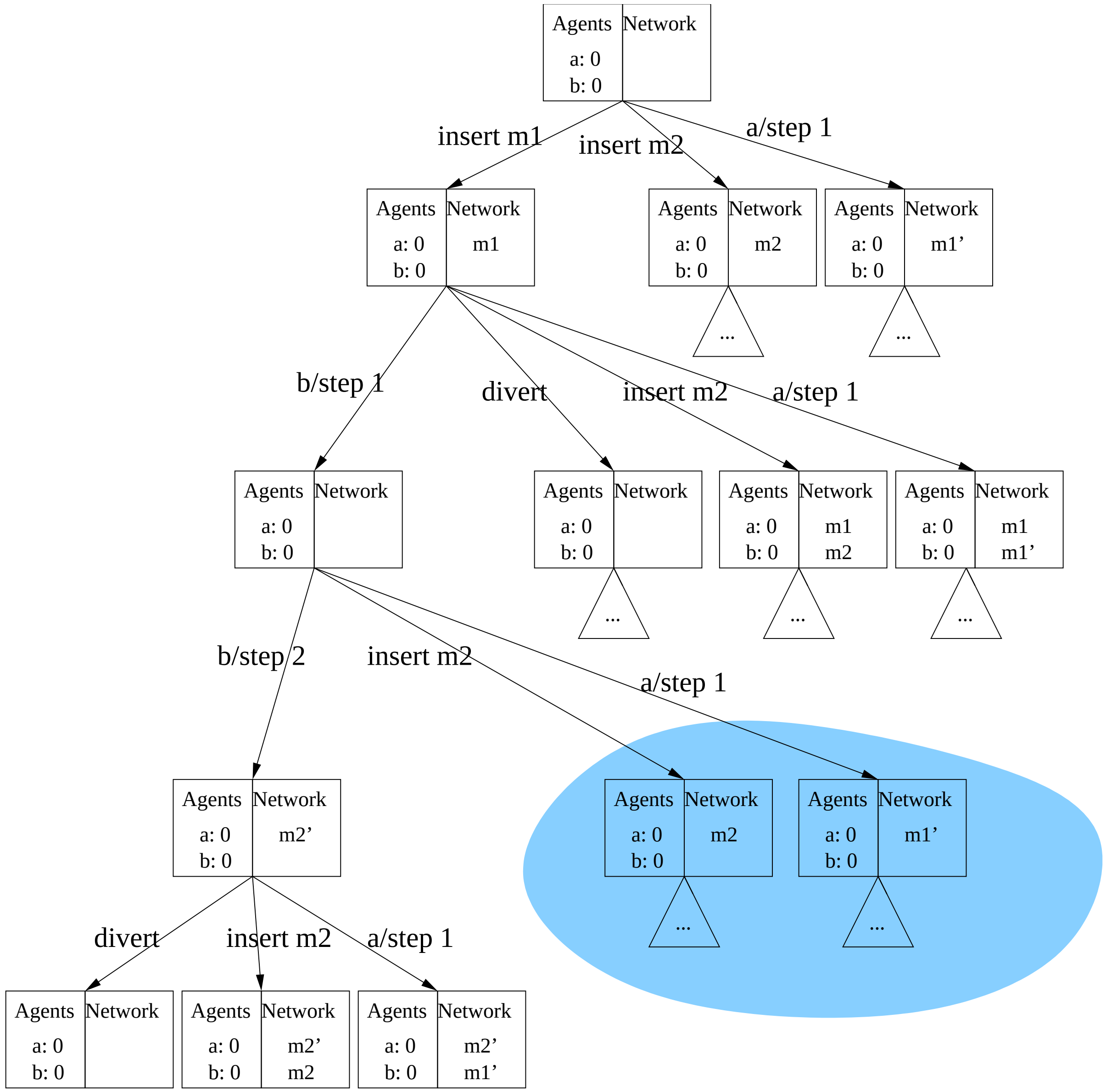
Example NSPK, role Alice

$$
\begin{array}{rll}
(\mathit{insert}) & & \mathsf{i\_knows}(\mathsf{M}) \\
\Rightarrow & & \mathsf{msg}(M) \\[1ex]
(\mathit{step}\ 2/3) & \mathsf{state}(\mathsf{roleA}, 1, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}). & \mathsf{msg}(\{\mathsf{NA}, \mathsf{NB}\}_{\mathsf{K(A)}}) \\
\Rightarrow & \mathsf{state}(\mathsf{roleA}, 3, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}, \mathsf{NB}). & \mathsf{msg}(\{\mathsf{NB}\}_{\mathsf{K(B)}}) \\[1ex]
(\mathit{intercept}) & & \mathsf{msg}(\mathsf{M}) \\
\Rightarrow & & \mathsf{i\_knows}(\mathsf{M})
\end{array}
$$

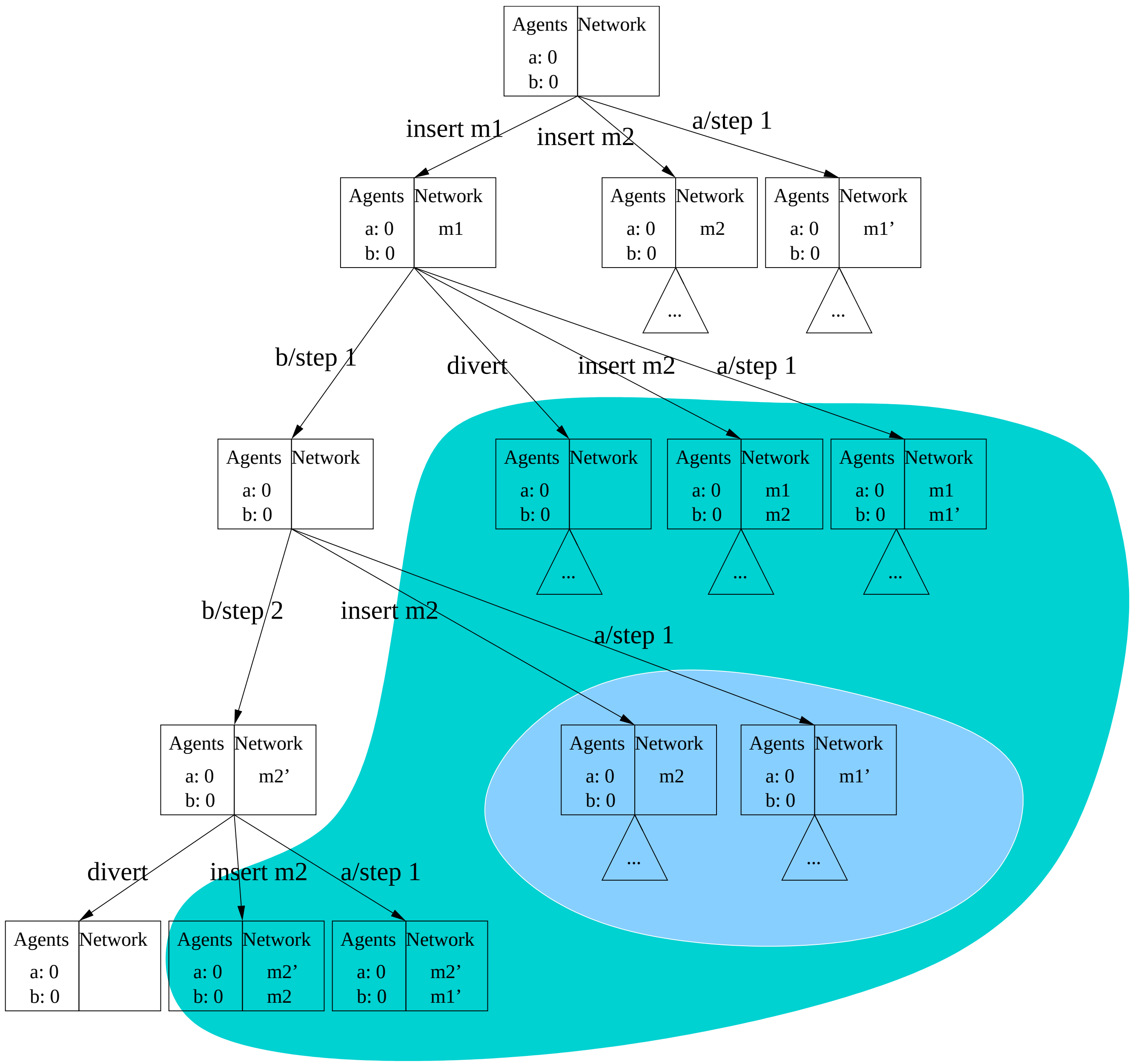
$$\downarrow$$

$$
\begin{array}{lll}
\multicolumn{3}{l}{(\mathit{compressed\ step}\ 2/3)} \\
 & \mathsf{state}(\mathsf{roleA}, 1, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}). & \mathsf{i\_knows}(\{\mathsf{NA}, \mathsf{NB}\}_{\mathsf{K(A)}}) \\
\Rightarrow & \mathsf{state}(\mathsf{roleA}, 3, \mathsf{A}, \mathsf{B}, \mathsf{SID}, \mathsf{K}, \mathsf{NA}, \mathsf{NB}). & \mathsf{i\_knows}(\{\mathsf{NB}\}_{\mathsf{K(B)}})
\end{array}
$$

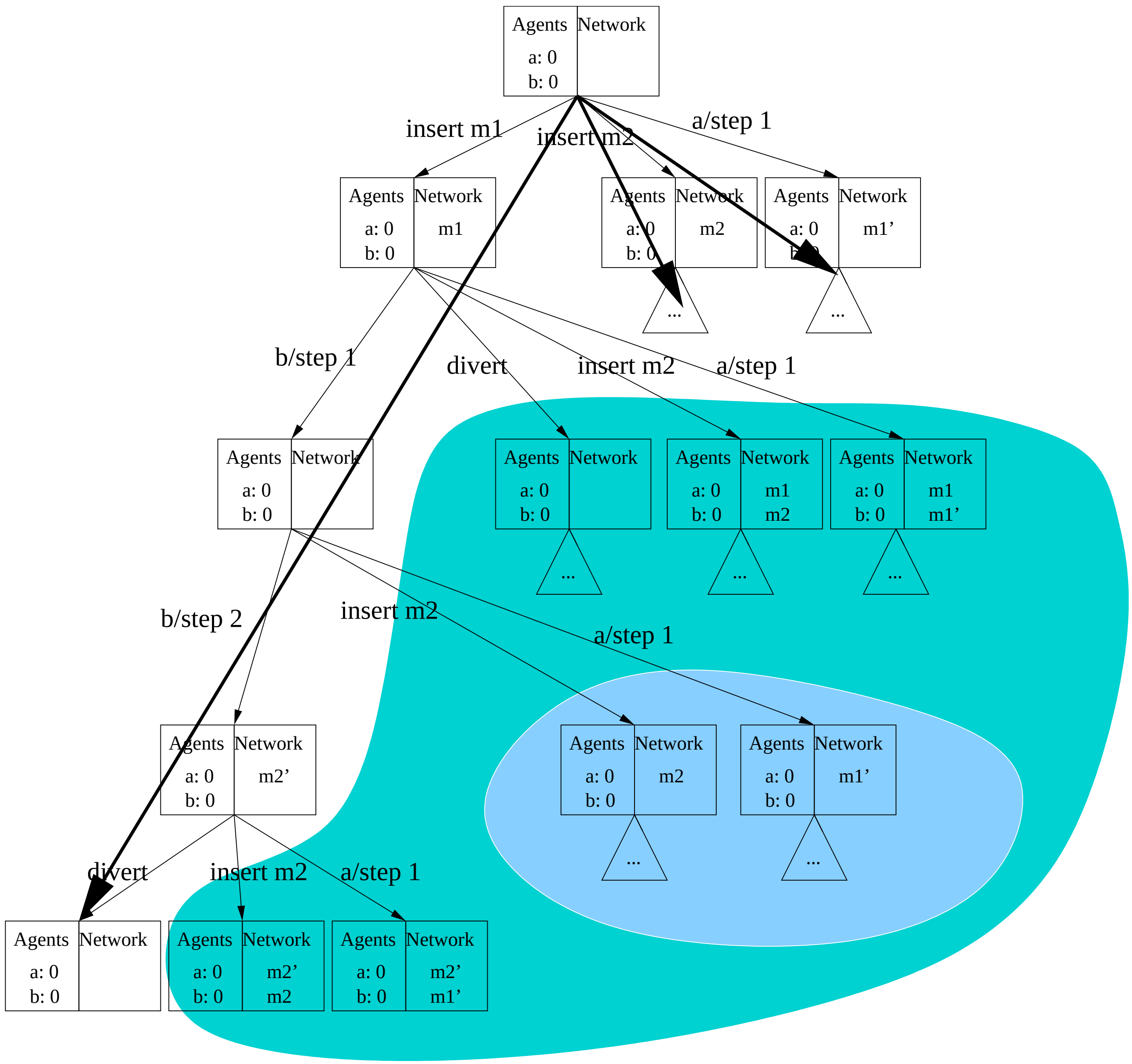
Step-Compression: Effects



Step-Compression: Effects

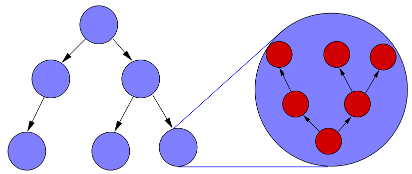


Step-Compression: Effects

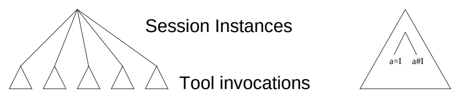


Overview

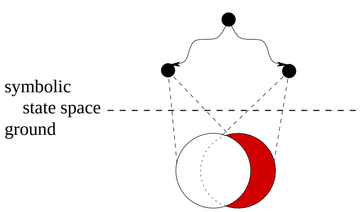
- Introduction: IF
- Compressions
- The Lazy Intruder

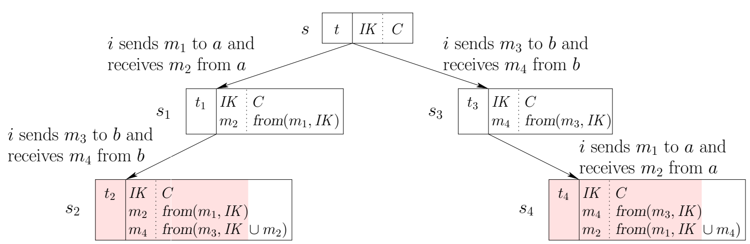


- Symbolic Sessions



- Constraint Differentiation





The Lazy Intruder: Idea

$$1.\ \mathsf{A} \rightarrow \mathsf{B} : \mathsf{M}, \mathsf{A}, \mathsf{B}, \{\!|\mathsf{N_A}, \mathsf{M}, \mathsf{A}, \mathsf{B}|\!\}_{\mathsf{K_{AS}}}$$

Which concrete value is chosen for these parts makes a difference only later.

The Lazy Intruder: Idea

$$1.\ \mathsf{A} \rightarrow \mathsf{B} : \mathsf{M}, \mathsf{A}, \mathsf{B}, \{|\mathsf{N_A}, \mathsf{M}, \mathsf{A}, \mathsf{B}|\}_{\mathsf{K_{AS}}}$$

Which concrete value is chosen for these parts makes a difference only later.

Idea: postpone this decision.

$$1.\ \mathsf{i}(\mathsf{X_2}) \rightarrow \mathsf{b} : \mathsf{X_1}, \mathsf{X_2}, \mathsf{b}, \mathsf{X_3} \quad \textit{from}(\{\mathsf{X_1}, \mathsf{X_2}, \mathsf{X_3}\}; IK)$$

IK: current Intruder Knowledge

The Lazy Intruder: Idea

$$1.\ \mathsf{A} \rightarrow \mathsf{B} : \mathsf{M}, \mathsf{A}, \mathsf{B}, \{\!|\mathsf{N_A}, \mathsf{M}, \mathsf{A}, \mathsf{B}|\!\}_{\mathsf{K_{AS}}}$$

Which concrete value is chosen for these parts makes a difference only later.

Idea: postpone this decision.

$$1.\ \mathsf{i}(\mathsf{X_2}) \rightarrow \mathsf{b} : \mathsf{X_1}, \mathsf{X_2}, \mathsf{b}, \mathsf{X_3} \quad \textit{from}(\{\mathsf{X_1}, \mathsf{X_2}, \mathsf{X_3}\}; IK)$$

IK: current Intruder Knowledge

from-constraints are evaluated in a demand-driven way,
hence lazy intruder.

Lazy Intruder: Formally

- Constraints of the lazy intruder: $\textit{from}(T; IK)$
- $[\![\textit{from}(T; IK)]\!] = \{\sigma \mid \mathrm{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$

  where $\mathcal{DY}(IK)$ is the closure of IK under Dolev-Yao rules.
- Semantics hence relates *from*-constraints to the Dolev-Yao model.

Lazy Intruder: Formally

- Constraints of the lazy intruder: $\mathit{from}(T; IK)$

- $[\![\mathit{from}(T; IK)]\!] = \{\sigma \mid \mathrm{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$

  where $\mathcal{DY}(IK)$ is the closure of IK under Dolev-Yao rules.

- Semantics hence relates *from*-constraints to the Dolev-Yao model.

- Simple constraints: the T-part contains only variables
  $\Rightarrow$ simple constraints are always satisfiable
  $\Rightarrow$ solved form for constraints.

- Calculus of reduction rules for constraints to obtain simple constraints.
  $\Rightarrow$ simple constraints are not reduced — lazy.

- Adding Inequalities

$$\mathit{from}(\mathsf{X}_1, \ldots, \mathsf{X}_\mathsf{n}; IK) \wedge t_1 \neq t_2 \wedge t_3 \neq t_4 \ldots$$

  If the inequalities are satisfiable then the entire constraint set is satisfiable.

Lazy Intruder: Formally

- Constraints of the lazy intruder: $\mathit{from}(T; IK)$

- $[\![\mathit{from}(T; IK)]\!] = \{\sigma \mid \mathrm{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$

  where $\mathcal{DY}(IK)$ is the closure of IK under Dolev-Yao rules.

- Semantics hence relates *from*-constraints to the Dolev-Yao model.

- Simple constraints: the T-part contains only variables
  $\Rightarrow$ simple constraints are always satisfiable
  $\Rightarrow$ solved form for constraints.

- Calculus of reduction rules for constraints to obtain simple constraints.
  $\Rightarrow$ simple constraints are not reduced — lazy.

- Adding Inequalities

$$\mathit{from}(\mathsf{X}_1, \ldots, \mathsf{X}_\mathsf{n}; IK) \wedge t_1 \neq t_2 \wedge t_3 \neq t_4 \ldots$$

  If the inequalities are satisfiable then the entire constraint set is satisfiable.Just choose different messages for each X_i!

Lazy Intruder: Reduction Rules

$$\frac{\mathit{from}(m_1 \cup m_2 \cup T; IK) \cup C, \sigma}{\mathit{from}(\{\!|m_2|\!\}_{m_1} \cup T; IK) \cup C, \sigma}\, G^l_{\text{scrypt}}\,,$$

$$\frac{(\mathit{from}(T; m_2 \cup IK) \cup C)\tau, \sigma\tau}{\mathit{from}(m_1 \cup T; m_2 \cup IK) \cup C, \sigma}\, G^l_{\text{unif}}\ (\tau = mgu(m_1, m_2),\ m_1 \notin \mathcal{V})\,,$$

$$\frac{\mathit{from}(m_1; IK) \cup \mathit{from}(T; m_2 \cup \{\!|m_2|\!\}_{m_1} \cup IK) \cup C, \sigma}{\mathit{from}(T; \{\!|m_2|\!\}_{m_1} \cup IK) \cup C, \sigma}\, A^l_{\text{scrypt}}\ (m_2 \notin IK)\,,$$

Lazy Intruder: An Example

The intruder knows an old message $\{|\mathsf{a}, \mathsf{b}, \mathsf{n}_1, \mathsf{n}_2|\}_{\mathsf{k(b,s)}} \in IK$, as well as the contained nonces $\mathsf{n}_1, \mathsf{n}_2 \in IK$.

Agent b expects to receive the final message of the protocol:

$$\{|\mathsf{a}, \mathsf{b}, \mathsf{KAB}|\}_{\mathsf{k(b,s)}}, \{|\mathsf{n}_2|\}_{\mathsf{KAB}}$$

$$\cfrac{\cfrac{\cfrac{\mathit{from}(\emptyset; IK)\ ; [\mathsf{KAB} \mapsto <\mathsf{n}_1, \mathsf{n}_2>]}{\mathit{from}(\mathsf{n}_1 \cup \mathsf{n}_2; IK)\ ; [\mathsf{KAB} \mapsto <\mathsf{n}_1, \mathsf{n}_2>]}\ G^l_{\mathrm{unif}}}{\mathit{from}(\{|\mathsf{n}_2|\}_{<\mathsf{n}_1,\mathsf{n}_2>}; IK)\ ; [\mathsf{KAB} \mapsto <\mathsf{n}_1, \mathsf{n}_2>]}\ G^l_{\mathrm{pair}}, G^l_{\mathrm{scrypt}}}{\mathit{from}(\{|\mathsf{a}, \mathsf{b}, \mathsf{KAB}|\}_{\mathsf{k(b,s)}} \cup \{|\mathsf{n}_2|\}_{\mathsf{KAB}}; IK)\ ; \mathrm{id}}\ G^l_{\mathrm{unif}}$$

Lazy Intruder: Completeness

- **Theorem.** Satisfiability of (well-formed) *from*-constraints is decidable.

$$\begin{array}{c} T_1 \qquad T_2 \\ \vdots \qquad\quad \vdots \\ \dfrac{t_1\tau \quad t_2\tau}{\{|t_2|\}_{t_1}\tau} \qquad T_0 \\ \nabla \qquad\qquad \vdots \\ \textit{from}(\ \{|t_2|\}_{t_1} \ \cup E_0; IK) \end{array}$$

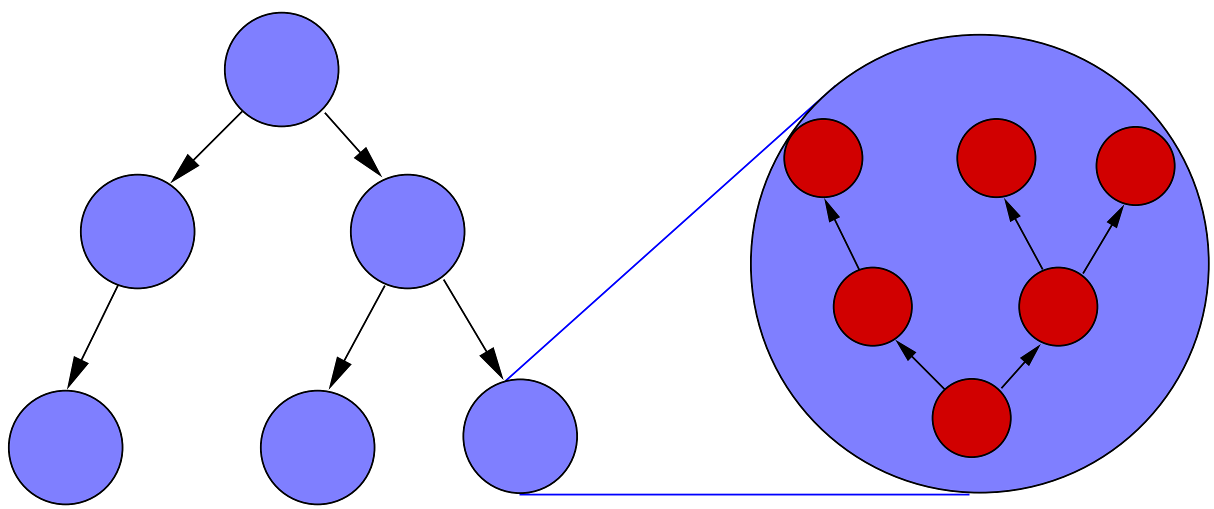
$$\downarrow_{G^l_{\text{scrypt}}}$$

$$\begin{array}{c} T_1 \qquad T_2 \\ \vdots \qquad\quad \vdots \\ t_1\tau \qquad t_2\tau \qquad T_0 \\ \nabla \qquad\quad \nabla \qquad\quad \vdots \\ \textit{from}(\ t_1 \ \cup \ t_2 \ \cup E_0\sigma; IK\sigma) \ . \end{array}$$

Integration: Symbolic Transition System

- Symbolic state = term with variables + constraint set
- $[\![(t, C)]\!] = \{t\sigma \mid \sigma \in [\![C]\!]\}$ (a set of ground states).
- Two layers of search:

  Layer 1: search in the symbolic state space
  Layer 2: constraint reduction



Lazy Intruder: History

[Huima 1999] First paper with the idea. Formalization extremely complex.

[Amadio & Lugiez 2001] Much simpler presentation of the idea and proofs.

[Rusinowitch & Turuani 2001] The insecurity problem for a bounded number of sessions is NP-complete, even without restriction to atomic keys.

[Chevalier & Vigneron 2001] First lazy intruder without the restriction to atomic keys.

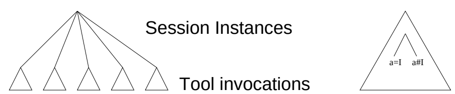
[Millen & Shmatikov 2001] Similar (independent) approach with non-atomic keys, including formal proofs.

. . .

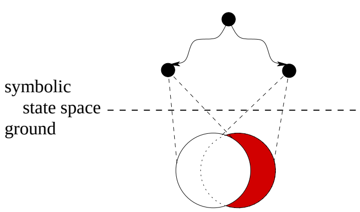
The approaches get more powerful and at the same time simpler, for instance . . .

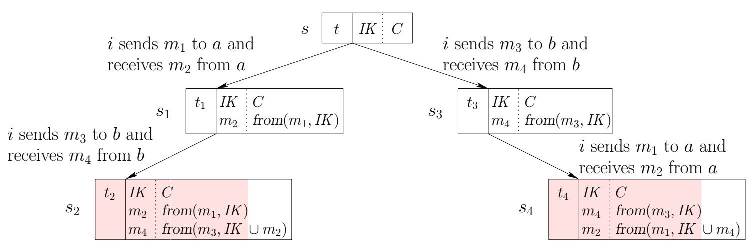
Overview

- Introduction: IF
- Compressions
- The Lazy Intruder
- Symbolic Sessions



- Constraint Differentiation





Session Instances — The Model

- Session: instantiation of all roles with an agent name.

- Example: $\begin{array}{l}[A : a, B : b]\\ [A : a, B : i]\end{array}$

  means that the initial state contains

$$\begin{array}{l}\mathsf{state}(\mathsf{roleA}, 0, \mathsf{a}, \mathsf{b})\\ \mathsf{state}(\mathsf{roleB}, 0, \mathsf{b}, \mathsf{a})\\ \mathsf{state}(\mathsf{roleA}, 0, \mathsf{a}, \mathsf{i})\\ \mathsf{state}(\mathsf{roleB}, 0, \mathsf{i}, \mathsf{a})\end{array}$$

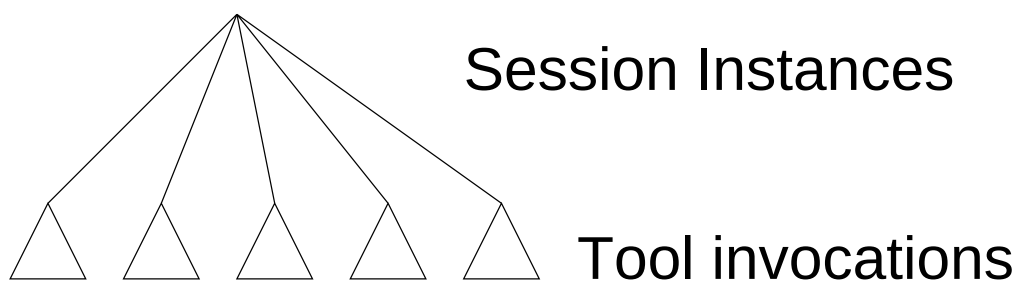
- We also call this a scenario.

Automated Session Generation

- What scenarios to examime?
- *Bouallagui et al, 2002*: given a bound n, generate all instances of n parallel sessions avoiding redundancies like isomorph instances.
- E.g. $n = 2$:

$$\begin{array}{llll} [\mathsf{A}:\mathsf{a},\mathsf{B}:\mathsf{b}] & [\mathsf{A}:\mathsf{a},\mathsf{B}:\mathsf{b}] & [\mathsf{A}:\mathsf{a},\mathsf{B}:\mathsf{b}] & \\ [\mathsf{A}:\mathsf{a},\mathsf{B}:\mathsf{b}] & [\mathsf{A}:\mathsf{a},\mathsf{B}:\mathsf{i}] & [\mathsf{A}:\mathsf{b},\mathsf{B}:\mathsf{a}] & \ldots \end{array}$$

- Parameter Search:



Symbolic Sessions

- Let the lazy intruder take care of the instantiation problem.

- Ag is the set of agent names.

- Initial state containts variables ranging over Ag, e.g.:

$$\begin{array}{ll} \mathsf{state}(\mathsf{roleA}, 0, \mathsf{A}_1, \mathsf{B}_1) & \mathsf{A}_1 \neq \mathsf{i} \\ \mathsf{state}(\mathsf{roleB}, 0, \mathsf{B}'_1, \mathsf{A}'_1) & \mathsf{B}'_1 \neq \mathsf{i} \end{array}$$

- Constraint sets must be well-formed, in particular, all variables must be bound by a constraint.

- $\{\mathit{from}(\mathsf{A}_i; IK_0)\}$ where A_i are the variables occuring in the initial state.

- We therefore assume the intruder initially knows all agent names: $\mathsf{Ag} \subseteq IK_0$.

$\Rightarrow$ The agent names are chosen by the intruder as well.

Symbolic Sessions

- Let the lazy intruder take care of the instantiation problem.
- Ag is the set of agent names.
- Initial state containts variables ranging over Ag, e.g.:

$$\begin{array}{ll} \mathsf{state}(\mathsf{roleA}, 0, \mathsf{A}_1, \mathsf{B}_1) & \mathsf{A}_1 \neq \mathsf{i} \\ \mathsf{state}(\mathsf{roleB}, 0, \mathsf{B}'_1, \mathsf{A}'_1) & \mathsf{B}'_1 \neq \mathsf{i} \end{array}$$

- Constraint sets must be well-formed, in particular, all variables must be bound by a constraint.
- $\{\mathit{from}(\mathsf{A}_i; IK_0)\}$ where A_i are the variables occuring in the initial state.
- We therefore assume the intruder initially knows all agent names: $\mathsf{Ag} \subseteq IK_0$.

$\Rightarrow$ The agent names are chosen by the intruder as well. Lazily.

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$$\begin{array}{ccc} \boxed{\mathsf{A} : \mathsf{A}_1} & \longrightarrow & \mathsf{B} : \mathsf{B}_1 \\ \mathsf{A} : \mathsf{A}_2 & \longrightarrow & \boxed{\mathsf{B} : \mathsf{B}_2} \end{array} \qquad \boxed{IK_0}$$

$$\mathsf{A}_1 \neq \mathsf{i} \quad \mathsf{B}_2 \neq \mathsf{i}$$

Trace:

$$1. \quad \mathsf{A}_1 \rightarrow \mathsf{i}(\mathsf{B}_1) : \quad \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k}(\mathsf{B}_1)}$$

$\Rightarrow$ The intruder can read n_1 iff $\mathsf{B}_1 = \mathsf{i}$.
$\Rightarrow$ Case split $\mathsf{B}_1 = \mathsf{i}$ and $\mathsf{B}_1 \neq \mathsf{i}$.
Let's follow the case $\mathsf{B}_1 = \mathsf{i}$.

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$\boxed{A : A_1} \longrightarrow B : i$

$A : A_2 \longrightarrow \boxed{B : B_2}$

$A_1 \neq i \quad B_2 \neq i$

$\boxed{IK_0}$

$\{n_1, A_1\}_{k(i)} \quad n_1 \quad IK_1$

Trace:

$$1. \quad A_1 \longrightarrow i(i) : \quad \{n_1, A_1\}_{k(i)}$$

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$$\begin{array}{ccc} \boxed{\mathsf{A} : \mathsf{A}_1} & \longrightarrow & \mathsf{B} : \mathsf{i} \\ \mathsf{A} : \mathsf{A}_2 & \longrightarrow & \boxed{\mathsf{B} : \mathsf{B}_2} \end{array}$$

$$\mathsf{A}_1 \neq \mathsf{i} \quad \mathsf{B}_2 \neq \mathsf{i}$$

$$\boxed{IK_0}$$

$$\{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k(i)}} \quad \mathsf{n}_1 \quad IK_1$$

Trace:

$$\begin{array}{lll} 1. & \mathsf{A}_1 \rightarrow \mathsf{i} : & \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k(i)}} \\ 1.' & \mathsf{i}(\mathsf{A}_2) \rightarrow \mathsf{B}_2 : & \{\mathsf{NA}, \mathsf{A}_2\}_{\mathsf{k}(\mathsf{B}_2)} \end{array}$$

with the new constraint store

$$\begin{array}{l} \mathit{from}(\mathsf{A}_1, \mathsf{A}_2, \mathsf{B}_2; IK_0) \\ \mathit{from}(\{\mathsf{NA}, \mathsf{A}_2\}_{\mathsf{k}(\mathsf{B}_2)}; IK_1) \end{array}$$

Solutions: either replay old messages or generate the new message from subterms:

$$\mathit{from}(\mathsf{k} \cup \mathsf{B}_2 \cup \mathsf{A}_2 \cup \mathsf{NA}; IK_1)$$

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$$\begin{array}{lcl} \boxed{\mathsf{A : A_1}} & \longrightarrow & \mathsf{B : i} \\ \mathsf{A : A_2} & \longrightarrow & \boxed{\mathsf{B : B_2}} \end{array}$$

$$\mathsf{A_1} \neq \mathsf{i} \quad \mathsf{B_2} \neq \mathsf{i}$$

$$\begin{array}{lll} \boxed{IK_0} & & \\ \{\mathsf{n_1, A_1}\}_{\mathsf{k(i)}} & \mathsf{n_1} & IK_1 \\ \{\mathsf{NA, n_2}\}_{\mathsf{k(A_2)}} & & IK_2 \end{array}$$

Trace:

$$\begin{array}{lll} 1. & \mathsf{A_1} \rightarrow \mathsf{i} : & \{\mathsf{n_1, A_1}\}_{\mathsf{k(i)}} \\ 1.' & \mathsf{i(A_2)} \rightarrow \mathsf{B_2} : & \{\mathsf{NA, A_2}\}_{\mathsf{k(B_2)}} \\ 2.' & \mathsf{B_2} \rightarrow \mathsf{i(A_2)} : & \{\mathsf{NA, n_2}\}_{\mathsf{k(A_2)}} \end{array}$$

Again, the intruder can decrypt this message iff he is the intended recipient, i.e. iff $\mathsf{A_2} = \mathsf{i}$.

Let's follow the case $\mathsf{A_2} \neq \mathsf{i}$ here.

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$\boxed{\mathsf{A} : \mathsf{A}_1} \longrightarrow \mathsf{B} : \mathsf{i}$

$\mathsf{A} : \mathsf{A}_2 \longrightarrow \boxed{\mathsf{B} : \mathsf{B}_2}$

$\mathsf{A}_1 \neq \mathsf{i} \quad \mathsf{B}_2 \neq \mathsf{i} \quad \mathsf{A}_2 \neq \mathsf{i}$

$\boxed{IK_0}$

$\{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k}(\mathsf{i})} \quad \mathsf{n}_1 \quad IK_1$

$\{\mathsf{NA}, \mathsf{n}_2\}_{\mathsf{k}(\mathsf{A}_2)} \quad IK_2$

Trace:

$$\begin{array}{lll} 1. & \mathsf{A}_1 \rightarrow \mathsf{i} : & \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k}(\mathsf{i})} \\ 1.' & \mathsf{i}(\mathsf{A}_2) \rightarrow \mathsf{B}_2 : & \{\mathsf{NA}, \mathsf{A}_2\}_{\mathsf{k}(\mathsf{B}_2)} \\ 2.' & \mathsf{B}_2 \rightarrow \mathsf{i}_{\mathsf{A}_2} : & \{\mathsf{NA}, \mathsf{n}_2\}_{\mathsf{k}(\mathsf{A}_2)} \\ 2. & \mathsf{i} \rightarrow \mathsf{A}_1 : & \{\mathsf{n}_1, \mathsf{NB}\}_{\mathsf{k}(\mathsf{A}_1)} \end{array}$$

with the new constraint store

$$\begin{array}{l} \mathit{from}(\mathsf{A}_1, \mathsf{A}_2, \mathsf{B}_2; IK_0) \\ \mathit{from}(\mathsf{NA}; IK_1) \\ \mathit{from}(\{\mathsf{n}_1, \mathsf{NB}\}_{\mathsf{k}(\mathsf{A}_1)}; IK_2) \end{array}$$

Solutions: generate the new message from its subterms, or replay an old one:

$$\begin{array}{rl} & \{\mathsf{n}_1, \mathsf{NB}\}_{\mathsf{k}(\mathsf{A}_1)} = \{\mathsf{NA}, \mathsf{n}_2\}_{\mathsf{k}(\mathsf{A}_2)} \\ \Rightarrow & \mathsf{A}_1 = \mathsf{A}_2,\ \mathsf{n}_1 = \mathsf{NA},\ \mathsf{NB} = \mathsf{n}_2 \end{array}$$

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$\boxed{\mathsf{A} : \mathsf{A}_1} \longrightarrow \mathsf{B} : \mathsf{i}$

$\mathsf{A} : \mathsf{A}_1 \longrightarrow \boxed{\mathsf{B} : \mathsf{B}_2}$

$\mathsf{A}_1 \neq \mathsf{i} \quad \mathsf{B}_2 \neq \mathsf{i}$

$\boxed{IK_0}$

$\{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k}(\mathsf{i})} \quad \mathsf{n}_1 \quad IK_1$

$\{\mathsf{n}_1, \mathsf{n}_2\}_{\mathsf{k}(\mathsf{A}_1)} \quad IK_2$

Trace:

$$\begin{array}{lll}
1. & \mathsf{A}_1 \longrightarrow \mathsf{i} : & \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k}(\mathsf{i})} \\
1.' & \mathsf{i}(\mathsf{A}_1) \longrightarrow \mathsf{B}_2 : & \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k}(\mathsf{B}_2)} \\
2.' & \mathsf{B}_2 \longrightarrow \mathsf{i}(\mathsf{A}_1) : & \{\mathsf{n}_1, \mathsf{n}_2\}_{\mathsf{k}(\mathsf{A}_1)} \\
2. & \mathsf{i} \longrightarrow \mathsf{A}_1 : & \{\mathsf{n}_1, \mathsf{n}_2\}_{\mathsf{k}(\mathsf{A}_1)}
\end{array}$$

with the new constraint store

$$\begin{array}{l}
\mathit{from}(\mathsf{A}_1, \mathsf{B}_2; IK_0) \\
\mathit{from}(\mathsf{n}_1; IK_1) \\
\mathit{from}(\mathsf{n}_1; IK_2)
\end{array}$$

The answer from A_1 is $\{\mathsf{n}_2\}_{\mathsf{k}(\mathsf{i})}$, so the intruder now knows n_2 and can finish the protocol with B_2.

Symbolic Sessions: Example

```
1. A -> B: {NA,A}KB
2. B -> A: {NA,NB}KA
3. A -> B: {NB}KB
```

$\boxed{\mathsf{A} : \mathsf{A}_1} \longrightarrow \mathsf{B} : \mathsf{i}$

$\mathsf{A} : \mathsf{A}_1 \longrightarrow \boxed{\mathsf{B} : \mathsf{B}_2}$

$\mathsf{A}_1 \neq \mathsf{i} \quad \mathsf{B}_2 \neq \mathsf{i}$

$\boxed{IK_0}$

$\{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k(i)}}$ n_1 IK_1

$\{\mathsf{n}_1, \mathsf{n}_2\}_{\mathsf{k(A_1)}}$ IK_2

$\{\mathsf{n}_2\}_{\mathsf{k(i)}}$ n_2 IK_3

Trace:

$$
\begin{array}{lll}
1. & \mathsf{A}_1 \longrightarrow \mathsf{i} : & \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k(i)}} \\
1.' & \mathsf{i(A_1)} \longrightarrow \mathsf{B}_2 : & \{\mathsf{n}_1, \mathsf{A}_1\}_{\mathsf{k(B_2)}} \\
2.' & \mathsf{B}_2 \longrightarrow \mathsf{i(A_1)} : & \{\mathsf{n}_1, \mathsf{n}_2\}_{\mathsf{k(A_1)}} \\
2. & \mathsf{i} \longrightarrow \mathsf{A}_1 : & \{\mathsf{n}_1, \mathsf{n}_2\}_{\mathsf{k(A_1)}} \\
3. & \mathsf{A}_1 \longrightarrow \mathsf{i} : & \{\mathsf{n}_2\}_{\mathsf{k(i)}} \\
3.' & \mathsf{i(A_1)} \longrightarrow \mathsf{B}_2 : & \{\mathsf{n}_2\}_{\mathsf{k(b)}}
\end{array}
$$

where A_1 and B_2 are arbitrary honest agents.

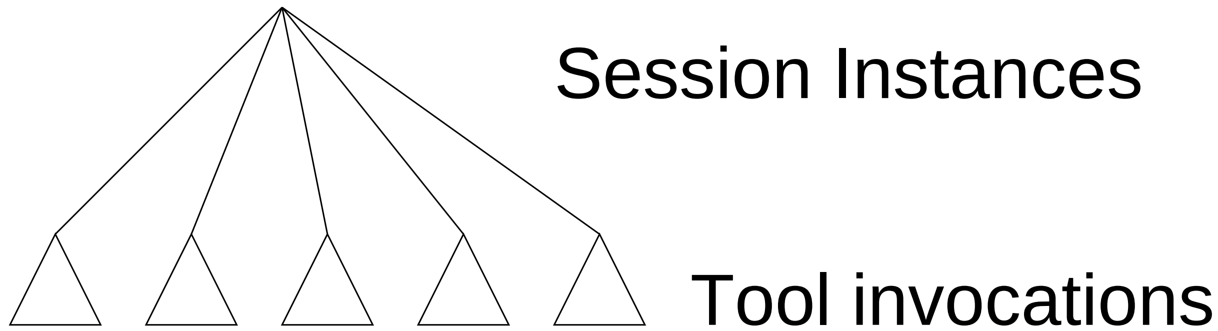
Two agents are sufficient

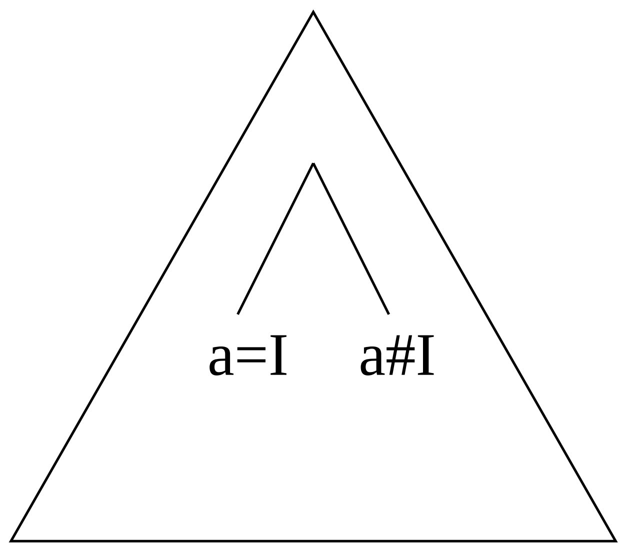
All substitutions for the variables for agent names are substituted either

- with each other (e.g. $A_1 = A_2$)
- or with the intruder (e.g. $B_1 = i$).

Thus: isn't it sufficient to have only two agents, alice and intruder?

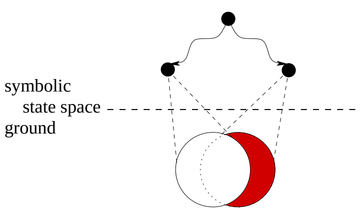
Intuition

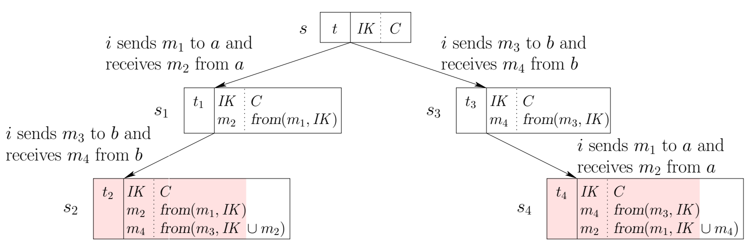




Overview

- Introduction: IF
- Compressions
- The Lazy Intruder
- Symbolic Sessions
- Constraint Differentiation





Two Key Challenges and their Solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.

2. Concurrency: number of parallel sessions executed by honest agents.

Two Key Challenges and their Solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.
   - No bound on the messages the intruder can compose.
   - Lazy Intruder: symbolic representation of intruder.
     "Often just as if there were no intruder!"

2. Concurrency: number of parallel sessions executed by honest agents.

Two Key Challenges and their Solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.
   - No bound on the messages the intruder can compose.
   - Lazy Intruder: symbolic representation of intruder.
     "Often just as if there were no intruder!"

2. Concurrency: number of parallel sessions executed by honest agents.
   - Often addressed using Partial-Order Reduction (POR).
   - POR is limited when using the lazy intruder technique.
   - Constraint Differentiation: general, POR-inspired reduction technique extending the lazy intruder.

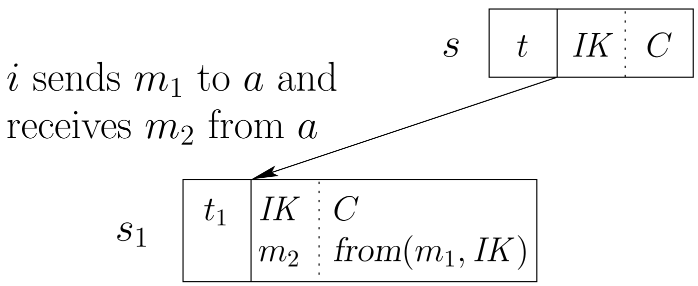
Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:

s	t	IK	C

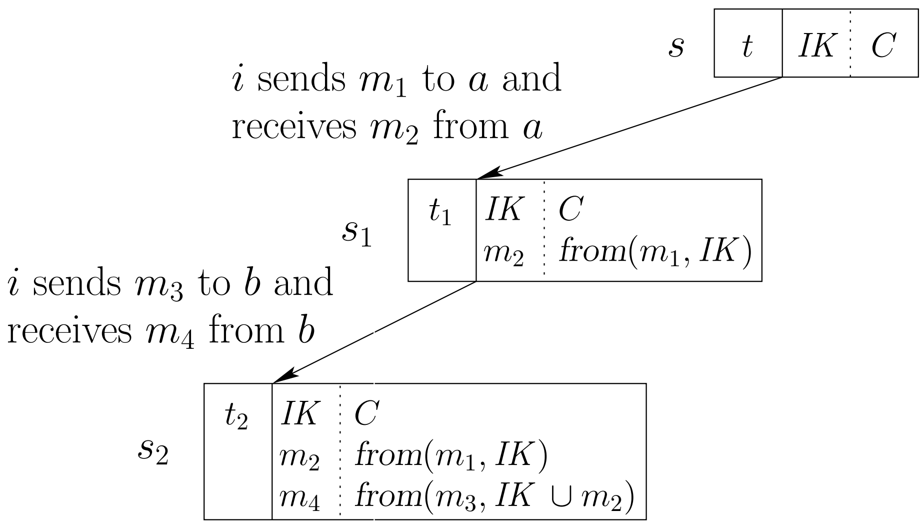
Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



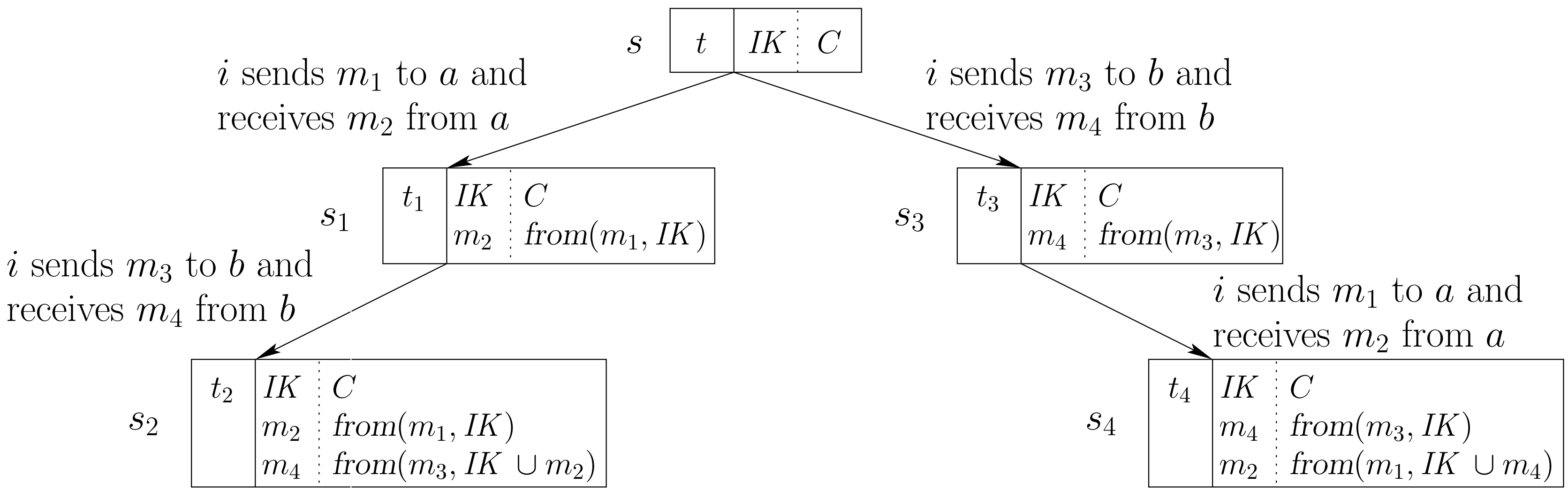
Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



Constraint Differentiation: Idea

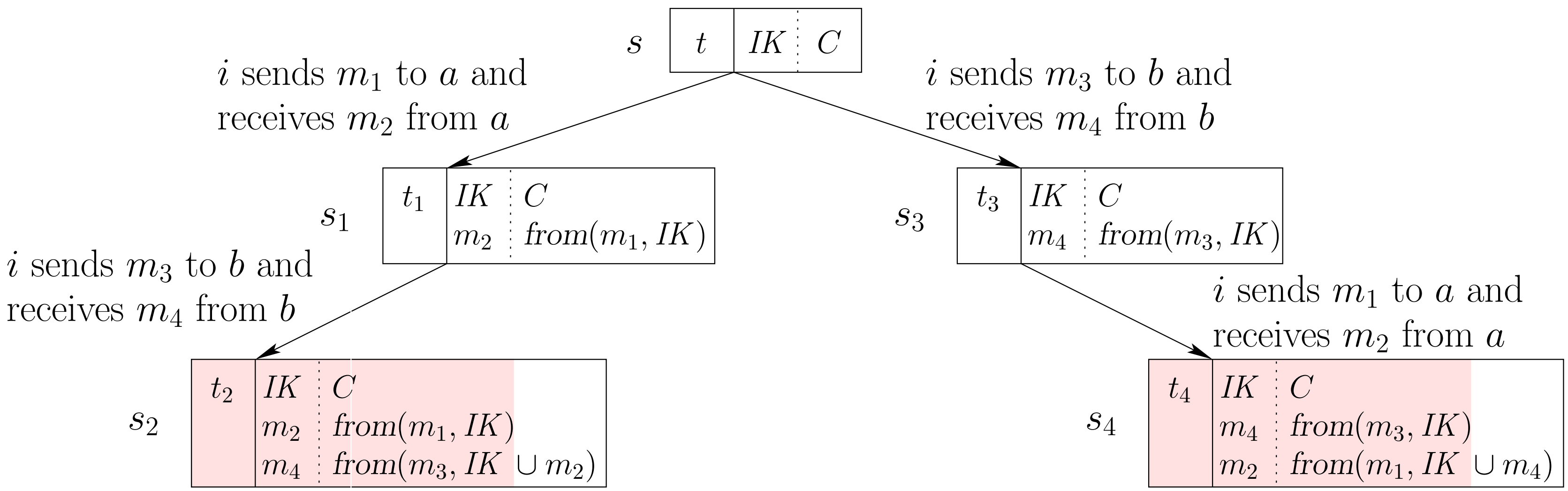
Typical situation: 2 independent actions executable in either order:



(where $t_2 = t_4$)

Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:

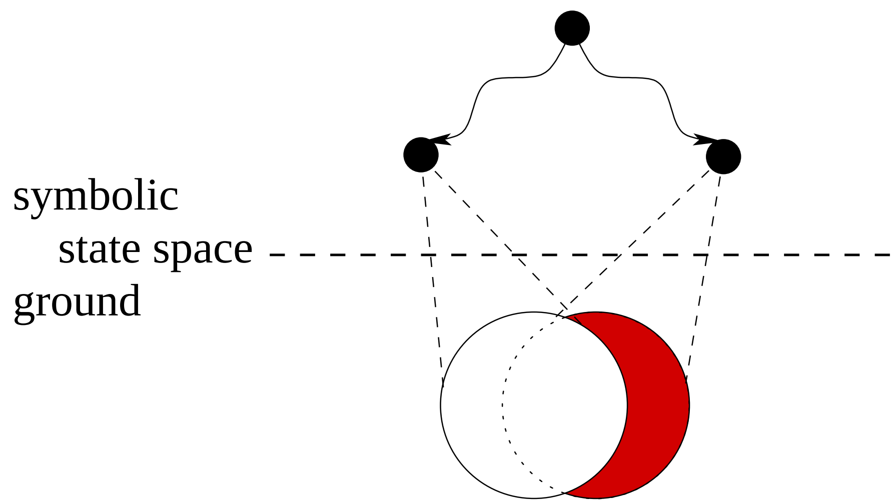


(where $t_2 = t_4$)

Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.

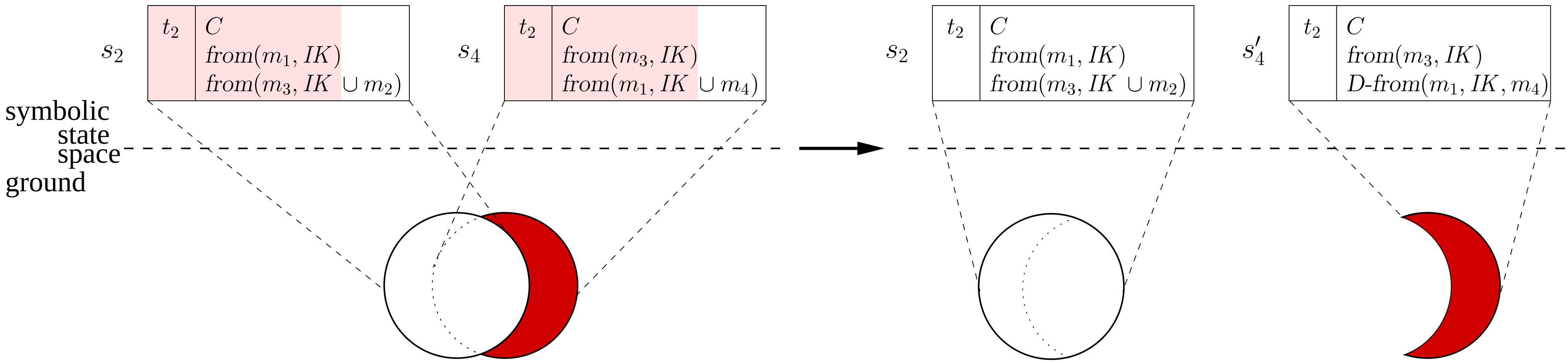
Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



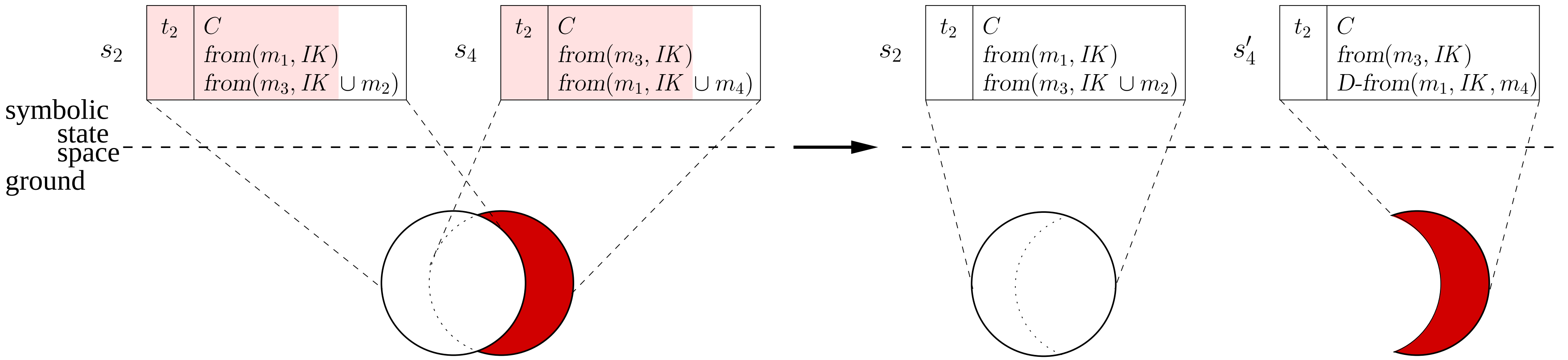
Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.

Constraint Differentiation (1)



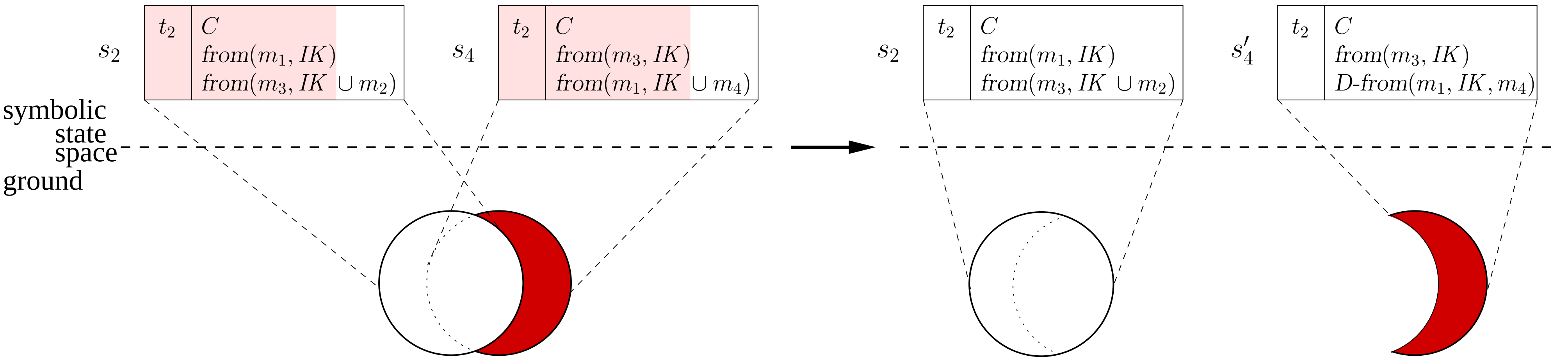
- New kind of constraints: $D\text{-}from(T; IK; NIK)$.
- Intuition:
  - Intruder has just learned some new intruder knowledge NIK.

Constraint Differentiation (1)



- New kind of constraints: *D-from*$(T; IK; NIK)$.
- Intuition:
  - Intruder has just learned some new intruder knowledge NIK.
  - All solutions $[\![\mathit{from}(T; IK \cup NIK)]\!]$ are "correct"

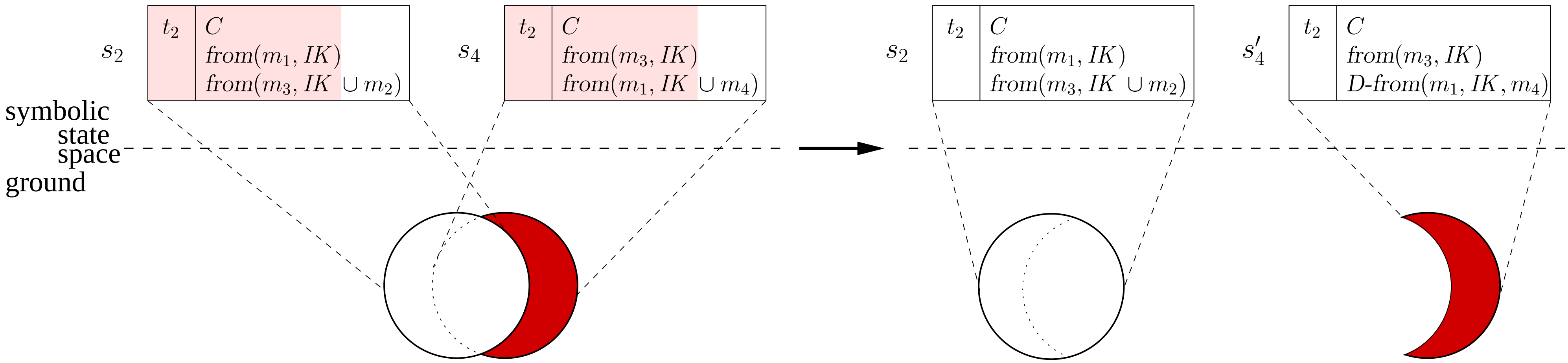
Constraint Differentiation (1)



- New kind of constraints: *D-from*$(T; IK; NIK)$.
- Intuition:
  - Intruder has just learned some new intruder knowledge NIK.
  - All solutions $[\![\textit{from}(T; IK \cup NIK)]\!]$ are "correct" but a solution is interesting only if it requires NIK.

$$[\![\textit{D-from}(T; IK; NIK)]\!] = [\![\textit{from}(T; IK \cup NIK)]\!] \setminus [\![\textit{from}(T; IK)]\!].$$

Constraint Differentiation (2)



- $[\![D\text{-}from(T; IK; NIK)]\!] = [\![from(T; IK \cup NIK)]\!] \setminus [\![from(T; IK)]\!]$
- **Theorem.** *Satisfiability of (well-formed) D-from constraints is decidable.*
- **Theorem.** $[\![s_2]\!] \cup [\![s_4]\!] = [\![s_2]\!] \cup [\![s_4']\!]$

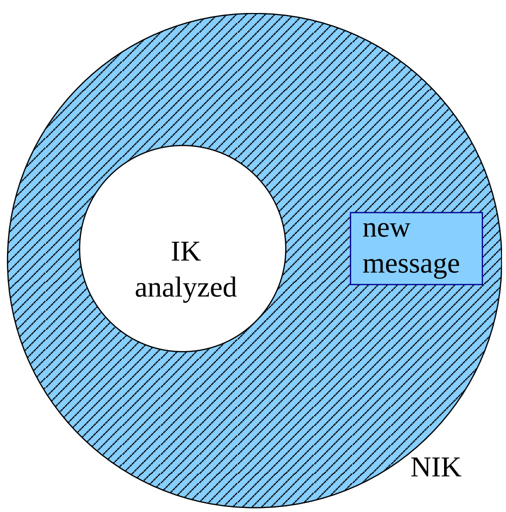
Constraint Differentiation: Reduction Rules

$$\frac{\textit{D-from}(m_1 \cup m_2 \cup M; IK; NIK) \cup C, \sigma}{\textit{D-from}(\{\!|m_1|\!\}_{m_2} \cup M; IK; NIK) \cup C, \sigma} G^{LD}_{\text{scrypt}} ,$$

$$\frac{(\textit{D-from}(M; m_2 \cup IK; NIK) \cup C)\tau, \sigma\tau}{\textit{D-from}(m_1 \cup M; m_2 \cup IK; NIK) \cup C, \sigma} G^{LD}_{\text{unif1}} \; (\tau = mgu(m_1, m_2), m_1 \notin \mathcal{V}) ,$$

$$\frac{(\textit{from}(M; m_2 \cup IK \cup NIK) \cup C)\tau, \sigma\tau}{\textit{D-from}(m_1 \cup M; IK; m_2 \cup NIK) \cup C, \sigma} G^{LD}_{\text{unif2}} \; (\tau = mgu(m_1, m_2), m_1 \notin \mathcal{V}) ,$$

Analysis: either specialized rules,



analysis of IK and new message

. . . or by normalization:

Conclusions

- Introduction: IF
  Simple, powerful formalism to describe protocols and intruder.

- Compressions
  We can optimize specifications by compressing rules.

- The Lazy Intruder
  Efficient representation of the prolific Dolev-Yao intruder.

- Symbolic Sessions
  Leaving the instantiation problem to the intruder.

- Constraint Differentiation
  Removing redundancies by "POR for the lazy intruder".