Protokollsicherheit

Pi Kalkül, Horn Klauseln, Korrespondenzeigenschaften

Der Pi Kalkül in Proverif

Proverif verwendet ein eingeschränktes Fragment des angewandten Pi-Kalküls.

Grund: Kosten der Unifikation modulo Gleichheit bezüglich einer Gleichungstheorie.

Terme:

$$M, N ::= a, b, c, \ldots, m, n, k, \ldots \mid x, y, z, \ldots \mid f(M_1, \ldots, M_n)$$

Prozesse:

$$P, Q ::= 0 \mid u(x).P \mid \overline{u}\langle M \rangle.P \mid \nu a.P \mid P|Q \mid !P \mid \text{if } M = N \text{ then } P \text{ else } Q$$
$$\mid \text{let } x = M \text{ in } P \mid \text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q$$

Der Pi Kalkül in Proverif

Anstelle einer Gleichungstheorie werden **Konstruktoren** und **Destruktoren** benutzt.

Konstruktoren können in den Termen vorkommen. Es gibt keine Gleichungen für Konstruktoren.

Destruktoren können nur in der let-Konstruktion vorkommen:

$$\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q$$

Die Bedeutung jedes Destruktors g wird durch (endlich viele) Reduktionsregeln der Form

$$g(M_1, \ldots, M_n) \to M$$

festgelegt.

Der Pi Kalkül in Proverif

Reduktionssemantik:

Wenn $g(M_1, \ldots, M_n) \to M$, dann:

$$\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q \longrightarrow P\{M/x\}$$

Wenn es kein M mit $g(M_1, \ldots, M_n) \to M$ gibt, dann:

$$\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q \longrightarrow Q$$

Erzeugung von Horn Klauseln

Wir wollen aus einer gegebenen Prozessspezifikation im Pi Kalkül Klauseln erzeugen, um zu überprüfen, dass ein Geheimnis nicht über einen öffentlichen Kanal geschickt wird.

Patterns:

$$p, q ::= x \mid a[p_1, \ldots, p_n] \mid f(p_1, \ldots, p_n)$$

Wir benutzen zwei Prädikate:

- $\mathsf{att}(p)$ — Der Angreifer könnte Nachricht p bekommen.
- $\mathsf{mess}(p_1, p_2)$ — Die Nachricht p_2 könnte über den Kanal p_1 geschickt werden.

Klauseln für den Angreifer

Für jeden Konstruktor $f(M_1, \ldots, M_n)$:

$$\mathsf{att}(x_1) \wedge \cdots \wedge \mathsf{att}(x_n) \supset \mathsf{att}(f(x_1, \ldots, x_n))$$

Für jede Destruktor-Reduktionsregel $g(M_1, \ldots, M_n) \to M$:

$$\mathsf{att}(M_1) \wedge \cdots \wedge \mathsf{att}(M_n) \supset \mathsf{att}(M)$$

Der Angreifer kann Namen erzeugen:

$$\supset \mathsf{att}(b_0[])$$

Nachrichten:

$$\mathsf{mess}(x, y) \wedge \mathsf{att}(x) \supset \mathsf{att}(y)$$

$$\mathsf{att}(x) \wedge \mathsf{att}(y) \supset \mathsf{mess}(x, y)$$

Anfangswissen (Menge S von Namen):

$$\supset \mathsf{att}(a[]) \quad \text{für jeden Namen } a \in S$$

Klauseln für das Protokoll

$$
\begin{aligned}
[\![0]\!]\rho h &= \emptyset \\
[\![P \mid Q]\!]\rho h &= [\![P]\!]\rho h \cup [\![Q]\!]\rho h \\
[\![!P]\!]\rho h &= [\![P]\!]\rho h \\
[\![\nu a.\, P]\!]\rho h &= [\![P]\!](\rho[a \mapsto a[p_1, \ldots, p_n]])h \\
&\qquad \text{wenn } h = \mathsf{mess}(q_1, p_1) \wedge \cdots \wedge \mathsf{mess}(q_n, p_n) \\
[\![M(x).P]\!]\rho h &= [\![P]\!][\rho[x \mapsto x']](h \wedge \mathsf{mess}(M\rho, x')) \;\; (x' \text{ frisch}) \\
[\![\overline{M}\langle N\rangle.P]\!]\rho h &= [\![P]\!]\rho h \cup \{h \supset \mathsf{mess}(M\rho, N\rho)\}
\end{aligned}
$$

ρ bildet Variablen und Namen auf Patterns ab.
h ist eine Konjunktion von Annahmen der Form $\mathsf{mess}(q, p)$. Der Prozess kann nur zur Ausführung kommen, wenn h erfüllt ist.

Klauseln für das Protokoll

$[\![\text{let } x = M \text{ in } P]\!]\rho h = [\![P]\!](\rho[x \mapsto M\rho])h$

$[\![\text{let } x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q]\!]\rho h =$
$\bigcup\{[\![P]\!]((\rho\sigma)[x \mapsto p\sigma'])(h\sigma) \mid g(p_1, \ldots, p_n) \to p$ ist Red.-Regel und (σ, σ') ist das allgemeinste Substitutionspaar mit $M_1\rho\sigma = p_1\sigma'$, …, $M_n\rho\sigma = p_n\sigma'\ \} \cup [\![Q]\!]\rho h$.

$[\![\text{if } M = N \text{ then } P \text{ else } Q]\!]\rho h = [\![P]\!](\rho\sigma)(h\sigma) \cup [\![Q]\!]\rho h$
wenn M und N unifizierbar sind und σ der allgemeinste Unifikator von M und N ist.

$[\![\text{if } M = N \text{ then } P \text{ else } Q]\!]\rho h = [\![Q]\!]\rho h$
wenn M und N nicht unifizierbar sind.

Beispiel: Vereinfachtes Denning-Sacco Protokoll

1. $A \to B$: $\{\{k\}_{\mathsf{sk}(B)}\}_{\mathsf{pk}(B)}$
2. $B \to A$: $\{\mathsf{secret}\}_k$

```
let processA(skA) =
        in(c, pkB);
        new k;  out(c, encrypt(sign(k, skA), pkB));
        in(c, x);  let s = decrypt(x, k) in 0.
let processB(skB, pkA) =
        in(c, km);
        let ks = decrypt(km, skB) in
        let k = checksign(ks, pkA) in
        out(c, sencrypt(secret, k)).
process new skA; let pkA = pk(skA) in
        new skB; let pkB = pk(skB) in
        out(c, pkA); out(c, pkB);
        (!processA(skA)) | (!processB(skB, pkA))
```

```
let processB(skB, pkA) =
      in(c, km);
      let ks = decrypt(km, skB) in
      let k = checksign(ks, pkA) in
      out(c, sencrypt(secret, k)).
```

$[\![M(x).P]\!]\rho h = [\![P]\!][\rho[x \mapsto x']](h \wedge \mathsf{mess}(M\rho, x'))$ (x' frisch)

$[\![\overline{M}\langle N\rangle.P]\!]\rho h = [\![P]\!]\rho h \cup \{h \supset \mathsf{mess}(M\rho, N\rho)\}$

$[\![\mathsf{let}\ x = g(M_1, \ldots, M_n)\ \mathsf{in}\ P\ \mathsf{else}\ Q]\!]\rho h =$

$\bigcup\{[\![P]\!]((\rho\sigma)[x \mapsto p\sigma'])(h\sigma) \mid g(p_1, \ldots, p_n) \to p$ ist Red.-Regel, (σ, σ') allgemeinstmöglich für $M_1\rho\sigma = p_1\sigma', \ldots, M_n\rho\sigma = p_n\sigma'\ \} \cup [\![Q]\!]\rho h$.

Beispiel: Vereinfachtes Denning-Sacco Protokoll

1. $A \to B$: $\{\{k\}_{\mathsf{sk}(B)}\}_{\mathsf{pk}(B)}$
2. $B \to A$: $\{\mathsf{secret}\}_k$

$\mathsf{mess}(c[\,], pkB) \supset \mathsf{mess}(c[\,], \mathsf{encrypt}(\mathsf{sign}(k[pkB], skA[\,]), pkB))\}$

```
let proc
        in(c, pkB);
        new k;  out(c, encrypt(sign(k, skA), pkB));
        in(c, x);  let s = decrypt(x, k) in 0.
```

$\mathsf{mess}(c[\,], \mathsf{encrypt}(\mathsf{sign}(k, skA[\,]), \mathsf{pk}(skB[\,])))\} \supset \mathsf{mess}(c[\,], \mathsf{sencrypt}(\mathsf{secret}, k))$

```
        in(c, km);
        let ks = decrypt(km, skB) in
        let k = checksign(ks, pkA) in
        out(c, sencrypt(secret, k)).
process new skA; let pkA = pk(skA) in
        new skB; let pkB =
        out(c, pkA); out(c
        (!processA(skA)) | (!processB(skB, pkA))
```

$\mathsf{mess}(c[\,], \mathsf{pk}(skA[\,])), \mathsf{mess}(c[\,], \mathsf{pk}(skB[\,]))$

Korrektheit

Wenn $\texttt{att}(s)$ aus den Klauseln eines Prozesses (sowie den allgemeinen Klauseln für den Angreifer) nicht hergeleitet werden kann, dann bleibt s in P geheim.

(D.h. es gibt keinen Lauf von P in Gegenwart eines Angreifers, in dem s über einen öffentlichen Kanal geschickt wird.)

Korrespondenzeigenschaften

Prozesse mit Ereignissen:

$$P, Q ::= \cdots \mid \mathsf{event}(M).P$$

Im Prinzip können wir nun ein neues Prädikat $\mathsf{event}(p)$ zu den Klauseln hinzunehmen und damit zeigen, ob ein Ereignis möglich ist.

Korrespondenzeigenschaften

$$\mathsf{event}(e_1(x)) \rightsquigarrow \mathsf{event}(e_2(x))$$

(wenn das Ereignis $e_1(M)$ passiert, dann auch $e_2(M)$) können so nicht direkt ausgedrückt werden.

Korrespondenzeigenschaften

Korrespondenzeigenschaften

$$\mathsf{event}(e_1(x)) \rightsquigarrow \mathsf{event}(e_2(x))$$

(wenn das Ereignis $e_1(M)$ passiert, dann auch $e_2(M)$) können so nicht direkt ausgedrückt werden.

Grund: Die Horn-Klauseln liefern eine *Überapproximation* der Ereignisse.
Man kann nicht folgern, dass $\mathsf{event}(e_2(x))$ wirklich passieren muss, oder ob er nur aus Gründen der Überapproximation herleitbar ist.

Beispiel: $\overline{a}\langle M\rangle.0 \mid a(x).\mathsf{event}(e_1).0 \mid a(x).\mathsf{event}(e_2).0$
Die naive Übersetzung liefert Klauseln, die $\mathsf{event}(e_1)$ und $\mathsf{event}(e_2)$ beweisen.

Notwendige Ereignisse

$$\mathsf{event}(e_1(x)) \rightsquigarrow \mathsf{event}(e_2(x))$$

Parametrisiere das Problem durch eine beliebige Menge $\mathcal{E}$ von erlaubten Ereignissen.

Passe die Übersetzung so an, dass $\mathsf{event}(M).P$ in etwa wie

$$\mathsf{if}\ M \in \mathcal{E}\ \mathsf{then}\ \mathsf{event}(M).P\ \mathsf{else}\ 0$$

interpretiert wird.

Zeige dann ohne weitere Annahmen an $\mathcal{E}$ durch Resolution: Das Ereignis $\mathsf{event}(e_1(M))$ kann nur eintreten, wenn $e_2(M) \in \mathcal{E}$ gilt.

Für jeden konkreten Lauf des Systems kann man für $\mathcal{E}$ die im Lauf tatsächlich auftretenden Ereignisse wählen.

Prädikate

Wir benutzen Prädikate:

- att(p) — Der Angreifer kann Nachricht p erlangen.
- mess(p_1, p_2) — Die Nachricht p_2 kann über Kanal p_1 geschickt werden.
- event(p) — Das Ereignis p kann eintreten.
- m-event(p) — Es gilt $p \in \mathcal{E}$.

Das Prädikat m-event(p) wird nur auf der linken Seite von Klauseln vorkommen.

Die allgemeinen Klausel für den Angreifer sind genau wie vorher.

Notwendige Ereignisse

$$\mathsf{event}(e_1(x)) \rightsquigarrow \mathsf{event}(e_2(x))$$

Zeige dann ohne weitere Annahmen an $\mathcal{E}$ durch Resolution: Das Prädikat $\mathsf{event}(e_1(M))$ ist nur herleitbar, wenn auch $e_2(M) \in \mathcal{E}$ gilt.

Für jeden konkreten Lauf des Systems kann man für $\mathcal{E}$ die darin tatsächlich auftretenden Ereignisse wählen.

Dieses Argument ist so noch nicht ganz korrekt, da wir die Terme in den Klauseln zu Patterns abstrahieren. Dabei können verschiedene Namen identifiziert werden.

Beispiel: $!c(x).\nu a.\, \mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))$.
In *allen Kopien* der Replikation wird a zu $a[x]$.
Wir verlieren die Information, dass der Name in beiden Events der gleiche sein muss. Wir können lediglich zeigen, dass $\mathsf{event}(e_1(a[x]))$ nur eintreten kann, wenn $\mathsf{event}(e_2(a[x]))$ eintritt.

Instrumentierte Prozesse

Wir müssen die Namen in den verschiedenen Kopien einer Replikation unterscheiden.

Dazu führen wir noch zusätzliche Parameter für Session-IDs ein ($a[x, i]$ statt nur $a[x]$).

Es ist technisch günstig, diese bereits im Pi Kalkül einzuführen.

$$P, Q ::= !^i P \mid \nu a{:}p.P \mid \dots$$

i ist in P gebunden

Patterns:

$$p, q ::= x \mid i \mid a[p_1, \dots, p_n, i_1, \dots, i_m] \mid f(p_1, \dots, p_n)$$

Sitzungsvariablen

Instrumentierte Prozesse

Sei P ein Prozess. Den **instrumentierten Prozess** $\mathsf{instr}(P)$ erhält man, indem man

- jede Replikation $!Q$ darin mit einer frischen Session-Variable i annotiert: $!^i Q$; und
- jede Restriktion $\nu a.\, Q$ darin mit den gebundenen Variablen (Eingaben und Sitzungsvariablen) annotiert: $\nu a{:}a[x_1, \ldots, x_n, i_1, \ldots, i_m].\, Q$.

Beispiel:

$$!c(x).\nu a.\, \mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))$$

wird zu

$$!^i c(x).\nu a{:}a[x, i].\, \mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))$$

Klauseln für das Protokoll

$$
\begin{aligned}
[\![0]\!]\rho h &= \emptyset \\
[\![P \mid Q]\!]\rho h &= [\![P]\!]\rho h \cup [\![Q]\!]\rho h \\
[\![!^i P]\!]\rho h &= [\![P]\!](\rho[i \mapsto i'])h \quad (i' \text{ frisch}) \\
[\![\nu a{:}p.\, P]\!]\rho h &= [\![P]\!](\rho[a \mapsto p\rho])h \\
[\![M(x).P]\!]\rho h &= [\![P]\!][\rho[x \mapsto x']](h \wedge \mathsf{mess}(M\rho, x')) \quad (x' \text{ frisch}) \\
[\![\overline{M}\langle N\rangle.P]\!]\rho h &= [\![P]\!]\rho h \cup \{h \supset \mathsf{mess}(M\rho, N\rho)\} \\
[\![\mathsf{event}(M).P]\!] &= [\![P]\!]\rho(h \wedge \mathsf{m\text{-}event}(M\rho)) \cup \{h \supset \mathsf{event}(M\rho)\}
\end{aligned}
$$

Beispiel

$$
\begin{aligned}
&[\![!^i c(x).\nu a{:}a[x,i].\,\mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))]\!][]\emptyset \\
&\quad = [\![c(x).\nu a{:}a[x,i].\,0 \mid \mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))]\!][i \mapsto i']\emptyset \\
&\quad = [\![\nu a{:}a[x,i].\,\mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))]\!][i \mapsto i', x \mapsto x']\{\mathsf{mess}(c,x')\} \\
&\quad = [\![\mathsf{event}(e_2(a)).\mathsf{event}(e_1(a))]\!][i \mapsto i', x \mapsto x', a \mapsto a[x',i']]\{\mathsf{mess}(c,x')\} \\
&\quad = [\![\mathsf{event}(e_1(a))]\!][i \mapsto i', x \mapsto x', a \mapsto a[x',i']] \\
&\qquad\qquad \{\mathsf{mess}(c,x') \wedge \mathsf{m\text{-}event}(e_2(a[x',i']))\} \\
&\qquad \cup \{\mathsf{mess}(c,x') \supset \mathsf{event}(e_2(a[x',i']))\} \\
&\quad = \{\mathsf{mess}(c,x') \wedge \mathsf{m\text{-}event}(e_2(a[x',i'])) \supset \mathsf{event}(e_1(a[x',i'])), \\
&\qquad \mathsf{mess}(c,x') \supset \mathsf{event}(e_2(a[x',i']))\}
\end{aligned}
$$

Korrektheit

Gegeben sei P ein geschlossener Prozess.
Sei $\mathcal{R}_P$ die Klauselmenge für P, bestehend aus $[\![P]\!][]\emptyset$ und den allgemeinen Klauseln für den Angreifer.
Sei Q ein beliebiger geschlossener Prozess, der die Rolle des Angreifers spielt.
Betrachte eine beliebige Trace $\mathcal{T}$ von $P \mid Q$.
Sei $\mathcal{F}_{mp}$ eine Menge von Fakten mit

$$\{\text{m-event}(p) \mid \text{event}(p) \text{ wird in } \mathcal{T} \text{ ausgeführt}\} \subseteq \mathcal{F}_{mp}.$$

Dann ist jeder Event $\text{event}(p')$, der in $\mathcal{T}$ ausgeführt wird, auch mit der Klauselmenge $\mathcal{R}_P \cup \mathcal{F}_{mp}$ herleitbar.

Resolutionsalgorithmus

Wir wählen die Selektionsfunktion so, dass $\mathsf{sel}(F_1 \wedge \cdots \wedge F_n \supset F)$ eine Teilmenge von

$$\{F \in \{F_1, \ldots, F_n\} \mid (\neg\exists\, \text{Variable}\ x.F = \mathsf{att}(x)) \wedge (\neg\exists p.\, P = \text{m-event}(p))\}$$

ist. (Es gibt keine Klauseln, die $\text{m-event}(p)$ beweisen.)

Dann enthält $\mathsf{saturate}(\mathcal{R}_P)$ nur noch Klauseln ohne ausgewählte Hypothesen und $\mathsf{saturate}(\mathcal{R}_P) \cup \mathcal{F}_{mp}$ leitet die gleichen Klauseln her wie $\mathcal{R}_P \cup \mathcal{F}_{mp}$.

Resolutionsalgorithmus

Der Lösungsalgorithmus liefert für jede Anfrage der Form $\mathsf{event}(p)$ eine Menge von Klausel ohne ausgewählte Hypothesen, d.h. von Klauseln der Form

$$\mathsf{att}(x_1) \wedge \cdots \wedge \mathsf{att}(x_n) \wedge \mathsf{m\text{-}event}(p_1) \wedge \cdots \wedge \mathsf{m\text{-}event}(p_m) \supset \mathsf{event}(p').$$

Die Korrektheitsaussage ist:
Wenn eine geschlossene Instanz $\mathsf{event}(p')$ von $\mathsf{event}(p)$ herleitbar ist, dann gibt es eine Klausel $H \supset C$ in dieser Menge und eine Substitution σ, so dass $\mathsf{event}(p') = C\sigma$ gilt und so dass alle Fakten in $H\sigma$ aus der Klauselmenge ($\mathsf{saturate}(\mathcal{R}_P) \cup \mathcal{F}_{mp}$) herleitbar sind.

Die Fakten in $H\sigma$ müssen bereits aus $\mathcal{F}_{mp}$ herleitbar sein, da $\mathcal{R}_P$ überhaupt keine Fakten der Form $\mathsf{m\text{-}event}(q)$ beweist.

Korrespondenz und Beweissuche

Angenommen $\mathsf{event}(p)$ wird in $\mathcal{T}$ ausgeführt.

Sei $\mathcal{F}_{mp} = \{\mathsf{m\text{-}event}(p) \mid \mathsf{event}(p) \text{ in } \mathcal{T}\}$.

Dann ist $\mathsf{event}(p)$ aus $\mathsf{saturate}(\mathcal{R}_P) \cup \mathcal{F}_{mp}$ herleitbar.

Somit liefert die Beweissuche (bzgl. $\mathsf{saturate}(\mathcal{R}_P) \cup \mathcal{F}_{mp}$) eine Klausel $H \supset C$ mit folgenden Eigenschaften:

- $C\sigma = \mathsf{event}(p)$;
- alle Fakten in $H\sigma$ sind aus $\mathsf{saturate}(\mathcal{R}_P) \cup \mathcal{F}_{mp}$ herleitbar;
- $\mathsf{sel}(H) = \emptyset$
  (d.h. H enthält nur Fakten der Form $\mathsf{att}(x)$ oder $\mathsf{m\text{-}event}(p)$).

Die Fakten in $H\sigma$ müssen bereits aus $\mathcal{F}_{mp}$ herleitbar sein, da $\mathcal{R}_P$ überhaupt keine Fakten der Form $\mathsf{m\text{-}event}(q)$ beweist.

Für alle $\mathsf{m\text{-}event}(p) \in H$ muss $\mathsf{event}(p)$ in $\mathcal{T}$ vorkommen.

Korrespondenz und Beweissuche

Eine Korrespondenz

$$\mathsf{event}(e_1(M)) \rightsquigarrow \mathsf{event}(e_2(N))$$

kann also folgendermaßen bewiesen werden:

- Führe die Beweissuche mit Ziel $\mathsf{event}(e_1(M))$ aus.
- Überprüfe, dass jede zurückgelieferte Klausel die Form

$$H \wedge \mathsf{m\text{-}event}(e_2(N\sigma)) \supset \mathsf{event}(e_1(M\sigma))$$

  für eine geeignete Substitution σ hat.

(Genau genommen muss man M und N noch in Patterns umwandeln, indem man jeden Namen a durch $a[\,]$ ersetzt.)