Protokollsicherheit

Angewandter pi-Kalkül

π-Kalkül

CCS:

$$P, Q ::= 0 \mid a.P \mid \overline{a}.P \mid \nu a.P \mid P + Q \mid P|Q$$

π-Kalkül:

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

- Austausch von Nachrichten statt nur von Signalen
- Prozessmobilität durch Austausch von Kanalnamen
- Replikation $!P$

Angewandter pi-Kalkül:

- Im π-Kalkül werden *nur* Namen ausgetauscht.
- Der angewandte pi-Kalkül erlaubt verschiedene Typen, deren Operationen durch eine Gleichungstheorie spezifiziert sind.

π-Kalkül — Austausch von Nachrichten

Eingabe: Der Prozess $a(x).P$ empfängt einen Wert M auf Kanal a und verhält sich dann wie $P\{M/x\}$.
($\{M/x\}$ bezeichnet Substitution von M für x)

Im Term $a(x).P$ wird die Variable x in P gebunden.

Ausgabe: Der Prozess $a\langle M\rangle.P$ gibt den Wert M auf Kanal a aus.

Intendierte Semantik:

$$a(x).P \mid \overline{a}\langle M\rangle.Q \longrightarrow P\{M/x\} \mid Q$$

Beispiel:

$$\mathit{Buf} = in(x).\overline{out}\langle x\rangle.0$$

π-Kalkül — Replikation

Die rekursive Definition von Prozessen wird auf einfache Weise durch Replikation erlaubt.

$$!P \equiv P \mid !P$$

Erfasst typisches Muster der nebenläufigen Programmierung.

Beispiel: Puffer $!(in(x).\overline{out}\langle x\rangle.0)$

Beispiel: Zählprozess

- Rekursive Definition: $C(x) = \overline{out}\langle x\rangle.\tau.C(x+1)$
- Definition mit Replikation: $C(x) = \nu c.\, c\langle x\rangle \mid !(c(y).\overline{out}\langle y\rangle.\overline{c}\langle y+1\rangle)$

π-Kalkül — Prozessmobilität

Durch das Versenden von Kanalnamen kann man Kanal- und Prozessmobilität modellieren.

Beispiel:

$$Server = !\,\nu\, out.(\overline{get\_new\_counter}\langle out\rangle.C(0))$$

Dieser Prozess erzeugt bei jeder Anfrage einen neuen Zählprozess und gibt den Zugangskanal dazu zurück.

Die Zählprozesse haben alle verschiedene Kanäle.

π-Kalkül — Namen

Im Zusammenhang mit Namensaustausch funktioniert Restriktion wie ein Operator zur Erzeugung neuer Namen.

Beispiel:

$$N = \nu n.\,(\overline{a}\langle n\rangle.0)$$

Beispiel:

$$P = \nu s.\,(c(x).\mathsf{if}\ x = s\ \mathsf{then}\ \overline{c}\langle\mathsf{true}\rangle.0\ \mathsf{else}\ \overline{c}\langle\mathsf{false}\rangle.0)$$

verhält sich wie

$$c(x).\overline{c}\langle\mathsf{false}\rangle.0$$

Angewandter pi-Kalkül — Überblick

Prozesse:

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

Terme:

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

Erweiterte Prozesse:

$$A, B ::= P \mid A|B \mid \nu a.\, A \mid \nu x.A \mid \{M/x\}$$

Terme

$$M, N ::= \underbrace{a, b, c, \ldots, m, n, k, \ldots}_{\text{Namen}} \mid \underbrace{x, y, z, \ldots}_{\text{Variablen}} \mid \underbrace{f(M_1, \ldots, M_n)}_{\text{Funktionssymbole}}$$

Terme werden bezüglich einer mehrsortigen Signatur Σ gebildet.
Sorten:

- Basissorten, z.B. Bool, Int, Key, Bytes.
- Für jede Sorte τ ist $\mathsf{Channel}\langle\tau\rangle$ eine Sorte.

Σ definiert eine Menge von Funktionssymbolen und für jedes eine Arität $(\tau_1, \ldots, \tau_n) \to \tau$, wobei $\tau, \tau_1, \ldots, \tau_n$ Basissorten sind.
Beispiel: $\Sigma = \{\mathsf{senc} \colon (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}, \mathsf{sdec} \colon (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}\}$

Jedem Namen und jeder Variablen ist eine beliebige, aber feste Sorte zugeordnet; nur Terme müssen sortenkorrekt geformt sein.

Gleichungstheorie

Die Bedeutung der Funktionssymbole wird durch Gleichungen erklärt.

Beispiel:
Signatur:
$\Sigma = \{\mathsf{pair} : (\mathsf{Bytes}, \mathsf{Bytes}) \to \mathsf{Bytes}, \mathsf{fst} : \mathsf{Bytes} \to \mathsf{Bytes}, \mathsf{snd} : \mathsf{Bytes} \to \mathsf{Bytes}, \mathsf{senc} : (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}, \mathsf{sdec} : (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}\}$
Gleichungsmenge:

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x$$
$$\mathsf{snd}(\mathsf{pair}(x, y)) = y$$
$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

Gleichungstheorie

Die von einer Menge E von Termgleichungen induzierte Gleichungsrelation $=_E$ ist die kleinste Äquivalenzrelation mit folgenden Eigenschaften:

- Es gilt $M =_E N$ wenn $M = N$ in E ist.
- $M_1 =_E N_1, \ldots, M_n =_E N_n$ impliziert $f(M_1, \ldots, M_n) =_E f(N_1, \ldots, N_n)$ für jedes Funktionssymbol f.
- $M =_E N$ impliziert $M\{R/x\} =_E N\{R/x\}$ für jeden Term R und jede Variable x.
- $M =_E N$ impliziert $M\{b/a\} =_E M\{b/a\}$ für jeden Namen a und jeden Namen b, der weder in M noch in N vorkommt.

Gleichheit ist also abgeschlossen unter Substitution von Variablen und bijektiver Umbenennung von Namen.

Angewandter pi-Kalkül — Überblick

Prozesse:

Name

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

Term der Sorte Channel$\langle\cdots\rangle$.

Terme:

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

Erweiterte Prozesse:

Name

$$A, B ::= P \mid A|B \mid \nu a.\, A \mid \nu x.A \mid \{M/x\}$$

Variablen

Erweiterte Prozesse

Erweiterte Prozesse:

$$A, B ::= P \mid A|B \mid \nu a.\, A \mid \nu x.A \mid \{M/x\}$$

Der Prozess $\{M/x\}$ wird *aktive Substitution* genannt.

Bedeutung:

$$\{M/x\} \mid P \equiv \{M/x\} \mid P\{M/x\}$$

Beispiele:

- Bereitstellung von Werten, die nicht konstruiert werden können: $\nu k.\{\mathsf{senc}(k, y)/x\}$.
- $\nu x.(\{M/x\} \mid P)$ entspricht einem let-Konstrukt: let $x = M$ in P

Aktive Substitutionen

Beliebige Substitutionen können als Prozess ausgedrückt werden:

$$\{M_1/x_1, \ldots, M_n/x_n\} = \{M_1/x_1\} \mid \cdots \mid \{M_n/x_n\}$$

Annahmen:

- Ein erweiterter Prozess enhält für jede Variable x höchstens eine Substitution $\{M/x\}$.
- Der Körper jeder Restriktion νx enthält genau eine Substitution der Form $\{M/x\}$.
- Die Substitutionen in jedem erweiterten Prozess müssen *zyklenfrei* sein. Eine Substitution σ ist zyklenfrei falls für jedes M die Menge $\{M\sigma, M\sigma\sigma, M\sigma\sigma\sigma, \ldots\}$ endlich ist.
  (Beispiel: $\{g(y)/x\} \mid \{x/y\}$ ist nicht zyklenfrei; $\{g(y)/x\} \mid \{n/y\}$ ist zyklenfrei.)

Strukturelle Äquivalenz

Strukturelle Äquivalenz ist die kleinste Äquivalenzrelation $\equiv$ auf erweiterten Prozessen, für die gilt:

1.

$$A \mid 0 \equiv A \qquad A \mid B \equiv B \mid A \qquad (A \mid B) \mid C \equiv A \mid (B \mid C)$$

$$\nu n.\, 0 \equiv 0 \qquad \nu u.\, \nu w.\, A \equiv \nu w.\, \nu u.\, A \qquad \nu u.\, (A \mid B) \equiv (\nu u.\, A) \mid B$$

wenn $u \notin fn(B) \cup fv(B)$

$$!P \equiv P \mid !P \qquad \nu x.\{M/x\} \equiv 0$$

$$\{M/x\} \mid A \equiv \{M/x\} \mid A\{M/x\}$$

$$\{M/x\} \equiv \{N/x\} \text{ wenn } M =_E N$$

2. Wenn $A \equiv B$ dann $E[A] \equiv E[B]$ für jeden Auswertungskontext $E[-]$.

Strukturelle Äquivalenz

Ein erweiterter Prozess A ist *geschlossen* falls er keine freie Variable x enthält, die nicht durch eine aktive Substitution $\{M/x\}$ definiert wird.

Jeder geschlossene erweiterte Prozess A ist strukturell äquivalent zu einem Prozess der Form

$$\nu a_1 \ldots \nu a_k . (P \mid \{M_1/x_1, \ldots, M_n/x_n\}) .$$

Reduktionssemantik

Die Reduktionsrelation $\longrightarrow$ zwischen Prozesse ist nun induktiv durch folgende Regeln definiert.

$$\text{Kommunikation}\ \frac{}{a(x).P \mid \overline{a}\langle x\rangle.Q \longrightarrow P \mid Q}$$

$$\text{Then}\ \frac{}{\text{if } M = M \text{ then } P \text{ else } Q \longrightarrow P}$$

$$\text{Else}\ \frac{}{\text{if } M = N \text{ then } P \text{ else } Q \longrightarrow Q}\quad \begin{array}{l} M \neq_E N, \\ M, N \text{ variablenfrei} \end{array}$$

$$\text{Kontextabschluss}\ \frac{P \longrightarrow Q}{E[P] \longrightarrow E[Q]}$$

$$\text{Struktur}\ \frac{P' \equiv P \qquad P \longrightarrow Q \qquad Q \equiv Q'}{P' \longrightarrow Q'}$$

Beispiel — Datentypen

Funktionssymbole für Konstruktoren und Destruktoren des Datentyps: $\mathsf{pair} : (\mathsf{Bytes}, \mathsf{Bytes}) \to \mathsf{Bytes}$, $\mathsf{fst}, \mathsf{snd} : \mathsf{Bytes} \to \mathsf{Bytes}$

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x$$
$$\mathsf{snd}(\mathsf{pair}(x, y)) = y$$

Schreibe (M, N) für $\mathsf{pair}(M, N)$.

Beispiel:

$$\nu s.\, \big(\overline{a}\langle (M, s)\rangle \mid a(x).\mathsf{if}\ \mathsf{snd}(x) = s\ \mathsf{then}\ \overline{b}\langle \mathsf{fst}(x)\rangle\ \mathsf{else}\ 0\big)$$

Angreifer:

$$a(x).\overline{a}\langle (N, \mathsf{snd}(x))\rangle$$

Beispiel — Hash-Funktionen

Funktionssymbol $\mathsf{hash} : \mathsf{Bytes} \to \mathsf{Bytes}$, *keine Gleichungen*

$$\nu s. \left(\begin{array}{l} \overline{a}\langle (M, \mathsf{hash}(s, M)) \rangle \\ \quad \mid a(x).\mathsf{if}\ \mathsf{snd}(x) = \mathsf{hash}(s, \mathsf{fst}(x))\ \mathsf{then}\ \overline{b}\langle \mathsf{fst}(x) \rangle\ \mathsf{else}\ 0 \end{array} \right)$$

Analog: $\mathsf{mac} : (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}$, *keine Gleichungen*

Beispiel — Symmetrische Verschlüsselung

Funktionssymbole:

$$\mathsf{senc}\colon (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}, \qquad \mathsf{sdec}\colon (\mathsf{Bytes}, \mathsf{Key}) \to \mathsf{Bytes}$$

Gleichung:

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

Beispiel: Verschlüsselung von M mit einem neuen Schlüssel:

$$\nu k.\overline{a}\langle \mathsf{enc}(k, M)\rangle.0$$

Beispiel — Asymmetrische Verschlüsselung

Funktionssymbole:

$$\begin{aligned}\mathsf{pk}&\colon \mathsf{sKey} \to \mathsf{pKey},\\ \mathsf{aenc}&\colon (\mathsf{Bytes}, \mathsf{pKey}) \to \mathsf{Bytes},\\ \mathsf{adec}&\colon (\mathsf{Bytes}, \mathsf{sKey}) \to \mathsf{Bytes},\end{aligned}$$

Gleichung:

$$\mathsf{adec}(\mathsf{aenc}(y, \mathsf{pk}(x)), x) = y$$

Beispiel:

$$\nu s.\, (\overline{a}\langle \mathsf{pk}(s)\rangle \mid b(x).\overline{c}\langle \mathsf{adec}(s, x)\rangle)$$

Beispiel — Signaturen

Funktionssymbole:

$$\begin{gathered}\mathsf{pk}\colon \mathsf{sKey} \to \mathsf{pKey},\\ \mathsf{sign}\colon (\mathsf{Bytes}, \mathsf{sKey}) \to \mathsf{Bytes},\\ \mathsf{mess}\colon \mathsf{Bytes} \to \mathsf{Bytes},\\ \mathsf{checksign}\colon (\mathsf{Bytes}, \mathsf{pKey}) \to \mathsf{Bool},\\ \mathsf{true}\colon \mathsf{Bool}\end{gathered}$$

Gleichungen:

$$\begin{aligned}\mathsf{mess}(\mathsf{sign}(x, y)) &= x\\ \mathsf{checksign}(\mathsf{sign}(x, y), \mathsf{pk}(y)) &= \mathsf{true}\end{aligned}$$

Beispiel — Handshake-Protokoll

Client A möchte ein Geheimnis s dem Server B mitteilen.

Annahmen:

- A und B haben je ein Schlüsselpaar für die asymmetrische Verschlüsselung.
- A kennt Bs öffentlichen Schlüssel.

1. $A \to B : \mathsf{pk}_A$
2. $B \to A : \mathsf{aenc}(\mathsf{sign}((\mathsf{pk}_B, k), \mathsf{sk}_B), \mathsf{pk}_A)$
3. $A \to B : \mathsf{senc}(s, k)$

Beispiel — Handshake-Protokoll

1. $A \to B : \mathsf{pk}_A$
2. $B \to A : \mathsf{aenc}(\mathsf{sign}((\mathsf{pk}_B, k), \mathsf{sk}_B), \mathsf{pk}_A)$
3. $A \to B : \mathsf{senc}(s, k)$

$$
\begin{aligned}
P &= \nu s_A.\, \nu s_B.\, \nu s. \\
&\quad \text{let } pk_A = \mathsf{pk}(s_A) \text{ in let } pk_B = \mathsf{pk}(s_B) \text{ in} \\
&\quad (\overline{c}\langle pk_A \rangle \mid \overline{c}\langle pk_B \rangle \mid !P_A \mid !P_B) \\
P_A &= \overline{c}\langle pk_A \rangle.c(x). \\
&\quad \text{let } y = \mathsf{adec}(x, s_A) \text{ in} \\
&\quad \text{if } \mathsf{checksign}(y, s_A) = \mathsf{true} \text{ then } \overline{c}\langle \mathsf{senc}(s, k) \rangle \text{ else } 0 \\
P_B &= c(pkX).\nu k. \\
&\quad \overline{c}\langle \mathsf{aenc}(\mathsf{sign}((pk_B, k), s_B), pkX) \rangle. \\
&\quad c(x).\, \text{let } z = \mathsf{sdec}(x, k) \text{ in } 0
\end{aligned}
$$

Prozessäquivalenz

Definiere Prozessäquivalenz für den angewandten pi-Kalkül analog zu den Definitionen in CCS:

Observationsäquivalenz (= Weak Barbed Equivalence)

Weak Bisimulation

Beide Definitionen fallen wieder zusammen, Bisimulation kann als Beweismethode zum Zeigen von Observationsäquivalenzen benutzt werden.

Observationsäquivalenz

Schreibe wieder $\Longrightarrow$ für $\longrightarrow^*$.

Schreibe $A \Downarrow a$ falls $P \Longrightarrow E[\overline{a}\langle M\rangle.P]$ gilt für geeignete P und M sowie einen Evaluationskontext $E[-]$, der a nicht bindet.
(A kann nach ≥ 0 internen Schritten auf Kanal a senden.)

Definition:
Observationsäquivalenz $\approx$ ist die größte symmetrische Relation, mit folgenden Eigenschaften (für alle P, Q, P'):

1. Aus $A \approx B$ und $A \Downarrow a$ folgt $B \Downarrow a$ für alle a.
2. Gilt $A \approx B$ und $A \Longrightarrow A'$, so gibt es B' mit $B \Longrightarrow B'$ und $A' \approx B'$.
3. Aus $A \approx B$ folgt $E[A] \cong E[B]$ für jeden Auswertungskontext $E[-]$, so dass $C[A]$ und $C[B]$ jeweils keine freie Variable enthalten, die nicht durch eine aktive Substitution definiert wird.

Beispiel: $\nu s.(\overline{a}\langle s\rangle) \approx \nu s.\overline{a}\langle \mathsf{hash}(s)\rangle$

Statische Äquivalenz

Ein **Frame** ist ein erweiterter Prozess, der sich aus 0 und aktiven Substitutionen $\{M/x\}$ mittels paralleler Komposition und Restriktion aufbauen lässt.

Beispiele:

$$\phi_0 = \nu k.\{k/x\} \mid \nu s.\{s/y\}$$
$$\phi_1 = \nu k.\{f(k)/x, g(k)/y\}$$
$$\phi_2 = \nu k.\{k/x, f(k)/y\}$$

Jeder Frame ist bis auf strukturelle Äquivalenz von der Form

$$\nu a_1 \ldots . \nu a_n . \{M_1/x_1, \ldots, M_n/x_k\}.$$

Jeder erweiterte Prozess A hat einen Frame $\varphi(A)$, den man durch Ersetzen aller einfachen Prozesse durch 0 erhält.

Statische Äquivalenz

Zwei Terme M und N sind **gleich bezüglich des Frames** φ, geschrieben als $(M = N)\varphi$, falls $\varphi \equiv \nu\vec{a}.\sigma$ und $M\sigma = N\sigma$ und $\{\vec{a}\} \cap (fn(M) \cup fn(N)) = \emptyset$.

Zwei geschlossene Frames φ und ψ sind **statisch äquivalent**, falls sie die gleichen Variablen definieren und falls

$$(M = N)\varphi \qquad \text{gdw} \qquad (M = N)\psi$$

für alle Terme M und N gilt.

Zwei erweiterte Prozesse sind statisch äquivalent falls ihre Frames äquivalent sind.