Verhaltensorientierte Modellierung 4

Modul 13 (v1.0)

Kanonikvorlesung: Foundations of Computing

Heiko Mantel

MAIS, TU Darmstadt, WS10/11

Motivation

Erinnerung

- Modul 10 und 11: Modellierung von Systemen
  - Verhaltensorientierte Modellierung
- Modul 12: Spezifikation von Systemen
  - eine Teilsprache der Prozessalgebra CSP

Fokus dieses Moduls

- Wie modelliert man Eigenschaften von Systemen deklarativ?
- Wie modelliert man Anforderungen an Systeme?

Beispiele für Anforderungen an Informationssysteme:

- **funktionale Anforderungen**
  - z.B. die Ausgabe ist die in aufsteigender Reihenfolge sortierte Folge der Elemente der eingegebenen Liste
- **nichtfunktionale Anforderungen**
  - z.B. private Informationen werden nicht an Dritte weitergegeben

Übersicht: Modul 13

Formale Modellierung von Systemeigenschaften

- Vorgehen und Rahmenbedingungen

Beispiel: Sicherheitsanforderung an eine Robotersteuerung

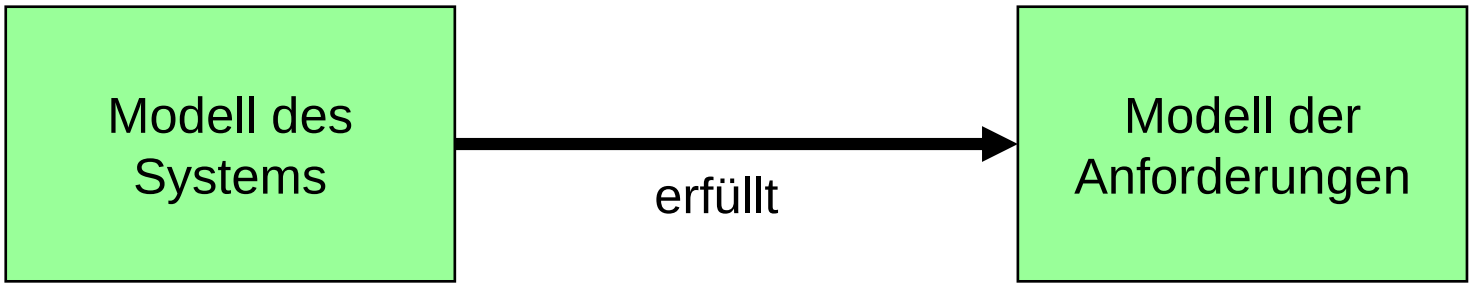
- informelle Beschreibung der Anforderung
- Vergleich von verschiedenen Formalisierungen der Anforderung

Beispiel: Sicherheitslücke in einem Kommunikationsprotokoll

- Spezifikation von Eigenschaften in CSP
- informelle Beschreibung eines Angriffs
- Spezifikation des Protokollablaufs
- Spezifikation der Sicherheitseigenschaft

Modellierung von Eigenschaften 1

Struktur der Modellierung



Verifikation von Anforderungen

- Erfüllt ein Systemmodell ein gegebenes Anforderungsmodell?

Was sind die Voraussetzung für eine formale Verifikation?

- Das System und die Anforderungen müssen formal modelliert sein.
  - In der Modellierung muss erkennbar sein, was das Systemmodell und was das Anforderungsmodell ist (entsprechend obigem Bild).
- Es muss formal definiert sein, unter welchen Bedingungen ein gegebenes Systemmodell ein Anforderungsmodell erfüllt.

Modellierung von Eigenschaften 2

Modellierung von Anforderungen in der Softwareentwicklung

- Die Anforderungsdefinition geschieht in frühen Entwicklungsphasen.
- Oft steht noch kein detailliertes Systemmodell zur Verfügung.

Wann kann man Anforderungen verifizieren?

- erst nachdem ein ausreichend detailliertes Modell vorhanden ist
  - Systemmodell und Anforderungsmodell müssen existieren
- Verifikation unterscheidet sich von Validierung (siehe Modul 9)

Wie modelliert man Anforderungen, ohne ein Systemmodell?

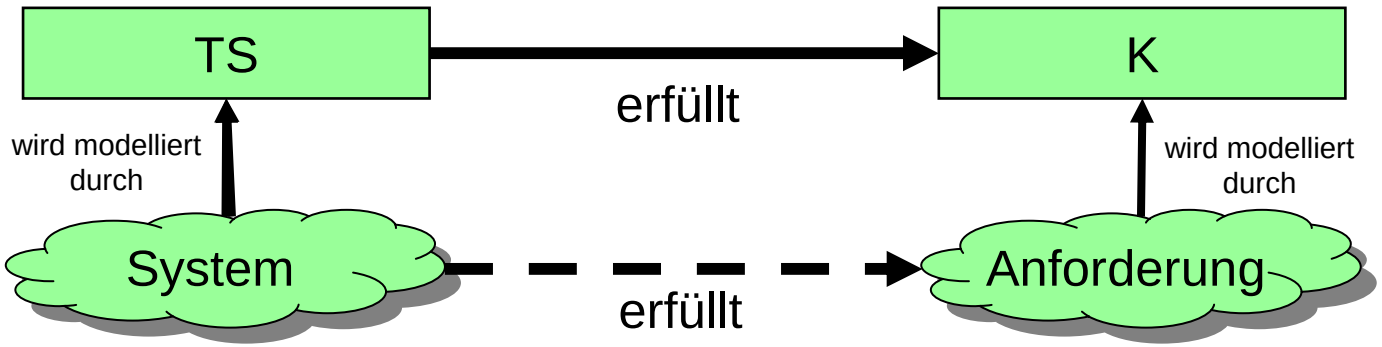
- indem man annimmt, dass bestimmte Strukturen im Systemmodell definiert werden (z.B. dass Zustände durch Funktionen von Programmvariablen nach Werten modelliert werden), und die Anforderungen basierend auf diesen Strukturen definiert (Beispiele folgen)

Modellierung von Eigenschaften 3

Beispiel für ein Vorgehen bei der Anforderungsmodellierung

- Wir legen fest, dass das Systemverhalten im Systemmodell durch ein Transitionssystem $TS = (S, S0, E, \rightarrow)$ modelliert wird.
- Systemanforderungen können jetzt durch Prädikate auf Transitionssystemen modelliert werden.
- Die Modellierung der Anforderungen durch ein Prädikat K ist **angemessen**, wenn folgendes gilt:
  - $K(TS)$ gilt für ein Transitionssystem $TS = (S, S0, E, \rightarrow)$

  genau dann wenn
  - das durch TS modellierte System die durch K modellierte Anforderung erfüllt.



Übersicht: Modul 13

Formale Modellierung von Systemeigenschaften

- Vorgehen und Rahmenbedingungen

Beispiel: Sicherheitsanforderung an eine Robotersteuerung

- informelle Beschreibung der Anforderung
- Vergleich von verschiedenen Formalisierungen der Anforderung

Beispiel: Sicherheitslücke in einem Kommunikationsprotokoll

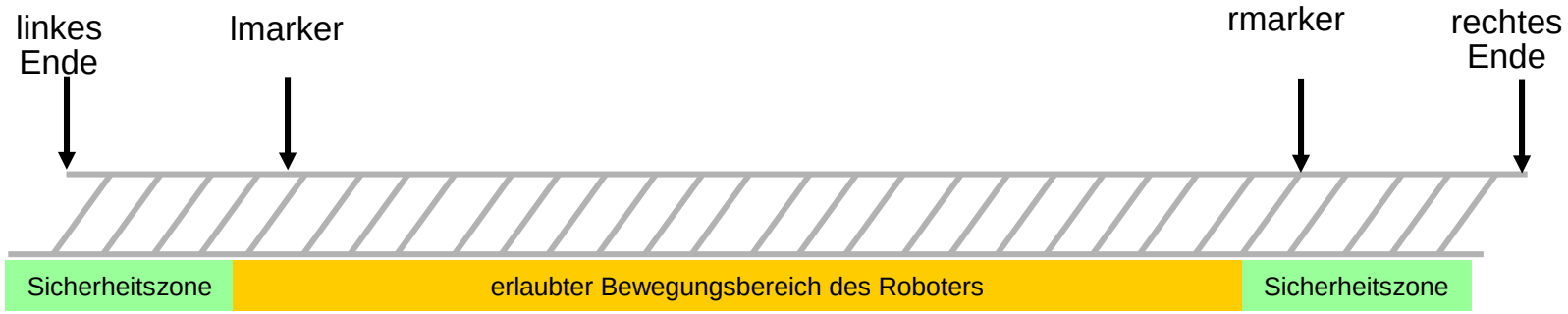
- Spezifikation von Eigenschaften in CSP
- informelle Beschreibung eines Angriffs
- Spezifikation des Protokollablaufs
- Spezifikation der Sicherheitseigenschaft

Modellierung einer Steuerung 1

Beispiel

Steuerung eines Produktionsroboters

- Der Roboter bewegt sich auf Schienen nach rechts oder links.
- Vor dem linken Ende der Schienen gibt es eine Stopmarkierung.
- Vor dem rechten Ende der Schienen gibt es eine Stopmarkierung.



Informelle Beschreibung der Sicherheitsanforderung

Der Roboter soll sich nur innerhalb des erlaubten Bereichs bewegen.

Wie präzisiert und formalisiert man diese Anforderung?

Modellierung einer Steuerung 2

Formalisierung der Anforderung

- Wir legen fest, dass die Steuerung durch ein Transitionssystem **TS = (S,S0,E,→)** modelliert wird. Die genaue Modellierung des Systemsverhalten wird noch nicht durchgeführt.
- Für einen gegebenen Systemzustand s sei die Position des Roboters durch **pos(s)∈ℤ** gegeben.
  - Anmerkung: Die Funktion **pos** kann genauer definiert werden, sobald die relevanten Teile des Systemmodells definiert sind.
  - Beispiel: Wenn Zustände als Funktionen von Programmvariablen zu Werten modelliert werden und es eine Programmvariable **current-pos** gib, die die aktuelle Position des Roboters angibt, so definieren wir **pos(s) = s(current-pos)**.
- Die Anforderung kann wie folgt als Prädikat formalisiert werden:
  - **K1(TS) = ∀h∈S-Hist(TS): ∀n∈ℕ:**
    (lmarker ≤ pos(h(n)) ∧ pos(h(n)) ≤ rmarker)

Ist dieses Anforderungsmodell angemessen?

Modellierung einer Steuerung 3

Erste Modellierung der Anforderung

- **K1(TS) = $\forall h \in$ S-Hist(TS): $\forall n \in \mathbb{N}$:**
  (lmarker $\leq$ pos(h(n)) $\wedge$ pos(h(n)) $\leq$ rmarker)

Interpretation der Modellierung:

- In jedem Zustand, der in einer möglichen Zustandshistorie des Systems vorkommt, ist die momentane Position zwischen den Markierungen **lmarker** und **rmarker**.

Wir betrachten eine weitere Modellierung der Anforderung:

- **K2(TS) = $\forall s \in$ S: (lmarker $\leq$ pos(s) $\wedge$ pos(s) $\leq$ rmarker)**

Interpretation der Modellierung:

- In jedem Zustand ist die momentane Position zwischen den Markierungen **lmarker** und **rmarker**.

Welche der beiden Modellierungen ist besser?

Modellierung einer Steuerung 4

Theorem

Seien $TS = (S,S0,E,\rightarrow)$ und $pos: S \rightarrow \mathbb{Z}$ beliebig.
Dann gilt: $K2(TS) \Rightarrow K1(TS)$.

Beweis

- Angenommen K2(TS) gelte.
- Dann gilt $lmarker \leq pos(s) \leq rmarker$ für jeden Zustand $s \in S$.
- Seien $h \in S\text{-}Hist(TS)$ und $n \in \mathbb{N}$ beliebig.
- Gemäß Modul 10 gilt $h: \mathbb{N} \rightarrow S$.
- Da $h(n) \in S$ gilt, gilt auch $lmarker \leq pos(h(n)) \leq rmarker$.
- Also gilt $\forall h \in S\text{-}Hist(TS): \forall n \in \mathbb{N}:$
  $(lmarker \leq pos(h(n)) \wedge pos(h(n)) \leq rmarker)$
- Somit gilt K1(TS).

Modellierung einer Steuerung 5

Theorem

K1(TS) $\Rightarrow$ K2(TS) gilt **nicht** für beliebige Transitionssysteme TS = (S,S0,E,$\rightarrow$) und Funktionen pos: S$\rightarrow\mathbb{Z}$.

Beweis

- Sei VAR eine Menge von Programmvariablen, die eine Variable current-pos mit Wertebereich $\mathbb{Z}$ enthält, die die momentane Position des Roboters angibt.
- Wir betrachten das Transitionssystem TS = (S,S0,E,$\rightarrow$) mit
  - S = VAR$\rightarrow\mathbb{Z}$
  - S0 = { s0 }, wobei s0(current-pos) = lmarker und lmarker $\leq$ rmarker
  - E = { e }
  - $\rightarrow = \emptyset$, d.h. keine Transitionen sind möglich.
- Für dieses Transitionsssystem gilt die Eigenschaft K1, d.h. K1(TS) gilt.
- Da lmarker$\in\mathbb{Z}$ gilt lmarker – 1$\in\mathbb{Z}$. Also gibt es einen Zustand s$\in$S mit s(current-pos) = lmarker – 1.
- Für den Zustand s gilt lmarker $\leq$ pos(s) nicht.
- Also gilt $\forall$s$\in$S: (lmarker $\leq$ pos(s) $\wedge$ pos(s) $\leq$ rmarker) nicht.
- Somit gilt K2(TS) nicht.

Modellierung einer Steuerung 6

Informelle Beschreibung der Sicherheitsanforderung

Der Roboter soll sich nur innerhalb des erlaubten Bereichs bewegen.

Die alternativen formalen Modellierungen:

- K1(TS) = $\forall h \in S\text{-Hist}(TS): \forall n \in \mathbb{N}:$
  $(lmarker \leq pos(h(n)) \ \wedge \ pos(h(n)) \leq rmarker)$
- K2(TS) = $\forall s \in S: (lmarker \leq pos(s) \ \wedge \ pos(s) \leq rmarker)$

Welche Probleme treten bei der zweiten Modellierung auf?

- Betrachte ein Systemmodell TS mit einem oder mehreren Zustände s, für die $lmarker \leq pos(s) \leq rmarker)$ nicht gilt.
- K2(TS) kann für ein solches Systemmodell nicht gelten, auch dann nicht wenn alle Zustände s mit $lmarker \leq pos(s) \leq rmarker)$ während keines möglichen Systemlaufs erreicht werden können (siehe Beispielsystem im Beweis auf der vorigen Folie).

Die erste Modellierung ist daher der zweiten vorzuziehen!

Modellierung einer Steuerung 7

Wir betrachten eine dritte Modellierung der Anforderung:

- K3(TS) =

$$\forall s \in S0: (lmarker \leq pos(s) \ \wedge \ pos(s) \leq rmarker)$$
$$\wedge \ \forall s,s' \in S: \forall e \in E: [(lmarker \leq pos(s) \ \wedge \ pos(s) \leq rmarker$$
$$\wedge \ (s,e,s') \in \rightarrow)$$
$$\Rightarrow (lmarker \leq pos(s') \ \wedge \ pos(s') \leq rmarker)$$

Wie vergleicht sich diese dritte Modellierung zu den anderen?

- **siehe Übung und Musterlösung**

Übersicht: Modul 13

Formale Modellierung von Systemeigenschaften

- Vorgehen und Rahmenbedingungen

Beispiel: Sicherheitsanforderung an eine Robotersteuerung

- informelle Beschreibung der Anforderung
- Vergleich von verschiedenen Formalisierungen der Anforderung

Beispiel: Sicherheitslücke in einem Kommunikationsprotokoll

- Spezifikation von Eigenschaften in CSP
- informelle Beschreibung eines Angriffs
- Spezifikation des Protokollablaufs
- Spezifikation der Sicherheitseigenschaft

Eigenschaften in CSP 1

Syntax, Intuition und Aussprache

P sat S spezifiziert die Aussage, dass der durch P spezifizierte Prozess die durch S modellierte Eigenschaft hat.

- P ist ein Prozessausdruck
- S ist ein Prädikat auf Spuren, d.h. S: E*→Bool
- P sat S wird „P erfüllt S“ gesprochen.

Semantik

P sat S

- Die Semantik von P sat S wird durch folgende prädikatenlogische Formel definiert: **∀tr∈ traces(P): S(tr)**

Eigenschaften in CSP 2

Definition

Seien E ein Alphabet und $F,G \subseteq E$ beliebig. Die Eigenschaft

- „Wenn ein Ereignis aus G geschieht, dann muss zuvor ein Ereignis aus F geschehen sein“

kann wie folgt in CSP für einen Prozess P spezifiziert werden

- **P sat $BEFORE_{F,G}$**

wobei das Prädikat $BEFORE_{F,G}$ wie folgt definiert ist:

- **$BEFORE_{F,G}(tr) = tr \upharpoonright G \neq () \Rightarrow tr \upharpoonright F \neq ()$**

Bemerkung

Die oben definierte Eigenschaft kann verwendet werden, um Sicherheitsanforderungen an Kommunkationsprotokolle zu spezifizieren (siehe die folgenden Folien).

Modellierung eines Protokolls

Definition von Wertebereichen

USER = { A, B, C }

- modelliert die Namen der Kommunikationspartner

KEY = { Pr(u) , Pu(u) | u∈USER }

- modelliert die kryptographischen Schlüssel der Nutzer
  - **Pr(u)** modelliert den privaten Schlüssel von Nutzer u
  - **Pu(u)** modelliert den öffentlichen Schlüssel von Nutzer u

NONCE

- modelliert frisch erzeugte Zufallswerte

PLAINTEXT

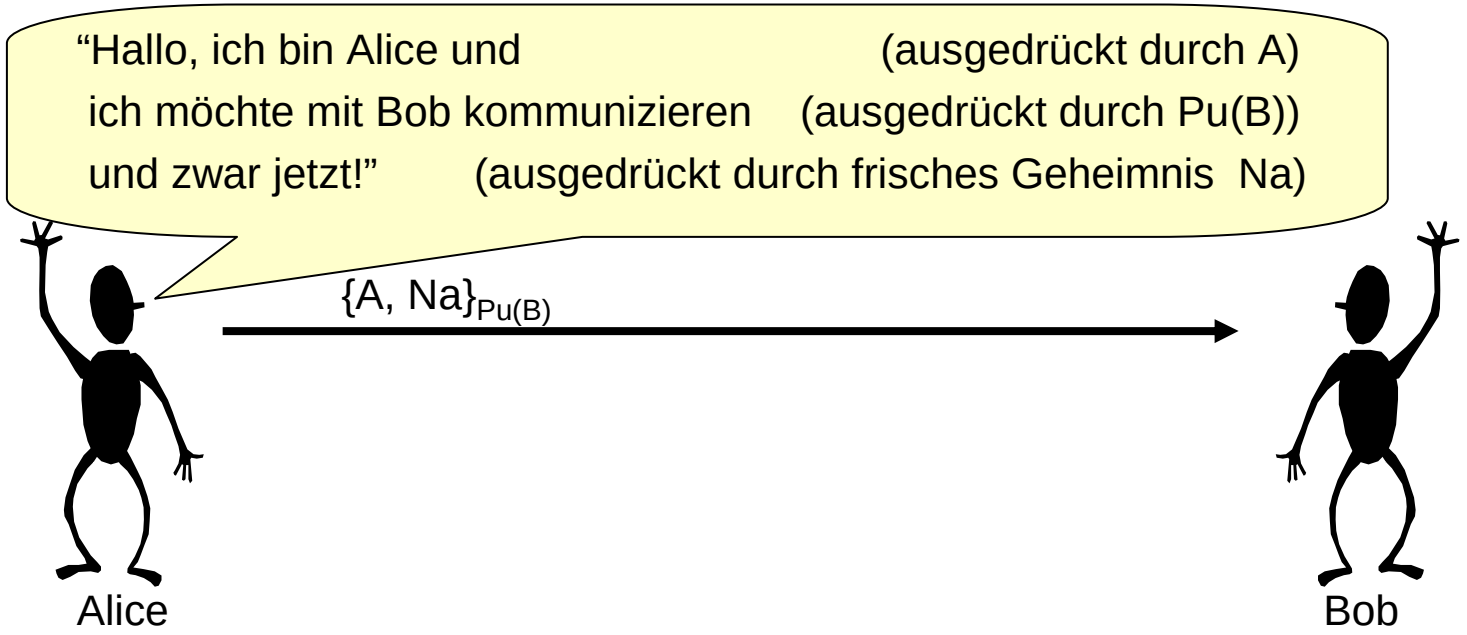
- modelliert unverschlüsselten Text (bleibt unterspezifiziert)

MSG ::= USER | KEY | NONCE | PLAINTEXT | $\{MSG\}_{KEY}$ | MSG,MSG

- modelliert die Nachrichten, die verschickt werden können
- Für k∈KEY und m∈MSG modelliert **$\{m\}_k$** die Nachricht, die durch Verschlüsselung der Nachricht m mit k entsteht.
- Für m1,m2∈MSG modelliert **m1,m2** die Nachricht, die durch Aneinanderhängen der Nachrichten m1 und m2 entsteht.

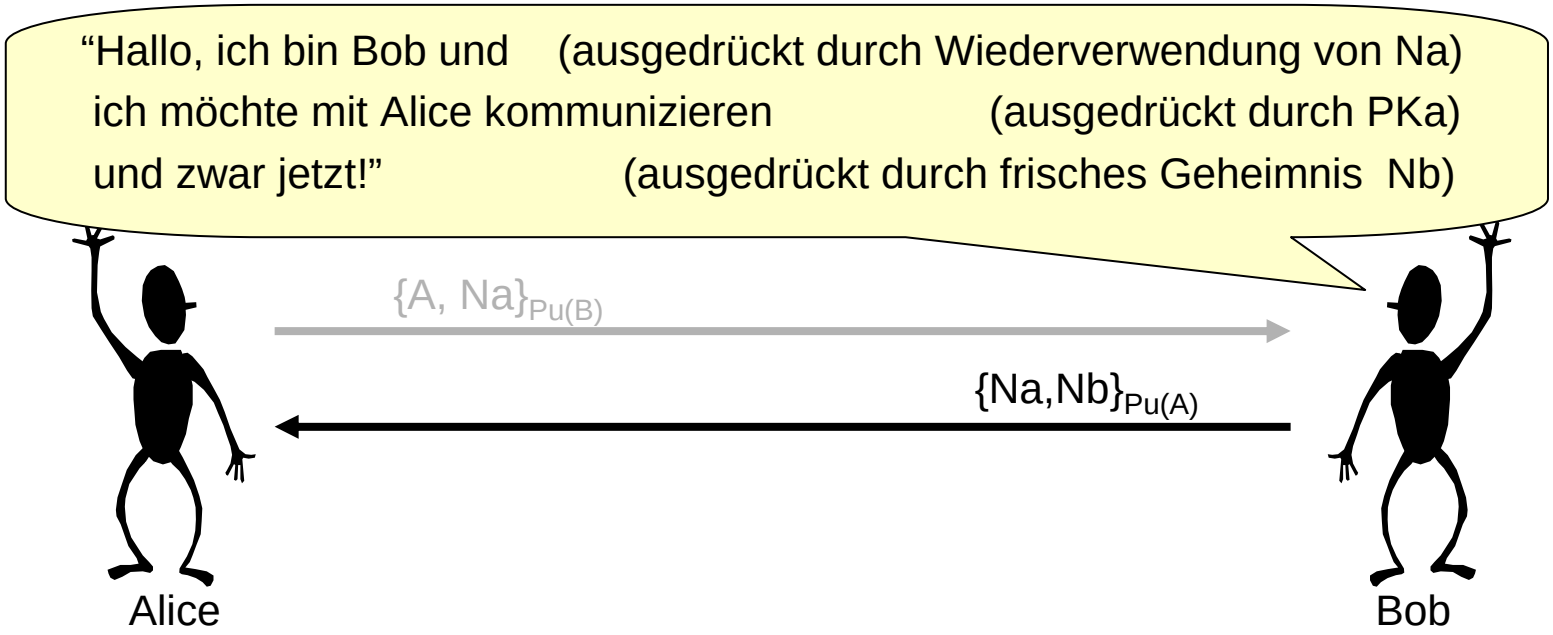
Beispiel: Sicherheitsprotokoll 1

Das Needham-Schroeder Public-Key Protokoll



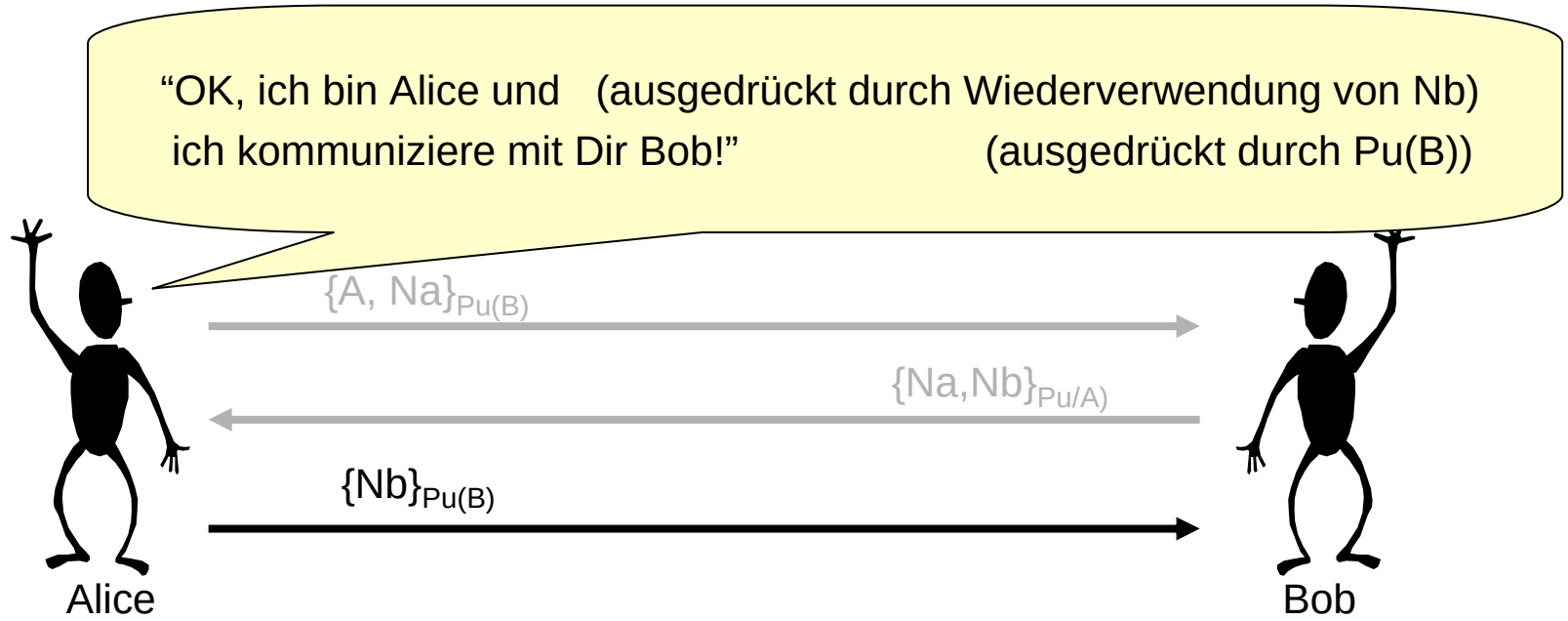
Beispiel: Sicherheitsprotokoll 2

Das Needham-Schroeder Public-Key Protokoll



Beispiel: Sicherheitsprotokoll 3

Das Needham-Schroeder Public-Key Protokoll



Situation nach einem Protokolllauf

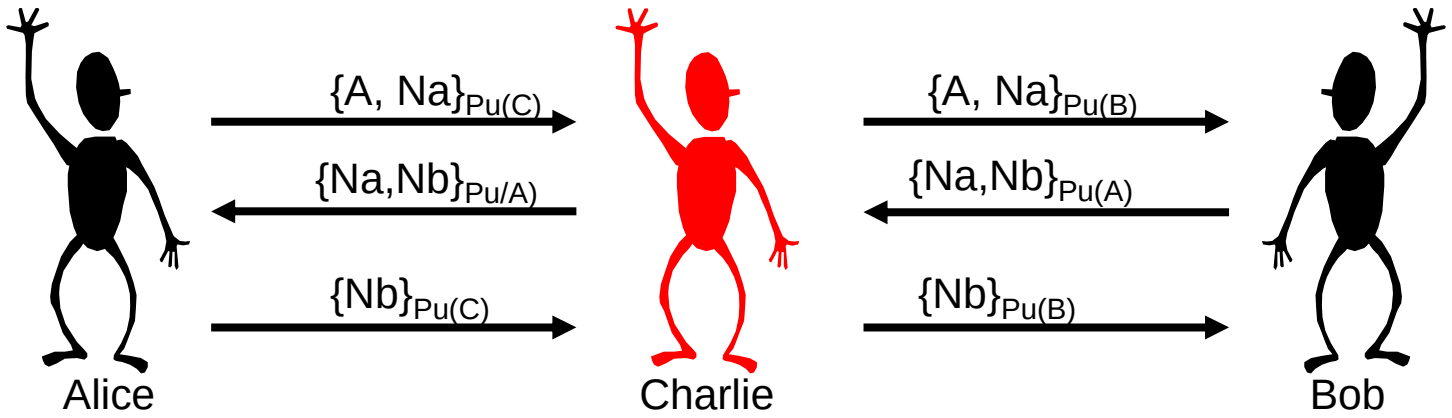
- Alice schließt, dass sie mit Bob kommuniziert.
- Bob schließt, dass er mit Alice kommuniziert.

Sind diese Schlüsse durch die Protokolllogik gerechtfertigt?

Beispiel: Sicherheitsprotokoll 4

Das Needham-Schroeder Public-Key Protokoll

Ein “Mann in der Mitte Angriff”



Situation nach einem Protokolllauf

- Alice mit kommuniziert mit Charlie (wie gewünscht).
- Aber Bob kommuniziert mit Charlie (nicht wie gewünscht mit Alice).

Gavin Lowe entdeckte den Angriff mit formalen Methoden.

Beispiel: Sicherheitsprotokoll 5

Das Needham-Schroeder Public-Key Protokoll

- wurde 1978 von Needham und Schroeder vorgeschlagen

Ziel des Protokolls

Wenn beide Kommunikationspartner das Protokoll erfolgreich beendet haben, können sie davon ausgehen, dass der jeweils andere Kommunikationspartner mit ihnen kommuniziert und auch wirklich mit ihnen kommunizieren will.

Sicherheitslücke

- wurde 1996 von Gavin Lowe mit formalen Methoden gefunden
- Beachte: 18 Jahre nach Entwicklung des Protokolls

Formale Methoden in der Protokollanalyse

- Es gibt viele unterschiedliche Sicherheitsprotokolle.
- Die möglichen Sicherheitslücken können recht subtil sein.
- Viele Sicherlücken wurde erst mit formalen Methoden entdeckt.

Spezifikation des Protokolls in CSP

Wie kann man ein solches Protokoll in CSP spezifizieren?

- Wir geben keine allgemeine Spezifikation des Protokolls, sondern konzentrieren uns auf den zuvor illustrierten Protokolllauf.
- Eine allgemeine Spezifikation des Protokolls ist möglich. Dazu siehe z.B. den Artikel von Steve Schneider in den Referenzen.

Spezifikation der Ereignisse im Beispielprotokoll

send.u.m

- modelliert: Nutzer $u \in USER$ verschickt die Nachricht $m \in MSG$

recv.u.m

- modelliert: Nutzer $u \in USER$ empfängt die Nachricht $m \in MSG$

$\surd_u$

- modelliert: Nutzer $u \in USER$ beendet das Protokoll erfolgreich

Erweiterung der Spezifikationssprache

Wir führen ein weiteren Operator der Prozessalgebra ein.

Syntax, Intuition und Aussprache

(P □ Q) spezifiziert einen Prozess, der sich entweder wie der Prozess P oder wie der Prozess Q verhält.

- (P □ Q) ist nur dann ein zulässiger Ausdruck, wenn $\alpha P = \alpha Q$ gilt.
- (P □ Q) wird **„P Wahl Q“** gesprochen.

Semantik von (P □ Q)

- **α (P □ Q) = αP = αQ** und
- **traces((P □ Q)) = traces(P) ∪ traces(Q) .**

Beachte: Der Operator □ hat die gleiche Semantik wie ⊓.

- **Intuition von P⊓Q:** Die Systemumgebung hat keine Kontrolle, ob sich der Prozess wie P oder wie Q verhält.
- **Intuition von P□Q:** Die Systemumgebung kann beeinflussen, ob sich der Prozess wie P oder wie Q verhält.

Formale Unterscheidung geschieht durch Failure Semantics (Buch).

Spezifikation des Protokolllaufs 1

Verallgemeinerung zu einem n-ären Operator

Syntax, Intuition und Aussprache

$(\Box_{j\in I}\ P(j))$ spezifiziert einen Prozess, der sich wie einer der Prozesse P(j) verhält.

- $(\Box_{j\in I}\ P(j))$ ist nur dann ein zulässiger Ausdruck, wenn $\forall i,j\in I: \alpha P(i) = \alpha P(j)$ gilt.

Semantik von $(\Box_{j\in I}\ P(j))$

- **$\alpha\ (\Box_{j\in I}\ P(j)) = \alpha P(j)$** und
- **$traces((\Box_{j\in I}\ P(j)) = \cup_{j\in I}\ traces(P(j))$** .

Spezifikation des Protokolllaufs 2

Spezifikation des Senders

Alice = $send.A.\{A,Na\}_{Pu(C)} \rightarrow \Box_{N \in NONCE} recv.A.\{Na,N\}_{Pu(A)}$
$\rightarrow send.A.\{N\}_{Pu(C)} \rightarrow \surd_A \rightarrow SKIP_{\{send.A.m, recv.A.m, \surd A \mid m \in MSG\}}$

Spezifikation des Empfängers

Bob = $\Box_{u \in USER,\ N \in NONCE}\ recv.B.\{u,N\}_{Pu(B)} \rightarrow send.B.\{N,Nb\}_{Pu(u)}$
$\rightarrow recv.B.\{Nb\}_{Pu(B)} \rightarrow \surd_B \rightarrow SKIP_{\{send.B.m, recv.B.m, \surd B \mid m \in MSG\}}$

Spezifikation des Angreifers

Charlie = $\Box_{N1 \in NONCE}\ recv.C.\{A,N1\}_{Pu(C)} \rightarrow send.C.\{A,N1\}_{Pu(B)}$
$\rightarrow \Box_{m \in MSG} recv.C.m \rightarrow send.C.m$
$\rightarrow \Box_{N2 \in NONCE} recv.C.\{N2\}_{Pu(C)} \rightarrow send.C.\{N2\}_{Pu(B)}$
$\rightarrow SKIP_{\{send.u.m, recv.u.m \mid u \in USER,\ m \in MSG\}}$

Spezifikation des Netzwerkes (für beliebige $M \subseteq MSG$)

NET(M) = $(\Box_{u \in USER, m \in MSG}\ send.u.m \rightarrow NET(M \cup \{m\}))$
$\sqcap$
$(\sqcap_{u \in USER, m \in M}\ recv.u.m \rightarrow NET(M))$

Spezifikation des Protokolllaufs 3

Spezifikation des Systems

Alice || Bob || Charlie || NET(∅)

Spezifikation der Sicherheitseigenschaft

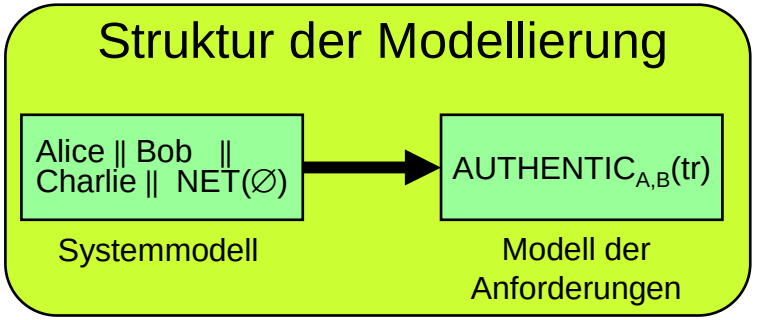
$AUTHENTIC_{A,B}(tr) = BEFORE_{E1,E2}(tr) \wedge BEFORE_{E3,E4}(tr)$

wobei

- **E1** = { send.A.$\{A,N\}_{Pu(B)}$, | N∈ NONCE }
- **E2** = { $\surd_B$ }

und

- **E3** = { send.B.$\{N1,N2\}_{Pu(A)}$, | N1,N2∈ NONCE }
- **E4** = { $\surd_A$ }



Beobachtung

(Alice || Bob || Charlie || NET(∅)) sat $AUTHENTIC_{A,B}$ gilt nicht

Wie begründet man diese Beobachtung am formalen Modell?

- **siehe Übung und Musterlösung**

Rückblick

Einige wesentliche Lernziele dieses Moduls

- Fähigkeit zur klaren Unterscheidung von
  - Systemverhalten und Systemeigenschaften

  in einem formalen Modell
- Fähigkeit formale Modellierungen von Eigenschaften zu verstehen
- Fähigkeit formale Modellierungen von Eigenschaften zu beurteilen
- Fähigkeit formale Modellierungen von Eigenschaften zu vergleichen
- Spezifikation von Eigenschaften
  - Prozessalgebra CSP

Literatur

C. A. R. Hoare

Communicating Sequential Processes; Prentice Hall 1985.

- enthält die hier definierte Spezifikationssprache

Steve Schneider

Security Properties and CSP. Konferenzband IEEE Symposium on Security and Privacy 1996, Seiten 174-187.

- Ansatz zur Spezifikation von Protokollen und Sicherheitsanforderungen in CSP

Steve Schneider

Verifying authentication protocols with CSP. Konferenzband IEEE Computer Security Foundations Workshop 1997, Seiten 3-17.

- Spezifikation und Verifikation von Protokollen und Sicherheitsanforderungen in CSP