

# Sicherheitsmodelle

Claas Lorenz

5. November 2010



# Inhalt

## Motivation

## Zugriffskontrollmodelle

HRU-Modell

Bell-LaPadula-Modell

Probleme mit Zugriffskontrollmodellen

## Informationsflussmodelle

Motivationsbeispiel: Nachrichtenübermittlung

Informationsflussmodelle

Noninterferenz für deterministische Systeme

Possibilistische Modelle für nichtdeterministische Systeme

Noninterferenz für nichtdeterministische Systeme

Nondeducibility

Generalisierte Noninterferenz

Restriktion

Probabilistische Modelle für nichtdeterministische Systeme

FM

AFM und PNI

**Motivation**

**Motivation**

Zugriffskontrollmodelle

Informationsflussmodelle

# Motivation

- ▶ Wozu werden Sicherheitsmodelle benötigt?
- ▶ Sicherheitsmodelle sind ein ganzheitlicher Ansatz alle sicherheitsrelevanten Eigenschaften eines Systems abzudecken und so die Sicherheit des Gesamtsystems zu bewerkstelligen.
- ▶ Der Sicherheitsbegriff bezieht sich im Folgenden ausschließlich auf die Aspekte Vertraulichkeit und Integrität.
- ▶ Ziel ist es, dass unprivilegierte Benutzer keinen Zugang zu schützenswerten Informationen erhalten.

# Zugriffskontrollmodelle

## Motivation

### Zugriffskontrollmodelle

- HRU-Modell

- Bell-LaPadula-Modell

- Probleme mit Zugriffskontrollmodellen

## Informationsflussmodelle

## Zugriffskontrollmodelle

- ▶ Ein Modell ist ein endlicher Automat mit Zuständen  $(S,O,M)$  und einer Menge von Zugriffsrechten  $A$
- ▶  $S$  - Menge von Subjekten
- ▶  $O$  - Menge von Objekten
- ▶  $M$  - Matrix  $M_{|S|,|O|}$  mit den Zugriffsrechten in  $M[s,o]$

# Zugriffskontrollmodelle

## HRU-Modell

- ▶ Aufrufe haben folgende Form:

if( $a_1 \in M[s_1, o_1] \wedge a_2 \in M[s_2, o_2] \wedge \dots \wedge a_n \in M[s_n, o_n]$ )

then  $op_1, op_2, \dots, op_n$

wobei  $op_i \in \{\text{enter a into (s,o), remove a from (s,o), create subject s, remove subject s, create object o, remove object o}\}$

- ▶ Die Semantik entspricht hier der Intuition.

# Zugriffskontrollmodelle

## HRU-Modell

- ▶ Gegeben seien ein System, eine Startkonfiguration  $Q_0$  und ein  $a \in A$ .
- ▶  $Q_0$  ist sicher für alle  $a$ , wenn es keine Sequenz von Aufrufen gibt, die, ausgeführt in  $Q_0$ ,  $a$  in eine Matrixzelle schreibt, welche  $a$  noch nicht enthält.
- ▶ HRU ist entscheidbar für mono-operationelle Systeme.
- ▶ Aber im Allgemeinen nicht entscheidbar.
- ▶ Beweisskizze: System als Turingmaschine. Anschließend Reduktion auf das Halteproblem.

# Zugriffskontrollmodelle

## Bell-LaPadula-Modell

- ▶ Das Bell LaPadula-Modell ist ein Mandatory Access Modell, d.h. Sicherheitseinstufungen entscheiden über die Zugriffsberechtigungen.
- ▶ Das Modell ist ein endlicher Automat, wobei
  - ▶  $S$  und  $O$  festgelegt sind und  $A=\{\text{read,write}\}$
  - ▶  $L$  - feste Menge von Sicherheitseinstufungen
  - ▶  $F:SUO\rightarrow L$  - Zugriffsfunktion liefert Zugriffsrechte
  - ▶  $V$  - Menge der Zustände mit  $(F,M)$ , wobei  $M$  die Zugriffsmatrix ist
- ▶ Ein System enthält
  - ▶ einen Startzustand  $v_0$
  - ▶  $R$  - Menge von Aufrufen
  - ▶  $T:(V\times R)\rightarrow V$  - Transitionsfunktion

# Zugriffskontrollmodelle

Bell-LaPadula-Modell

- ▶ Ein Zustand ist read-secure, gdw.  
 $\forall s \in S, o \in O. (\text{read} \in M(s, o) \rightarrow F(o) \leq F(s)).$
- ▶ Ein Zustand ist write-secure, gdw.  
 $\forall s \in S, o \in O. (\text{write} \in M(s, o) \rightarrow F(s) \leq F(o)).$
- ▶ Ein Zustand  $s$  ist state-secure, gdw. er read- und write-secure ist.
- ▶ Ein System  $(v_0, R, T)$  ist sicher, gdw.  $v_0$  ist state secure und jeder, durch eine endliche Sequenz von Aufrufen aus  $R$  erreichbare, Zustand ebenfalls state-secure ist.

# Zugriffskontrollmodelle

Bell-LaPadula-Modell

- ▶ Sicherheitseinstufungen in deutschen Behörden:
  - ▶ vier Stufen: VS-NfD, VS-V, geheim, streng geheim
  - ▶ Jeder Beamte kann Dokumente mit gleicher oder geringerer Einstufung aus seinem Aufgabenbereich lesen.
  - ▶ Jeder Beamte darf Dokumente mit gleicher oder höheren Sicherheitsstufe versehen.
  - ▶ Anekdote dazu später in der Diskussion

# Zugriffskontrollmodelle

## Probleme mit Zugriffskontrollmodellen

- ▶ Race Conditions beim Zugriff von Subroutinen auf Dateien können zu unerlaubtem Schreiben in low-level Dateien führen.
- ▶ Diese als Covert Channels bezeichneten Sicherheitslücken sind durch die Schwierigkeit bedingt, die Atome des Modells auf Computer(-netze) zu übertragen und treten vor allem in Multiprozesssystemen (insbesondere in verteilten Systemen) auf.
- ▶ Sie können oft erst am Ende des Systementwicklungsprozesses gefunden werden, sodass Änderungen am System umso teurer sind.

# Informationsflussmodelle

Motivation

Zugriffskontrollmodelle

## Informationsflussmodelle

Motivationsbeispiel: Nachrichtenübermittlung

Informationsflussmodelle

Noninterferenz für deterministische Systeme

Possibilistische Modelle für nichtdeterministische Systeme

- Noninterferenz für nichtdeterministische Systeme

- Nondeducibility

- Generalisierte Noninterferenz

- Restriktion

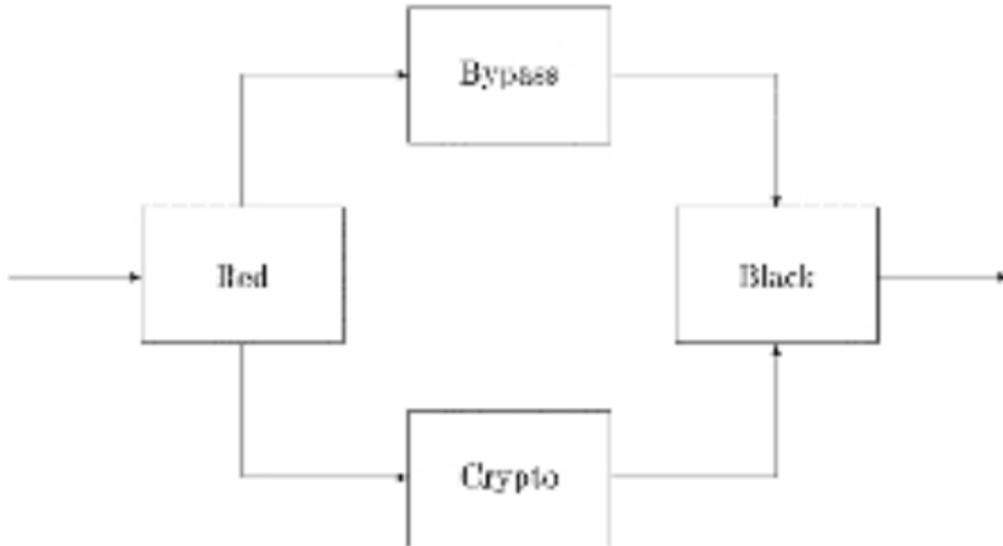
Probabilistische Modelle für nichtdeterministische Systeme

- FM

- AFM und PNI

# Informationsflussmodelle

Motivationsbeispiel: Nachrichtenübermittlung



- ▶ Nachricht von Rot nach Schwarz muss verschlüsselt übertragen werden
- ▶ Nachrichtenkopf muss unverändert weitergeleitet werden

# Informationsflussmodelle

## Informationsflussmodelle

- ▶ Kontrolle des Informationsflusses anstatt des Zugriffs
- ▶ Der Informationsfluss ist die Menge der Routen, welche eine Information im System nehmen darf.

# Informationsflussmodelle

Noninterferenz für deterministische Systeme

- ▶ System ist endlicher Automat mit
  - ▶  $S$  - Menge der Zustände
  - ▶  $A$  - Menge der Aktionen
  - ▶  $O$  - Menge der Ausgaben
  - ▶  $U$  - Menge der Benutzer
  - ▶  $L$  - Menge der Sicherheitslevel
- ▶ Ausgabefunktion  $\text{out}: U \times A^* \rightarrow A^*$  mit  $\text{out}(u, \text{hist.read}(u))$ 
  - ▶ wobei  $\text{hist.read}(u)$  ein Systempfad ist, dessen letzte Eingabe ein Lesekommando von Benutzer  $u$  war
- ▶ Sicherheitslevelfunktion:  $\text{cl}: U \rightarrow L$

# Informationsflussmodelle

Noninterferenz für deterministische Systeme

- ▶ Ausschlussfunktion  $\text{purge}: U \times A^* \rightarrow A^*$  mit  
 $\text{purge}(u, \langle \rangle) = \langle \rangle$ , wobei  $\langle \rangle$  der leere Pfad ist  
 $\text{purge}(u, \text{hist.command}(w)) = \text{purge}(u, \text{hist}).\text{command}(w)$ , falls  
 $\text{command}(w)$  ein Kommando vom Nutzer  $w$  und  $\text{cl}(u) \geq \text{cl}(w)$   
 $\text{purge}(u, \text{hist.command}(w)) = \text{purge}(u, \text{hist})$ , falls  $\text{command}(w)$  ein  
Kommando vom Nutzer  $w$  ist und  $\text{cl}(u) < \text{cl}(w)$
- ▶ Ein System erfüllt Noninterferenz gdw. für alle Benutzer  $u$ , alle  
Systempfade  $T$  und alle Ausgabekommandos  $c$   
 $\text{out}(u, T.c(u)) = (\text{out}(u, \text{purge}(u, T).c(u)))$  gilt.

# Informationsflussmodelle

## Noninterferenz für deterministische Systeme

- ▶ Noninterferenz ist zu stark! Es schließt bestimmte Typen von Systemen komplett aus.
- ▶ Beispiel: Gegeben seien die Akteure  $Y$ ,  $X$  und  $V$ .  $Y$  erhält als Ausgabe  $x \oplus v$ , wobei  $x \in X$  und  $v \in V$  sind. Wenn keine Eingabe vorhanden setze  $x$  oder  $v$  gleich 0.  
Es ist offensichtlich, dass  $X$  bzw.  $V$  mit  $Y$  interferiert. Durch geschicktes Setzen einer der beiden Variablen erlaubt die Interferenz eine Kommunikation der anderen Variablen mit  $Y$ .  
Solch ein System wird durch Noninterferenz als unsicher ausgeschlossen. Wenn nun allerdings  $V$  eine zufällige Folge von 0 und 1 erzeugt, dann wäre das System ein perfektes Kryptographiesystem. Diese werden somit durch Noninterferenz ausgeschlossen.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Ausgaben werden Teil der Eingabehistory
- ▶ Beispiel: Ein Nutzer kann 0 oder 1 eingeben und erhält postwendend seine Eingabe als Ausgabe zurück.  
mögliche Traces:  $\{\langle \rangle, \text{in}(0), \text{in}(1), \text{in}(0).\text{out}(0), \text{in}(1).\text{out}(1), \text{in}(0).\text{out}(0).\text{in}(1), \dots\}$
- ▶ Jeder Präfix eines akzeptierenden Pfades muss ebenfalls akzeptierend sein.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Ansatz: Die, durch Entfernung von high-level Eingaben, bereinigte Version eines akzeptierenden Pfades ist ebenfalls ein akzeptierender Pfad.
- ▶ Problem: Nur sicherheitsrelevante Pfade, welche die Sicherheit verletzen, dürfen nicht akzeptiert werden. Dies gilt aber nicht für andere Systempfade.
- ▶ Beispiel: Gegeben sei das System der vorigen Folie. Angenommen alle Ein- und Ausgaben seien high-level. Da das System keine low-level Ausgabe liefert, ist es trivialerweise sicher.  $\text{in}(0).\text{out}(0)$  ist ein akzeptierender Pfad.  $\text{purge}(\text{in}(0).\text{out}(0))=\text{out}(0)$  ist dies nicht mehr, weil das System keine unangeforderten Ausgaben liefern soll.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ weiterer Ansatz: Neudefinition des purge-Operators. Dieser soll nun auch noch alle high-level Ausgaben entfernen.
- ▶ Problem 1: Diese Anforderung ist zu stark, weil sie jedes System ausschließt, welches aus low-level Eingaben high-level Ausgaben erzeugt.
- ▶ Beispiel: Monitoringprogramme, welche low-level Benutzeraktivitäten überwachen und in eine high-level Datei schreiben.
- ▶ Problem 2: Die Anforderungen erlaubt sogar unsichere Systeme!

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Beispiel: Angenommen ein System hätte folgende Menge von Traces:  
 $\{\langle \rangle, \text{highin}(0), \text{highin}(1), \text{lowout}(0), \text{lowout}(1),$   
 $\text{highin}(0).\text{lowout}(0), \text{highin}(1).\text{lowout}(1)\}$   
 $\text{purge}(\text{highin}(0).\text{lowout}(0)) = \text{lowout}(0) \rightarrow \text{akzeptiert}$   
 $\text{purge}(\text{highin}(1).\text{lowout}(1)) = \text{lowout}(1) \rightarrow \text{akzeptiert}$
- ▶ Das System entspricht den Anforderungen der Noninterferenz.
- ▶ Ein high-level Benutzer kann in diesem System Informationen an einen low-level Benutzer weitergeben, was eine Sicherheitsverletzung darstellt.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Ansatz: Seien  $S$  und  $T$  akzeptierende Traces. Dann muss es ein Trace  $R$  geben, welches die low-level Ereignisse von  $T$  (in gleicher Anordnung), die high-level Eingaben von  $S$  (in gleicher Anordnung) und möglicherweise andere Ereignisse enthält, die weder low-level Ereignisse von  $T$  oder high-level Eingaben von  $S$  sind.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Nondeducibility ist äquivalent zur Noninterferenz bei deterministischen Systemen mit nur zwei Benutzern. Es ist schwächer, wenn mehr Benutzer angenommen werden.
- ▶ Beispiel: Benutzer  $Y$  erhält als Ausgabe  $\text{in}(X) \oplus \text{in}(V)$  der Nutzer  $X$  und  $V$ .  $X$  und  $V$  interferieren mit  $Y$ , was durch Noninterferenz ausgeschlossen werden würde. Nondeducibility erlaubt aber so ein System, weil zu jeder Ausgabe von  $Y$  und jeder Eingabe von  $X$  eine passende Eingabe von  $V$  gefunden werden kann.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Problem 1: Nondeducibility ist insgesamt zu schwach. In einem System, in welchem ein high-level Benutzer H beliebige Eingaben (z.B. ein zu verschlüsselnder Text) und ein low-level Benutzer L einen Befehl look eingibt, der ihm die verschlüsselte Version der Eingabe von H, sofern vorhanden, oder einen Zufallsstring liefert. Dieses System erfüllt Nondeducibility, weil L nichts über die Eingaben von H lernen kann. Problematisch ist aber, dass man das Verschlüsselungselement einfach entfernen kann, ohne dass dies etwas an der Erfüllung von Nondeducibility ändert.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Beispiel: Seien `highin(tierisch geheim)` und `lowin(look).lowout(???)` gegeben. Dann kann der akzeptierende Pfad `lowin(look).lowout(???.highin(tierisch geheim)` gebildet werden. Gleichzeitig seien die Pfade `<>` und `highin(tierisch geheim).lowin(look).lowout(tierisch geheim)` akzeptiert. Dann kann der legale Pfad `lowin(look).lowout(tierisch geheim)` gebildet werden, weil der String 'tierisch geheim' zufällig gebildet worden sein könnte.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Problem 2: Nondeducibility ist nicht abgeschlossen unter Komposition. Zwei Subsysteme, welche jeweils Nondeducibility erfüllen, können nicht miteinander verknüpft werden und zwangsläufig die Sicherheitseigenschaft aufrecht erhalten.
- ▶ Der Grund hierfür ist, dass Nondeducibility zu viel Freiheit bei der Konstruktion akzeptierender Pfade aus den high-level Eingaben aus  $T$  und den low-level Ereignissen aus  $S$  erlaubt.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Der Ansatz lautet nun nicht nur die Ordnung der Ereignisse aufrecht zu erhalten, sondern auch die Art und Weise der Konkatenation der Elemente aus beiden Pfaden zu beschränken.
- ▶ Die Lösung ist nun, dass es für jede Änderung  $T_1$  eines akzeptierenden Pfades  $T$ , welche durch Entfernen oder Hinzufügen einer high-level Eingabe entstanden ist, einen akzeptierenden Pfad  $T_2$  gibt, welcher durch Entfernen oder Hinzufügen von high-level Ausgaben nach der Änderung in  $T$  von  $T_1$  entstanden ist.
- ▶ Beispiel: Gegeben sei voriges System eines low-level Benutzers, der high-level Eingaben überwacht. Der Versuch den Pfad `highin(tierisch geheim).lowin(look).lowout(???)` lässt sich nicht mehr bilden, weil keine high-level Ausgabe hinzugefügt werden kann, sodass ein akzeptierender Pfad  $T_2$  gebildet werden kann.

# Informationsflussmodelle

Possibilistische Modelle für nichtdeterministische Systeme

- ▶ Leider löst generalisierte Noninterferenz nicht das Problem des fehlenden Abschlusses unter Komposition.
- ▶ Lösung: Die Regel für  $T_2$  muss erweitert werden, sodass nun entweder alle high-level Ausgaben nach der Änderung entfernt werden oder eine high-level Ausgabe direkt hinter einer, der Änderung folgenden, Sequenz von low-level Eingaben eingefügt wird.
- ▶ Beispiel: Sei  $wxyz$  ein akzeptierender Pfad, wobei  $w$  eine high-level Eingabe,  $x$  und  $y$  low-level Eingaben und  $z$  eine high-level Ausgabe sind. Nun wird eine high-level Eingabe  $h$  hinter  $w$  eingeführt, sodass der Pfad nun  $whxyz$  lautet. Um  $T_2$  zu bilden, kann nun entweder  $z$  entfernt oder aber eine high-level Eingabe direkt hinter  $xy$  eingefügt werden.

# Informationsflussmodelle

Probabilistische Modelle für nichtdeterministische Systeme

- ▶ Flow Model ist ein Modell für nichtdeterministische Systeme, welches mit Störsignalen auf den Kommunikationskanälen umgehen kann.
- ▶ FM ist ein Viertupel  $\langle S, I, C, O \rangle$  mit
  - ▶ S - Menge von Informationsquellen. Bringt Probabilismus ins System und enthält alle Eingabekanäle und mögliche interne Zufallszahlengeneratoren
  - ▶ C Menge der Ausgabekanäle
  - ▶ I - Eingabealphabet
  - ▶ O - Ausgabealphabet

# Informationsflussmodelle

Probabilistische Modelle für nichtdeterministische Systeme

- ▶ Der Einfachheit halber gilt für die folgenden Betrachtungen  $S = \{\text{lowin}, \text{highin}\}$  und  $C = \{\text{lowout}, \text{highout}\}$ . Zudem kennen low-level und high-level Benutzer die gesamte Ereignishistory ihrer jeweiligen Stufe.
- ▶ Zu einem Zeitpunkt  $t$  besteht die Eingabe aus dem geordneten Paar  $\text{in}_t = \langle \text{highin}_t, \text{lowin}_t \rangle$  und die Ausgabe aus  $\text{out}_t = \langle \text{highout}_t, \text{lowout}_t \rangle$
- ▶ Die History zu einem Zeitpunkt  $t$  sei  $\langle \text{in}_1, \text{in}_2, \dots, \text{in}_t \rangle$ .
- ▶ Gegeben sei ebenfalls eine Funktion  $\hat{O}(\text{out}_{t+1} | \langle \text{in}_1, \text{in}_2, \dots, \text{in}_t \rangle, \langle \text{out}_1, \text{out}_2, \dots, \text{out}_t \rangle)$ , welche die Wahrscheinlichkeit berechnet, dass  $\text{out}_{t+1}$  bezüglich der Ein- und Ausgabehistory ausgegeben wird.
- ▶ Die Funktion  $\hat{I}$  für die Eingabewahrscheinlichkeit sei analog definiert.
- ▶ Ein Wahrscheinlichkeitsmaßstab  $P$  kann nun für jedes mögliche Ereignis ermittelt werden.

# Informationsflussmodelle

Probabilistische Modelle für nichtdeterministische Systeme

- ▶ Im Folgenden werden nur, im high-level Bereich, abgeschlossene Systeme betrachtet. Folglich kann ein high-level Benutzer keine Informationen an einen low-level Benutzer außerhalb des Systems weitergeben. Hierzu dürfen low-level Eingaben nur auf der vorangegangenen low-level History beruhen und low-level Eingaben dürfen statistisch nicht von high-level Eingaben abhängen.

# Informationsflussmodelle

Probabilistische Modelle für nichtdeterministische Systeme

- ▶ Es werden zwei Wahrscheinlichkeitswerte der high-level und low-level Umgebungen H und L definiert:
- ▶  $H(\text{highin}_{t+1} | \langle \text{in}_1, \text{in}_2, \dots, \text{in}_t \rangle, \langle \text{out}_1, \text{out}_2, \dots, \text{out}_t \rangle)$  gibt die Wahrscheinlichkeit einer high-level Eingabe zum Zeitpunkt  $t+1$
- ▶  $L(\text{lowin}_{t+1} | \langle \text{lowin}_1, \text{lowin}_2, \dots, \text{lowin}_t \rangle, \langle \text{lowout}_1, \text{lowout}_2, \dots, \text{lowout}_t \rangle)$  gibt die Wahrscheinlichkeit einer low-level Eingabe zum Zeitpunkt  $t+1$
- ▶  $\hat{I}(\text{in}_{t+1} | \langle \text{in}_1, \text{in}_2, \dots, \text{in}_t \rangle, \langle \text{out}_1, \text{out}_2, \dots, \text{out}_t \rangle) = H(\text{highin}_{t+1} | \langle \text{in}_1, \text{in}_2, \dots, \text{in}_t \rangle, \langle \text{out}_1, \text{out}_2, \dots, \text{out}_t \rangle) \cdot L(\text{lowin}_{t+1} | \langle \text{lowin}_1, \text{lowin}_2, \dots, \text{lowin}_t \rangle, \langle \text{lowout}_1, \text{lowout}_2, \dots, \text{lowout}_t \rangle)$

# Informationsflussmodelle

Probabilistische Modelle für nichtdeterministische Systeme

- ▶ Die AFM-Anforderung für ein Framework  $\langle S, I, C, O \rangle$  mit gegebenem  $\hat{O}$  gilt, wenn alle  $\hat{I}$  die Sicherheitseigenschaft  $P(\text{lowout}_{t+1} | \langle \text{in}_1, \text{in}_2, \dots, \text{in}_t \rangle, \langle \text{out}_1, \text{out}_2, \dots, \text{out}_t \rangle) = P(\text{lowout}_{t+1} | \langle \text{lowin}_1, \text{lowin}_2, \dots, \text{lowin}_t \rangle, \langle \text{lowout}_1, \text{lowout}_2, \dots, \text{lowout}_t \rangle)$  gilt.
- ▶ Die PNI-Anforderung für ein Framework  $\langle S, I, C, O \rangle$  mit gegebenem  $\hat{O}$ , H und L gilt, wenn für alle low-level Ereignisse e für zwei beliebige high-level Umgebungen H1 und H2 gilt, dass  $P_{H1,L}(e) = P_{H2,L}(e)$ .
- ▶ PNI ist ausreichend um sicher zu stellen, dass kein Informationsfluss von einem high-level Benutzer zu einem low-level Benutzer stattfindet. Aber es gibt Systeme, welche PNI aber nicht AFM erfüllen. AFM ist also echt stärker als PNI.