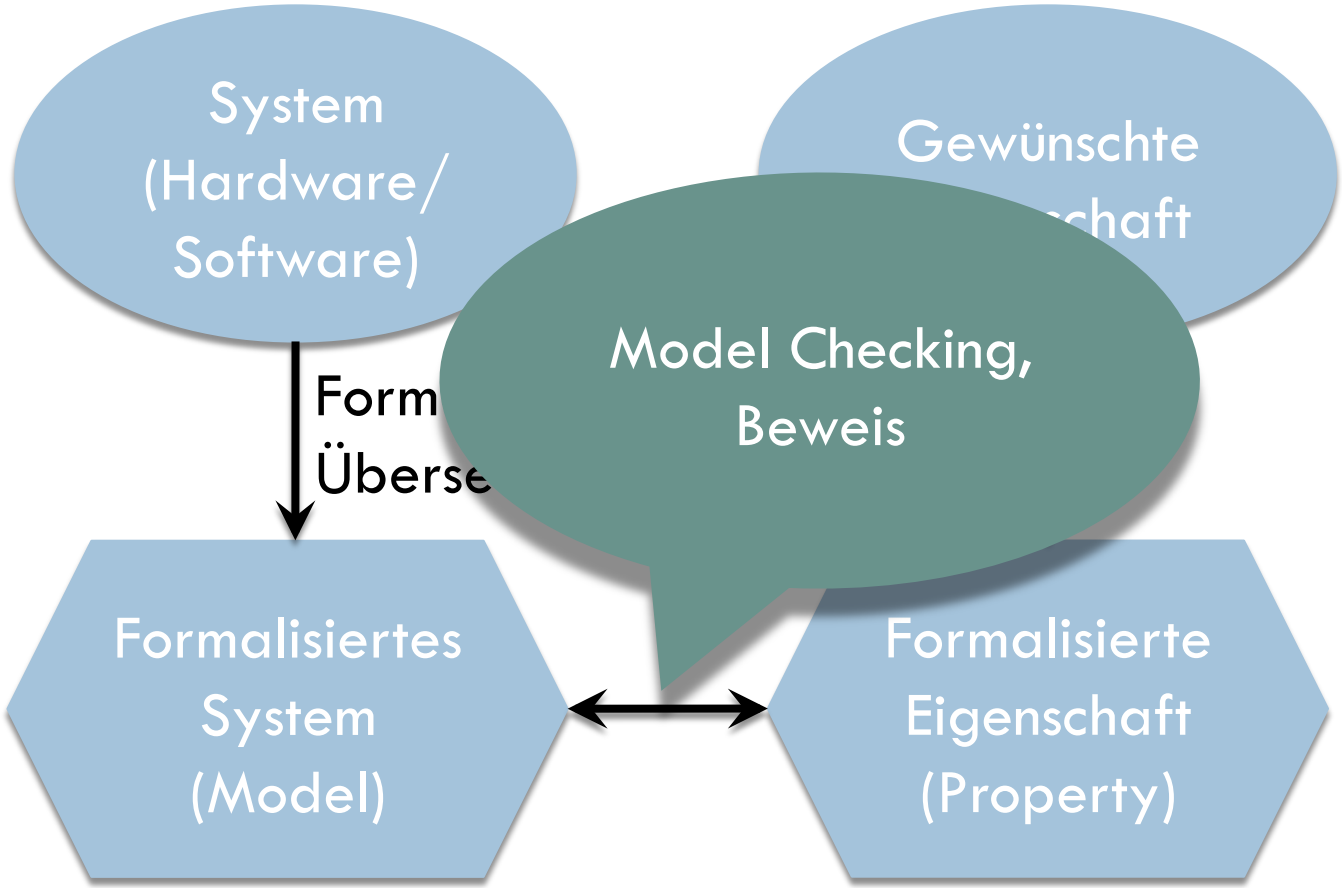
MODEL CHECKING
2 - AUTOMATEN

Sommersemester 2009

Dr. Carsten Sinz, Universität Karlsruhe

Model Checking



Systeme

- Vielfalt an Systemen:
  - Hardware / Software
  - Verschiedene Programmiersprachen
  - Verschiedene Typen von Systemen: reaktiv, offen / geschlossen
- Systembegriff
  - Geht auf Johann Heinrich Lambert (1728-1777) zurück
  - Kybernetik / Informationstheorie / Systembiologie
  - System: Menge von Elementen und deren **Vernetzung**

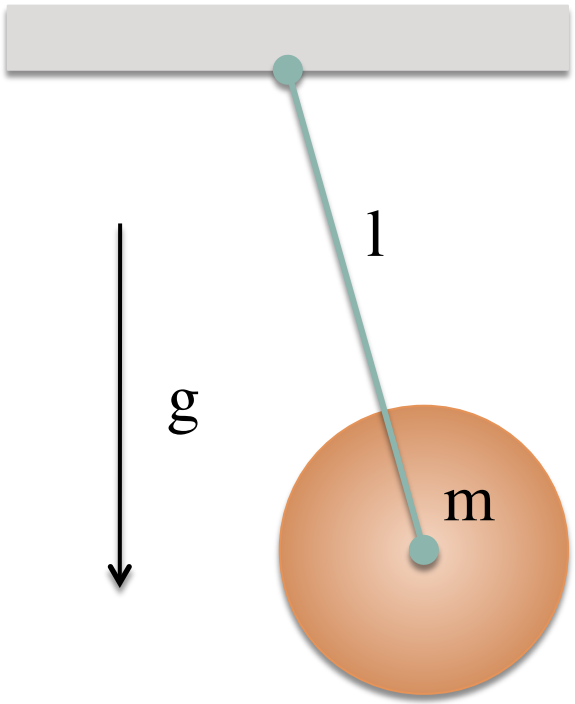
Modellbildung: Naturwissenschaften

Beispiel (mathematisches) Pendel

- *Idealisierungen*:
  - keine Reibung
  - Pendelmasse in einem Punkt konzentriert (Massenmittelpunkt)
  - Nur kleine Auslenkungen betrachten
- Dynamik beschrieben durch Differentialgleichung

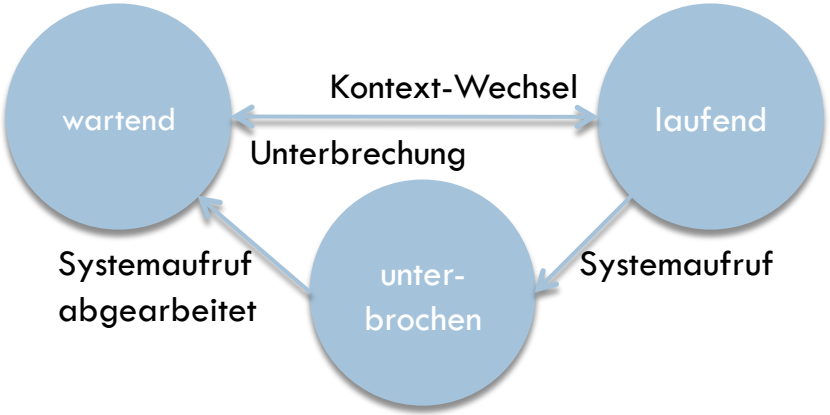
$$m \cdot l \cdot \ddot{\varphi} = -m \cdot g \cdot \sin(\varphi)$$

- Unabhängig von Masse!



Modellbildung

- Abstraktion
  - Z.B. hinsichtlich Implementationsdetails, Zeitverhalten, low-level-Funktionen
  - Restriktion auf bestimmte Komponenten des Systems
- Bsp.: Prozessmodell im Betriebssystem



Modellierungsfragen

- Offenes oder geschlossenes System?
  - **Offen**: Interaktion mit der Umgebung (z.B. Sensorwerte lesen, Netzwerkkommunikation, Ausgabe an Aktoren)
  - **Geschlossen**: keine Interaktion mit der Umgebung (Funktions*berechnung*; Umgebung mit modelliert)
- Grad der Abstraktion
  - Welche Aspekte sind wesentlich?
  - Bsp. Betriebssystem: Swapping? Prozesserzeugung? Fairness? Garantierte Antwortzeiten?

Model Checking

- Grundfrage: *Gilt eine Eigenschaft für ein gegebenes System?*
- Modellierung des Systems als (endlicher) **Automat**
- Modellierung der zu prüfenden Eigenschaft in passender Logik (bzw. ebenfalls als Automat)
- Definition Model Checking:

  Model Checking ist ein Verfahren zur vollautomatischen Verifikation einer Systembeschreibung (Modell) gegen eine Spezifikation (Eigenschaft, Formel).

Endliche Automaten

- **Definition:** Ein endlicher Automat (EA) ist ein Tupel

$$A = (S, I, \Sigma, T, F)$$

- S: Zustandsmenge (häufig endlich)
- I⊆S: Menge der Initialzustände
- ∑: Eingabealphabet
- T⊆S×∑×S: Übergangsrelation
  - Notation: $s \xrightarrow{a} s'$ für $(s, a, s') \in T$ [auch: "$T(s, a, s')$ gilt."]
- F⊆S: Menge der Endzustände

Akzeptierte Sprache eines EA

- **Definition:** Ein endlicher Automat A akzeptiert ein Wort $w \in \sum^*$ genau dann, wenn es eine Sequenz

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \cdots \xrightarrow{a_{n-1}} s_{n-1} \xrightarrow{a_n} s_n$$

  gibt mit $n \geq 0, s_0 \in I, s_n \in F, w = a_1 \cdots a_n$

- **Definition:** Die Sprache L(A) eines Automaten A ist die Menge der von A akzeptierten Worte.
- Verwendung zur Modellierung:
  - Offene Systeme (Verarbeitung von Ereginisstömen), endliche Läufe
  - Prüfung, ob implementierter Ereignisstrom Spezifikation erfüllt

Produktautomat

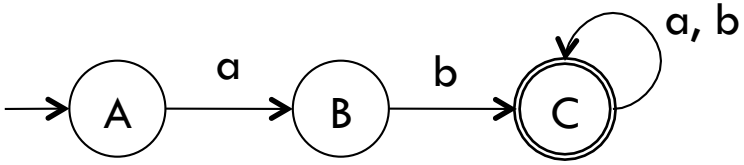
- **Definition:** Der Produktautomat $A = (A_1 \times A_2)$ zweier EAs $A_1 = (S_1, I_1, \Sigma_1, T_1, F_1)$ und $A_2 = (S_2, I_2, \Sigma_2, T_2, F_2)$ mit $\Sigma_1 = \Sigma_2$ ist definiert durch

$$S = S_1 \times S_2 \quad I = I_1 \times I_2$$
$$\Sigma = \Sigma_1 = \Sigma_2 \quad F = F_1 \times F_2$$
$$T((s_1, s_2), a, (s_1', s_2')) \quad \text{gdw.} \quad T_1(s_1, a, s_1') \text{ und } T_2(s_2, a, s_2')$$

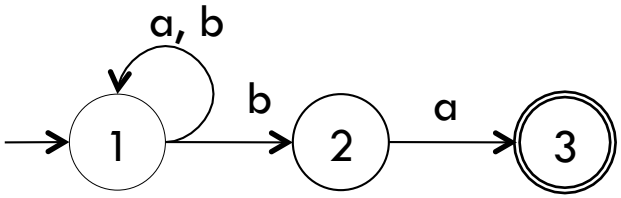
- **Theorem:** Sei $A = (A_1 \times A_2)$. Dann $L(A) = L(A_1) \cap L(A_2)$.
- Beispiel: Automat, der alle Worte mit Präfix *ab* und Suffix *ba* akzeptiert.

Produktautomat: $ab\Sigma^* \cup \Sigma^* ba$

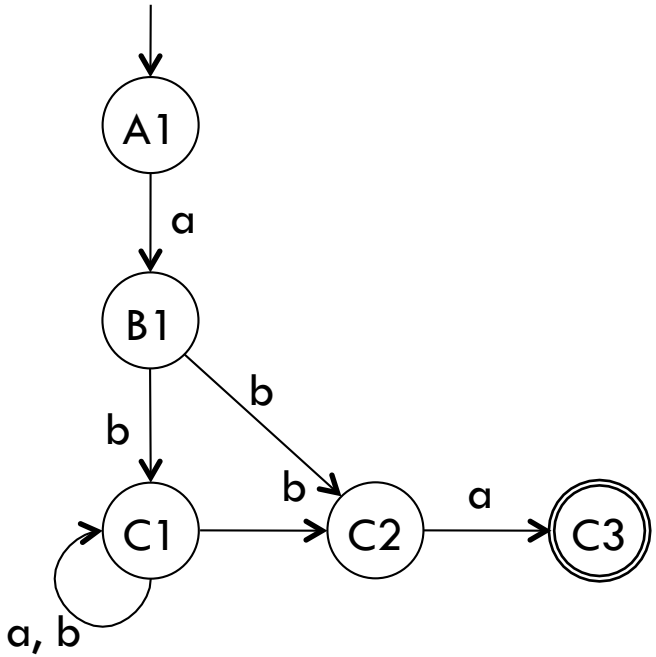
A_1 : $[L(A_1) = ab\Sigma^*]$



A_2 : $[L(A_2) = \Sigma^* ba]$



$A_1 \times A_2$:



Eigenschaften von Endl. Automaten

- **Definition:** Für $s \in S, a \in \Sigma$ ist $s \xrightarrow{a}$ die Menge der Nachfolger von s, d.h. $s \xrightarrow{a} = \{s' \in S \mid T(s, a, s')\}$.
- **Definition:** Ein EA ist *vollständig* gdw. $|I| > 0$ und $|s \xrightarrow{a}| > 0$ für alle $s \in S, a \in \Sigma$.
- **Definition:** Ein EA ist *deterministisch* gdw. $|I| \leq 1$ Und $|s \xrightarrow{a}| \leq 1$ für alle $s \in S, a \in \Sigma$.
- Daher: EA deterministisch und vollständig gdw. $|I| = 1$ und $|s \xrightarrow{a}| = 1$ für alle $s \in S, a \in \Sigma$.

Potenzautomat, Teilmengenkonstruktion

- **Definition:** Der Potenzautomat $A_p = \mathbb{P}(A)$ eines EA A ist wie folgt definiert:

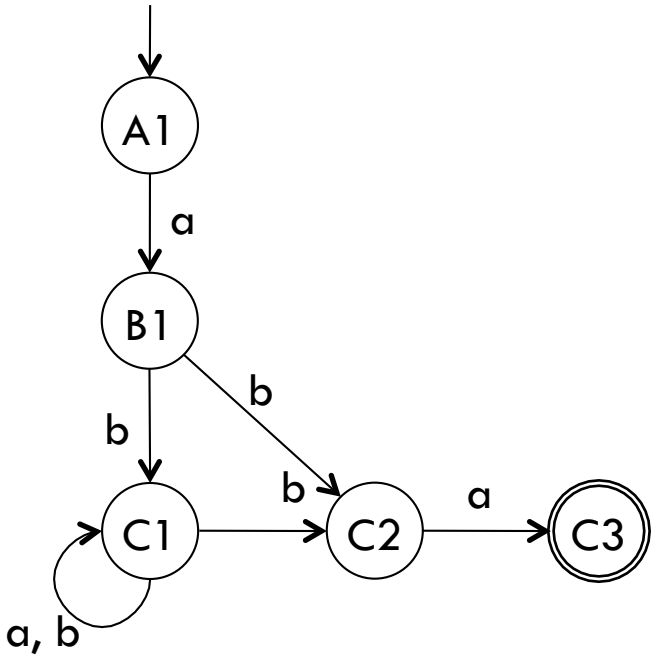
$$S_p = \mathbb{P}(S) \qquad I_p = \{I\}$$
$$\Sigma_p = \Sigma \qquad F_p = \{F' \subseteq S \mid F' \cap F \neq \emptyset\}$$

$$T_p(S', a, S'') \quad \text{gdw.} \quad S'' = \{s'' \in S \mid \exists s' \in S' \text{ mit } T(s', a, s'')\}$$

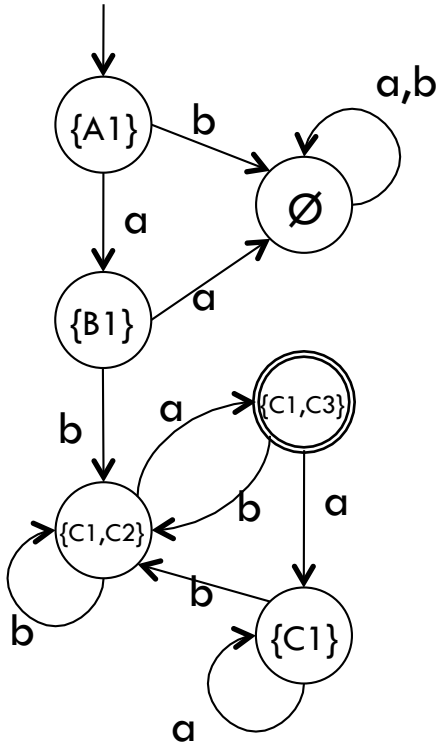
- **Satz:** Sei $A_p = \mathbb{P}(A)$ der Potenzautomat eines EA A. Dann ist $L(A_p) = L(A)$ und A_p ist deterministisch und vollständig.

Beispiel Potenzautomat

A :



A_p :



Komplementautomat

- **Definition:** Der Komplementautomat $A_k = K(A)$ eines EA A gleicht diesem bis auf die Menge der Finalzustände, für welche $F_k = S \setminus F$ gilt.
- **Satz:** Der Komplementautomat $A_k = K(A)$ ist deterministisch und vollständig, sofern A deterministisch und vollständig ist. Außerdem gilt $L(A_k) = \overline{L(A)} = \Sigma^* \setminus L(A)$.

Model Checking mit Endl. Automaten

- Idee:
  - Modellierung **und** Spezifikation mit Endl. Automaten
  - Ereignisströme einer Implementierung repräsentiert durch Automat A_I.
  - (Partielle) Spezifikation zulässiger Ereignisströme als Automat A_S.
- Model Checking: Stimmt Impl. mit Spez. überein?
  - $L(A_I) \subseteq L(A_S)$
  - gdw. $L(A_I) \cap \overline{L(A_S)} = \emptyset$
  - gdw. $A_I \times K(\mathbb{P}(A_S))$ besitzt keine erreichbaren Finalzustände

Algorithmus: MC mit Endl. Automaten

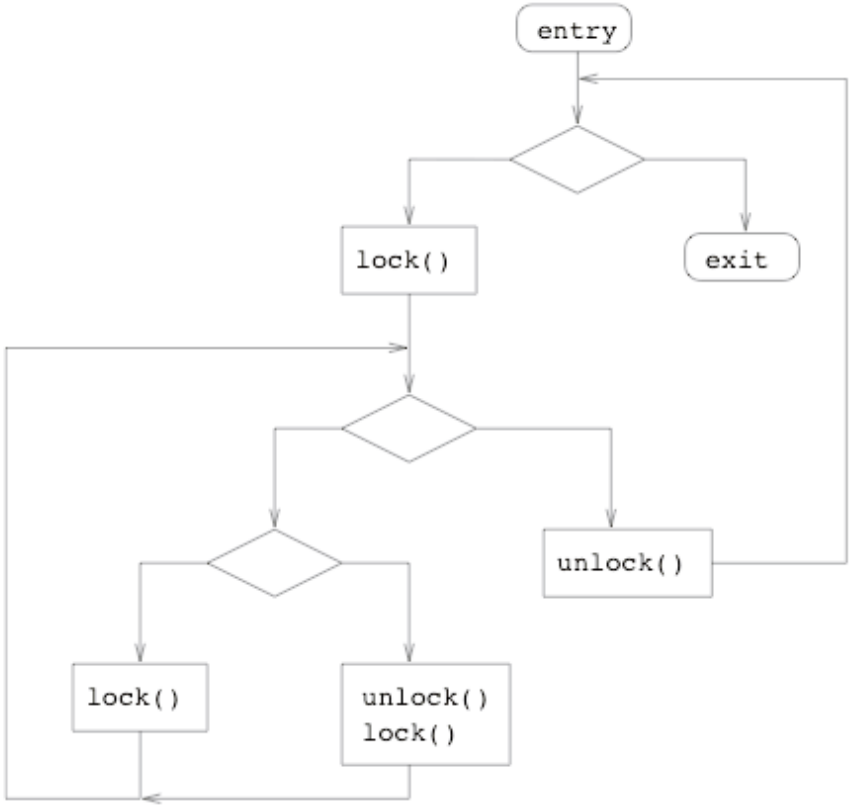
- Algorithmus:
  1. Gegeben: Implementation A_I , Spezifikation A_S .
  2. Berechne Produktautomat $A_{\mathrm{MC}} = A_I \times K(\mathbb{P}(A_S))$.
  3. Prüfe, ob es erreichbare Finalzustände in A_{MC} gibt.
- Wie kann Erreichbarkeitsanalyse durchgeführt werden?
  - Tiefensuche (DFS) in Automat A_{MC} .
- Beispiel: Implementation $I = ((s|c)^2)^*$, Spezifikation $S = (cs|sc|ss)^*$.

Beispiele Spezifikation

- Zeitliche (temporale) Eigenschaften:
  - Jedes 3. Zeichen im Eingabestrom ist a: $(\Sigma \cdot \Sigma \cdot a)^*$
  - Genau jedes 3. Zeichen ist a: $(\overline{a} \cdot \overline{a} \cdot a)^*$
  - Einem a (acknowledge) muss ein r (request) vorangehen: $\overline{\overline{r}^* \cdot a}$
  - Jedem a (acknowledge) muss ein r (request) vorangehen: $\overline{(\Sigma^* \cdot a)^* \cdot \overline{r}^* \cdot a}$
- Verfeinerung: (Prozessplanung für 3 Proz. a, b, c)
  - *Round robin*: $(abc|acb|bac|bca|cab|cba)^*$
  - *Round robin*, a höhere Priorität als b?

Beispiel: Locking


```
while (...) {
  lock ();
  ...
  while (...) {
    if (...) {
      lock ();
      ...
    } else {
      unlock ();
      ...
      lock ();
    }
    ...
  }
  ...
  unlock();
}
```



$$(l \cdot (l \mid u \cdot l)^* \cdot u)^*$$

[Quelle: A. Biere]

Büchi-Automaten

- **Definition:** Ein Büchi-Automat ist ein Tupel

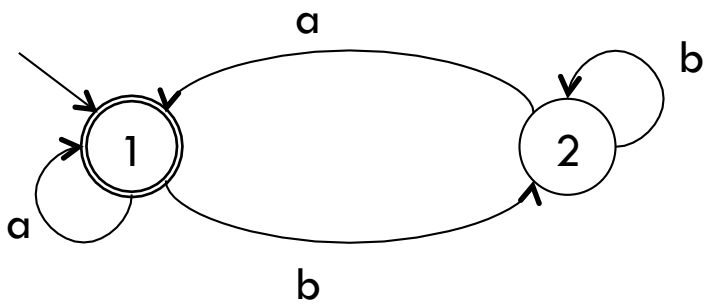
$$A = (S, I, \Sigma, T, F)$$

- S: Zustandsmenge (häufig endlich)
- I⊆S: Menge der Initialzustände
- ∑: Eingabealphabet
- T⊆S×∑×S: Übergangsrelation
  - Notation: $s \xrightarrow{a} s'$ für $(s, a, s') \in T$ [auch: "$T(s, a, s')$ gilt."]
- F⊆S: Menge der ***akzeptierenden*** *Zustände*

Akzeptierte Sprache eines Büchi-Automaten

- Läufe von Büchi-Automaten sind immer unendlich!
- Eingabeworte: $w \in \Sigma^\omega$ (unendlich lange Worte)
- **Definition**: Sei $\inf(\rho)$ die Menge der Zustände, die unendlich oft in einem Lauf ρ auftreten. Ein Büchi-Automat A akzeptiert ein Wort $w \in \Sigma^\omega$ genau dann, wenn auf dem zu w gehörigen Lauf ρ ein Finalzustand unendlich oft auftritt, d.h. $\inf(\rho) \cap F \neq \emptyset$
- **Definition**: Die Sprache L(A) eines Büchi-Automaten A ist die Menge der von A akzeptierten Worte.

Beispiel Büchi-Automat



Wird $(ab)^\omega$ akzeptiert? ja

Akzeptierte Sprache? $(b^* \cdot a)^\omega$

ω-reguläre Ausdrücke

Worte, in denen a unendlich oft vorkommt

Eigenschaften von Büchi-Automaten

- Abgeschlossen unter Schnitt und Komplement (wie Endliche Automaten)
- Berechnung des Komplements kompliziert

Model-Checking mit Büchi-Automaten

- Gleiche Idee wie bei endlichen Automaten
  - Modellierung **und** Spezifikation mit Büchi-Automaten
  - Ereignisströme einer Implementierung repräsentiert durch Büchi-Automat A_I .
  - (Partielle) Spezifikation zulässiger Ereignisströme als Büchi-Automat A_S.
- Model Checking: Stimmt Impl. mit Spez. überein?
  - $L(A_I) \subseteq L(A_S)$
  - gdw. $L(A_I) \cap \overline{L(A_S)} = \emptyset$

Algorithmus: MC mit Büchi-Automaten

- Wie im Falle von Endlichen Automaten
  - Notwendige Sub-Routinen: Komplement, Produkt von Büchi-Automaten; Test auf leere Sprache

$$L(A_I) \cap \overline{L(A_S)} = \emptyset$$

- Schnitt-Berechnung über Produktautomat
- Komplement-Berechnung schwierig
  - Daher häufig direkte Angabe des Automaten $A_{\overline{S}}$, der fehlerhafte Läufe beschreibt (d.h. $L(A_{\overline{S}}) = \overline{L(A_S)}$).

Produkt-Büchi-Automat

- **Definition:** Der Produktautomat $A = (A_1 \times A_2)$ zweier Büchi-Automaten $A_1 = (S_1, I_1, \Sigma_1, T_1, F_1)$ und $A_2 = (S_2, I_2, \Sigma_2, T_2, F_2)$ mit $\Sigma_1 = \Sigma_2$ ist definiert durch

$$S = S_1 \times S_2 \times \{0, 1, 2\} \qquad I = I_1 \times I_2 \times \{0\}$$

$$\Sigma = \Sigma_1 = \Sigma_2 \qquad F = F_1 \times F_2 \times \{2\}$$

Für die Übergangsrelation gilt $((s_1, s_2, x), a, (s_1', s_2', y)) \in T$, falls

- $(s_1, a, s_1') \in T_1$ und $(s_2, a, s_2') \in T_2$;
- wenn $x = 0$ und $s_1' \in F_1$, dann $y = 1$;
- wenn $x = 1$ und $s_2' \in F_2$, dann $y = 2$;
- wenn $x = 2$, dann $y = 0$; ansonsten : $y = x$.