
Vorlesung „Methoden des Software Engineering“

Block C „Formale Methoden“

Büchi-Automaten und Modellprüfung

Martin Wirsing

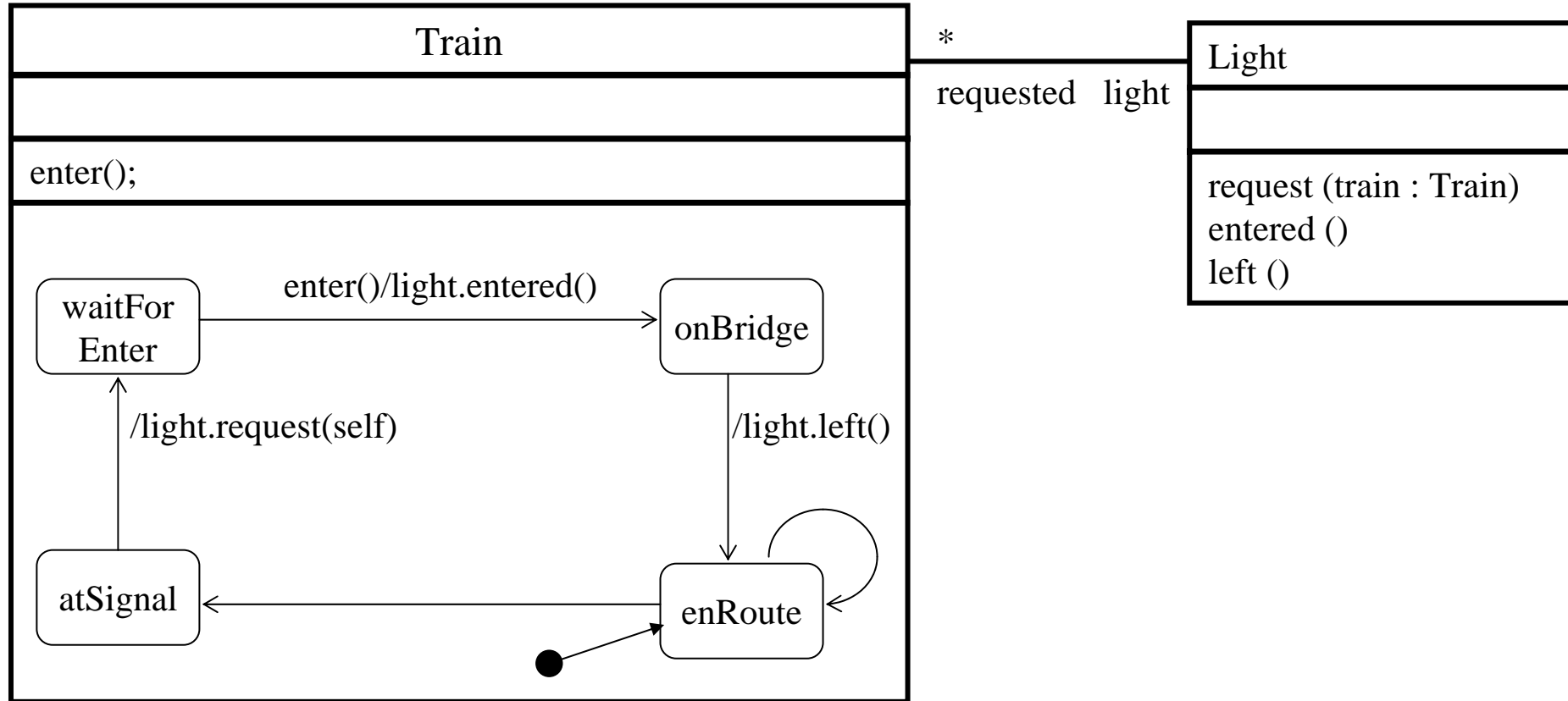
Einheit C.4, 12.12.2006

Ziele

- ω -Automaten zur operationellen Beschreibung reaktiver Systeme kennen lernen
- Zusammenhang zwischen LTL-Formeln und ω -Sprachen untersuchen
- Anwendung automatentheoretischer Verfahren beim Modelchecking kennen lernen

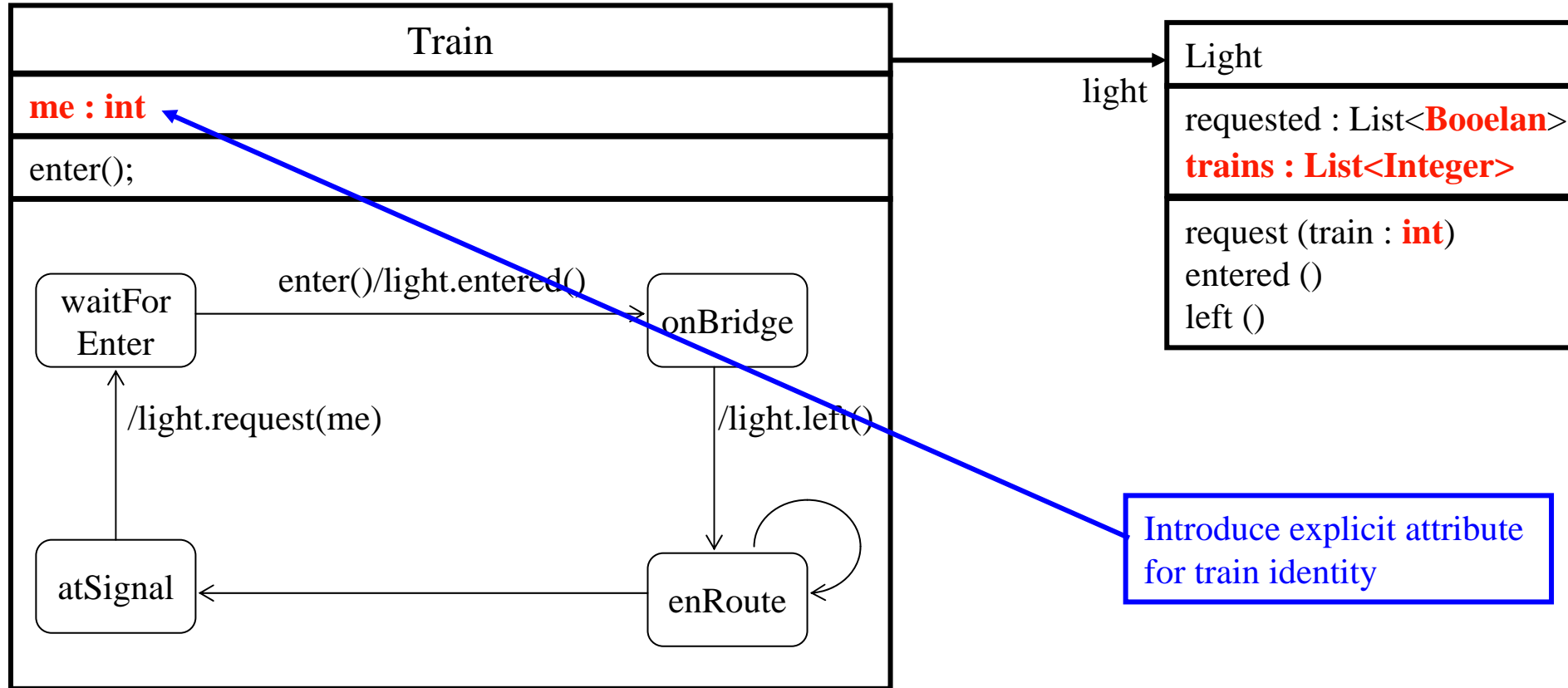
Beispiel Eisenbahnsteuerung: Statechart

Objekt-orientierte Modellierung (Wiederholung)



Beispiel Eisenbahnsteuerung: Statechart

Transformation in Modellierung mit expliziten Zugidentifikatoren



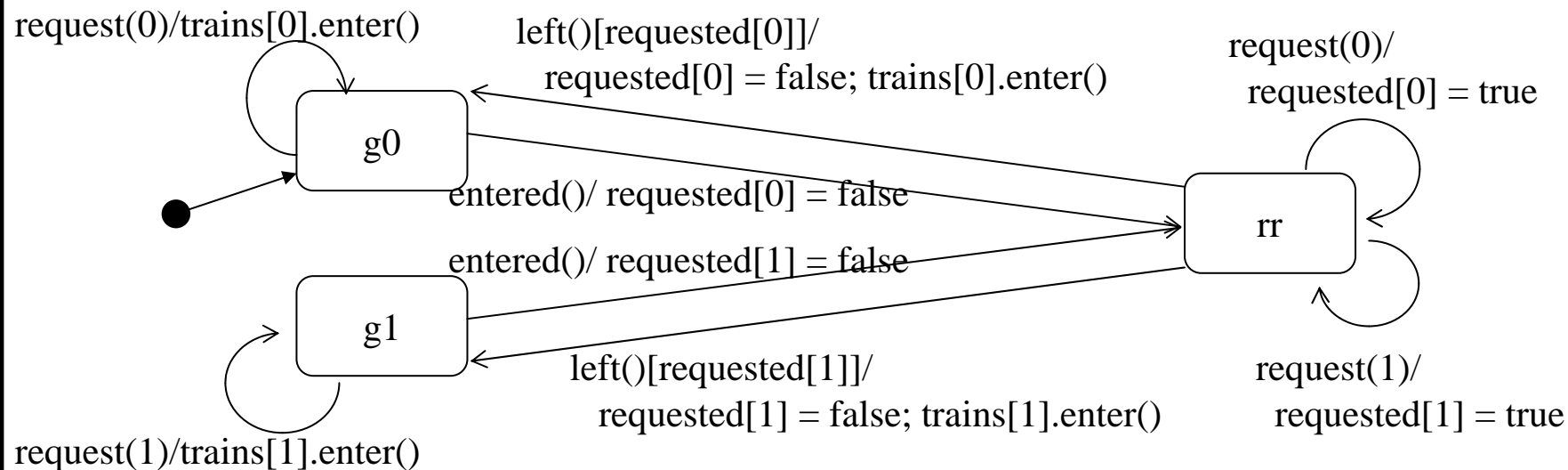
Beispiel Eisenbahnsteuerung: Statechart

Modellierung mit expliziten Zugidentifikatoren

Light

requested : List<Integer>
trains : List<Integer>

request (train : boolean)
entered ()
left ()



Beispiel Eisenbahnsteuerung: Eigenschaften

Modellprüfung folgender Eigenschaften:

- Zwei Züge sind niemals gleichzeitig auf der Brücke:

$$\square \neg(t1.ocInState(onBridge) \wedge t2.ocInState(onBridge))$$

- Jeder Zug kommt irgendwann auf die Brücke

$$\langle \rangle t0.inState(OnBridge) \text{ and } \langle \rangle t1.inState(OnBridge);$$

- Der Zug t1 ist immer wieder nicht auf der Brücke:

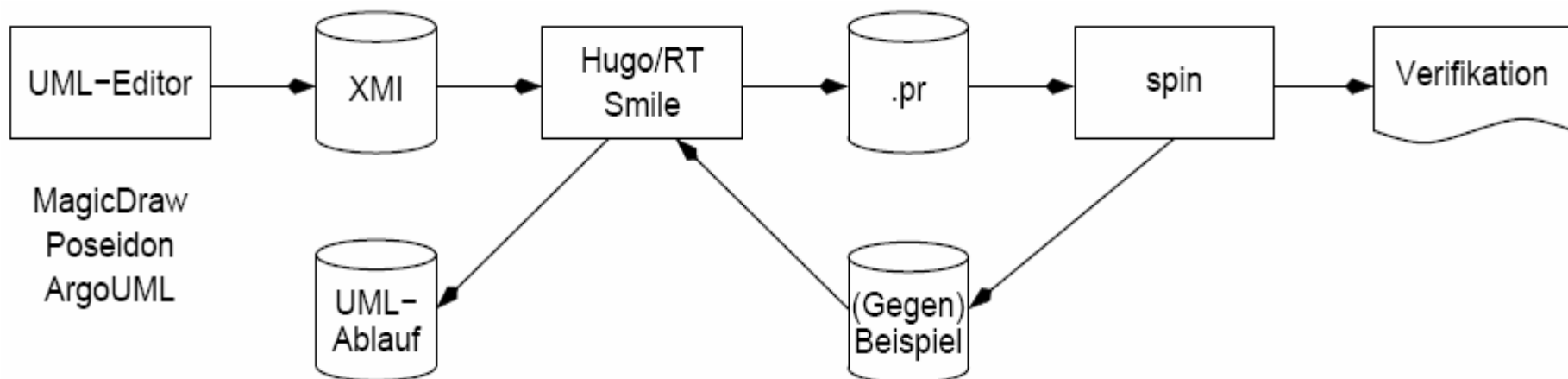
$$\square \langle \rangle \neg t1.ocInState(onBridge)$$

- Immer wenn t1 am Signal wartet, wird t1 irgendwann später auf der Brücke sein:

$$\square(t1.ocInState(atSignal) \Rightarrow \langle \rangle t1.ocInState(onBridge))$$

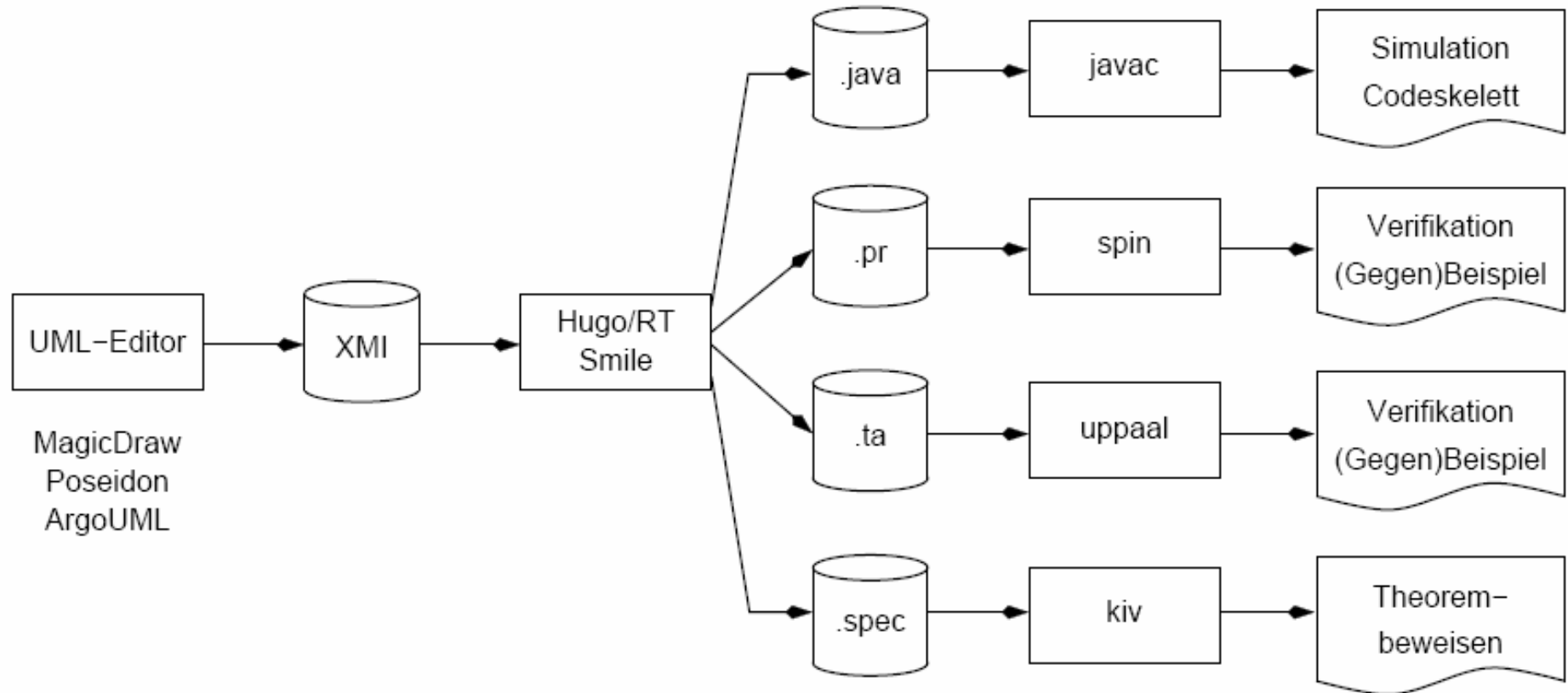
Modellprüfung mit Hugo

- **Ziel**
 - Automatisierte Analyse des Verhaltens einer Menge von aktiven Objekten
- **Ansatz: UML-Modelchecking**
 - Compilation von UML-Zustandsmaschinen für off-the-shelf Modelchecker
 - Eigenschaften: Deadlock, Invarianten, temporale Eigenschaften
 - Konsistenz: UML-Interaktionen
- **Werkzeug: Hugo/RT (Knapp, Merz, ab 2001)**



Hugo/RT Überblick

Gemeinsames semantisches Modell für Verifikation und Codegenerierung



<http://www.pst.ifi.lmu.de/projekte/hugo/>

Modellprüfung mit HUGO hier

- Textuelle Darstellung der Zustandsmaschine im textueller Zwischensprache Ute
- Darstellung der zu testenden Konfiguration und der LTL-Eigenschaften in Ute
- Aufruf von Spin

Ute-Darstellung der Eisenbahnklassen

```
model Bridge {
  class Train {
    signature {
      attr me : int;
      attr light : Light;
      reception enter();
    }
    behaviour {
      states {
        initial Init;
        simple WaitForEnter; simple OnBridge; simple EnRoute; simple AtSignal;
      }
      transitions {
        Init -> EnRoute;
        WaitForEnter -> OnBridge {
          trigger enter;
          effect light.entered();
        }
        OnBridge -> EnRoute {
          trigger wait;
          effect light.left();
        }
        ...
      }
    }
  }
}
```

Ute-Darstellung der Eisenbahnklassen

```
class Light {
  signature {
    attr trains : Train[2];
    attr requested : boolean[2];
    reception request(train : int); reception entered(); reception left();
  }
  behaviour {
    states {
      initial Init;
      simple Green0; simple Green1; simple Red { defer left; }
    }
    transitions {
      Init -> Green0;
      Green0 -> Green0 { trigger request; guard train == 0; effect trains[0].enter(); }
      Green0 -> Red { trigger entered; effect requested[0] = false; }
      Red -> Red { trigger request; effect requested[train] = true; }
      Red -> Green0 { trigger left; guard requested[0];
        effect requested[0] = false; trains[0].enter(); }
    }
  }
}
```

Ute-Darstellung der Kollaboration (Objekte und Eigenschaften)

```
collaboration test {  
  object t0 : Train {  
    me = 0;  
    light = a;  
  }  
  object t1 : Train {  
    me = 1;  
    light = a;  
  }  
  object a : Light {  
    trains = { t0, t1 };  
    requested = { false, false };  
  }  
  
  assertion safety {  
    G not (t0.inState(OnBridge) and t1.inState(OnBridge));  
  }  
}
```

Automatentheoretisches Modelchecking

Modellprüfung wird zurückgeführt auf ein **automatentheoretisches Problem**:
Prüfung auf Leerheit der Sprache eines Büchi-Automaten, der die
Zustandsmaschine und die Negation der zu prüfenden Formel beschreibt.

$$\begin{aligned} M \models \varphi \\ \text{gdw.} \\ \mathcal{L}(M) \subseteq \mathcal{L}(\varphi) \\ \text{gdw.} \\ \mathcal{L}(M) \cap \mathcal{L}(\neg\varphi) = \emptyset \\ \text{gdw.} \\ \mathcal{L}(M \times \mathcal{B}_{\neg\varphi}) = \emptyset \end{aligned}$$

ω -Automaten

Alphabet	endliche Menge Σ
ω -Wörter	$\alpha = a_0 a_1 \dots$ ($a_i \in \Sigma$) „unendlich“ lange Wörter
ω -Sprachen	Menge $L \subseteq \Sigma^\omega$ von ω -Wörtern
Speziell	$\Sigma = 2^V$ interpretiert als auss.log. Belegungen über V (endlich) ω -Wörter über Σ entsprechen LTL-Strukturen

Endliche Automaten über ω -Wörtern

- **Büchi-Automat (über Alphabet Σ) $B = (Q, I, \delta, F)$**

Q	endliche Menge von (Automaten-) Zuständen
$I \subseteq Q$	Anfangszustände
$\delta \subseteq Q \times \Sigma \times Q$	Übergangsrelation
$F \subseteq Q$	akzeptierende Zustände

wie
endlicher
Automat

- **Ablauf von B über ω -Wörtern**

Ablauf $\rho = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \dots$

Initialisierung $q_0 \in I$

Übergänge $(q_i, a_i, q_{i+1}) \in \delta$ für alle $i \in \mathbb{N}$

Akzeptanz $q_i \in F$ für unendlich viele $i \in \mathbb{N}$

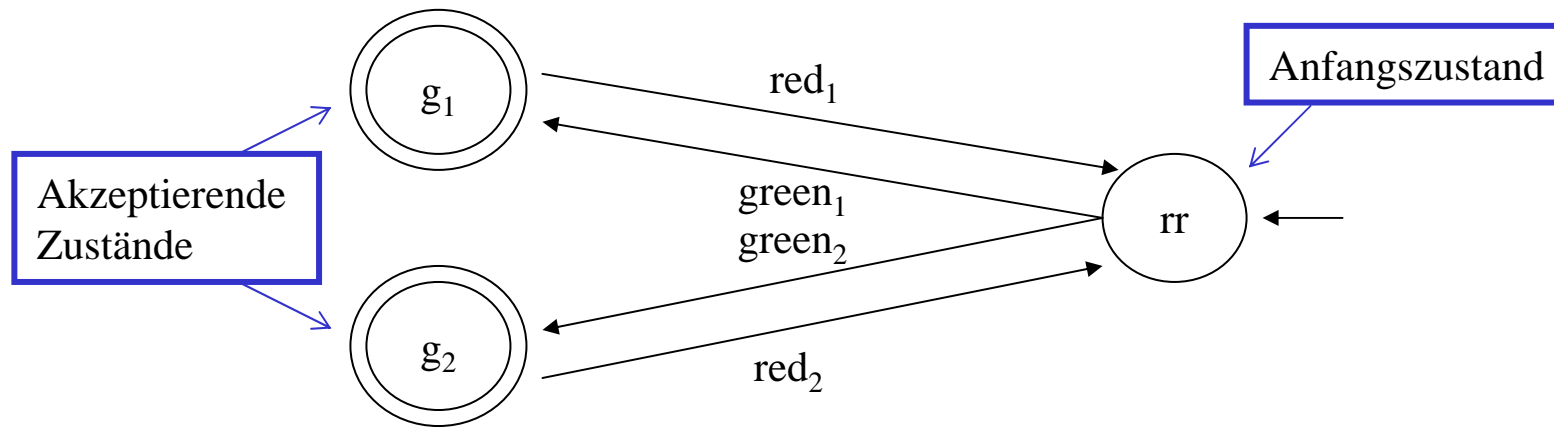
- **Sprache**

$L(B) = \{\alpha \in \Sigma^\omega : \text{es gibt einen akzeptierenden Ablauf von B über } \alpha = a_0 a_1 \dots\}$

ω -reguläre Sprachen:

Klasse der durch Büchi-Automaten definierbaren ω -Sprachen

Büchi-Automaten: Beispiel Ampel

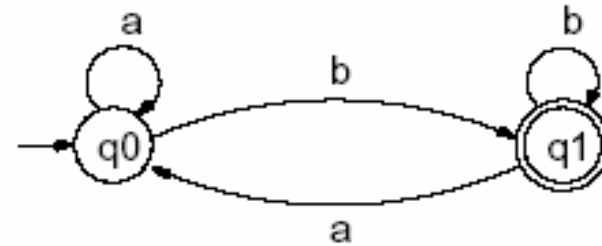


Der Büchi-Automat akzeptiert alle ω -Wörter

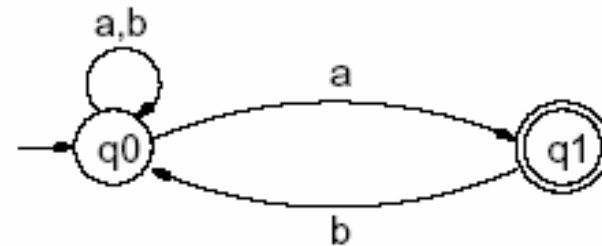
- über dem Alphabet $\{\text{red}_1, \text{red}_2, \text{green}_1, \text{green}_2\}$,
- die unendlich viele green_1 und unendlich viele green_2 enthalten.

Büchi-Automaten: Beispiele

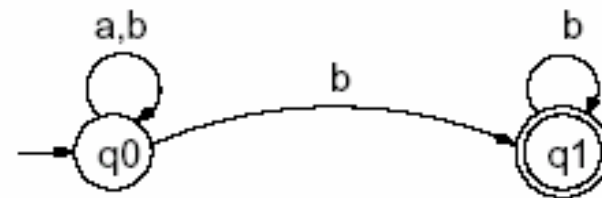
unendlich oft `b`



unendlich oft `ab`



irgendwann nur noch `b`



Von LTL zu Büchi-Automaten

Prinzip

- $L(\varphi)$ bezeichne Menge der Zustandsfolgen, die φ erfüllen
- Konstruiere Automat B_φ mit $L(B_\varphi) = L(\varphi)$ (über Alphabet 2^\vee)

Grundidee der Konstruktion

1) Überführe φ in **Negative Normalform**, bei der nur atomare Formeln negiert werden können.

2) **Konstruiere Büchi-Automaten:**

Zustände Mengen von „Teilformeln“ von φ , die „jetzt“ erfüllt werden müssen

Anfangszustände Zustände, die φ enthalten

Übergangsrelation

- sichert Erfülltheit nicht-temporalen Formeln im Ausgangszustand
- ersetzt temporale Formeln im Ausgangs- durch andere im Zielzustand

Akzeptierende Zustände definiert aus „eventualities“ $\diamond \varphi$ oder φ until ψ

3) **Optimierung:** Entferne überflüssige Kanten und nicht erreichbare Zustände

Übersetzung von LTL-Formeln in Büchi-Automaten

LTL-Formel φ in NNF \rightsquigarrow verallgemeinerter Büchi-Automat $\mathcal{B}_\varphi = (Q, I, \delta, \mathcal{F})$

Alphabet $\Sigma = 2^{\mathcal{V}}$

Zustände $Q \subseteq 2^{cl(\varphi)}$ mit $q \in Q \iff$

1. $false \notin q$
2. $\varphi_1 \wedge \varphi_2 \in q \Rightarrow \varphi_1 \in q$ und $\varphi_2 \in q$
3. $\varphi_1 \vee \varphi_2 \in q \Rightarrow \varphi_1 \in q$ oder $\varphi_2 \in q$

Anfangszustände $I = \{q \in Q \mid \varphi \in q\}$

Transitionsrelation $(q, P, q') \in \delta \iff$

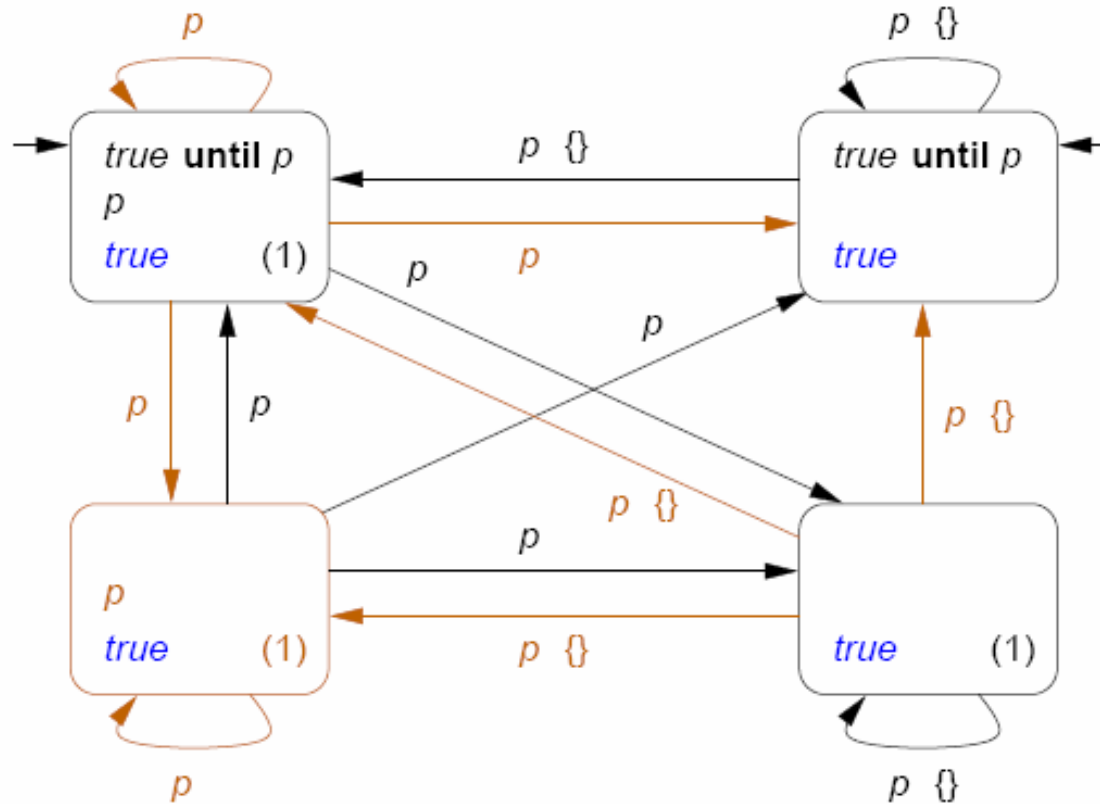
1. $p \in q \Rightarrow p \in P$, $\neg p \in q \Rightarrow p \notin P$
2. $\circ\varphi_1 \in q \Rightarrow \varphi_1 \in q'$
3. $\varphi_1 \text{ until } \varphi_2 \in q \Rightarrow (\varphi_2 \in q)$ oder $((\varphi_1 \in q)$ und $(\varphi_1 \text{ until } \varphi_2 \in q'))$.
4. $\varphi_1 \text{ release } \varphi_2 \in q \Rightarrow (\varphi_2 \in q)$ und $((\varphi_1 \in q)$ oder $(\varphi_1 \text{ release } \varphi_2 \in q'))$.

Akzeptanzmengen $\mathcal{F} = \{F_{\varphi_1 \text{ until } \varphi_2} \subseteq Q \mid \varphi_1 \text{ until } \varphi_2 \in cl(\varphi)\}$ mit

$$F_{\varphi_1 \text{ until } \varphi_2} = \{q \in Q \mid (\varphi_2 \in q \wedge \varphi_1 \text{ until } \varphi_2 \in q) \vee (\varphi_1 \text{ until } \varphi_2 \notin q)\}$$

Übersetzung von LTL-Formeln in Büchi-Automaten

Übersetzung von $\varphi = \diamond p = \text{true until } p$



Optimierung Seien $(q, P, q_1), (q, P, q_2) \in \delta$ mit $q_2 \subseteq q_1$, sodaß für alle $\varphi_1 \text{ until } \varphi_2 \in q_1$ gilt: Ist $\varphi_1 \text{ until } \varphi_2 \in q_2$ und $\varphi_2 \in q_1$, so ist auch $\varphi_2 \in q_2$. Dann kann (q, P, q_1) aus δ gestrichen werden.

Automatentheoretisches Modelchecking

Transitionssystem aufgefasst als Büchi-Automat, bei dem alle Zustände akzeptieren

Automatentheoretisches Modelchecking

$$\begin{aligned} M \models \varphi \\ \text{gdw.} \\ \mathcal{L}(M) \subseteq \mathcal{L}(\varphi) \\ \text{gdw.} \\ \mathcal{L}(M) \cap \mathcal{L}(\neg\varphi) = \emptyset \\ \text{gdw.} \\ \mathcal{L}(M \times \mathcal{B}_{\neg\varphi}) = \emptyset \end{aligned}$$

Komplexität: $O(|M| \cdot |\mathcal{B}_{\neg\varphi}|) = O(|M| \cdot 2^{|\varphi|})$ $|M|$ i.a. kritischer als $2^{|\varphi|}$!

Zustandsexplosion

- Produktautomat $M \times B_{\neg\varphi}$ ist zu groß

problematisch ab ca. 10^6 Zuständen (des Produktautomaten)

- Lösungen

- Reduktion ignoriere irrelevante Teile des Zustandsraums
- Kompression kompakte Darstellung von $M \times B_{\neg\varphi}$ berechnen
- Abstraktion Projektion auf kleineren Zustandsraum (von M)

Zusammenfassung

- ω -Automaten sind eine Sprache zur operationellen Beschreibung reaktiver Systeme
- Zu jeder LTL-Formel ϕ gibt es einen Büchi-Automaten, der genau die Zustandsfolgen akzeptiert, die ϕ erfüllen.
- Beim Modelchecking wird die Gültigkeit einer temporal-logischen Formel f in einem Transitionssystem M durch Test auf die Leerheit des Büchi-Automaten $M \times \mathcal{B}_{\neg\phi}$ überprüft.

Appendix: LTL-Formeln in Negationsnormalform

Reduktion der LTL-Syntax auf \mathcal{V} , *true*, *false*, \neg , \wedge , \vee , \circ , **until**, **release**

- **release** dual zu **until**
- $\sigma \models \varphi \text{ **release** } \psi \iff$ für alle $i \geq 0$: falls $\sigma|_j \not\models \varphi$ für alle $j < i$, dann $\sigma|_i \models \psi$
- $\diamond\varphi = (\textit{true} \text{ **until** } \varphi)$, $\square\varphi = (\textit{false} \text{ **release** } \varphi)$

Negation nur vor atomaren Formeln, insbesondere

$$nnf(\neg p) = \neg p$$

$$nnf(\neg\neg\varphi) = nnf(\varphi)$$

$$nnf(\neg(\varphi \wedge \psi)) = nnf(\neg\varphi) \vee nnf(\neg\psi)$$

$$nnf(\neg(\varphi \vee \psi)) = nnf(\neg\varphi) \wedge nnf(\neg\psi)$$

$$nnf(\neg\circ\varphi) = \circ nnf(\neg\varphi)$$

$$nnf(\neg(\varphi \text{ **until** } \psi)) = nnf(\neg\varphi) \text{ **release** } nnf(\neg\psi)$$

$$nnf(\neg(\varphi \text{ **release** } \psi)) = nnf(\neg\varphi) \text{ **until** } nnf(\neg\psi)$$

Appendix: Abschluss von LTL-Formeln in Negationsnormalform

LTL-Formel φ in NNF Teilformelmengemenge (Abschluß) $cl(\varphi)$

$$\varphi \in cl(\varphi)$$

$$\varphi_1 \wedge \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$$

$$\varphi_1 \vee \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$$

$$\circ\varphi_1 \in cl(\varphi) \Rightarrow \varphi_1 \in cl(\varphi)$$

$$\varphi_1 \text{ **until** } \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$$

$$\varphi_1 \text{ **release** } \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$$

Bsp.: $cl(\text{true until } \neg p) = \{\text{true until } \neg p, \text{true}, \neg p\}$

Literatur

- Wirsing: Grundlagen der Systementwicklung, Kap. 8
- Knapp: Model Checking Praktikum,
Büchi-Automaten, Modelchecking LTL
- Kastens: Petrinetze, Vorlesungsmitschrift
- G. Goos: Vorlesungen über Informatik, Band 1, Springer-Verlag,
1995