Formale Modellierung
Vorlesung 12 vom 05.07.2015: Temporale Logik und Modellprüfung

Christoph Lüth

Universität Bremen

Sommersemester 2015

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung von Software
  - Temporale Logik und Modellprüfung
  - Zusammenfassung, Rückblick, Ausblick

Organisatorisches

- Übung am Donnerstag fällt aus — Ersatztermin?

Tagesmenu: Temporale Logik und Modellprüfung

- Modellierung des Programmes als endliche Zustandsmaschine
- Abstraktion über Zuständen, Zustandsübergang als primäres Konzept
- Temporale Logik: Logik über Pfade von Zustandsübergängen
  - Temporal im Sinne von *tempus fugit*

Endliche Zustandsmaschine

Definition (Finite State Machine, FSM)

Eine FSM ist $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$ mit

- Σ eine endliche Menge von Zuständen, und
- $\rightarrow \subseteq \Sigma \times \Sigma$ eine Zustandsübergangsrelation, mit $\rightarrow$ linkstotal:

$$\forall s \in \Sigma. \exists s' \in \Sigma. s \rightarrow s'$$

- Varianten dieser Definition: Anfangszustände; Zustandsvariablen oder benannte Zustandsübergänge
- NB: Kein Endzustand, und keine Ein/Ausgabe (Unterschied zu Automaten)
- Wenn $\rightarrow$ eine Funktion ist (rechtseindeutig), dann ist die FSM deterministisch, ansonsten nicht-deterministisch.
- Jede nicht-deterministische FSM kann durch die Power-State-Konstruktion deterministisch gemacht werden.

Einfaches Beispiel

- Getränkemaschine für Kaffee
- Nimmt $10c$ oder $20c$ Münzen
- Kleiner Kaffe $10c$, großer Kaffee $20c$
- Nimmt nicht mehr als zwei Münzen
- Geldrückgabe

Linear Temporal Logic (LTL) and Pfade

- LTL ist die Logik über Ausführungspfade in einer FSM.
- Wir definieren erst Pfade, dann LTL-Formeln, dann eine Erfülltheitsrelation.

Definition (Pfade)

Für eine FSM $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$ ist ein Pfad in $\mathcal{M}$ eine (unendliche) Sequenz $\langle s_1, s_2, s_3, \ldots \rangle$ mit $s_i \in \Sigma$ und $s_i \rightarrow s_{i+1}$ für alle i.

- Notation: Sei $p = \langle s_1, s_2, s_3, \ldots \rangle$ ein Pfad, dann ist $p_i \stackrel{def}{=} s_i$ (Selektion) und $p^i \stackrel{def}{=} \langle s_i, s_{i+1}, \ldots \rangle$ (Suffix ab Position i).

Lineare Temporale Logik (LTL)

$$
\begin{array}{rll}
\phi ::= & \top \mid \bot \mid q & \text{— True, false, atomar} \\
\mid & \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \longrightarrow \phi_2 & \text{— Aussagenlog. Formeln} \\
\mid & \mathsf{X}\,\phi & \text{— Nächster Zustand} \\
\mid & \mathsf{F}\,\phi & \text{— Irgendwann} \\
\mid & \mathsf{G}\,\phi & \text{— Immer} \\
\mid & \phi_1\, U\, \phi_2 & \text{— Bis}
\end{array}
$$

- Präzendenzen: unäre Operatoren; dann U; dann $\wedge$, $\vee$; dann $\longrightarrow$.
- Eine atomare Formel p ist ein Zustandsprädikat. Andere (äquivalente) Möglichkeit: Zustände mit atomaren Prädikaten zu benennen (Kripke-Stuktur).
- Andere Operatoren wie $\phi\, R\, \psi$ (release) oder $\phi\, W\, \psi$ (schwaches *until*).

Erfüllung und Modelle für LTL

Die Erfüllbarkeitsrelation für einen Pfad p und eine LTL-Formel ϕ ist induktiv wie folgt definiert:

$$
\begin{array}{lcll}
p & \models & \top & \\
p & \not\models & \bot & \\
p & \models & q & \text{gdw} \quad q(p_1) \\
p & \models & \neg\phi & \text{gdw} \quad p \not\models \phi
\end{array}
\qquad
\begin{array}{lcll}
p & \models & \phi \wedge \psi & \text{gdw} \quad p \models \phi \text{ und } p \models \psi \\
p & \models & \phi \vee \psi & \text{gdw} \quad p \models \phi \text{ oder } p \models \psi \\
p & \models & \phi \longrightarrow \psi & \text{gdw} \quad \text{wenn } p \models \phi \\
 & & & \quad \text{dann } p \models \psi
\end{array}
$$

$$
\begin{array}{lcll}
p & \models & \mathsf{X}\,\phi & \text{gdw} \quad p^2 \models \phi \\
p & \models & \mathsf{G}\,\phi & \text{gdw} \quad \text{für alle } i \text{ gilt } p^i \models \phi \\
p & \models & \mathsf{F}\,\phi & \text{gdw} \quad \text{es gibt } i \text{ mit } p^i \models \phi \\
p & \models & \phi\, U\, \psi & \text{gdw} \quad \text{es gibt } i\ p^i \models \psi \text{ und für } j = 1, \ldots, i-1,\ \ p^j \models \phi
\end{array}
$$

Definition (Modell einer LTL-Formel)

Eine FSM $\mathcal{M}$ erfüllt eine LTL-Formel ϕ, $\mathcal{M} \models \phi$, gdw. jeder Pfad p in $\mathcal{M}$ ϕ erfüllt.

Äquivalenzen

Definition (Äquivalenz)

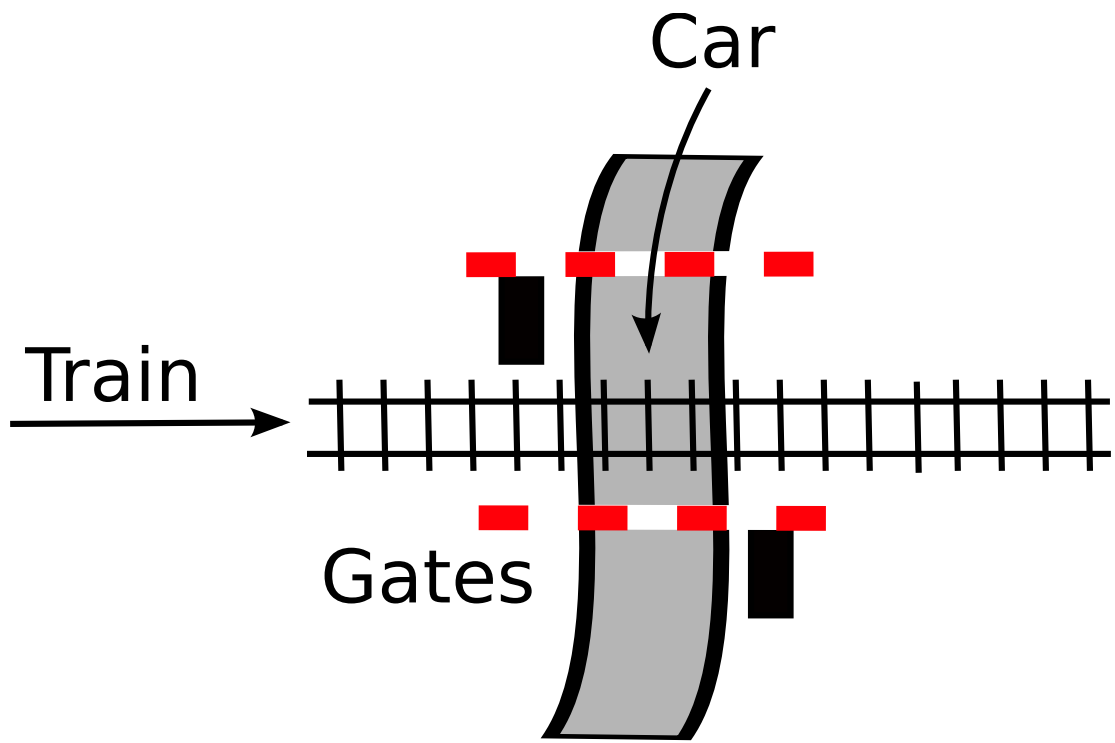
Zwei Formeln sind äquivalent, $\phi \equiv \psi$ gdw. für alle FSM $\mathcal{M}$ und Pfade p in $\mathcal{M}$, $p \models \phi \longleftrightarrow p \models \psi$

- Es gelten aussagenlogische Tautologien z.B. $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$

$$\mathsf{F}(\phi \vee \psi) \equiv \mathsf{F}\,\phi \vee \mathsf{F}\,\psi \qquad \neg\,\mathsf{F}\,\phi \equiv \mathsf{G}(\neg\phi) \qquad \mathit{FGF}\,\phi \equiv \mathit{GF}\,\phi$$
$$\mathsf{G}(\phi \wedge \psi) \equiv \mathsf{G}\,\phi \wedge \mathsf{G}\,\psi \qquad \neg\,\mathsf{G}\,\phi \equiv \mathsf{F}(\neg\phi) \qquad \mathit{GFG}\,\phi \equiv \mathit{FG}\,\phi$$
$$\neg\,\mathsf{X}\,\phi \equiv \mathsf{X}(\neg\phi)$$

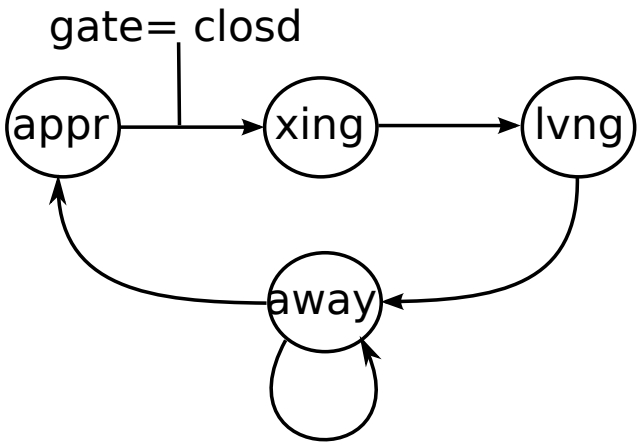
$$\mathsf{X}\,\mathsf{F}\,\phi \equiv \mathsf{F}\,\mathsf{X}\,\phi \qquad \mathsf{F}\,\phi \equiv \phi \vee \mathsf{X}\,\mathsf{F}\,\phi$$
$$\mathsf{X}\,\mathsf{G}\,\phi \equiv \mathsf{G}\,\mathsf{X}\,\phi \qquad \mathsf{G}\,\phi \equiv \phi \wedge \mathsf{X}\,\mathsf{G}\,\phi$$
$$\mathsf{X}(\phi\,U\,\psi) \equiv \mathsf{X}\,\phi\,U\,\mathsf{X}\,\psi \qquad \phi\,U\,\psi \equiv \psi \vee (\phi \wedge \mathsf{X}(\phi\,U\,\psi))$$

Längeres Beispiel: der Bahnübergang

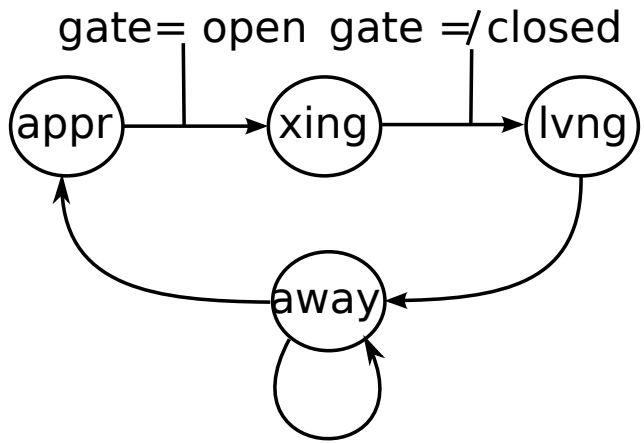


Modellierung des Bahnübergangs

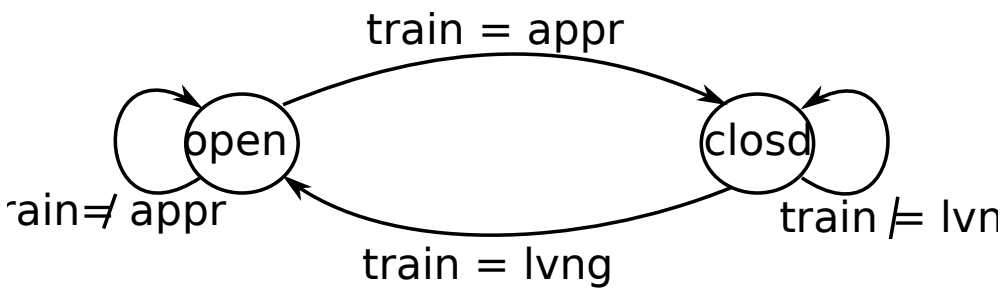
Zustände des Zuges:



Zustände des Autos:



Zustände der Schranke:



Die FSM

- Zustände sind eine endliche Abbildung der Variablen *Car*, *Train*, *Gate* auf Wertebereiche:

$$\begin{aligned} \Sigma_{Car} &= \{appr, xing, lvng, away\} \\ \Sigma_{Train} &= \{appr, xing, lvng, away\} \\ \Sigma_{Gate} &= \{open, clsd\} \end{aligned}$$

  oder ein Tripel $S \in \Sigma = \Sigma_{Car} \times \Sigma_{Train} \times \Sigma_{Gate}$.

- Zustandsübergang komponentenweise, bspw:

$$\begin{aligned} &\langle away, open, away\rangle \rightarrow \langle appr, open, away\rangle \\ &\langle appr, open, away\rangle \rightarrow \langle xing, open, away\rangle \\ &\ldots \end{aligned}$$

Bahnübergang — Formalisierung von Eigenschaften

- Bahn und Auto überqueren den Übergang nie zur selben Zeit:

Bahnübergang — Formalisierung von Eigenschaften

- Bahn und Auto überqueren den Übergang nie zur selben Zeit:

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

Bahnübergang — Formalisierung von Eigenschaften

- Bahn und Auto überqueren den Übergang nie zur selben Zeit:

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

$$\mathsf{G}(\mathit{car} = \mathit{xing} \longrightarrow \mathsf{F}(\mathit{car} = \mathit{lvng}))$$

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

Bahnübergang — Formalisierung von Eigenschaften

- Bahn und Auto überqueren den Übergang nie zur selben Zeit:

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

$$\mathsf{G}(\mathit{car} = \mathit{xing} \longrightarrow \mathsf{F}(\mathit{car} = \mathit{lvng}))$$

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

$$\mathsf{G}(\mathit{train} = \mathit{appr} \longrightarrow \mathsf{F}(\mathit{train} = \mathit{xing}))$$

- Es gibt Autos, die den Bahnübergang passieren:

Bahnübergang — Formalisierung von Eigenschaften

- Bahn und Auto überqueren den Übergang nie zur selben Zeit:

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

$$\mathsf{G}(\mathit{car} = \mathit{xing} \longrightarrow \mathsf{F}(\mathit{car} = \mathit{lvng}))$$

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

$$\mathsf{G}(\mathit{train} = \mathit{appr} \longrightarrow \mathsf{F}(\mathit{train} = \mathit{xing}))$$

- Es gibt Autos, die den Bahnübergang passieren:

$$\mathsf{F}(\mathit{car} = \mathit{xing}) \text{ ist etwas anderes!}$$

  - Nicht in LTL auszudrücken!

Computational Tree Logic (CTL)

- Grenzen der LTL: Quantifikation über Pfaden
  - z.B. Existenz eines Pfades mit einer bestimmten Eigenschaft
- Computational Tree Logic (CTL): Erweiterung der LTL um existentielle/universelle Quantoren über modalen Pfadoperatoren.
  - Modale Operatoren: die Zustandsübergänge betreffend
  - Beispiel: $\mathsf{AF}\, p, \mathsf{EG}\, q, \mathsf{A}[p\, U\, q]$
- Name: Pfade im Berechnungsbaum durch Auffalten der FSM.
  - Beispiel Berechnungsbäume für die Getränkemaschine

LTL und CTL

- CTL ist ausdrucksstärker als LTL, aber das gilt auch anders herum!
  - D.h. es gibt Eigenschaften, die in LTL ausgedrückt werden können, aber nicht in CTL.
- Beispiel: in allen Pfade, in denen p auftritt, tritt auch q auf.
- LTL: $\mathsf{F}\, p \longrightarrow \mathsf{F}\, q$
- CTL: Weder $\mathsf{AF}\, p \longrightarrow \mathsf{AF}\, q$ noch $\mathsf{AG}(p \longrightarrow \mathsf{AF}\, q)$
- Die Logik CTL^* kombiniert die Mächtigkeit von LTL and CTL.

Modellprüfung (Model-Checking)

- Das Model-Checking Problem:

  Gegeben Modell $\mathcal{M}$ und Eigenschaft ϕ, gilt $\mathcal{M} \models \phi$?

- Das Grundproblem beim Model-Checking ist Zustandsexplosion.
  - Eine typische 32-Bit Ganzzahlvariable hat über 4 Mrd. Zustände!
- Die Theorie bietet wenig Anlass zu Hoffnung:

Theorem (Komplexität von Modellprüfung)

(i) *Model-Checking für LTL ohne U ist NP-vollständig.*

(ii) *Model-Checking für LTL ist PSPACE-vollständig.*

(iii) *Model-Checking für CTL ist EXPTIME-vollständig.*

- Gute Nachricht: wenigstens entscheidbar
  - Schlüsseltechnik: Zustandsabstraktion und Zustandskompression

Model-Checking Werkzeuge

- NuSMV2 (Edmund Clarke, Ken McMillan)
  - Web Seite: `http://nusmv.fbk.eu/`
- Spin (Gerard Holzmann)
  - Web Seite: `http://spinroot.com/`
- NuSMV vs. Spin:
  - Spin (Promela) ist näher an einer Programmiersprache
  - NuSMV unterstützt auch CTL

Zusammenfassung

- Temporale Logik: Pfade in Zustandsautomaten
- Aussagenlogik plus modale Operatoren:
  - LTL — linear über einen Pfad: X, G, F, U
  - CTL — verzweigend: AX, EX, AG, EG, AF, EF, A[U], E[U]
  - LTL für Sicherheitseigenschaften, CTL für Verfügbarkeit.
- In der Praxis: LTL/CTL für Modellprüfung (model checking)
  - Modellierung des Systems als FSM $\mathcal{M}$, Eigenschaften als LTL/CTL-Formel ϕ, Überprüfung ob $\mathcal{M} \models \phi$.
  - Entscheidbar, aber mit hoher Komplexität (Zustandsexplosion)
- Model-Checker wie NuSMV entscheiden das Model-Checking-Problem
  - Bei negativer Antwort Gegenbeispiel.
  - Vertrauenswürdigkeit: bei positiver Antwort? Wie gut ist das Modell?