Reaktive Programmierung
Vorlesung 14 vom 12.07.2022
Theorie der Nebenläufigkeit

Christoph Lüth, Martin Ring

Universität Bremen

Sommersemester 2022

Fahrplan

Theorie der Nebenläufigkeit

- Nebenläufige Systeme sind **kompliziert**
  - Nicht-deterministisches Verhalten
  - Neue Fehlerquellen wie **Deadlocks**
  - Schwer zu testen
- Reaktive Programmierung kann diese Fehlerquellen **einhegen**
- **Theoretische Grundlagen** zur Modellierung nebenläufiger Systeme
  - Eigenschaften und ihre Prüfung (**model checking**)

Temporale Logik, Prozessalgebren und Modelchecking

- Temporale (und modale) Logik beschreiben **Systeme** anhand ihrer **Zustandsübergänge**
- Ein System ist dabei im wesentlichen eine **endliche Zustandsmaschine**.

Finite State Machine (FSM)

Eine endliche Zustandsmaschine $\mathcal{M} = \langle S, \Sigma, \rightarrow \rangle$ ist gegeben durch

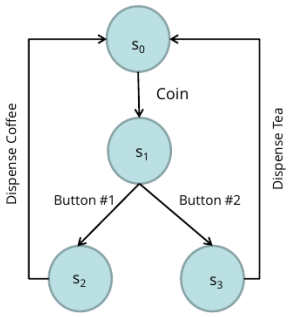
- eine Menge Σ von **Zuständen**,
- eine Menge $I \subseteq \Sigma$ von **initialen** Zuständen, und
- eine **Zustandsübergangsrelation** $\rightarrow \subseteq \Sigma \times \Sigma$

so dass

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

Einfache Beispiele

1. Münze einwerfen
2. Knopf drücken: Tee oder Kaffee
3. Tee oder Kaffee ausschenken
4. Zurück zu (1)



Model Checking

Das Model-Checking Problem

Gegeben ein Model $\mathcal{M}$ und eine Eigenschaft ϕ, gilt

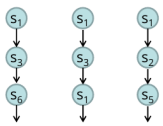
$$\mathcal{M} \models \phi?$$

- System $\mathcal{M}$ wird als FSM modelliert
- Wie beschreiben wir ϕ?
  - Temporale Logik
- Wie beweisen wir das?
  - Indem wir die Zustände aufzählen und das Modell **prüfen**

Temporal-Logiken

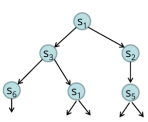
Linear Time

- Jeder Zeitpunkt hat genaue **einen** Nachfolger
- Systemzustand: Menge von Zustandssequenzen



Branching Time

- Jeder Zeitpunkt hat **mehrere** Nachfolger
- Systemzustand: unendlicher **Baum**



Formeln der LTL

$$\phi ::= \underbrace{p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2}_{\text{Aussagenlogik}} \mid \mathsf{X}\phi \mid \Box\phi \mid \Diamond\phi \mid \phi_1 \mathsf{U} \phi_2$$

- $\mathsf{X}\phi$: ϕ gilt im **nächsten** Zustand
- $\Box\phi$: ϕ gilt in **allen** Zuständen
- $\Diamond\phi$: ϕ gilt in **mindestens einem** Zustand
- $\phi \mathsf{U} \psi$: ϕ gilt in allen Zuständen, bis ψ gilt.

Einfache Aussagen

- Irgendwann gibt es Kaffee:

$$\Diamond \mathit{Kaffee}$$

- Wenn ich Knopf-1 drücke, gibt es danach Kaffee:

$$\Box(\mathit{Knopf\text{-}1} \longrightarrow \mathrm{X}\, \mathit{Kaffee})$$

- Wenn ich eine Münze einwerfe, gibt es irgendwann entweder Kaffee oder Tee:

$$\Box(\mathit{Coin} \longrightarrow (\Diamond \mathit{Kaffee} \vee (\Diamond \mathit{Tee})))$$

Entscheidbarkeit und Zustandsexplosion

Entscheidbarkeit

- LTL ohne U ist NP-vollständig;
- LTL ist PSPACE-vollständig;
- CTL ist EXPTIME-vollständig.

- Schlüssel zur **praktischen** Handhabbarkeit: **Zustandsabstraktion**
- Werkzeuge: Spin, nuSMV/nuXMV, UPPAAL, ...
- Erweiterungen: **hybrid state machines** — Zustände sind **kontinuierlich** und werden durch **Differentialgleichungen** beschrieben.