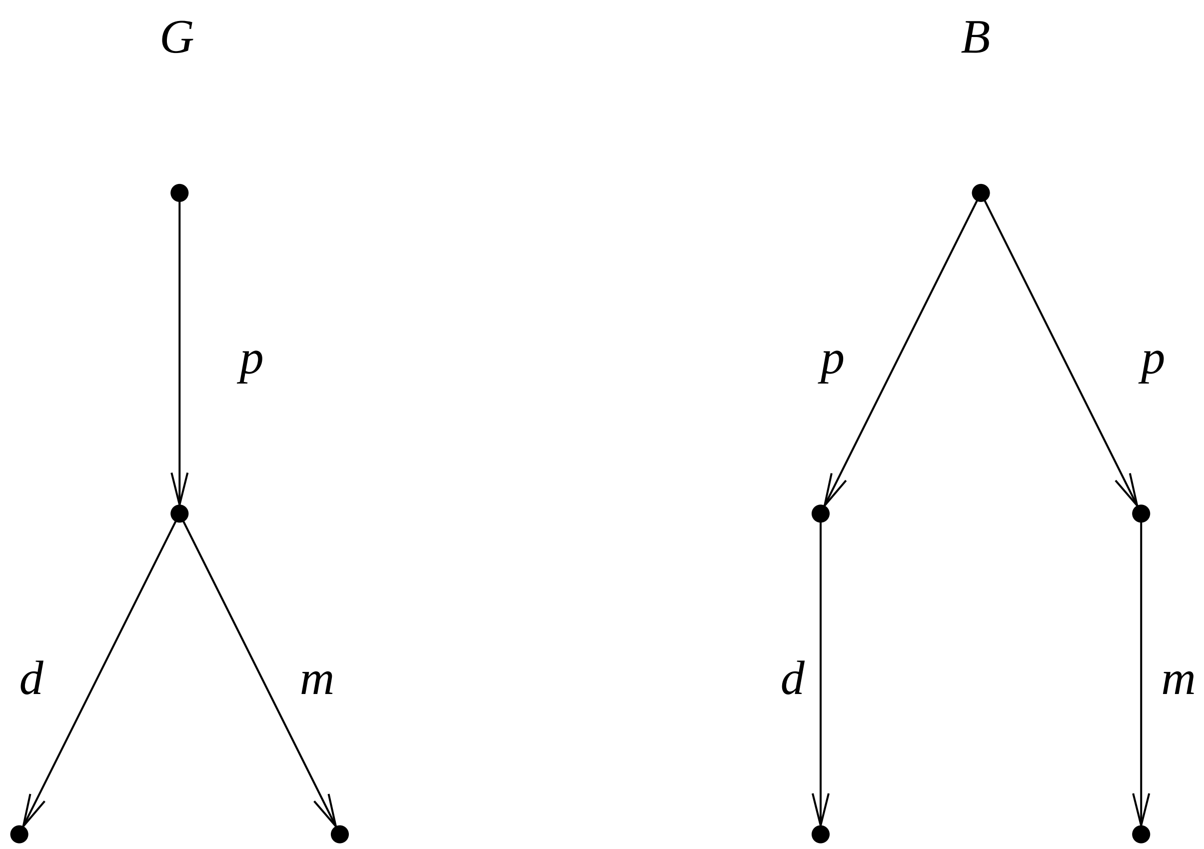
Labelled Transition Systems (LTS)

- semantisches Modell im Kontext von **Process Algebra**
  - Fokus auf **Reaktive Systeme** oder **Offene Systeme**
  - dabei Konzept der **Umgebung** mit **externen** Ereignissen
  - Implementierung (auf einer Abstraktionsebene) bestimmt die **internen** Ereignisse
- ein LTS $A = (S, I, \Sigma, T)$ ist im wesentlichen ein EA:
  - das Verhalten, die Möglichkeit von Übergängen zählt
  - ohne Final-Menge, d.h. auch ohne “explizite” Sprache
  - “implizite” Sprache $L(A)$ durch $F = S$

Schokolade-Automat von Milner



p = Einwurf des Geldbetrages (pay)

d = Auswahl und Ausgabe der dunklen Schokolade

m = Auswahl und Ausgabe der Milch-Schokolade

Sprachäquivalenz Inadequat

- sicherlich sollte die Semantik der beiden LTS “unterschiedlich” sein
  - G erlaubt nach dem Geldeinwurf Wahl der Schokoladenart
  - B setzt nicht-deterministisch beim Geldeinwurf die Schokoladenart fest
- aber B und G sind **sprachäquivalent**:
  - $L(B) \; = \; p \cdot (d \mid m) \; = \; L(G)$
- Problem überträgt sich auf Konformitäts-Test:
  - Sprach-basierter Konformitäts-Test identifiziert B und G
  - Sprach-Konformität ignoriert das “Branching-Verhalten”

Intuition des Simulationsbegriffes

- Verhalten der Implementierung A_1 sollte gültiges Verhalten der Spezifikation A_2 sein
  - jeder **Übergang** in A_1 hat eine Entsprechung in A_2
  - A_2 **simuliert** A_1
  - A_2 kann möglicherweise mehr

- Vereinfachung der formalen Notation durch Vereinigung zu einem LTS A
  - gemeinsames Alphabet Σ
  - (disjunkte) Vereinigung der anderen Komponenten:

    $S = S_1 \dot\cup S_2$, $I = I_1 \dot\cup I_2$, $T = T_1 \dot\cup T_2$
  - Schreibweise: $\quad A = A_1 \dot\cup A_2$

Definition Simulations-Relation

Definition eine Relation $\lesssim \; \subseteq \; S \times S$ über LTS A ist eine **Simulation** gdw.

(man ließt $s \lesssim t$ als t simuliert s)

$$s \lesssim t \quad \text{dann} \quad \forall a \in \Sigma,\, s' \in S\, [s \xrightarrow{a} s' \;\Rightarrow\; \exists\, t' \in S\, [\, t \xrightarrow{a} t' \,\wedge\, s' \lesssim t']]$$

Fakt es gibt genau eine maximale Simulation über jedem LTS A

Beweisskizze (S endlich)

- die Vereinigung zweier Simulationen ist wiederum eine Simulation

- die Menge der Simulationen über A ist nicht leer (enthält die Identität)

Konstruktive Bestimmung der Maximalen Simulation

- Ausgangspunkt: $\lesssim_0 = S \times S$ (normalerweise keine Simulation)

- verfeinere $\lesssim_i$ zu $\lesssim_{i+1}$ wie folgt

  $s \lesssim_{i+1} t$ gdw. $s \lesssim_i t$ und $\forall a \in \Sigma,\ s' \in S\ [s \xrightarrow{a} s' \Rightarrow \exists\, t' \in S\ [\, t \xrightarrow{a} t' \wedge s' \lesssim_i t']]$

- bei endlichem S gibt es ein n mit $\lesssim_n = \lesssim_{n+1}$

  - $\lesssim_n$ ist offensichtlich eine Simulation

  - Maximalität schwerer einzusehen

- kann als Fixpunkt-Prozess reformuliert werden

Sei $\lesssim$ eine Simulation. Zeige $\lesssim \subseteq \lesssim_i$ durch Induktion über i.

Induktionsanfang ist trivial, Induktionsschritt folgt.

Annahme (indirekter Beweis): $\quad \lesssim \not\subseteq \lesssim_{i+1}$.

Dann gibt es s und t mit $s \lesssim t$ aber $s \not\lesssim_{i+1} t$.

Somit muss es s' und a geben mit $s \xrightarrow{a} s'$, aber $t \stackrel{a}{\not\rightarrow} t'$ oder $t \not\lesssim_i t'$ für alle t'.

Mit der Induktionshypothese $\lesssim \subseteq \lesssim_i$ folgt: $\quad t \stackrel{a}{\not\rightarrow} t'$ oder $t \not\lesssim t'$ für alle t'.

Widerspruch zur Voraussetzung $\lesssim$ Simulation.

Weitere Fakten über Simulationen

Fakt maximale Simulation ist eine Halb-Ordnung (insbesondere transitiv)

Beweisskizze

- Reflexivität siehe vorige Beweisskizze

- Transitivität folgt aus unterem Lemma

Lemma transitive Hülle einer Simulation ist wieder eine Simulation

Beweisskizze folgender Operator erhält die Simulationseigenschaft

$$\Psi\colon P(S \times S) \to P(S \times S) \qquad \Psi(\lesssim)(r,t) \quad \text{gdw.} \quad r \lesssim t \quad \text{oder} \quad \exists s[r \lesssim s \wedge s \lesssim t]$$

Simulation von LTS

Definition LTS A_2 simuliert LTS A_1 gdw. es eine Simulation $\lesssim$ über $A_1 \cup A_2$ gibt, so dass für jeden Anfangszustand $s_1 \in S_1$ von A_1, es einen Anfangszustand $s_2 \in S_2$ von A_2 gibt, mit $s_1 \lesssim s_2$. Man schreibt dann auch $A_1 \lesssim A_2$.

Fakt Simulation von LTS ist eine Halb-Ordnung (insbesondere transitiv)

Beweisskizze

- bilde maximale Simulationsrelation über alle drei LTS
- zeige Existenz von simulierenden Anfangszuständen
- Projektion auf äussere LTS liefert gewünschte Simulation

Simulation von Traces

Definition Ein *Trace* eines LTS A ist ein Wort $w = a_1 \cdots a_n \in \Sigma^*$ mit

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} s_{n-1} \xrightarrow{a_n} s_n,$$

wobei $s_0 \in I$ und $n \geq 0$.

Fakt $\quad L(A) = \{w \mid w \text{ Trace von } A\}$

Satz (Simulation ist eine konservative Abstraktion)

LTS A_2 simuliere A_1 via Simulation $\lesssim$ (also $A_1 \lesssim A_2$), dann gilt $L(A_1) \subseteq L(A_2)$.

Anwendung $\quad P \lesssim A \leq S \quad \Rightarrow \quad L(P) \subseteq L(S)$

(P = Programm, A Abstraktion, S Spezifikation)

Wenn man nur an der Sprache bzw. den Traces interessiert ist, kann man dennoch die Abstraktion immer so konstruieren, dass das Programm simuliert wird.

Schwache Simulation

- $\tau \in \Sigma$ bezeichnet ein nicht beobachtbares *internes Ereignis*

- vorherige Definition der Simulation ist dann eine **starke Simulation**

$$s \lesssim t \quad \text{dann} \quad \forall a \in \Sigma,\ s' \in S\,[s \xrightarrow{a} s' \;\Rightarrow\; \exists\, t' \in S\,[\,t \xrightarrow{a} t' \,\wedge\, s' \lesssim t']]$$

- man schreibe $s \xrightarrow{\tau^* a} t$ falls es $s_0, \cdots, s_n$ gibt mit

$$s = s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_{n-1} \xrightarrow{a} s_n = t$$

- eine Relation $\lesssim$ ist eine **schwache Simulation** gdw.

$$s \lesssim t \quad \text{dann} \quad \forall a \in \Sigma\backslash\{\tau\},\ s' \in S\,[s \xrightarrow{\tau^* a} s' \;\Rightarrow\; \exists\, t' \in S\,[\,t \xrightarrow{\tau^* a} t' \,\wedge\, s' \lesssim t']]$$

Zur Schwachen Simulation

- Man verwende τ für *abstrahierte* Ereignisse
  - z.B. nebensächliche Berechnungen/Datenfluss im Programm

- τ-bereinigtes LTS A für ein LTS A_1 mit τ: $\Sigma = \Sigma_1 \backslash \{\tau\}$, $T(s,t)$ gdw. $s \stackrel{\tau^* a}{\rightarrow} t$ in A_1.
  - τ-Bereinigung macht aus schwacher Simulation eine starke (und umgekehrt)
  - damit lassen sich die vorherigen Algorithmen auch hier anwenden

- Transitivität und Anwendungen wie im starken Fall

- **Divergenz** $s \stackrel{\tau^+}{\rightarrow} s$ wird noch unzulänglich behandelt
  - $A_1 \lesssim A_2$ erlaubt A_1 divergent und A_2 nicht

Idee: Implementierung des spezifizierten Verhaltens und nicht mehr!

Definition eine Relation $\approx$ ist eine **starke Bisimulation** gdw.

$$s \approx t \quad \text{dann} \quad \forall a \in \Sigma,\, s' \in S\, [s \xrightarrow{a} s' \Rightarrow\, \exists\, t' \in S\, [\, t \xrightarrow{a} t' \,\wedge\, s' \approx t']] \text{ und}$$

$$\forall a \in \Sigma,\, t' \in S\, [t \xrightarrow{a} t' \Rightarrow\, \exists\, s' \in S\, [\, s \xrightarrow{a} s' \,\wedge\, s' \approx t']]$$

Definition eine Relation $\approx$ ist eine **schwache Bisimulation** gdw.

$$s \approx t \quad \text{dann} \quad \forall a \in \Sigma \backslash \{\tau\},\, s' \in S\, [s \xrightarrow{\tau^* a} s' \Rightarrow\, \exists\, t' \in S\, [\, t \xrightarrow{\tau^* a} t' \,\wedge\, s' \approx t']] \text{ und}$$

$$\forall a \in \Sigma \backslash \{\tau\},\, t' \in S\, [t \xrightarrow{\tau^* a} t' \Rightarrow\, \exists\, s' \in S\, [\, s \xrightarrow{\tau^* a} s' \,\wedge\, s' \approx t']]$$

Insbesondere die schwache Bisimulation bei Abstraktion von internen Ereignissen der Implementierung durch τ ist in der Praxis sehr nützlich!

Theorie-Anwendung: bisimulations-äquivalente LTS haben die gleiche Eigenschaften

Exkurs über Minimierung von EA

Geg. deterministischer und vollständiger EA $A = (S, I, \Sigma, T, F)$

- Ausgangspunkt: $\sim_0 = (F \times F) \cup (\overline{F} \times \overline{F})$
  - Partitionierung bezüglich “Endzustands-Flag”
  - Äquivalenzrelation

- verfeinere $\sim_i$ zu $\sim_{i+1}$

  $s \sim_{i+1} t$ gdw. $s \sim_i t$ und

  $$\forall a \in \Sigma,\ s' \in S\ [s \xrightarrow{a} s' \Rightarrow\ \exists\, t' \in S\ [\ t \xrightarrow{a} t'\ \wedge\ s' \sim_i t']] \quad \text{und}$$

  $$\forall a \in \Sigma,\ t' \in S\ [t \xrightarrow{a} t' \Rightarrow\ \exists\, s' \in S\ [\ s \xrightarrow{a} s'\ \wedge\ s' \sim_i t']]$$

- Terminierung $\sim_{n+1} = \sim_n$ spätestens für $n = |S|$

- Äquivalenzrelation $\sim = \sim_n$ erzeugt minimalen Automaten $A/\sim$