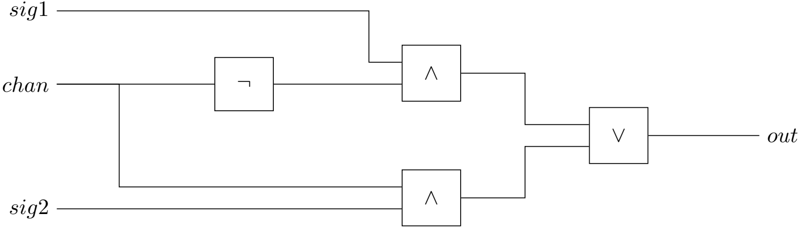
Teil 3.1: Modellierung von Verhalten

Falk Howar

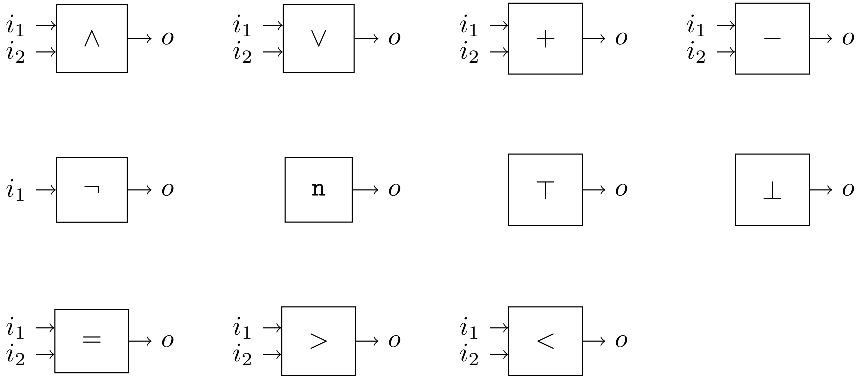
Softwarekonstruktion WS 2018/19

LS14

Multiplexer Schaltung



Bausteine



Formale Semantik?

Variablen / Typen

Typen:

- `Nat`: Ganze Zahlen
- `Boolean`: Boolesche Werte $\{0, 1\}$
- Aufzählungen. Z.B. Typ `State` mit Werten $\{\texttt{on}, \texttt{off}\}$

Variablen:

- Menge X von getypten Variablen

Belegungen

Belegung

Eine Belegung v von X ist eine Typ-konsistente Zuweisung von Werten zu Variablen in X. v ist eine Funktion und für $x \in X$ ist $v(x)$ der Wert mit dem x in v belegt ist.

- Typ-konsistent: $v(x)$ gehört zu Typ von x.
- Z.B., $v(x) \in \mathbb{Z}$ wenn x von Typ `Nat`.

Sei V_X die Menge aller möglichen typ-konsistenten Belegungen von X.

Beispiel

Variablen:

- $X = \{x_1, x_2\}$

Typen:

- x_1 von Typ `Nat`
- x_2 von Typ `State` mit Werten $\{\texttt{on}, \texttt{off}\}$

Belegung:

- v_1 mit $v_1(x_1) = 101$ und $v_1(x_2) = \texttt{on}$

Getypte Ausdrücke

Für einen fixen Aufzählungstyp, z.B. `State`:

$$e \in \mathbf{EExp} \text{ (Aufzälungs-Ausdrücke)}$$

$$e ::= \; z \mid c$$

$z \in \mathbf{Var}$ Variable vom Aufzähzlungstyp `State`

$c \in$ `State` konkreter Wert vom passenden Typ

Getypte Ausdrücke

$a \in \mathbf{AExp}$ (Arithmetische Ausdrücke)

$a ::= \; x \mid \mathtt{n} \mid (a_1 \; op_a \; a_2)$

$b \in \mathbf{BExp}$ (Boolesche Ausdrücke)

$b ::= \; y \mid \mathtt{true} \mid \mathtt{false} \mid \neg \, b \mid (b_1 \; op_b \; b_2) \mid (a_1 \; op_r \; a_2) \mid (z \; = \; c)$

$x \in X$ Variable Typ `Nat`

$y \in X$ Variable Typ `Boolean`

$z \in X$ Variable Typ `State`

$\mathtt{n}$ Literal für $n \in \mathbb{Z}$

$op_a \in \mathbf{Op_a}$ arithmetische Op.: $+, \; -$

$op_b \in \mathbf{Op_b}$ Boolesche Op.: $\wedge, \; \vee$

$op_r \in \mathbf{Op_r}$ Relationale Op.: $>, <, =$

Beispiel

Variablen:

- $X = \{x_1, x_2\}$

Typen:

- x_1 von Typ `Nat`
- x_2 von Typ `State` mit Werten $\{\texttt{on}, \texttt{off}\}$

Ausdruck:

$$((x_2 = \texttt{off}) \ \wedge \ (x_1 > \texttt{50}))$$

Semantik

Verbindung zu Metamodellierung:

- Definition bis jetzt: konkrete Syntax (Sprache)
- Jetzt: Semantische Abbildung

- Aufgabe: Auswertung / Werte von Ausdrücken.
- Menge der Ausdrücke $\mathbf{Exp} = (\mathbf{BExp} \cup \mathbf{AExp} \cup \mathbf{EExp})$

Auswertung von Ausdrücken

- Set von Variablen X
- Ausdrücke aus $\mathbf{Exp}$ über Variablen aus X
- Fixer Aufzählungstyp z.B., $\mathtt{State}$ mit Werten $\{\mathtt{on}, \mathtt{off}\}$

Auswertung:

- Funktion $\alpha_X : \mathbf{AExp} \times V_X \mapsto \mathbb{Z}$
- Funktion $\beta_X : \mathbf{BExp} \times V_X \mapsto \{0, 1\}$
- Funktion $\gamma_X : \mathbf{EExp} \times V_X \mapsto \mathtt{State}$

- Belegung v von X

Auswertung von Ausfzählungsausdrücken

Atome:

- $\gamma_X[\mathtt{on}, v] = \mathtt{on}$
- $\gamma_X[\mathtt{off}, v] = \mathtt{off}$
- $\gamma_X[z, v] = v(z)$

Auswertung von Booleschen Ausdrücken

Atome:

- $\beta_X[\mathtt{true}, v] = 1$
- $\beta_X[\mathtt{false}, v] = 0$
- $\beta_X[y, v] = v(y)$

Negation:

- $\beta_X[(\neg b), v] = 1$ gdw. $\beta_X[b, v] = 0$ (sonst 0)

Komposition:

- $\beta_X[(b_1 \wedge b_2), v] = 1$ gdw. $\beta_X[b_1, v] = 1$ und $\beta_X[b_2, v] = 1$ (sonst 0)
- $\beta_X[(b_1 \vee b_2), v] = 0$ gdw. $\beta_X[b_1, v] = 0$ und $\beta_X[b_2, v] = 0$ (sonst 1)

Auswertung von arithmetischen Ausdrücken

Atome:

- $\alpha_X[\mathtt{n}, v] = n$
- $\alpha_X[x, v] = v(x)$

Komposition:

- $\alpha_X[(a_1 + a_2), v] \;=\; \alpha_X[a_1, v] \;+\; \alpha_X[a_2, v]$
- $\alpha_X[(a_1 - a_2), v] \;=\; \alpha_X[a_1, v] \;-\; \alpha_X[a_2, v]$

Wir trennen hier nicht formal sauber zwischen Syntax und Semantik von +,- !

Boolesche Kombination:

Aufzählung:

- $\beta_X[(z = c), v] = 1$ gdw. $\gamma_X[z, v] \; = \; \gamma_X[c, v]$ (sonst 0)

Arithmetisch:

- $\beta_X[(a_1 \circ a_2), v] = 1$ gdw. $\alpha_X[a_1, v] \; \circ \; \alpha_X[a_2, v]$

Für $\circ = \{<, =, >\}$

Beispiel

- Ausdruck: $((x_2 = \texttt{on}) \ \wedge \ (x_1 > \texttt{50}))$
- Belegung: v mit $v(x_1) = 101$ und $v(x_2) = \texttt{off}$

$$\beta_X[((x_2 = \texttt{on}) \ \wedge \ (x_1 > \texttt{50})),\ v] = 0$$

$$\beta_X[(x_2 = \texttt{on}), v] = 0 \qquad \beta_X[(x_1 > \texttt{50}), v] = 1$$

$$\gamma_X[x_2, v] \neq \gamma_X[\texttt{on}, v] \qquad \alpha_X[x_1, v] > \alpha_X[\texttt{50}, v]$$

$$\gamma_X[x_2, v] = \texttt{off} \qquad \gamma_X[\texttt{on}, v] = \texttt{on} \qquad \alpha_X[x_1, v] = v(x_1) = 101 \quad \alpha_X[\texttt{50}, v] = v(x_1) = 50$$

Umgang mit Variablen

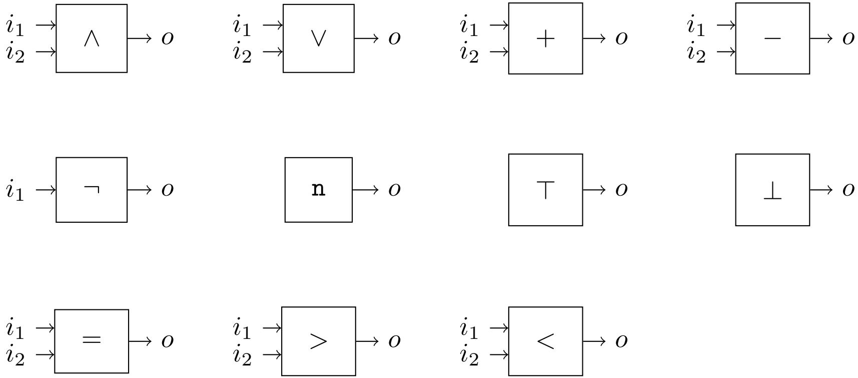
Ausdrücke mit Variablen:

- Wir schreiben $\varphi[\mathbf{x}]$ um anzuzeigen, dass der Ausdruck φ über einer Menge $\mathbf{x}$ von Variablen formuliert ist

Beispiel:

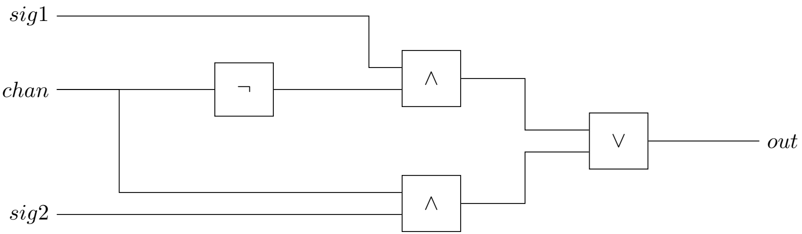
- Für $\varphi = ((x_2 = \texttt{on}) \ \wedge \ (x_1 > 50))$
- $\varphi[\mathbf{x}]$ mit $\mathbf{x} = \{x_1, x_2\}$

Blockschaltdiagramme



Semantik

- Durch Abbildung auf Ausdrücke:



$$(sig1 \;\wedge\; (\neg chan)) \;\vee\; (chan \;\wedge\; sig2)$$

Anmerkungen

- In dieser Semantik: Kein Speicher!
- Keine "Unterprogramme"
- Keine Zwischenergebnisse
- Mehrfache Verwendung eines Signals: Kopieren des Diagrammteils vor dem Signal

$\Rightarrow$ Mehr Features, kompliziertere Semantik ...

Von Blockschaltdiagrammen zu Ausdrücken

Zusammenfassung

Wir haben bis jetzt:

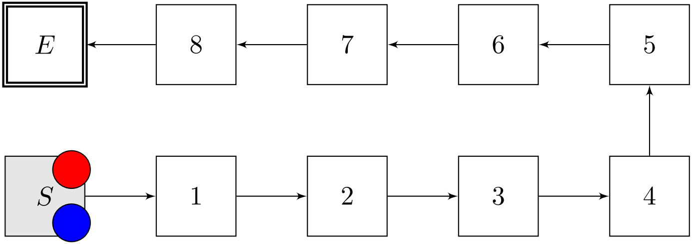
(1) Getypte Variablen
(2) Belegungen von Variablen
(3) Ausdrücke über Variablen

- Syntax: Grammatik
- Sematik: Auswertung von Ausdrücken unter Belegungen

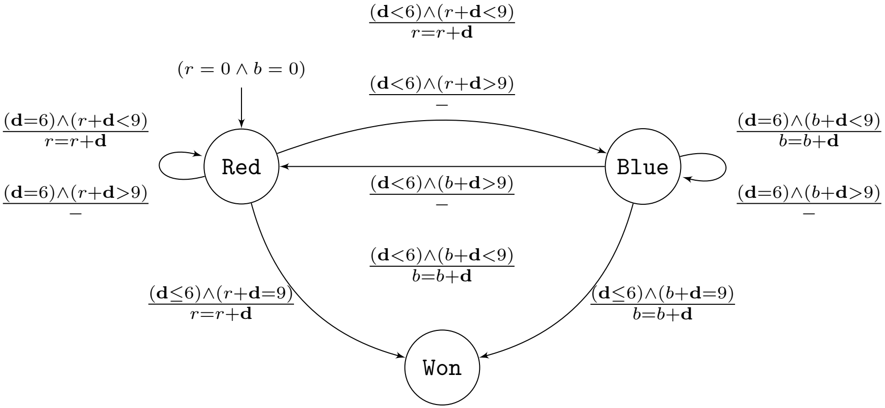
(4) Weitere konkrete Syntax: kombinatorische Komponenten

Bei Typen waren wir nicht ganz formal ...

Verhalten des Leiterspiels



Erweiterte endliche Automaten



Was wir noch brauchen

(1) Zusätzliche Notation für Komposition von Belegungen
(2) Modelle
(3) Zuweisungen
(4) Erweiterte Endliche Automaten

- Syntax
- Semantik (Transitionssysteme)

Komposition von Belegungen

Komposition:

Für Belegungen v_1 und v_2 von disjunkten Mengen von Variablen X_1 und X_2 definieren wir die Komposition v_1, v_2:

$$v_1, v_2(x) = \begin{cases} v_1(x) & \text{für } x \in X_1, \\ v_2(x) & \text{für } x \in X_2. \end{cases}$$

Beispiel:

- $X_1 = \{x_1\}$ und $v_1 = \{x_1 \mapsto 2\}$
- $X_2 = \{x_2\}$ und $v_2 = \{x_2 \mapsto 5\}$

- $v_1, v_2 = \{x_1 \mapsto 2,\ x_2 \mapsto 5\}$

Modelle für Boolesche Ausdrücke

Modell

Eine Belegung v von X ist ein Modell für $\varphi[\mathbf{x}]$ wenn (typkonsistent) $\mathbf{x} \subseteq X$ und $\beta_X[\varphi, v] = 1$. Wir schreiben dann

$$v \models \varphi.$$

Beispiel:

- Ausdruck: $\varphi = ((x_2 = \texttt{on}) \ \wedge \ (x_1 > \texttt{50}))$
- Belegung: v mit $v(x_1) = 101$ und $v(x_2) = \texttt{on}$ und $v(x_3) = 0$

- In diesem Fall: $v \models \varphi$

Zustände / Updates

Um erweiterte Automaten definieren zu können, müssen wir Updates von Variablen ausdrücken können.

Zuweisung (Syntax):

- Für eine Variable x_1 und einen Ausdruck e vom gleichen Typen schreiben wir eine Zuweisung $x_1 := e$
- Mehrere parallele Zuweisungen: $x_1 := e_1; x_2 = e_2; \ldots$ bilden ein Update u

- Wir nutzen Mengenschreibweise: $(x_1 := e_1) \in u$

Semantik von Zuweisungen

- Menge von Variablen X, Belegung v und Update u

- Die Belegung v' resultiert aus Anwendung von u auf v
- Wir definieren $v' = u(v)$ für $x \in X$ als:

$$v'(x) = \begin{cases} \alpha_X[e, v] & \text{für } (x := e) \in u \text{ und } x, e \text{ Typ } \texttt{Nat}, \\ \beta_X[e, v] & \text{für } (x := e) \in u \text{ und } x, e \text{ Typ } \texttt{Boolean}, \\ \gamma_X[e, v] & \text{für } (x := e) \in u \text{ und } x, e \text{ Typ } \texttt{State}, \\ v(x) & \text{sonst.} \end{cases}$$

Beispiel

- Belegung: v mit $v(x_1) = 101$ und $v(x_2) = \texttt{on}$ und $v(x_3) = 0$
- Update $u \ = \ x_1 := 50; x_3 := x_1 + x_3$

- Für $v' = u(v)$:

- $v'(x_1) = 50$
- $v'(x_2) = \texttt{on}$
- $v'(x_3) = 101$

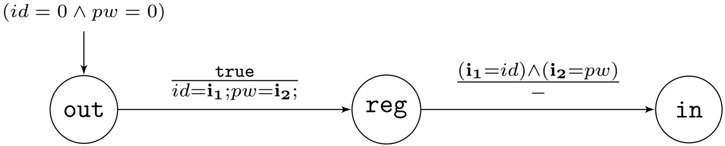
Erweiterte Endliche Automaten

Erweiterter Endlicher Automat

Ein erweiterter endlicher Automat A ist ein Tupel $\langle I, Q, X, (q_0, v_0), T, \texttt{State}, L\rangle$ mit

- einer Menge I von Eingabevariablen,
- einer Menge Q von Zuständen,
- einer Menge X von Zustandsvariablen von Typ `Nat` und `Boolean`,
- einer initialen Konfiguration $(q_0, v_0) \in Q \times V_X$,
- einer Menge T von Transitionen der Form $\langle q, g, u, q'\rangle$ mit
  - Quelle q und Ziel q' aus Q,
  - Bedingung (Boolescher Ausdruck) $g[\mathbf{x}]$ mit $\mathbf{x} \subseteq I \cup X$
  - Update u für eine Teilmenge der Variablen aus X durch Ausdrücke über $I \cup X$,
- einem Aufzählungstypen `State`,
- einer bijektiven Abbildung $L : Q \mapsto \texttt{State}$.

Beispiel



$(id = 0 \wedge pw = 0)$
out
$\frac{\texttt{true}}{id{=}\mathbf{i_1};pw{=}\mathbf{i_2};}$
reg
$\frac{(\mathbf{i_1}{=}id)\wedge(\mathbf{i_2}{=}pw)}{-}$
in

Konfigurationen / Zustandsübergänge

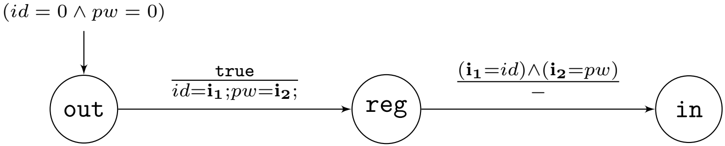
- Eine Konfiguration ist ein Paar $(q, v) \in Q \times V_X$
- Der Automat $A = \langle I, Q, X, (q_0, v_0), T, \texttt{State}, L\rangle$ erlaubt den Übergang von (q, v) nach (q', v') gdw. es eine Belegung $i \in V_I$ der Eingabevariablen gibt und eine Transition $t \in T$ gibt mit $t = \langle q, g, u, q'\rangle$ für die

$$v, i \models g \qquad \wedge \qquad v' = u(v, i)$$

Wir schreiben

$$(q, v) \xrightarrow{i} (q', v')$$

Beispiel



$$(\texttt{out}, \{id \mapsto 0, pw \mapsto 0\}) \xrightarrow{\{\mathbf{i_1}\mapsto 1, \mathbf{i_2}\mapsto 2\}} (\texttt{reg}, \{id \mapsto 1, pw \mapsto 2\})$$

$$(\texttt{reg}, \{id \mapsto 1, pw \mapsto 2\}) \xrightarrow{\{\mathbf{i_1}\mapsto 1, \mathbf{i_2}\mapsto 2\}} (\texttt{in}, \{id \mapsto 1, pw \mapsto 2\})$$

(Wir nutzen Labels als Namen von Zuständen)

Semantik erweiterter endlicher Automaten

Eine Folge $i_1,\ i_2,\ \ldots,\ v_n$ von Belegungen der Eingabevariablen von A hat eine Ausführung in $A = \langle I, Q, X, (q_0, v_0), T, \texttt{State}, L\rangle$, wenn es eine Folge von Konfigurationen (q_i, v_i) für $1 \leq i \leq n$ gibt, so dass:

$$(q_0, v_0) \xrightarrow{i_1} (q_1, v_1) \xrightarrow{i_2} (q_2, v_2) \rightarrow\ \ldots\ \rightarrow (q_{n-1}, v_{n-1}) \xrightarrow{i_n} (q_n, v_n)$$

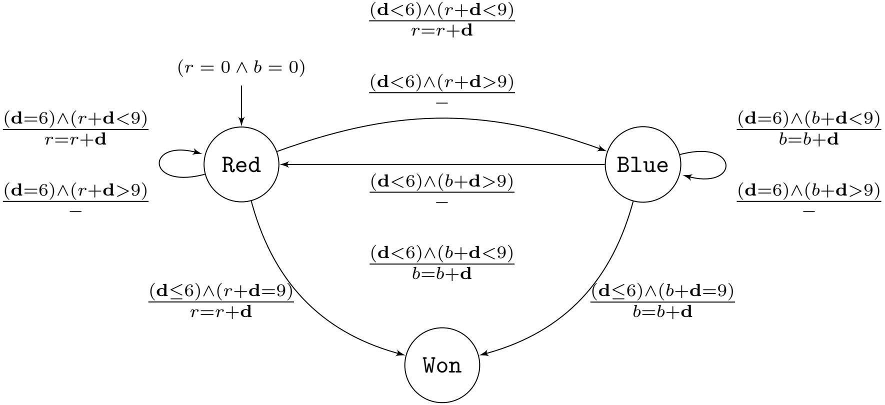
Die Semantik eines erweiterten endlichen Automaten A ist die Menge aller Ausführungen in A.

Für Codegenerierung werden wir später etwas formaler ...

Determinismus

Ein erweiterter endlicher Automat $A = \langle I, Q, X, (q_0, v_0), T, \texttt{State}, L\rangle$ ist deterministisch, wenn jede Folge $i_1,\ i_2,\ \ldots,\ v_n$ von Belegungen der Eingabevariablen in I höchstens einen Lauf in A hat.

Erweiterter Endlicher Automat für Wettrennen



Frage: Ist die Konfiguration $(\texttt{Won}, \{r \mapsto 9, b \mapsto 9\})$ Teil einer Ausführung?

Diskussion

Vereinfachtes Modell:

- Keine Ausgaben
- Fixe Parallelität in Zuweisungen
- Keine Komposition / hierarchische Strukturen
- Nur wenige Datentypen und Operationen

Komplementäre Arten von formalen Modellen:

- Petrinetze (Parallelität von Kontrolle)

Semantisches Funktional

Für erweiterten endlichen Automaten $A = \langle I, Q, X, (q_0, v_0), T, \texttt{State}, L\rangle$ sei

$$[\![A]\!] : V_I^* \mapsto \{0, 1\}$$

mit

$$[\![A]\!](\vec{i}\,) \;=\; \begin{cases} 1 & \text{wenn } \vec{i} \text{ eine Ausführung in } A \text{ hat,} \\ 0 & \text{sonst.} \end{cases}$$

für $\vec{i} \in V_I^*$

Template

Für $L(q_0) = \texttt{lbl}$

```
class Automaon {
  enum  State {...};
  State q=State.lbl;
  <<vars>>
  boolean step(Input i) {
    if (q==null) return false;
    switch (q) {
        <<block>>
    }
    return true;
  }
}
```

Ausfüllen von Templates

Für $x \in X$ von Typ `Nat` und $n = v_0(x)$

```
int x = n;
```

(Boolesche Variablen analog)

Für q mit $L(q) =$ `lbl`

```
case State.lbl:
  <<trans>>
  q = null;
  break;
```

Für $t = (q, g, u, q')$ mit $L(q') =$ `lbl2`

```
if (g.holdsFor(i, this)) {
    eval(u,i); // parallel Update
    q = State.lbl2;
  break;
}
```

Worte

```
public static boolean semofA(Automaton A, Input ... ivec) {
  boolean ret;
  for (Input i : ivec) {
    ret = a.step(i);
  }
  return ret;
}
```