Berechenbarkeit

- Endliche Automaten erkennen nicht alle algorithmisch erkennbaren Sprachen.
- Kontextfreie Grammatiken erzeugen nicht alle algorithmisch erzeugbaren Sprachen.

- Welche Berechnungsmodelle erlauben die Berechnung aller algorithmisch beschreibbaren Sprachen?
- Was können allgemeinere Berechnungsmodelle berechnen?
- Wo sind die Grenzen?
- Was heisst überhaupt Berechenbarkeit?

Berechenbarkeitsmodelle

- Präzisieren den Begriff der berechenbaren Funktionen
- Syntax: Sprache zum Beschreiben von Algorithmen
- Semantik: Kalkül zur Ausführung der Algorithmen

Beispiele für Berechenbarkeitsmodelle

- Spezielle Programmier- oder Spezifikationssprachen
- Turingmaschinen
- Typ-0-Grammatiken
- Registermaschinen, λ-Kalkül, μ-rekursive Funktionen

PASCALchen

- Kleine imperative Programmierspache
- Besteht aus $while$-Programmen
- Einziger Datentyp: natürliche Zahlen
- Keine Rekursion

Syntax

Zeichen in PASCALchen

- Variablen: Xn mit $n \in \mathbb{N}$
- Operatoren: $succ$, $pred$, 0
- Zuweisungssymbol: $:=$
- Schlüsselwörter: $begin$, end, $while$, do
- Hilfssymbole: $($, $)$, ;

- Ein $while$-Programm ist eine Reihung.

$$<prog> ::= <comp>$$

- Eine Reihung hat die Form $begin\ S_1; S_2; \ldots; S_m\ end$, wobei $S_1, \ldots, S_m$ für $m \geq 0$ Anweisungen sind.

$$<comp> ::= begin\ <stmtlist>\ end \mid begin\ end$$
$$<stmtlist> ::= <stmt> \mid <stmt>; <stmtlist>$$

- Eine Anweisung ist eine Reihung, eine Zuweisung oder eine Wiederholung

$$<stmt> ::= <comp> \mid <assign> \mid <while>$$

- Eine Zuweisung hat die Form $Xi := 0$, oder $Xi := succ(Xj)$ oder $Xi := pred(Xj)$, wobei Xi, Xj zwei Variablen sind.

$$<assign> ::= <var> := <expr>$$
$$<expr> ::= 0 \mid \mathrm{succ}(<var>) \mid \mathrm{pred}(<var>)$$

- Eine Wiederholung hat die Form $while\ Xi \neq Xj\ do\ S$, wobei S eine Anweisung ist und Xi, Xj zwei Variablen.

$$<while> ::= while\ <var> \neq <var>\ do\ <stmt>$$

- Variablen haben die Form Xn mit $n \in \mathbb{N}$.

$$<var> ::= X<nat>$$
$$<nat> ::= 0 \mid <one-nine><cipherseq>$$
$$<cipherseq> ::= \lambda \mid <cipher><cipherseq>$$
$$<cipher> ::= 0 \mid <one-nine>$$
$$<one-nine> ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Beschreibung der PASCALchen-Syntax mit kontextfreien Regeln

$$
\begin{aligned}
&<prog> ::= <comp> \\
&<comp> ::= \text{begin } <stmtlist> \text{ end} \mid \text{begin end} \\
&<stmtlist> ::= <stmt> \mid <stmt>; <stmtlist> \\
&<stmt> ::= <comp> \mid <assign> \mid <while> \\
&<assign> ::= <var> := <expr> \\
&<while> ::= \text{while } <var> \neq <var> \text{ do } <stmt> \\
&<expr> ::= 0 \mid \text{succ}(<var>) \mid \text{pred}(<var>)
\end{aligned}
$$

$$
\begin{aligned}
&<var> ::= X <nat> \\
&<nat> ::= \mid <one-nine><cipherseq> \\
&<cipherseq> ::= \lambda \mid <cipher><cipherseq> \\
&<cipher> ::= 0 \mid <one-nine> \\
&<one-nine> ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Beispiel

$$
\begin{aligned}
&begin \\
&\quad X1 := succ(X2); \\
&\quad X1 := pred(X1) \\
&end
\end{aligned}
$$

Weist der Variable $X1$ den Wert von $X2$ zu.

Makros

while-Programme sind Anweisungen, die in anderen *while*-Programmen verwendet werden können.

$$\left.\begin{array}{l} begin \\ \quad X1 := succ(X2); \\ \quad X1 := pred(X1) \\ end \end{array}\right\} X1 := X2$$

$$
\left.
\begin{array}{l}
\mathit{begin} \\
\quad X4 := 0;\ X1 := X2; \\
\quad \mathit{while}\ X4 \neq X3\ \mathit{do}\ \mathit{begin} \\
\qquad X4 := \mathit{succ}(X4); \\
\qquad X1 := \mathit{succ}(X1) \\
\quad \mathit{end} \\
\mathit{end}
\end{array}
\right\} X1 := X2 + X3
$$

$$\left.\begin{array}{l} begin \\ \quad Z := 0; V := 0; \\ \quad while\ V \neq Y\ do\ begin \\ \quad\quad Z := Z + X; \\ \quad\quad V := succ(V) \\ \quad end \\ end \end{array}\right\} Z := X * Y$$

Beachte: Wir benennen Variablen durch beliebige Großbuchstaben, wenn die spezielle Form Xn nicht gebraucht wird.

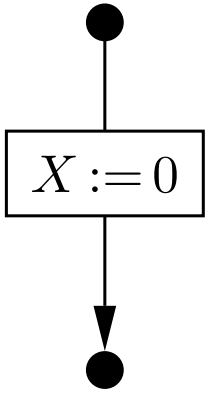
Weitere mögliche Makros

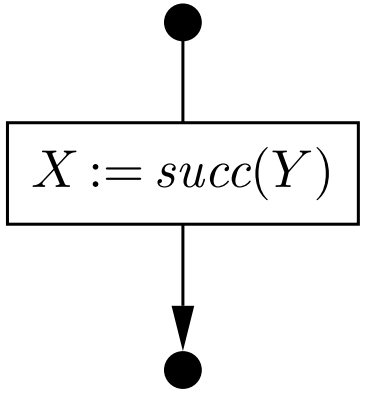
- $X1 := n$,
- $X1 := X2 \dot{-} X3$,
- $X1 := 2^{X2}$,
- $\mathit{while}\ B\ \mathit{do}\ S$,
- $\mathit{if}\ B\ \mathit{then}\ S_1\ \mathit{else}\ S_2$,
- $\mathit{repeat}\ S\ \mathit{until}\ B$ . . .

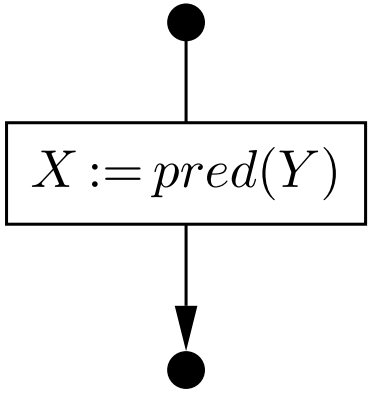
mit B: Boolescher Ausdruck, der aus Variablen, Konstanten, $=$, $\neq$, $<$, and, or, not aufgebaut ist.

Flussdiagramme

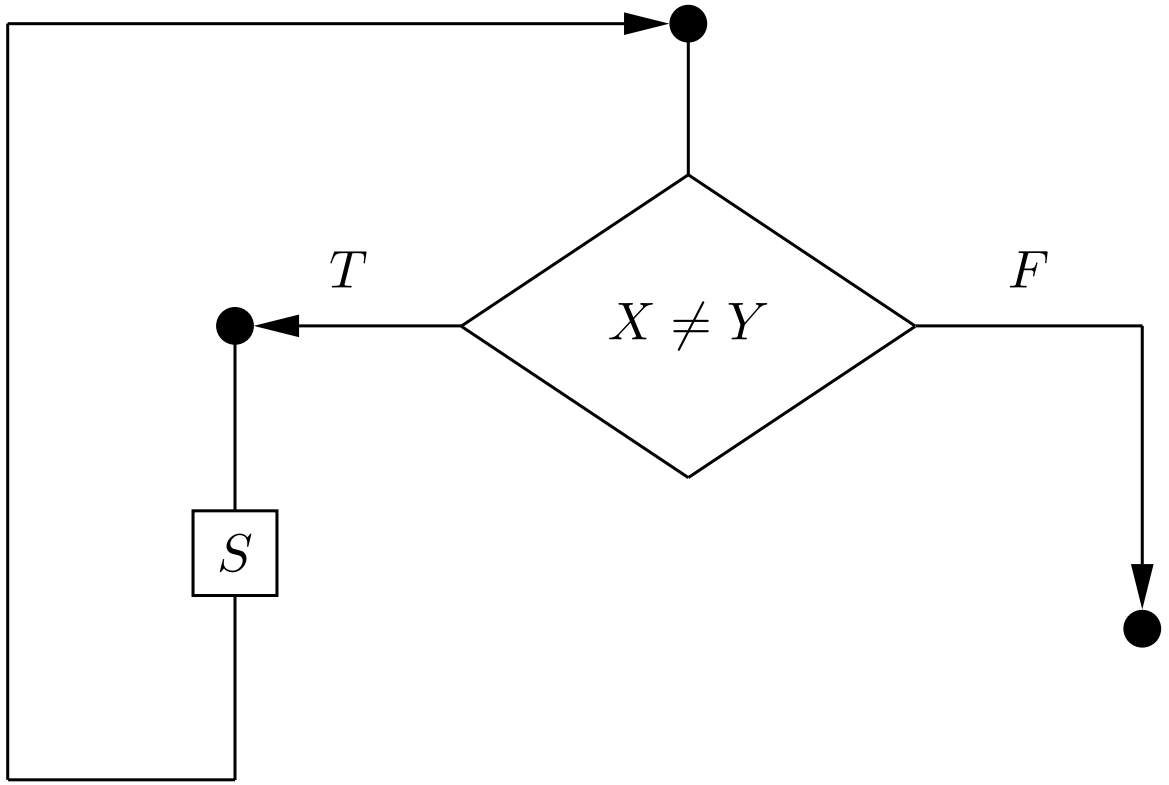
- Zuweisungen

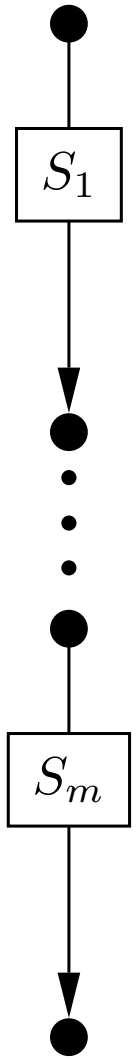






- Wiederholung und Reihung





Semantik (operationell)

Ein *while*-Programm wird k-variabel genannt, wenn höchstens die Variablen $X1, \ldots, Xk$ darin vorkommen.

Ein Berechnungszustand eines k-variablen *while*-Programms ist ein k-dimensionaler Vektor über den natürlichen Zahlen

$$a = (x_1, \ldots, x_k) \in \mathbb{N}^k,$$

der auch Zustand oder Zustandsvektor genannt wird.

Berechnung

$$a_0 A_1 a_1 A_2 a_2 \cdots a_{i-1} A_i a_i A_{i+1} a_{i+1} \cdots$$

- a_i: Berechnungszustände
- a_0 wird Eingabe genannt
- A_i: Instruktionen (d.h. Tests oder Zuweisungen)
- die im Folgenden genannten Bedingungen müssen erfüllt sein.

Bedingungen für $a_0 A_1 a_1 A_2 a_2 \cdots$ (1)

- Es gibt einen Weg durch das zugehörige Flussdiagramm, der beim Eingang beginnt und genau die Instruktionen $A_1, A_2, \ldots, A_i, \ldots$ in dieser Reihenfolge durchläuft.
- a_0 ist frei wählbar.
- $A_i = Xu \neq Xv \Longrightarrow a_i = a_{i-1}$.

Bedingungen für $a_0 A_1 a_1 A_2 a_2 \cdots$ (2)

$$A_i = \left\{ \begin{array}{l} Xu := 0 \\ Xu := succ(Xv) \\ Xu := pred(Xv) \end{array} \right\} \text{ und } a_{i-1} = (x_1, \ldots, x_k)$$

$$\Longrightarrow$$

$$a_i = (x_1, \ldots, x_{u-1}, \left\{ \begin{array}{c} 0 \\ succ(Xv) \\ pred(Xv) \end{array} \right\}, x_{u+1}, \ldots, x_k).$$

Bedingungen für $a_0A_1a_1A_2a_2\cdots$ (3)

- A_i: letzte Instruktion der Berechnung $\Longrightarrow$
  hinter A_i ist der Ausgang.
- $A_i = Xu \neq Xv$ und $a_{i-1} = (x_1, \ldots, x_k)$ $\Longrightarrow$
  - Falls $x_u \neq x_v$, ist A_{i+1} die erste Instruktion nach der herausführenden T-Kante.
  - Falls $x_u = x_v$, ist A_i die letzte Instruktion der Berechnung, oder A_{i+1} ist die erste Instruktion nach der herausführenden F-Kante.

Endliche Berechnung

$$a_0 A_1 a_1 \cdots a_{n-1} A_n a_n \qquad (n \geq 0)$$

- n: Länge der Berechnung
- a_n: Ausgabe
- $n = 0 \Longrightarrow$ Flussdiagramm enthält keine Instruktion

Beobachtung

Zu jedem k-variablen $while$-Programm und jedem Berechnungszustand a_0 gibt es genau eine Berechnung mit a_0 als Anfang.