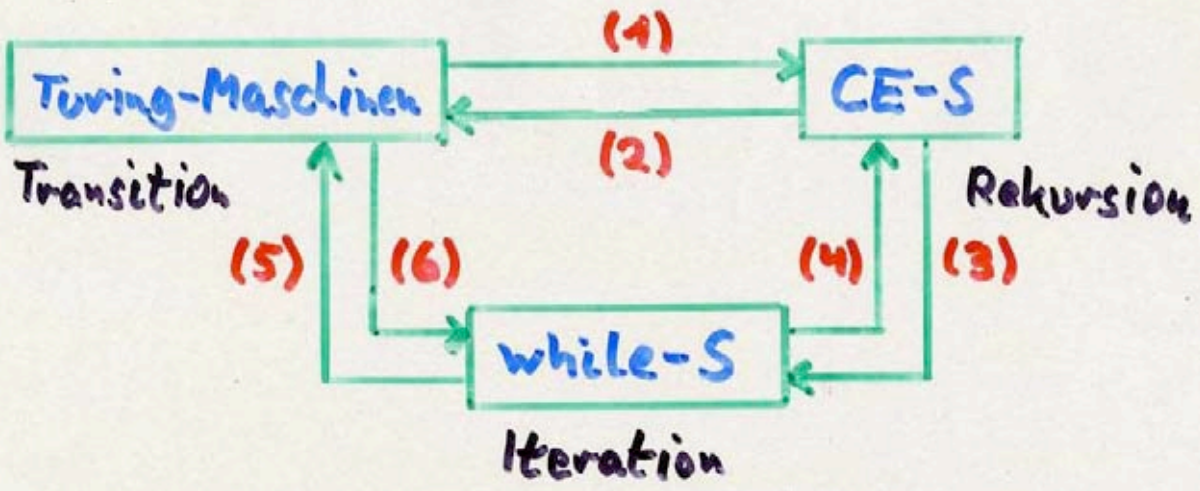
Vergleich von Berechenbarkeitsmodellen



(1) Zustände als rekursive Funktionen auf dem Band
(2) CE-S-Vorwärtsinterpreter als Turing-Maschine
(3) CE-S-Vorwärtsinterpreter iteriert (z.B. mit Stack)
(4) rekursive Form der Berechnung
(5) Berechnung auf Band einer Turing-Maschine
(6) while-Schleife, die die Schritte einer TM vollzieht
(2) & (5) nach Turingscher These (Church'sche-These)
(i) Komposition von (j) & (k) (geeignet gewählt)

Syntax von while-S

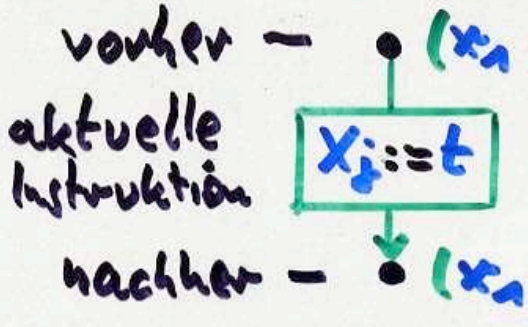
- Programmform : name
  vars: $X_1 \in D_1, \ldots, X_k \in D_k$
  {<compound> $\stackrel{*}{\rightarrow}$} body

- mit folgenden Syntaxregeln (BNF):

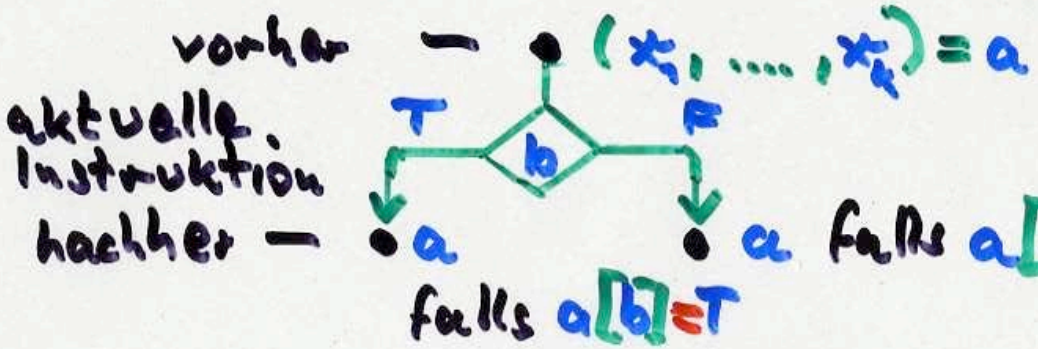
<compound> ::= begin <stmtList> end
<stmtlist> ::= <stmt> | <stmt> ; <stmtlist>
<stmt> ::= <assign> | <compound> | <loop> | <case>
<assign> ::= $X_i := t$ {Term des Typs D_i}
<loop> ::= while b do <stmt> {BOOL-Term}
<case> ::= if b then <stmt> [else <stmt>]
{optional}

Zustandstransformationssemantik

Eingang — ○ $a_0 = (x_{01}, \ldots, x_{0k}) \in D_1 \times \cdots \times D_k$
Anfangszustand



Berechnungszustand ist Wertzuweisung



Ausgang — □ $a_{end} = (y_1, \ldots, y_k)$ Endzustand
(nur bei Termination)

zum Beispiel....

flip

opns: $flip : \{0,1\}^* \to \{0,1\}^*$

vars: $u \in \{0,1\}^*$

eqns: $flip(\lambda) = \lambda$

$flip(0u) = 1\,flip(u)$

$flip(1u) = 0\,flip(u)$

... in CE-S

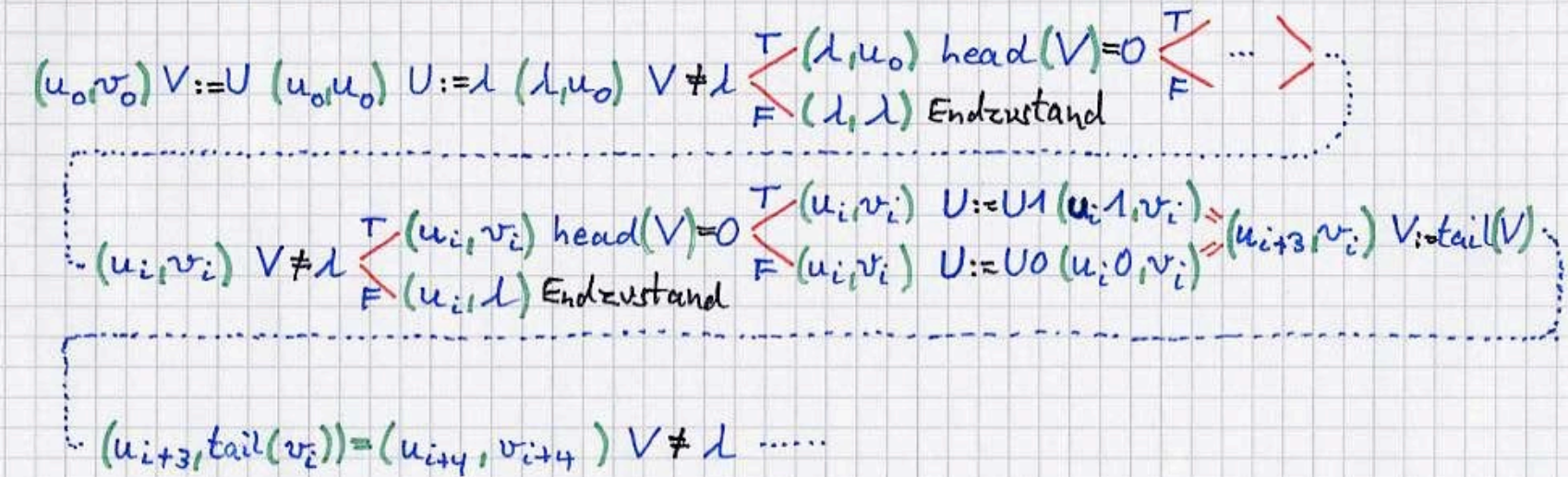
flip

vars: $U, V \in \{0,1\}^*$

```
begin
  V := U ; U := λ;
  while V ≠ λ do
    begin
      if head(V)=0 then U := U1
                   else U := U0;
      V := tail(V)
    end
end
```

.... in while-S

Berechnung von flip



Eigenschaften von flip

(1) für $k = 0, \ldots, \mathrm{length}(u_0)$ gilt:

(i) $u_{2+4k}\, \mathrm{flip}(v_{2+4k}) = \mathrm{flip}(u_0)$

(ii) $\mathrm{length}(u_{2+4k}) = k$

(2) $(u_{2+4\,\mathrm{length}(u_0)}, \ell)$ ist Endzustand

(3) insbes.: $u_{2+4\,\mathrm{length}\,u_0} = \mathrm{flip}(u_0)$

Beweis von (1) mit Induktion über k

IA: $k=0$: $(u_2, v_2) = (\lambda, u_0)$ nach Konstruktion und somit
$u_2\, flip(v_2) = \lambda\, flip(u_0) = flip(u_0)$ & $length(u_2) = length(\lambda) = 0$

IV: Behauptung gelte für $k \geq 0$

IS: $k+1$: nach Konstruktion gilt:

$(u_{2+4(k+1)}, v_{2+4(k+1)}) \underset{i=2+4k}{=} (u_{i+4}, v_{i+4}) = (u_{i+3}, tail(v_i)) = (u_i x_i, tail(v_i))$
für $x_i = 1$, falls $head(v_i) = 0$, und $x_i = 0$ sonst (d.h. $head(v_i) = 1$)

nach IV gilt: $u_i\, flip(v_i) = flip(v_0)$ & $length(u_i) = k$

daraus folgt zusammen mit der CE-S-Spezifikation von $flip$:

$u_{2+4(k+1)}\, flip(v_{2+4(k+1)}) = u_i x_i\, flip(tail(v_i))$
$u_i\, flip(head(v_i)\, tail(v_i)) = u_i\, flip(v_i) \underset{IV}{=} flip(v_0)$

$length(u_{2+4(k+1)}) = length(u_i x_i) = length(u_i) + length(x_i) = k+1$

Berechnung von while-S-Programmen in CE-S

- Zustandstransformationssemantik von while-S-Programm P ist (partielle) Funktion $st(P): D_1 \times \cdots \times D_k \to D_1 \times \cdots \times D_k$
- sei $st(P)(x_1, \ldots, x_k) = (y_1, \ldots, y_k)$ und $st_i(P)(x_1, \ldots, x_k) = y_i$ für $i = 1, \ldots, k$; dann gilt:

  $st(P)(x_1, \ldots, x_k) = (st_1(P)(x_1, \ldots, x_k), \ldots, st_k(P)(x_1, \ldots, x_k))$
- Variablen von bestimmtem Definitionsbereich; Rumpf den Ablauf der Berechnung
- Anweisung (wie der Rumpf) sind o.B.d.A. Elemente von $CHAR^*$

state transformation

opns: $st_i : CHAR^* \times D_1 \times \cdots \times D_k \to D_i \quad (i = 1, \ldots, k)$

vars: $x_1 \in D_1, \ldots, x_k \in D_k$; $stmt, stmtlist, b, xpr \in CHAR^*$

eqns:

$st_i(\text{begin } stmtlist \text{ end}, x_1, \ldots, x_k) = st_i(stmtlist, x_1, \ldots, x_k)$

$st_i(stmt\,;\,stmtlist, x_1, \ldots, x_k) =$
$\quad st_i(stmtlist, st_1(stmt, x_1, \ldots, x_k), \ldots, st_k(stmt, x_1, \ldots, x_k))$

$st_i(\text{while } b \text{ do } stmt, x_1, \ldots, x_k) = \text{if } eval(b, x_1, \ldots, x_k)$ [1]
$\quad$ then $st_i(stmt\,;\,\text{while } b \text{ do } stmt, x_1, \ldots, x_k)$ else x_i

$st_i(X_j := xpr, x_1, \ldots, x_k) = \begin{cases} eval(xpr, x_1, \ldots, x_k) & \text{für } i = j \\ x_i & \text{sonst} \end{cases}$

---

1) CE-S-Spezifikation für Termauswertung vorausgel.

```
while (TM)
  vars: W ∈ A*, LB, RB ∈ (A ∪ {□})*, Z ∈ S
   begin
    LB := λ ; Z := s0 ; RB := W ;
    while Z ∉ F do next
   end
```

d gegeben als Tabelle

s_1	a_1	s_1'	b_1	m_1
⋮				
s_i	a_i	s_i'	b_i	m_i
⋮				
s_l	a_l	s_l'	b_l	m_l

```
begin
 if Z=s_1 ∧ head(RB)=a_1 then stmt_1;
 if Z=s_i ∧ head(RB)=a_i then stmt_i;
 if Z=s_l ∧ head(RB)=a_l then stmt_l
end
```

next

1. Fall: s_i' b_i m_i

```
begin Z:=s_i'; RB:=b_i tail(RB) end
```

2. Fall: s_i' b_i l_i

```
begin Z:=s_i'; LB:=cutlastletter(LB);
      RB:=lastletter(LB) b_i tail(RB) end
```

3. Fall: s_i' b_i r_i

```
begin Z:=s_i'; LB:=LB b_i;
      RB:=tail(RB) end
```

stmt i