Grundzüge der Theoretischen Informatik

Markus Bläser

Universität des Saarlandes

8.12.2021

Kapitel 15: Der Satz von Rice

Fixpunktsatz

$f(x) = x+1$

Theorem (15.4, Fixpunktsatz)

Für alle WHILE-berechenbaren totalen Funktionen $f : \mathbb{N} \to \mathbb{N}$ *und alle* $n \in \mathbb{N} \setminus \{0\}$, *gibt es ein* $e \in \mathbb{N}$ *mit*

$$\varphi^n_{f(e)} = \varphi^n_e.$$

g und $g+1$ berechnen dieselbe Funktion.

Indexmengen

$i=j$: $i \in I$ und $\varphi_i = \varphi_i \Rightarrow i \in I$

Definition (15.5, Indexmenge)

$I \subseteq \mathbb{N}$ heißt *Indexmenge*, falls

$$\text{für alle } i, j \in \mathbb{N} \text{ gilt: } i \in I \text{ und } \varphi_i = \varphi_j \Longrightarrow j \in I.$$

Eine Indexmenge I ist *nicht-trivial*, falls zusätzlich $I \neq \emptyset$ und $I \neq \mathbb{N}$ gilt.

Indexmengen sind durch semantische Eigenschaften definiert:

Bemerkung

I ist Indexmenge genau dann, wenn es eine Menge F von WHILE-berechenbaren Funktionen gibt mit $I = \{i \in \mathbb{N} \mid \varphi_i \in F\}$.

Hot or not?

Welche Mengen sind Indexmengen?

1. $V_0 = \{i \in \mathbb{N} \mid \varphi_i(x) = 0 \text{ für alle } x \in \mathbb{N}\}$.
2. $N_1 = \{g \in \mathbb{N} \mid g \leq 10000\}$
3. $N_2 = \{g \in \mathbb{N} \mid \varphi_g(0) = 0 \text{ und } g \geq 10000\}$
4. $T = \{i \in \mathbb{N} \mid \varphi_i \text{ ist total}\}$
5. H_0, das spezielle Halteproblem
6. $D_c = \{i \in \mathbb{N} \mid |\operatorname{dom} \varphi_i| \geq c\}$ für alle $c \in \mathbb{N}$,
7. $\mathrm{Mon} = \{i \in \mathbb{N} \mid \varphi_i \text{ ist monoton}\}$
8. H, das Halteproblem

Der Satz von Rice

$H_0 \notin REC$ aber H_0 ist rekursiv Indexmenge

Theorem (15.8, Satz von Rice)

Jede nicht-triviale Indexmenge ist unentscheidbar

"Jede nicht-triviale semantische Programmeigenschaft ist unentscheidbar"

Der Satz von Rice liefert einen alternativen Beweis, dass V_0, V, T, D_c,... unentscheidbar sind.

$c \geq 1$

Beweis 15.8.

Sei I nicht-triviale Indexmenge

Seien $i \in I$ und $j \notin I$

Aus: I ist entscheidbar

Dann ist die Fkt:

$$h(x) = \begin{cases} i & \text{falls } x \notin I \\ j & \text{sonst} \end{cases}$$

WHILE-berechenbar.

Es gibt eine Gödelnummer e nach dem Fixpkt satz

mit $\varphi_e = \varphi_{h(e)}$

1) Falls $e \in I$ ist, dann ist $h(e) = j$, also $\varphi_e = \varphi_j$.

aber I ist eine Indexmenge, also ist $j \in I$ da $e \in I$.

↯ da $j \notin I$

2) Falls $e \notin I$, dann ist $h(e) = i$, also $\varphi_e = \varphi_i$.

Da aber $i \in I$ und I Indexmenge, ist auch $e \in I$ ↯

also muss I unentscheidbar sein. □


Even when it's not
funny, simply repeat it.

Bitte passen Sie auf!

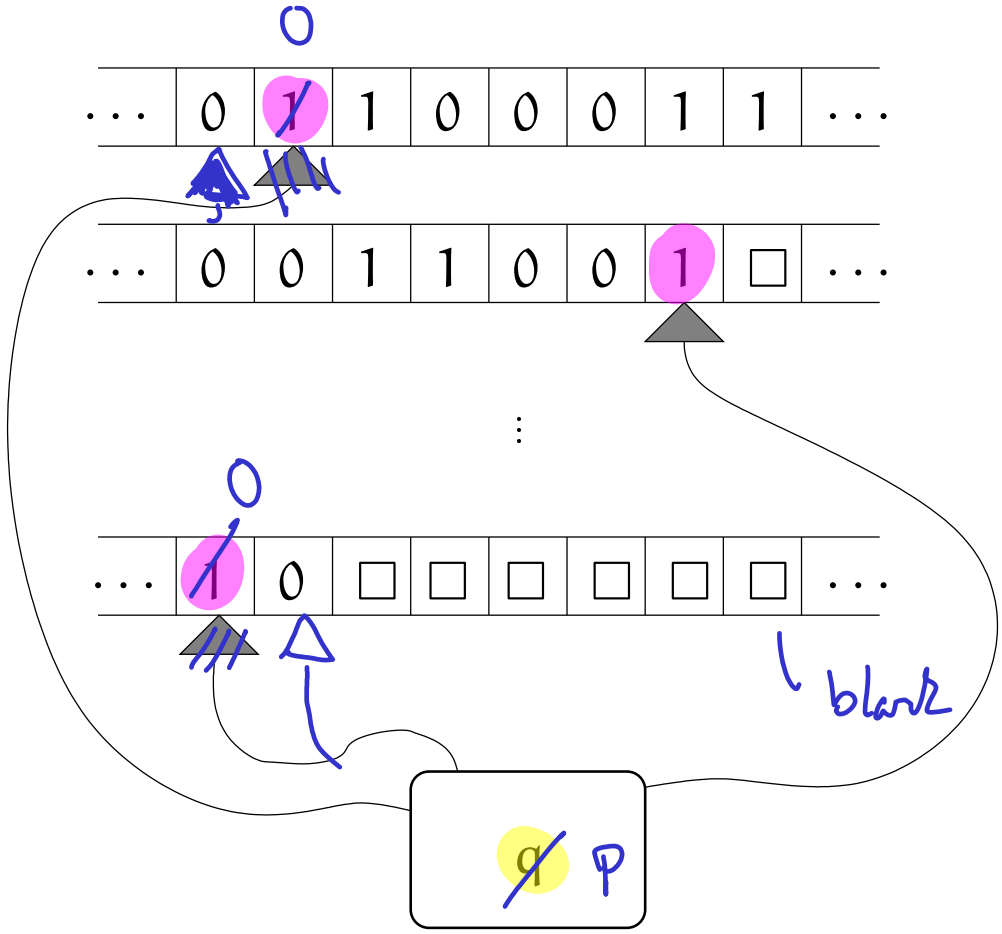
Methoden, um zu zeigen, dass $\mathrm{L} \subseteq \mathbb{N}$ unentscheidbar ist:
(Nicht alle sind von mir zertifiziert, werden dennoch gerne in Abgaben und Klausuren angewandt.)

- **Universelle Methode:** Reduzieren von H_0 auf L.
- **Gute Methode:** Beweisen, dass L nicht-triviale Indexmenge ist. Satz von Rice anwenden.
- **Akzeptable Methode:** Beweisen, dass L nicht-triviale Indexmenge ist. Gehirn ausschalten. Satz von Rice anwenden.
- **Inakzeptable Methode:** Gehirn ausschalten, Satz von Rice anwenden.
- **Schlechte Methode:** Gehirn ausschalten, Satz von Rice anwenden, behaupten, dass $\mathrm{L} \notin \mathrm{RE}$.

Kapitel 16: Turingmaschinen

Turingmaschinen

DEA: $\delta: Q \times \Sigma \to Q$



$\Gamma \supseteq \Sigma$

$\delta(q, 1, 1, 1) = (p, 0, 1, 0, L, S, R)$

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, S, R\}^k$$

δ ist partiell, $\delta(q, \gamma_1, \ldots, \gamma_2)$ undef $\Rightarrow$ TM hält

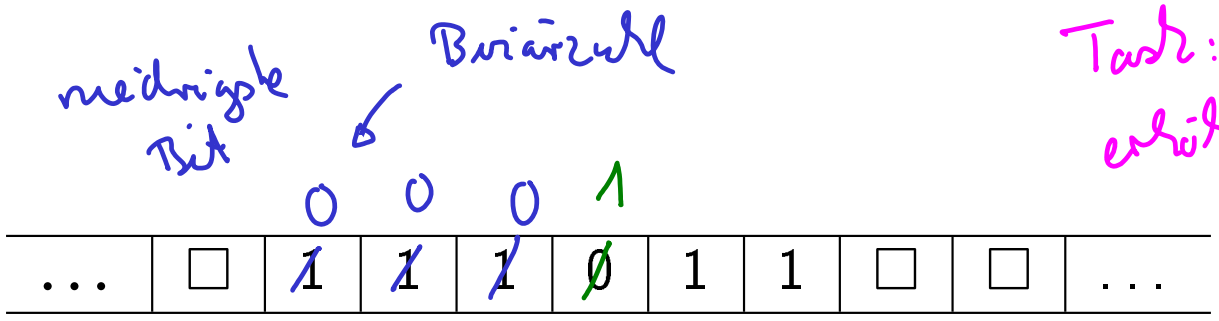
Definition

Definition (16.1)

Eine k-Band-Turingmaschine M ist ein Tupel $(Q, \Sigma, \Gamma, \delta, q_0)$ mit:

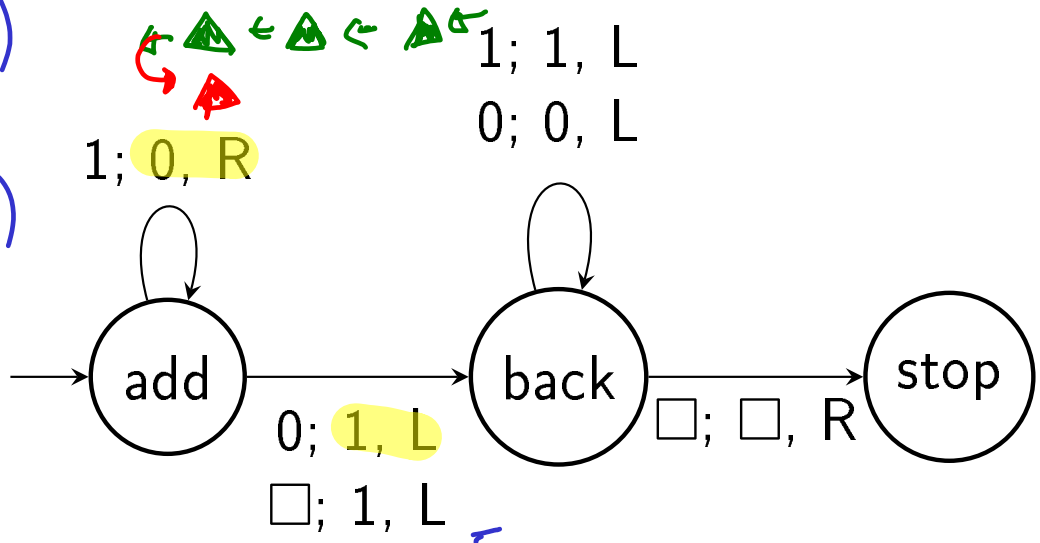
1. Q ist eine endliche Menge, die Menge der *Zustände*.
2. Σ ist eine endliche Menge, das *Eingabealphabet*.
3. Γ ist eine endliche Menge, das *Bandalphabet*.
   $\square \in \Gamma$ ist das Leerzeichen, $\Sigma \subseteq \Gamma \setminus \{\square\}$.
4. $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, S, R\}^k$ ist die *Übergangsfunktion*.
5. $q_0 \in Q$ ist der *Startzustand*.

Beispiel



Task: erhöhe die Zahl um 1 gehe wieder nach links

(add, (1, 111011))
⊢
(add, (2, 011011))
⊢
⋮



Zustand

Γ

δ	0	1	□
add	(back, 1, L)	(add, 0, R)	(back, 1, L)
back	(back, 0, L)	(back, 1, L)	(stop, □, R)
stop	—	—	—

Σ

Γ = Σ ∪ {□}

q

δ(add, 0) = (back, 1, L)

Konfigurationen und Berechnungen

Endliche Automaten:

- Berechnung = Folge von Zuständen
- darin implizit: wieviel von der Eingabe gelesen wurde
- kennt man den Zustand und den Rest der Eingabe, so kennt man das weitere Vehalten des Automaten

Turingmaschinen:

- Input kann mehrfach gelesen werden
- Bandinhalte können geändert werden
- Um das weitere Verhalten der Turingmaschine zu kennen, benötigt man:
  - den aktuellen Zustand
  - die aktuellen Bandinhalte
  - die Positionen der Köpfe

Konfigurationen

speichert alle Informationen, die nötig, um das weitere Verhalten der TM zu steuern.

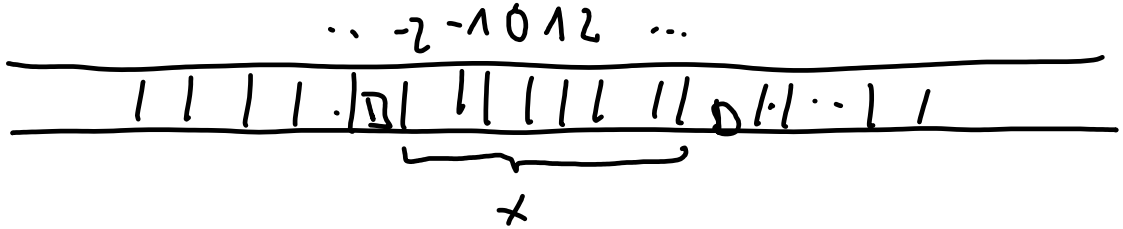
- Modellierung eines Bands: Funktion $t : \mathbb{Z} \to \Gamma$
- Am Anfang: $t(i) = \square$ für alle $i \in \mathbb{Z}$
- Die Turingmaschine kann nur einen endlichen Teil beschreiben.
- Die absolute Position ist irrelevant.

gibt, wollen wir aber nicht

Modellierung eines Bandes

$(p, x) \in \mathbb{N} \times \Gamma^*$

- x ist der Bandinhalt (bislang besuchte Zellen)
- p $(1 \leq p \leq |x|)$ ist die relative Position auf x



Konfigurationen (2)

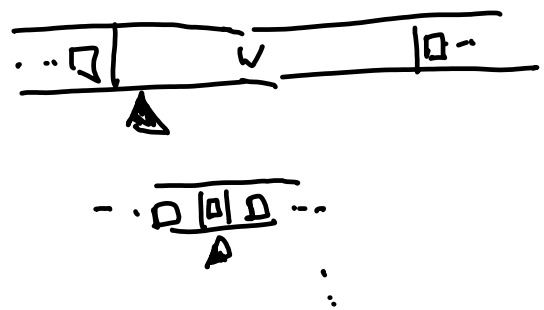
Definition (Konfiguration)

$(q, (p_1, x_1), \ldots (p_k, x_k)) \in Q \times (\mathbb{N} \times \Gamma^*)^k$

- q ist der Zustand
- (p_κ, x_κ) beschreibt das Band κ $\quad 1 \leq p_\kappa \leq |x_\kappa|$

Definition (Startkonfiguration auf w)

$(q_0, (1, w), (1, \square), \ldots, (1, \square))$.



Berechnungen

- $C = (q, (p_1, x_1), \ldots (p_k, x_k))$
- $C' = (q', (p'_1, x'_1), \ldots (p'_k, x'_k))$
- $x_\kappa = u_\kappa \alpha_\kappa v_\kappa$, wobei $|u_\kappa| = p_\kappa - 1$ und $\alpha_\kappa \in \Gamma$, $1 \leq \kappa \leq k$.

C' heißt *Nachfolgekonfiguration* von C, falls C' durch einen Schritt von M von C erreicht wird.

D.h. falls $\delta(q, \alpha_1, \ldots, \alpha_k) = (q', \beta_1, \ldots, \beta_k, r_1, \ldots, r_k)$, dann ist

$$x'_\kappa = u_\kappa \beta_\kappa v_\kappa, \quad 1 \leq \kappa \leq k$$

und

$$p'_\kappa = \begin{cases} p_\kappa - 1 & \text{falls } r_\kappa = L, \\ p_\kappa & \text{falls } r_\kappa = S, \\ p_\kappa + 1 & \text{falls } r_\kappa = R. \end{cases}$$

Berechnungen (2)

Randfälle:

- Falls $p_\kappa = 1$ und $r_\kappa = L$, dann ist

$$x'_\kappa = \square \beta_\kappa v_\kappa$$

und

$$p'_\kappa = 1.$$

- Falls $p_\kappa = |x_\kappa|$ and $r_\kappa = R$, dann ist

$$x'_\kappa = u_\kappa \beta_\kappa \square$$

und

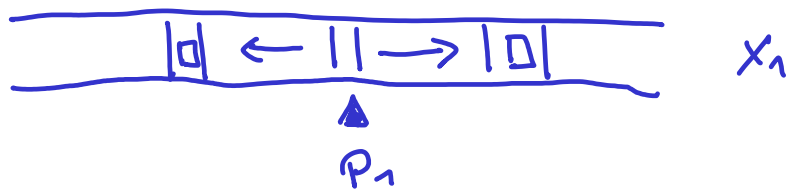
$$p'_\kappa = |x_\kappa| + 1.$$

Berechnungen (3)

C' ist (direkte) Nachfolgekonf. von C

- Notation: $C \vdash_M C'$
- $\vdash_M^*$ bezeichnet die reflexiv-transitive Hülle
- $C \vdash_M^* C'$ falls es $C_1, \ldots, C_\ell$ gibt mit $C \vdash_M C_1 \vdash_M \ldots \vdash_M C_\ell \vdash_M C'$. oder $C = C'$
- Eine Konfiguration ohne Nachfolger heißt *haltend*.
- M hält auf w, falls $\mathrm{SC}_M(w) \vdash_M^* C_t$ und C_t ist haltend.
- $\mathrm{SC}_M(w) \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_t$ heißt *Berechnung* von M auf w.
- Falls C_t nicht existiert, so hält M nicht auf w.
  Die zugehörige Berechnung ist unendlich.

Berechnungen (4)



- Sei $SC_M(w) \vdash_M^* C_t$, $C_t = (q, (p_1, x_1), \ldots, (p_k, x_k))$ haltend.
- Sei $i \leq p_1$ der größte Index mit $x_1(i) = \square$.
  ($i = 0$ falls der Index nicht existiert.)
- Sei $j \geq p_1$ der kleinste Index mit $x_1(j) = \square$.
  ($j = |x_1| + 1$, falls der Index nicht existiert.)
- $x_1(i+1)x_1(i+2)\ldots x_1(j-1)$ ist die *Ausgabe* von M auf w.
- Berechnete Funktion: $\varphi_M : \Sigma^* \to (\Gamma \setminus \{\square\})^*$

$$\varphi_M(w) = \begin{cases} \text{Ausgabe von } M \text{ auf } w & \text{falls } M \text{ auf } w \text{ hält,} \\ \text{undefiniert} & \text{sonst.} \end{cases}$$

Berechnete Funktionen und Sprachen

Definition (16.3)

$f : \Sigma^* \to \Sigma^*$ ist *Turing-berechenbar*, falls $f = \varphi_M$ für eine Turingmaschine $M = (Q, \Sigma, \Gamma, \delta, q_0, Q_{acc})$.

- Wir könnten $L \subseteq \Sigma^*$ Turing-entscheidbar nennen, falls $\chi_L : \Sigma^* \to \{0, 1\}$ Turing-berechenbar ist. (0, 1 als Elemente von Σ aufgefasst.)
- Stattdessen arbeiten wir mit akzeptierenden Zuständen $Q_{acc} \subseteq Q$.
- Eine haltende Konfiguration $(q, (p_1, x_1), \ldots, (p_k, x_k))$ heißt *akzeptierend*, falls $q \in Q_{acc}$. Sonst heißt sie *verwerfend*.

Berechnete Funktionen und Sprachen (2)

Definition (16.4)

Sei $L \subseteq \Sigma^*$.

1. $M = (Q, \Sigma, \Gamma, \delta, q_0, Q_{acc})$ *erkennt* $L \subseteq \Sigma^*$, falls für alle $w \in L$ die Berechnung von M in einer akzeptierenden Konfiguration endet und für alle $w \notin L$ nicht.
   (D.h. sie endet entweder in einer verwerfenden Konfiguration oder M hält nicht auf w.)
2. M *entscheidet* L, falls zusätzlich M auch auf alle $w \notin L$ hält.
3. $L(M)$ bezeichnet die von M erkannte Sprache.