Berechenbarkeit/Entscheidbarkeit

Frage: Ist eine algorithmische Problemstellung lösbar?

- was ist eine algorithmische Problemstellung?
  → *formale Sprachen*
- benötigen einen Berechenbarkeitsbegriff
  → *Maschinenmodelle* (unterschiedlich mächtig!)
  - bisher: endliche Automaten (nicht mächtig genug!)
  - jetzt: Turingmaschine

Bsp.: Postsches Korrespondenzproblem

Geg.: Folge von k Wortpaaren ("*Korrespondenzen*")

$$K = \{(x_1,y_1), (x_2,y_2), \ldots, (x_k,y_k)\}$$

über einem endlichen Alphabet Σ wobei x_i, y_i nichtleere Wörter sind

Ges.: Gibt es eine Zahl $n \geq 1$ und Indizes $i_1,\ldots i_n \in \{1,\ldots,k\}$, so dass

$$x_{i_1}x_{i_2} \ldots x_{i_n} = y_{i_1}y_{i_2} \ldots y_{i_n}$$

- **Bsp.**: $K = \{(1, 111), (10111, 10), (10, 0)\}$ hat Lösung
  $x_2x_1x_1x_3 = 10111|1|1|10 = 10|111|111|0 = y_2y_1y_1y_3$

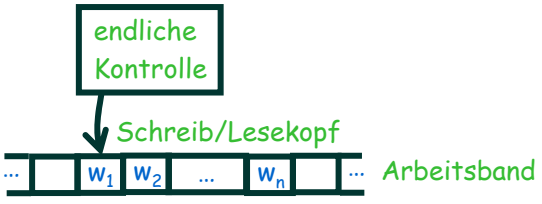
Algorithmische Probleme

- Übersetzung algorithmischer Probleme in Wortprobleme für formale Sprachen:
  - L_{prim} = { pϵ{0,1}* | p ist Binärdarstellung einer Primzahl}
    = {10, 11 ,101 ,111 , 1011, …}
  - L_{JAVA} = Menge aller syntaktisch korrekten JAVA Programme
  - L_{PKP} = { $x_1\#y_1\#x_2\#y_2\#\ldots\#x_k\#y_k \in (\Sigma \cup \{\#\})^*$ |
    es gibt $n \geq 1$ und $i_1,\ldots i_n \in \{1,\ldots,k\}$, so dass
    $x_{i_1}x_{i_2} \ldots x_{i_n} = y_{i_1}y_{i_2} \ldots y_{i_n}$}

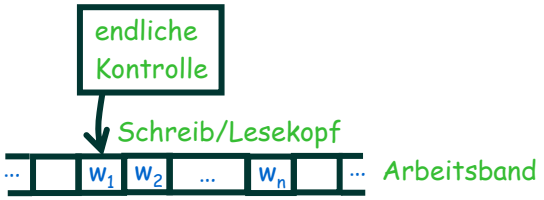
Deterministische Turing-Maschine

Turingmaschine

- hat endlich viele innere Zustände
- hat ein beidseitig unendliches Arbeitsband
  (1 Zeichen pro Zelle, zu jedem Zeitpunkt nur endlich viele Zellen beschriftet)
- hat einen beweglichen Schreib/Lesekopf
  (zeigt jeweils auf eine Zelle des Arbeitsbandes)



Arbeitsweise der Turing-Maschine



vom Arbeitsband gelesenes Symbol an aktueller Position +
aktueller innerer Zustand
→ schreibe neues Symbol an aktuelle Position +
bewege Schreib /Lesekopf um ≤1 Feld +
gehe in neuen inneren Zustand

Deterministische Turing-Maschine

Def.: Eine deterministische Turing-Maschine M (*dtm*)
ist ein 7-Tupel $(Q,\Sigma,\Gamma,\delta,q_0,\underline{b},F)$ mit

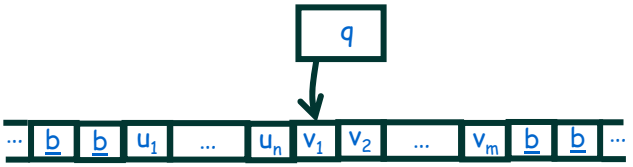
- Q endliche Menge von inneren Zuständen
- Σ endliches Eingabealphabet
- Γ endliches Bandalphabet $\Sigma \subset \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R,S\}$
  Zustandsüberführungsfunktion
  (L=links, R=rechts, S=stehenbleiben)
- $q_0 \in Q$ Anfangszustand
- $\underline{b} \in \Gamma \setminus \Sigma$ Blanksymbol
- $F = F_{ja} \cup F_{nein} \subset Q$ Menge der Endzustände mit
  - F_{ja} akzeptierenden Endzustände
  - F_{nein} nicht akzeptierenden Endzustände

Konfiguration einer dtm

Def.: Eine Konfiguration einer dtm $M=(Q,\Sigma,\Gamma,\delta,q_0,\underline{b},F)$ ist ein Wort aus $\Gamma^*Q\Gamma^+$

("Momentaufnahme" von M)

Bsp.:



→ Konfiguration $u_1...u_nqv_1..v_m\underline{b}\in\Gamma^*Q\Gamma^+$

Def.: Die Anfangskonfiguration einer dtm $M=(Q,\Sigma,\Gamma,\delta,q_0,\underline{b},F)$ auf der Eingabe $w\in\Sigma^*$ ist $c(w)=\epsilon q_0 w\underline{b}$

Nachfolgekonfiguration

Def.: Sei $c=u_1...u_nqv_1..v_m\in\Gamma^*Q\Gamma^+$ mit $n\geq 0$, $m\geq 1$ Konfiguration einer dtm $M=(Q,\Sigma,\Gamma,\delta,q_0,\underline{b},F)$. Die Nachfolgekonfiguration $N_M(c)$ von c ist

c	falls	$q\in F$
$u_1...u_{n-1}q'u_nv'_1..v_m$	falls	$\delta(q,v_1)=(q',v'_1,L)$ und $n>0$
$q'\underline{b}v'_1..v_m$	falls	$\delta(q,v_1)=(q',v'_1,L)$ und $n=0$
$u_1...u_nv'_1q'v_2..v_m$	falls	$\delta(q,v_1)=(q',v'_1,R)$ und $m>1$
$u_1...u_nv'_1q'\underline{b}$	falls	$\delta(q,v_1)=(q',v'_1,R)$ und $m=1$
$u_1...u_nq'v'_1v_2..v_m$	falls	$\delta(q,v_1)=(q',v'_1,S)$

Berechnungspfad einer dtm M

Def.: Der Berechnungspfad von M auf $w \in \Sigma^*$ ist die Folge der Konfigurationen $B_M(w) = c_0, c_1, c_2, \ldots$ die M bei der Abarbeitung von w durchläuft:

- $c_0 = c(w)$
- $c_{i+1} = N_M(c_i)$ falls $c_i = uqv$ mit $q \in F^c$

Bem.: $B_M(w)$ kann unendlich sein !! → *"M hält nicht auf w"*
$B_M(w)$ endlich → *"M hält auf w"*

Akzeptanzkriterium einer dtm M

Def.:

- M akzeptiert $w \in \Sigma^*$, falls M auf w in einer Konfiguration uqv mit $q \in F_{ja}$ hält
- M verwirft $w \in \Sigma^*$, falls M auf w in einer Konfiguration uqv mit $q \in F_{nein}$ hält

(Semi-)Entscheidbare Sprachen

Def.:

- $L \subseteq \Sigma^*$ heißt entscheidbar (oder rekursiv), falls es eine dtm M gibt, die auf allen Eingaben stoppt und genau die $w \in L$ akzeptiert (und damit genau die $w \in L^c$ verwirft)

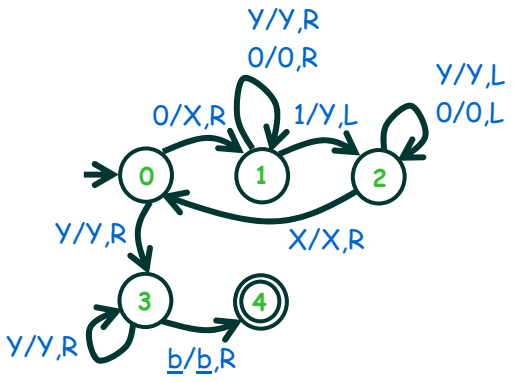
- $L \subseteq \Sigma^*$ heißt semi-entscheidbar, falls es eine dtm M gibt, die genau die $w \in L$ akzeptiert
  (**ACHTUNG**: M muss auf $w \in L^c$ nicht halten!)

Eine rekursive Sprache

Bsp.: $L = \{0^n1^n \mid n \geq 1\}$ ist rekursiv

Idee: Ändere alternierend eine 0 in ein X und eine 1 in ein Y bis alle 0/1en überschrieben sind

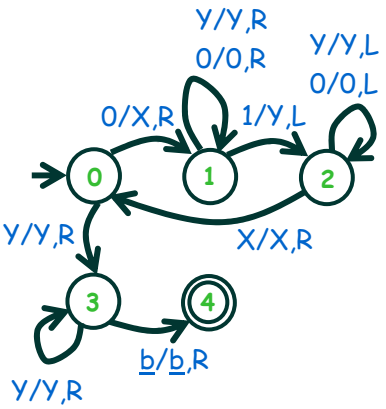
0 : linkeste 0
1 : gehe nach rechts
(überlese 0 und Y)
2 : gehe nach links
(überlese 0 und Y)
3 : teste, ob nur noch Y auf dem Band steht



Eine rekursive Sprache

Bsp.:

- Berechnungspfad auf w=0011



00011 → X1011 → X0111 →X20Y1 →
2X0Y1 → X00Y1 → XX1Y1 →XXY11 →
XX2YY → X2XYY → XX0YY →
XXY3Y → XXYY3b → XXYYb4b

- Berechnungspfad auf w=0010

00010 → X1010 → X0110 → X20Y0 →
2X0Y0 → X00Y0 → XX1Y0 → XXY10 → XXY01b

Abschlußeigenschaften rekursiver Sprachen

Satz: L entscheidbar → L^c entscheidbar

Bew.:

vertausche akzeptierende und verwerfende Endzustände

Frage: Was ist mit semi ~~entscheidbaren~~ Sprachen ??

Von dtm M berechnete Funktion

Def.: M berechnet $uv \in \Gamma^*$ auf Eingabe $w \in \Sigma^*$, falls M auf w in einer Konfiguration uqv mit $q \in F$ hält

- partielle Funktion $f_M : \Sigma^* \rightarrow \Gamma^*$ mit
  - $f_M(w)$ undefiniert, falls M auf w nicht hält
  - $f_M(w)=u$, falls u von M auf w berechnet wird

Turing-berechenbare Funktionen

Def.:

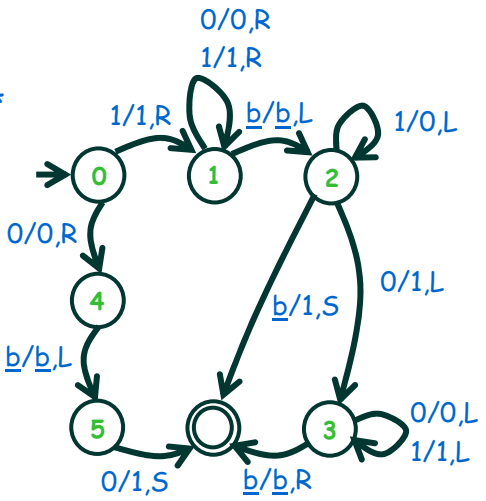
- $f : \Sigma^* \rightarrow \Sigma^*$ heisst Turing berechenbar, falls es eine dtm M gibt, so dass $f = f_M$
- $f : \mathbb{N}^k \rightarrow \mathbb{N}$ heisst Turing berechenbar, falls es eine dtm M gibt, so dass für alle $n_1,\ldots n_k,m$ gilt:
  $f(n_1,\ldots n_k)=m$ gdw.
  M berechnet bin(m) auf $bin(n_1)\#bin(n_2)\#\ldots\#bin(n_k)$

Eine Turing-berechenbare Funktion

Bsp.: succ : $\mathbb{N} \to \mathbb{N}$ mit succ(w)=w+1 ist Turing berechenbar

- $f_M(w)$=succ(w), $w \in 0 \cup 1(0 \cup 1)^*$
- $f_M(w)$ undefiniert, sonst

1 : nach rechts gehen
2 : Übertrag bilden
3 : nach links gehen
4,5: für w=0



Charakteristische Funktion

Satz:

- $L \subseteq \Sigma^*$ entscheidbar gdw.
  $\chi_L : \Sigma^* \to \{0,1\}$ mit
  $\chi_L(w)=1$, falls $w \in L$ und $\chi_L(w)=0$, falls $w \in L^c$
  ("charakteristische Funktion von L")
  Turing berechenbar

- $L \subseteq \Sigma^*$ semi- entscheidbar gdw.
  $\chi^*_L : \Sigma^* \to \{0,1\}$ mit
  $\chi^*_L(w)=1$, falls $w \in L$ und $\chi^*_L(w)$ undefiniert, sonst
  Turing berechenbar

„High-level" Beschreibung von dtm

Bsp.: $L = \{a^i b^j c^k \mid ij = k \text{ und } i,j,k \geq 1\} \subset \{a,b,c\}^*$ ist rekursiv

Arbeitsweise einer dtm M die L erkennt:

1. Prüfe, ob die Eingabe die Form a*b*c* hat
   (verwerfe, falls nicht)
2. Gehe ans linke Ende der Eingabe
3. Streiche ein a, dann abwechselnd ein b und ein c bis alle b gestrichen sind
4. Stelle die b wieder her
5. Wiederhole Phase 3, falls es noch ein a gibt
6. Falls es kein a mehr gibt, prüfe, ob alle c gestrichen sind.
   Falls ja akzeptiere, ansonsten verwerfe

Rechenzeit & Platzbedarf einer dtm M

Def.:

- $t_M(w)$:= Rechenzeit von M auf $w \in \Sigma^*$,
  = Länge des Berechnungspfades $B_M(w)$
- $s_M(w)$:= Platzbedarf von M auf $w \in \Sigma^*$,
  = maximale Länge einer Konfiguration
  auf dem Berechnungspfad $B_M(w)$
- $t_M(n) := \max_{w \in \Sigma^n} t_M(w)$ heisst Rechenzeit von M
- $s_M(n) := \max_{w \in \Sigma^n} s_M(w)$ heisst Platzbedarf von M
  "worst-case"

These von Church und Turing

Die Klasse der Turing-berechenbaren Funktionen ist genau die Klasse der im intuitiven Sinn überhaupt berechenbaren Funktionen

Bem.:

- These → *nicht beweisbar*
- "vorherrschende" Meinung
- gibt keine "Gegenbeispiele"
- bisher wurden keine realistischen stärkeren Maschinenmodelle gefunden
- völlig andersartige Formalisierungen des intuitiven Berechenbarkeitsbegriffs liefern die gleiche Klasse von Funktionen (λ Kalkül, μ rekursive Funktionen, RAM Modell)