8. Turingmaschinen und kontextsensitive Sprachen

Turingmaschinen (TM) von A. Turing vorgeschlagen, um den Begriff der Berechenbarkeit formal zu präzisieren.

Intuitiv: statt des Stacks bei Kellerautomaten verfügt eine TM über ein potenziell unendliches Band, dessen Felder Zeichen des Arbeitsalphabets enthalten können. Ein Lese/Schreibkopf kann sich an einer beliebigen Position des Bandes befinden und jeweils eine Position nach rechts oder links wandern. Die Eingabe befindet sich im Startzustand auf dem Band.

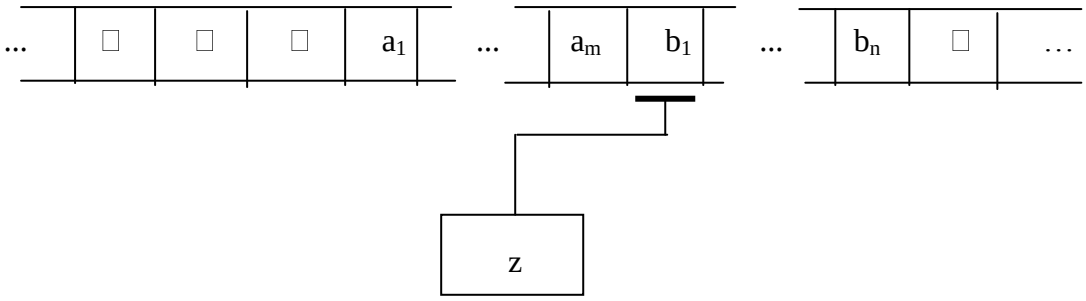
Def.: Eine Turingmaschine ist ein 7-Tupel M = (Z, Σ, Γ, δ, z_0, □, E). Hierbei ist

Z endliche Zustandsmenge
Σ Eingabealphabet
Γ Arbeitsalphabet
δ Überführungsfunktion
z_0 Anfangszustand
□ Leerzeichen (Blank)
E Endzustände

falls M deterministisch: $\delta: Z \times \Gamma \rightarrow Z \times \Gamma \times \{L,R,N\}$
falls M nichtdeterministisch: $\delta: Z \times \Gamma \rightarrow Pot(Z \times \Gamma \times \{L,R,N\})$

Def.: Eine Konfiguration einer TM ist ein Wort $k \in \Gamma^* Z \Gamma^*$.

Intuitiv repräsentiert die Konfiguration $a_1...a_m z b_1...b_n$ den Zustand



Die Eingabe x der TM steht auf dem Band: Startkonfiguration z_0x

Def.: Die Konfigurationsübergangsrelation |-- ist wie folgt definiert:

$a_1...a_m z b_1...b_n$ |-- $a_1...a_m z' c b_2...b_n$ falls $(z', c, N) \in \delta(z,b_1)$, $m \geq 0$, $n \geq 1$
$a_1...a_m z b_1...b_n$ |-- $a_1...a_m c z' b_2...b_n$ falls $(z', c, R) \in \delta(z,b_1)$, $m \geq 0$, $n \geq 2$
$a_1...a_m z b_1...b_n$ |-- $a_1...a_{m-1} z' a_m c b_2...b_n$ falls $(z', c, L) \in \delta(z,b_1)$, $m \geq 1$, $n \geq 1$

Sonderfälle, bei denen neues Feld besucht wird:

$a_1...a_m z b_1...b_n$ |-- $a_1...a_m c z'$□ falls $(z', c, R) \in \delta(z,b_1)$, $m \geq 0$, $n = 1$
$z b_1...b_n$ |-- z'□$c b_2...b_n$ falls $(z', c, L) \in \delta(z,b_1)$, $m = 0$, $n \geq 1$

Def.: Die von einer TM M akzeptierte Sprache ist:

$$T(M) = \{x \in \Sigma^* \mid z_0x \vdash^* \alpha z\beta, z \in E, \alpha, \beta \in \Gamma^* \}$$

Ein einfaches Beispiel:

$M_1 = (\{q_0, q_1, q_f\}, \{0,1\}, \{0,1, \square\}, \delta, q_0, \square, \{q_f\})$

mit $\delta(q_0,0) = (q_1,0,R)$
$\delta(q_1,1) = (q_0,1,R)$
$\delta(q_1,\square) = (q_f,\square,N)$

Übergänge:

$q_0$01010 |-- 0$q_1$1010 |-- 01$q_0$010 |-- 010$q_1$10 |-- 0101$q_0$0 |-- 01010$q_1\square$ |-- 01010$q_f\square$

erkennt (01)*0

weiteres Beispiel: deterministische TM M_2, die Zeichenketten aus 0,1 erkennt, in denen genauso viele Nullen wie Einsen vorkommen.

Idee: wir lesen das erste Symbol und merken uns im Zustand, welches es war. Dann lesen wir das zweite. Wenn es das andere Symbol ist, dann löschen wir es und machen rechts weiter. Wenn nicht, dann löschen wir es auch, merken uns aber, dass wir eine Position suchen müssen, die das andere Symbol enthält. Diese wird durch das erste Symbol ersetzt, wir gehen an den neuen linken Rand zurück usw.

q_a Anfangszustand	q_f Endustand
q_0 Null gelesen	q_1 Eins gelesen
s_0 Null zu überschreiben	s_1 Eins zu überschreiben
q_r suche linken Rand	

$\delta(q_a,0) = (q_0,\square,R)$	$\delta(q_r,0) = (q_r,0,L)$
$\delta(q_a,1) = (q_1,\square,R)$	$\delta(q_r,1) = (q_r,1,L)$
$\delta(q_a,\square) = (q_f,\square,N)$	$\delta(q_r,\square) = (q_a,\square,R)$
$\delta(q_1,0) = (q_a,\square,R)$	$\delta(s_0,1) = (s_0,1,R)$
$\delta(q_0,1) = (q_a,\square,R)$	$\delta(s_0,0) = (q_r,1,L)$
$\delta(q_1,1) = (s_0,\square,R)$	$\delta(s_1,1) = (q_r,0,L)$
$\delta(q_0,0) = (s_1,\square,R)$	$\delta(s_1,0) = (s_1,0,R)$

Konfigurationsübergänge bei Eingabe 110100:
q_a110100 |-- $\square$ $q_1$10100 |-- $\square\square$ $s_0$0100 |-- $\square q_r \square$1100 |-- $\square\square q_a$1100 |-- $\square\square\square q_1$100 |-- $\square\square\square\square s_0$00 |-- $\square\square\square q_r\square$10 |-- $\square\square\square\square q_a$10 |-- $\square\square\square\square\square q_1$0 |-- $\square\square\square\square\square\square q_a\square$
|-- $\square\square\square\square\square\square q_f\square$

Bisher: TMs, die Sprachen akzeptieren. Man kann auch die im Endzustand auf dem Band stehende Zeichenkette als Rechenergebnis betrachten.

Beispiel:

$M_3 = (\{q_0, q_1, q_2, q_f\}, \{0,1\}, \{0,1, \square\}, \delta, q_0, \square, \{q_f\})$

mit $\delta(q_0,0) = (q_0,0,R)$
$\delta(q_0,1) = (q_0,1,R)$
$\delta(q_0,\square) = (q_1,\square,L)$

$\delta(q_1,0) = (q_2,1,L)$
$\delta(q_1,1) = (q_1,0,L)$
$\delta(q_1,\square) = (q_f,1,N)$

$\delta(q_2,0) = (q_2,0,L)$
$\delta(q_2,1) = (q_2,1,L)$
$\delta(q_2,\square) = (q_f, \square,R)$

Intuitiv: Wandern an rechtes Ende der Eingabe, Ersetzen der am weitesten rechts stehenden 0 durch 1, aller rechts davon stehenden 1en durch 0; wenn keine 0 vorkommt, dann 1 vor Eingabe schreiben und alle 1 en in 0 umwandeln: entspricht +1 in Binärdarstellung

$q_010 \vdash 1q_00 \vdash 10q_0\square \vdash 1q_10\square \vdash q_211\square \vdash q_2\square11\square \vdash \square q_f11\square$

$q_011 \vdash 1q_01 \vdash 11q_0\square \vdash 1q_11\square \vdash q_110\square \vdash q_1\square00\square \vdash q_f100\square$

Satz1: Die durch Turingmaschinen akzeptierbaren Sprachen sind genau die Typ-0 Sprachen.
Beweis: später.

Eine Turingmaschine heißt linear beschränkt (LBA, linear bounded automaton), wenn sie bei ihrer Berechnung immer mit dem Teil des Bandes auskommt, auf dem die Eingabe steht.

Bei solchen TMs wird das Eingabealphabet Σ verdoppelt: zu jedem Eingabesymbol a gibt es ein neues Symbol â: $\Sigma' = \Sigma \cup \{ \hat{a} \mid a \in \Sigma\}$ (manchmal verwenden wir auch a'). Die neuen Symbole dienen dazu, den rechten Rand der Eingabe zu markieren, den die TM nicht überschreiten darf.

Def.: Eine nichtdeterministische Turingmaschine M heißt linear beschränkt, wenn für alle $a_1...a_{n-1}a_n \in \Sigma^+$ und alle Konfigurationen $\alpha z\beta$ mit $z_0a_1...a_{n-1}\hat{a}_n \vdash^* \alpha z\beta$ gilt: $|\alpha\beta| \leq n$.

Von LBA M akzeptierte Sprache:

$$T(M) = \{ a_1...a_{n-1}a_n \in \Sigma^* \mid z_0a_1...a_{n-1}\hat{a}_n \vdash^* \alpha z\beta, z \in E, \alpha, \beta \in \Gamma^* \}$$

Unsere TM M_1 ist fast linear beschränkt (1 Blank zuviel). Lässt sich beheben durch geeignete zusätzliche Regeln für 1' und 0'.

$M = (\{q_0, q_1, q_f\}, \{0,1,0',1'\}, \{0,1, 0', 1', \square\}, \delta, q_0, \square, \{q_f\})$

mit $\delta(q_0,0) = (q_1,0,R)$
$\delta(q_1,1) = (q_0,1,R)$
$\delta(q_0,0') = (q_f,0',N)$

Übergänge:

$q_0 01010' \vdash 0q_1 1010' \vdash 01q_0 010' \vdash 010q_1 10' \vdash 0101q_0 0' \vdash 0101q_f 0'$

Satz 2: Die von linear beschränkten, nichtdeterministischen TMs akzeptierten Sprachen sind genau die kontextsensitiven (Typ 1) Sprachen.

Beweis (Skizze): (<=)
Sei A = L(G) für die kontextsensitive Grammatik G. Man kann eine TM M konstruieren, die nichtdeterministisch für jede Regel u -> v in der Grammatik prüft, ob v auf dem Band steht, und falls das der Fall ist v durch u ersetzt. Falls u kürzer ist als v, so werden die Bandsymbole, die rechts von v vorkamen, entsprechend nach links verschoben. Genau dann, wenn - abgesehen von Blanks - nur noch das Startsymbol S auf dem Band steht, wird das ursprüngliche Wort aus G erzeugt.

(=>) Sei $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ LBA. Wir definieren eine kontextsensitive Grammatik G, die T(M) generiert.
Um Konfigurationen zu beschreiben, brauchen wir Symbole aus $\Delta = \Gamma \cup (Z \times \Gamma)$.
Konfigurationen $a_1...a_{n-1}za_n...a_m$ werden dargestellt als Wörter (der Länge m!) der Gestalt $a_1...a_{n-1}(z, a_n)...a_m$ über Δ.
δ-Übergänge von M können durch kontextsensitive Regeln beschrieben werden. So ergibt etwa $(q',b, R) \in \delta(q,a)$ die Regeln

$$(q,a)c \rightarrow b(q',c) \text{ für alle } c \in \Gamma.$$

Wir nennen die auf diese Weise insgesamt entstehende Regelmenge P'. Mit P' lassen sich genau die Konfigurationsübergänge von M modellieren.

Allerdings müssen wir uns in der Grammatik aber auch die Eingabe von M "merken" und diese nach Erreichen eines Zielzustands in M erzeugen. Das ist nötig, weil die TM ja die Eingabe möglicher Weise überschreibt. Wir müssen deshalb nicht nur die aktuelle Konfiguration durch ein Wort $a_1...a_{n-1}(z, a_n)...a_m$ über Δ beschreiben, sondern auch, was anfangs auf dem Band gestanden hat. Das macht man, indem man jeweils Paare bestehend aus einem Symbol aus Δ und einem Symbol aus Σ als Variablen der Grammatik verwendet. Die Zeichenkette der Länge m:

$$(a_1,x_1)...(a_{n-1},x_{n-1})((z, a_n), x_n)...(a_m,x_m)$$

beschreibt die aktuelle Konfiguration $a_1...a_{n-1}za_n...a_m$ sowie gleichzeitig, dass das Eingabewort $x_1...x_m$ war (und damit auf dem Band anfangs $x_1...x'_m$ gestanden hat).

G wird nun auf folgende Weise definiert: $G = (V, \Sigma, P, S)$ mit

$V = \{S, A\} \cup (\Delta \times \Sigma)$

$P = \{S \to A(\hat{a},a) \mid a \in \Sigma\} \cup$ (1)
$\{S \to ((z_0 ,\hat{a}),a) \mid a \in \Sigma\} \cup$ (2)
$\{A \to A(a,a) \mid a \in \Sigma\} \cup$ (3)
$\{A \to (z_0 ,a),a) \mid a \in \Sigma\} \cup$ (4)
$\{(\alpha_1, a)(\alpha_2, b) \to \{(\beta_1, a)(\beta_2, b) \mid a, b \in \Sigma, \alpha_1\alpha_2 \to \beta_1\beta_2 \in P'\} \cup$ (5)
$\{((z ,a),b) \to b \mid z \in E, a \in \Gamma, b \in \Sigma\} \cup$ (6)
$\{(a,b) \to b \mid a \in \Gamma, b \in \Sigma\}$ (7)

Wie bereits erläutert werden Konfigurationen und Eingabe durch Paare repräsentiert, wobei die 1. Komponente zur Beschreibung der aktuellen Konfiguration gehört, die 2. Komponente zur Beschreibung der Eingabe.

Zunächst (Phase I) wird mit Regeln aus (1)-(4) eine beliebige Anfangskonfiguration erzeugt:

$$S \Rightarrow^* ((z_0, a_1), a_1)\ (a_2, a_2)\ \ldots\ (a_{n-1}, a_{n-1})\ (\hat{a}_n, a_n)$$

Dann (Phase II) werden mit Regeln aus (5) Konfigurationsübergänge von M modelliert.

Wenn ein Endzustand erreicht wird, so wird (Phase III) durch Entfernen der jeweils 1. Komponente die ursprüngliche Eingabe von M mit der Grammatik generiert.

Beispiel: wir betrachten die TM für (01)*0:

$M = (\{q_0, q_1, q_f\}, \{0, 1, 0', 1'\}, \{0, 1, 0', 1', \square\}, \delta, q_0, \square, \{q_f\})$

mit $\delta(q_0,0) = (q_1,0,R)$
$\delta(q_1,1) = (q_0,1,R)$
$\delta(q_0,0') = (q_f,0',N)$

Wir definieren zunächst auf Basis dieser 3 Regeln P':

$(q_0,0)c \to 0(q_1,c)$ für alle $c \in \Gamma$:

$(q_0,0)0 \to 0(q_1,0)$ $(q_0,0)1 \to 0(q_1,1)$ $(q_0,0)0' \to 0(q_1,0')$ $(q_0,0)1' \to 0(q_1,1')$
$(q_0,0)\square \to 0(q_1,\square)$

$(q_1,1)c \to 1(q_0,c)$ für alle $c \in \Gamma$:

$(q_1,1)0 \to 1(q_0,0)$ $(q_1,1)1 \to 1(q_0,1)$ $(q_1,1)0' \to 1(q_0,0')$ $(q_1,1)1' \to 1(q_0,1')$
$(q_1,1)\square \to 1(q_0,\square)$

sowie:

$(q_0,0') \to (q_f,0')$

G = (V, Σ, P, S) besitzt folgende Regeln P:

S -> A(0',0) S -> A(1',1) (1)
S -> ((q_0 ,0'),0) S -> ((q_0 ,1'),1) (2)
A -> A(0,0) A -> A(1,1) (3)
A -> ((q_0,0),0) A -> (q_0,1),1) (4)

$\{(\alpha_1, a)(\alpha_2, b) \rightarrow \{(\beta_1, a)(\beta_2, b) \mid a, b \in \Sigma, \alpha_1\alpha_2 \rightarrow \beta_1\beta_2 \in P'\}$ (5)

Also für jede der 11 Regeln in P' genau 4 Regeln. Beispielsweise für (q_1,1)0 -> 1(q_0,0):

((q_1,1),0)(0,0) -> (1,0)((q_0,0),0) ((q_1,1),0)(0,1) -> (1,0)((q_0,0),1)
((q_1,1),1)(0,1) -> (1,1)((q_0,0),1) ((q_1,1),1)(0,0) -> (1,1)((q_0,0),0)

((q_f ,1),0) -> 0 ((q_f ,0),0) -> 0 ((q_f , □),0) -> 0 (6)
((q_f ,1'),0) -> 0 ((q_f ,0'),0) -> 0
((q_f ,1),1) -> 1 ((q_f ,0),1) -> 1 ((q_f , □),1) -> 1
((q_f ,1'),1) -> 1 ((q_f ,0'),1) -> 1

(1,1) -> 1 (1',1) -> 1 (0,1) -> 1 (0',1) -> 1 (□,1) -> 1 (7)
(1,0) -> 0 (1',0) -> 0 (0,0) -> 0 (0',0) -> 0 (□,0) -> 0

Beispiel-Übergänge in der TM bei Eingabe 010:

$q_0$010' |-- 0$q_1$10' |-- 01$q_0$0' |-- 01q_f 0'

Entsprechende Ableitung von 010 in G:

S |-- A(0',0) |-- A(1,1)(0',0) |-- ((q_0,0),0) (1,1) (0',0) Phase I
|-- (0,0) ((q_1,1),1) (0',0) |-- (0,0) (1,1) ((q_0, 0'),0) |-- (0,0) (1,1) ((q_f, 0'),0) Phase II
|-- 0 (1,1) ((q_f, 0'),0) |-- 01 ((q_f, 0'),0) |-- 010 Phase III

Bemerkung: aus diesem Beweis lässt sich sofort ein Beweis für Satz 1 herleiten: wir müssen nur die Bedingung der linearen Beschränkung und die entsprechende Bedingung der Grammatikregeln (Typ 1) weglassen: bei allgemeinen Grammatiken muss möglicherweise neuer Platz auf dem Band belegt werden, und es sind entsprechende Verschiebungen des Bandinhaltes notwendig.

Hinweis: Es ist bis heute ein offenes Problem, ob deterministische linear beschränkte Turingmaschinen dieselben Sprachen akzeptieren wie nichtdeterministische oder nur eine echte Teilmenge

Bei beliebigen TMs sind deterministische und nichtdeterministische äquivalent (in dem Sinn, dass sie dieselbe Klasse von Sprachen akzeptieren)!

9. Überblick über die wichtigsten Ergebnisse der Vorlesung

Typ	Grammatik	Automat	Abschl.				
			$\cap$	$\cup$	Kompl.	Produkt	*
3	regulär	DEA, NEA	ja	ja	ja	ja	ja
det.kon.fr.	LR(k)	DPDA	nein	nein	ja	nein	nein
2	kontextfrei	PDA	nein	ja	nein	ja	ja
1	kontextsensitiv	LBA	ja	ja	ja	ja	ja
0	allgemein	TM	ja	ja	nein	ja	ja

Äquivalenz von deterministischen/nichtdeterministischen Automaten:

DEA	NEA	ja
DPDA	PDA	nein
DLBA	LBA	? ungelöst
DTM	TM	ja

Entscheidbarkeit:

	Wortproblem $w \in L$?	Leerheitsproblem $L = \varnothing$?	Äquivalenz $L_1 = L_2$?	Schnittproblem $L_1 \cap L_2 = \varnothing$?
Typ 3	ja (O(n))	ja	ja	ja
det. kontextfrei	ja (O(n))	ja	ja	nein
Typ 2	ja ($O(n^3)$)	ja	nein	nein
Typ 1	ja (exp.)	nein	nein	nein
Typ 0	nein	nein	nein	nein