Języki, automaty i obliczenia

Wykład 12: Modele równoważne maszynom Turinga. Wstęp do złożoności obliczeniowej

Sławomir Lasota

Uniwersytet Warszawski

1 czerwca 2016

Plan

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki
typ 0	rekurencyjnie przeliczalne	maszyny Turinga	?
typ 1	kontekstowe	?	?
typ 2	bezkontekstowe	automaty ze stosem	bezkontekstowe
typ 3	regularne	automaty skończone	regularne

Gramatyka (typu 0)

$$\mathcal{G} \;=\; (A, N, S, \alpha)$$

- A to skończony zbiór symboli końcowych (*terminalnych*)
- N to skończony zbiór symboli niekońcowych (*nieterminalnych*), $A \cap N = \emptyset$
- $S \in N$ to symbol początkowy
- $\alpha \subseteq (A \cup N)^+ \times (A \cup N)^*$ to skończony zbiór reguł przepisywania (*produkcji*)

Notacja

Reguły $(v, v') \in \alpha$ będziemy zapisywać $v \longrightarrow_{\mathcal{G}} v'$ albo $v \longrightarrow v'$.

Reguły rozszerzamy do relacji $\longrightarrow\!\!\!\!\rightarrow_{\mathcal{G}} \subseteq (A \cup N)^* \times (A \cup N)^*$:

$$w \longrightarrow\!\!\!\!\rightarrow_{\mathcal{G}} w' \iff \exists (v \longrightarrow_{\mathcal{G}} v') \in \alpha,\ \exists u, t \in (A \cup N)^*,\quad w \;=\; u\,v\,t,\quad v \;=\; u\,v'\,t$$

i domykamy zwrotno-tranzytywnie: $w \longrightarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow_{\mathcal{G}} w'$ wtw. gdy istnieje ciąg

$$w \;=\; v_0 \longrightarrow\!\!\!\!\rightarrow_{\mathcal{G}} v_1 \longrightarrow\!\!\!\!\rightarrow_{\mathcal{G}} \ldots \longrightarrow\!\!\!\!\rightarrow_{\mathcal{G}} v_n \;=\; w', \qquad n \geq 0$$

zwany *wyprowadzeniem* (wywodem) w' z w (w gramatyce $\mathcal{G}$).

Język generowany przez gramatykę

Język *generowany* przez gramatykę:

$$L(\mathcal{G}) \;=\; \{\, w \in A^* \;:\; S \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} w \,\}$$

Przykład

Gramatyka $\mathcal{G}$:

$$A = \{a, b, c\} \qquad S \longrightarrow \varepsilon \qquad X \longrightarrow aBXc \qquad Ba \longrightarrow aB$$
$$N = \{S, X, B\} \qquad S \longrightarrow X \qquad X \longrightarrow abc \qquad Bb \longrightarrow bb$$

$L(\mathcal{G}) \;=\; ?$

Twierdzenie

Języki częściowo rozstrzygalne to języki generowane przez gramatyki.

Dowód:

- gramatyka $\mathcal{G}$ $\rightsquigarrow$ maszyna Turinga $\mathcal{M}$: maszyna symuluje gramatykę „wstecz"

$$v \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} w \qquad \Longleftrightarrow \qquad q\,w \longrightarrow^*_{\mathcal{M}} q\,v,$$

zaczyna w konfiguracji $q\,w$, akceptuje „w konfiguracji qS". Więc $L(\mathcal{G}) = L(\mathcal{M})$.

- maszyna Turinga z taśmą jednostronnie nieograniczoną $\rightsquigarrow$ gramatyka:

Faza 1: $S \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} \$\,w\,[q_f, a]\,v\,\mathbb{B}$ $\qquad w, v \in (T - \{\mathbb{B}\})^*,\ a \in T - \{\mathbb{B}\}$

Faza 2:

$$\begin{aligned}
b\,[q', c] &\longrightarrow [q, a]\,c && \text{jeśli } (q, a, q', b, \rightarrow) \in \delta \\
[q', c]\,b &\longrightarrow c\,[q, a] && \text{jeśli } (q, a, q', b, \leftarrow) \in \delta \\
b\,[q', \mathbb{B}] &\longrightarrow [q, \mathbb{B}] && \text{jeśli } (q, \mathbb{B}, q', b, \rightarrow) \in \delta \\
[q', c]\,b\,\mathbb{B} &\longrightarrow c\,[q, \mathbb{B}] && \text{jeśli } (q, \mathbb{B}, q', b, \leftarrow) \in \delta \\
\$\,[q_0, a] &\longrightarrow \$\,a \\
a\,\mathbb{B} &\longrightarrow a
\end{aligned}$$

Problem słów

Dane: gramatyk $\mathcal{G}$ i dwa słowa v, w

Wynik: czy $v \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} w$?

Wniosek

Problem słów jest nierozstrzygalny.

Dowód:

Redukujemy problem stopu do problemu słów:

funkcja obliczalna:	maszyna $\mathcal{M}$, słowo w	$\longmapsto$	$\mathcal{G}$ i słowa S, w
poprawność:	$w \in L(\mathcal{M})$	$\Longleftrightarrow$	$S \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} w$

Pytanie

Czy problem słów jest częściowo rozstrzygalny?

Plan

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki
typ 0	rekurencyjnie przeliczalne	maszyny Turinga	?
typ 1	kontekstowe	?	?
typ 2	bezkontekstowe	automaty ze stosem	bezkontekstowe
typ 3	regularne	automaty skończone	regularne

Gramatyka $\mathcal{G} = (A, N, S, \alpha)$ jest *kontekstowa* jeśli produkcje są postaci:

- $S \longrightarrow \varepsilon$
- $uXv \longrightarrow uwv \qquad X \in N, \quad u, v, w \in (A \cup N)^*, \quad w \neq \varepsilon, \quad S \notin w$

Pytanie

Gramatyka $\mathcal{G}$:

$$A = \{a, b, c\} \qquad S \longrightarrow \varepsilon \qquad X \longrightarrow aBXc \qquad Ba \longrightarrow aB$$
$$N = \{S, X, B\} \qquad S \longrightarrow X \qquad X \longrightarrow abc \qquad Bb \longrightarrow bb$$

$$L(\mathcal{G}) = \{ a^n b^n c^n \ : \ n \in \mathbb{N} \}$$

Jak przerobić tę gramatykę na gramatykę kontekstową?

Odpowiedź

Pomysł jest następujący:

$$Ba \longrightarrow B\bar{a} \qquad \bar{B}\bar{a} \longrightarrow \bar{B}B$$
$$B\bar{a} \longrightarrow \bar{B}\bar{a} \qquad \bar{B}B \longrightarrow aB$$

Gramatyka $\mathcal{G} = (A, N, S, \alpha)$ jest *kontekstowa* jeśli produkcje są postaci:

- $S \longrightarrow \varepsilon$
- $u\,X\,v \longrightarrow u\,w\,v \qquad X \in N, \quad u, v, w \in (A \cup N)^*, \quad w \neq \varepsilon, \quad S \notin w$

Gramatyka $\mathcal{G} = (A, N, S, \alpha)$ jest *monotoniczna* jeśli produkcje są postaci:

- $S \longrightarrow \varepsilon$
- $v \longrightarrow w \qquad |v| \leq |w|, \quad S \notin w$

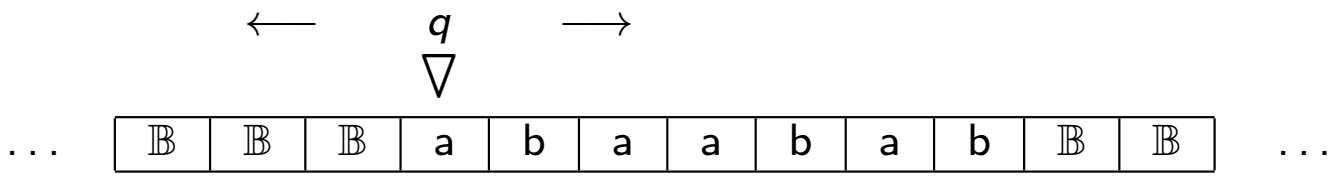
Twierdzenie

Gramatyki kontekstowe i monotoniczne rozpoznają te same języki.

Dowód:

j.w.

Maszyny liniowo ograniczone



Maszyna Turinga jest *liniowo ograniczona* jeśli nigdy nie pisze na pozycji taśmy zajętej przez symbol $\mathbb{B}$. Czyli zabraniamy przejść postaci:

$$(q, \mathbb{B}, q', a, x) \qquad a \neq \mathbb{B}$$

Twierdzenie

Języli kontekstowe to języki rozpoznawane przez niedet. maszyny liniowo ograniczone.

Dowód:

- gramatyka $\mathcal{G}$ $\rightsquigarrow$ maszyna Turinga $\mathcal{M}$: maszyna symuluje gramatykę „wstecz''

$$v \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} w \qquad \Longleftrightarrow \qquad q\,w \longrightarrow^{*}_{\mathcal{M}} q\,v,$$

zaczyna w konfiguracji $q\,w$, akceptuje „w konfiguracji qS''. Więc $L(\mathcal{G}) = L(\mathcal{M})$.

- maszyna Turinga z taśmą jednostronnie nieograniczoną $\rightsquigarrow$ gramatyka:

Faza 1: $S \longrightarrow\!\!\!\!\!\rightarrow_{\mathcal{G}} \$\,w\,[q_f, a]\,v\,\mathbb{B}$ $\qquad w, v \in (T - \{\mathbb{B}\})^*,\ a \in T - \{\mathbb{B}\}$

Faza 2:

$b\,[q', c] \longrightarrow [q, a]\,c$	jeśli $(q, a, q', b, \rightarrow) \in \delta$
$[q', c]\,b \longrightarrow c\,[q, a]$	jeśli $(q, a, q', b, \leftarrow) \in \delta$
$b\,[q', \mathbb{B}] \longrightarrow [q, \mathbb{B}]$	jeśli $(q, \mathbb{B}, q', b, \rightarrow) \in \delta$
$[q', c]\,b\,\mathbb{B} \longrightarrow c\,[q, \mathbb{B}]$	jeśli $(q, \mathbb{B}, q', b, \leftarrow) \in \delta$
$\$\,[q_0, a] \longrightarrow \$\,a$	
$a\,\mathbb{B} \longrightarrow a$	

(Nie)rozstrzygalność gramatyk monotonicznych

Fakt

Problem słów jest rozstrzygalny dla gramatyk monotonicznych.

Więc problem stopu jest rozstrzygalny dla liniowo ograniczonych maszyn Turinga.

Twierdzenie

Problem (nie)pustości jest nierozstrzygalny dla języków kontekstowych.

Dowód:

Dla ustalonej instancji PCP, zbiór rozwiązań jest językiem kontekstowym.

$$\begin{array}{rcl} \text{funkcja obliczalna:} & \text{instancja PCP } x \longmapsto & \mathcal{G}_x \\ \text{poprawność:} & x \text{ ma rozwiązanie} \iff & L(\mathcal{G}_x) \neq \emptyset \end{array}$$

Pytanie

Czy problem (nie)pustości jest częściowo rozstrzygalny?

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki
typ 0	rekurencyjnie przeliczalne	maszyny Turinga	typu 0
typ 1	kontekstowe	maszyny liniowo ogr.	kontekstowe/monotoniczne
typ 2	bezkontekstowe	automaty ze stosem	bezkontekstowe
typ 3	regularne	automaty skończone	regularne

Plan

Automaty z licznikami

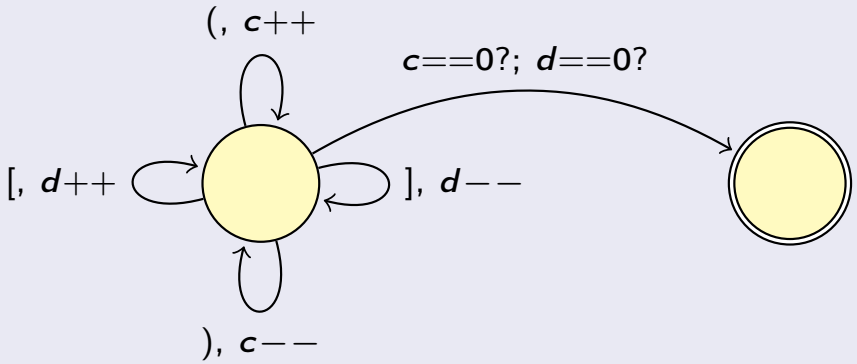
$$\mathcal{A} = (A, Q, I, F, C, \delta)$$

- A – alfabet
- Q – skończony zbiór stanów
- $I \subseteq Q$ – stany początkowe
- $F \subseteq Q$ – stany akceptujące
- C – skończony zbiór liczników
- $\delta \subseteq Q \times (A \cup \{\varepsilon\}) \times \{ c{+}{+}, c{-}{-}, c{=}{=}0? \ : \ c \in C \} \times Q$ – relacja przejścia

Przykład

$A = \{(,), [,]\}$

$C = \{c, d\}$



$L(\mathcal{A}) = ?$

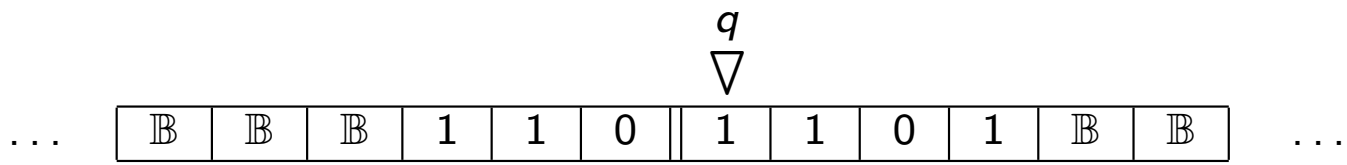
Automaty z licznikami są równoważne maszynom Turinga

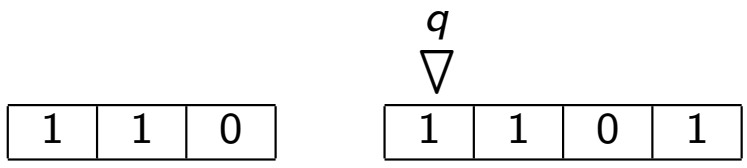
Twierdzenie

Języki częściowo rozstrzygalne to języki rozpoznawane przez automaty z licznikami.

Dowód (idea):

Maszyna Turinga $\rightsquigarrow$ automat z 3 licznikami:





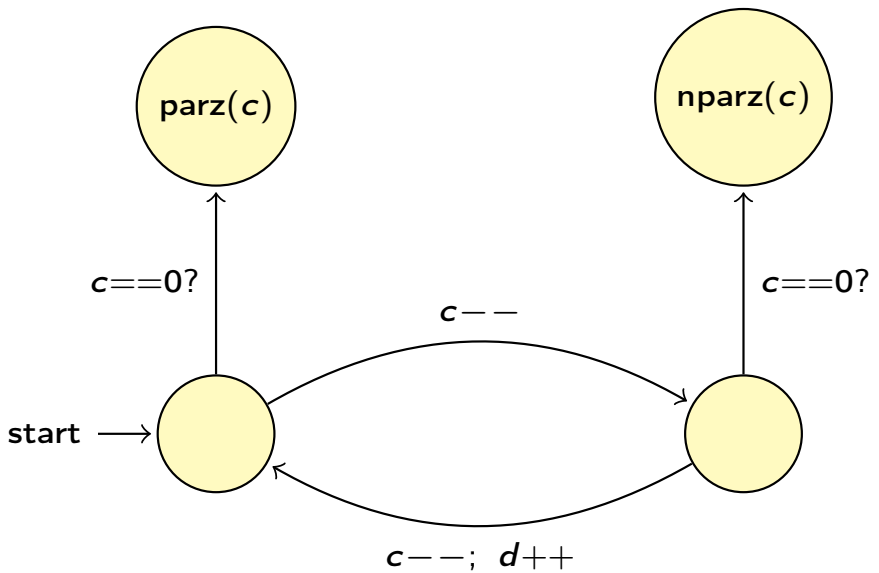
$6 = 4 + 2 + 0$ $\qquad$ $1 + 2 + 0 + 8 = 11$

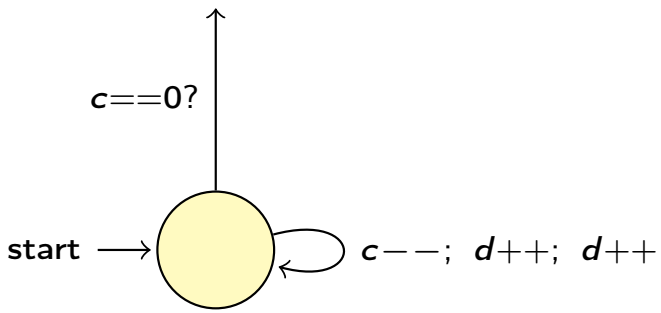
Potrzebne operacje na licznikach:

$\text{parz}(c)?$ $\qquad$ $\text{nparz}(c)?$ $\qquad$ $c = c >> 1$ $\qquad$ $c = c << 1$

Potrzebne operacje na licznikach:

$$\text{parz}(c)? \qquad \text{nparz}(c)? \qquad c = c >> 1 \qquad c = c << 1$$





Automaty z 2 licznikami

Twierdzenie

Automaty z 2 licznikami potrafią symulować automaty z 3 licznikami.

Dowód:

?

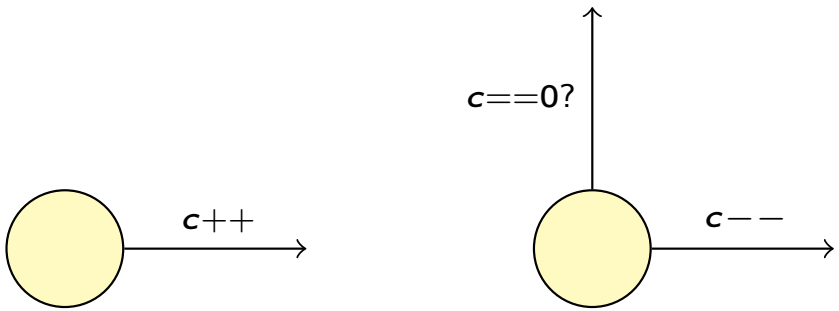
„Najprostszy" problem nierozstrzygalny

Dane: $\mathcal{A}$ – automat deterministyczny z 2 licznikami c_1, c_2

Wynik: czy $\mathcal{A}$ zatrzyma się, jeśli rozpocznie w konfiguracji $(q_0, c_1 = 0, c_2 = 0)$?

Pytanie

Kiedy automat z licznikami jest deterministyczny?



Plan

Czas i pamięć

n – długość słowa wejściowego. Rozważamy funkcje $f : \mathbb{N} \to \mathbb{N}$ t.że $f(n) \geq \log n$.

klasa języków	języki rozpoznawane przez
DTIME($f(n)$)	deterministyczną maszynę w czasie $f(n)$
NTIME($f(n)$)	niedeterministyczną maszynę w czasie $f(n)$
DSPACE($f(n)$)	deterministyczną maszynę w pamięci $f(n)$
NSPACE($f(n)$)	niedeterministyczną maszynę w pamięci $f(n)$

Pytanie

Czy wystarczy ograniczyć się do obliczalnych funkcji f?

Twierdzenie

Wszystkie klasy są właściwymi podklasami języków rozstrzygalnych.

Dowód:

$$L_p = \{ w : w \notin L_{\leq f}(\mathcal{M}_w) \}.$$

Niech $L(\mathcal{M}_w) = L_p$. Jeśli $L_{\leq f}(\mathcal{M}_w) = L_p$, to

$$w \in L_{\leq f}(\mathcal{M}_w) \iff w \notin L_{\leq f}(\mathcal{M}_w).$$

Hierarchia

Twierdzenie

- $DTIME(f(n)) \subseteq NTIME(f(n)) \subseteq DSPACE(f(n)) \subseteq NSPACE(f(n))$
- $NSPACE(f(n)) \subseteq \bigcup_c DTIME(2^{c \cdot f(n)}) = DTIME(2^{\mathcal{O}(f(n))})$ *założenie!*

$$
\begin{aligned}
&\dots \\
\text{EXPSPACE} &= \bigcup_c \text{DSPACE}(2^{n^c}) &&= \bigcup_c \text{NSPACE}(2^{n^c}) \\
\text{NEXPTIME} &= \bigcup_c \text{NTIME}(2^{n^c}) \\
\text{EXPTIME} &= \bigcup_c \text{DTIME}(2^{n^c}) \\
\text{PSPACE} &= \bigcup_c \text{DSPACE}(n^c) &&= \bigcup_c \text{NSPACE}(n^c) \\
NP = \text{NPTIME} &= \bigcup_c \text{NTIME}(n^c) \\
P = \text{PTIME} &= \bigcup_c \text{DTIME}(n^c) \\
NL = \text{NLOGSPACE} &= \text{NSPACE}(\log(n)) \\
L = \text{LOGSPACE} &= \text{DSPACE}(\log(n))
\end{aligned}
$$

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki	klasa złożoności
typ 0	rekurencyjnie przeliczalne	maszyny Turinga	dowolne	
typ 1	kontekstowe	maszyny liniowo ograniczone	kontekstowe/ monotoniczne	NSPACE(n)
typ 2	bezkontekstowe	automaty ze stosem	bezkontekstowe	$\subseteq$ PTIME
typ 3	regularne	automaty skończone	regularne	NSPACE(1)/ DSPACE(1)

W następnym odcinku:

Czas wielomianowy i pamięć wielomianowa:

$$\text{PSPACE} = \bigcup_c \text{DSPACE}(n^c) = \bigcup_c \text{NSPACE}(n^c)$$

$$NP = \text{NPTIME} = \bigcup_c \text{NTIME}(n^c)$$

$$P = \text{PTIME} = \bigcup_c \text{DTIME}(n^c)$$