Języki, automaty i obliczenia

Wykład 13: Klasa PSPACE. Hierarchia Chomsky'ego

Sławomir Lasota

Uniwersytet Warszawski

5 czerwca 2024

Plan

1. PSPACE
2. Hierarchia Chomsky'ego
3. Języki kontekstowe

Klasy złożoności

Klasy złożoności z dokładnością do wielomianu:

$$
\begin{aligned}
& & & \dots & & \\
& & \mathrm{EXPSPACE} &= \bigcup_p \mathrm{DSPACE}(2^{p(n)}) &&= \bigcup_p \mathrm{NSPACE}(2^{p(n)}) \\
& & \mathrm{NEXPTIME} &= \bigcup_p \mathrm{NTIME}(2^{p(n)}) & & \\
& & \mathrm{EXPTIME} &= \bigcup_p \mathrm{DTIME}(2^{p(n)}) & & \\
& & \mathrm{PSPACE} &= \bigcup_p \mathrm{DSPACE}(p(n)) &&= \bigcup_p \mathrm{NSPACE}(p(n)) \\
NP &= & \mathrm{NPTIME} &= \bigcup_p \mathrm{NTIME}(p(n)) & & \\
P &= & \mathrm{PTIME} &= \bigcup_p \mathrm{DTIME}(p(n)) & &
\end{aligned}
$$

Determinizacja klas pamięciowych

Twierdzenie (Savitch 1970)

Jeśli $f(n) \geq n$ *to* $NSPACE(f(n)) \subseteq DSPACE(f(n)^2)$. *założenie!*

Dowód:

Niech $\mathcal{M}$ – maszyna niedet. działająca w pamięci $f(n)$. Zaprojektujemy algorytm deterministyczny sprawdzający czy $c_0 \longrightarrow^*_{\mathcal{M}} c$, dla konfiguracji akceptującej c.

Liczba konfiguracji $\leq 2^m$, $m = \mathcal{O}(f(n))$, więc wystarczy sprawdzić czy $c_0 \longrightarrow^{\leq 2^m}_{\mathcal{M}} c$.

Algorytm

```
bool p(c, c', k) {
    if  (k == 0) { return ((c==c') || c ⟶_M c'); }
    else {
        for each c'' {
            if (p(c, c'', k-1) && p(c'', c', k-1)) return true;
        }
        return false;
    }
}
```

Pamięć: $m \cdot f(n) = \mathcal{O}(f(n)^2)$.

Wniosek

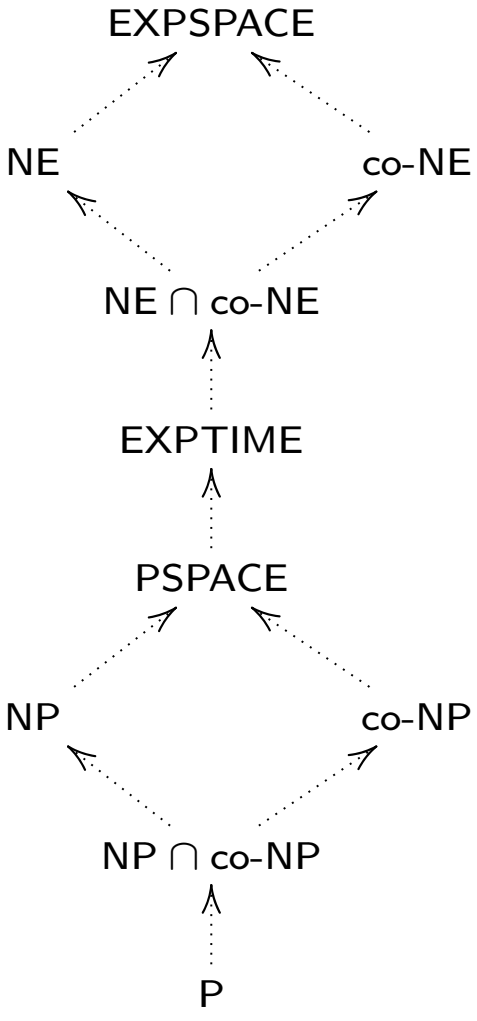
NPSPACE = PSPACE.

Ale nie wiemy, czy NSPACE(n) = DSPACE(n).

co-$\mathcal{C}$ = problemy, których dopełnienie należy do $\mathcal{C}$

Wniosek

co-PSPACE = PSPACE.



Pytania

Wiemy, że

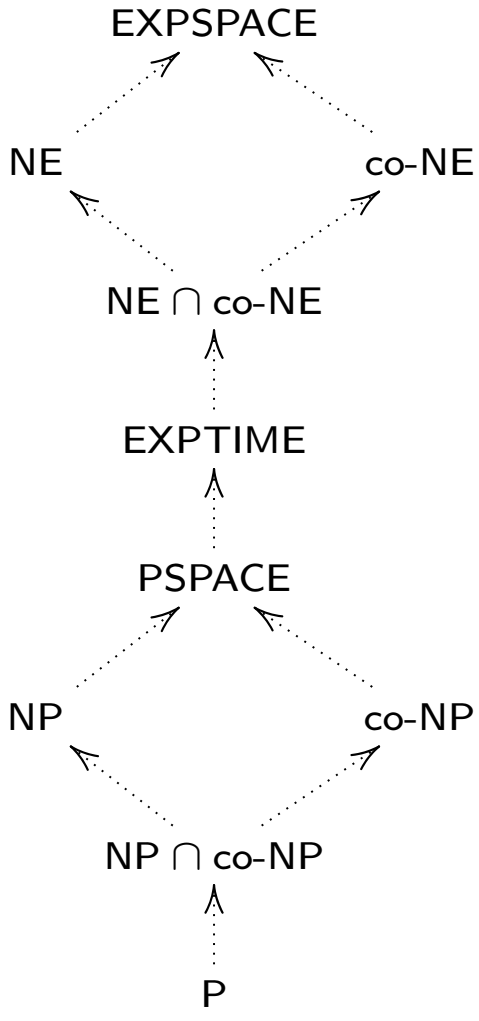
$$P \neq EXPTIME \neq 2\text{-}EXPTIME \neq \dots$$

$$NP \neq NEXPTIME \neq 2\text{-}NEXPTIME \neq \dots$$

$$PSPACE \neq EXPSPACE \neq \dots$$

$P \neq NP$?

$P \neq PSPACE$?



Problemy PSPACE-zupełne

Problem spełnialności kwantyfikowanej formuły zdaniowej (QBF)

Dane: formuła postaci $\forall x_1 \exists x_2 \forall x_3 \ldots \phi$.

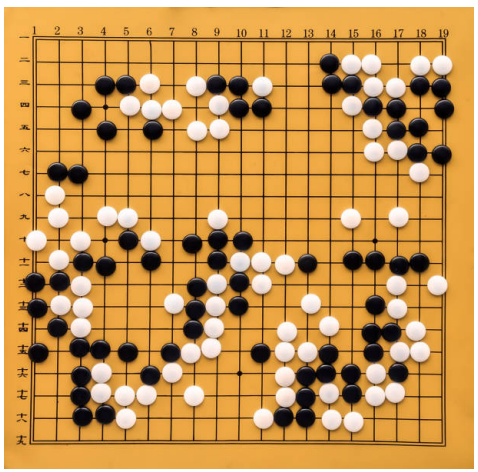
Wynik: czy formuła jest spełnialna (tautologią)?

np. $\forall x \exists y.\ (x \wedge y) \vee (x \wedge (\neg y \vee z))$

Kto wygrywa w Go?

Dane: pozycja w grze Go na dowolnie dużej planszy.

Wynik: czy białe wygrywają?



Problemy PSPACE-zupełne (dla języków regularnych)

- Uniwersalność wyrażenia regularnego (albo automatu).
- Równość dwóch wyrażeń regularnych (albo automatów).
- Niepustość przecięcia wyrażeń regularnych (albo automatów).
- ...

Plan

1. PSPACE
2. Hierarchia Chomsky'ego
3. Języki kontekstowe

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki
typ 0	częściowo rozstrzygalne	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: $w \longrightarrow\!\!\!\!\rightarrow^*_{\mathcal{G}} w'$ 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

Gramatyki $\equiv$ *maszyny Turinga.*

Dowód:

- gramatyka $\mathcal{G}$ $\rightsquigarrow$ maszyna $\mathcal{M}$: 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 $\mathcal{M}$ nad $\{0,1\}$ z taśmą jednostronnie nieograniczoną $\rightsquigarrow$ gramatyka $\mathcal{G}$:

$$A = \{0,1\} \qquad N = Q \times \{0,1,\mathbb{B}\} \cup \{\$, \mathbb{B}, S\}$$

Faza 1: $S \longrightarrow\!\!\!\!\!\rightarrow^*_{\mathcal{G}} \$[q_{\mathsf{tak}}, \mathbb{B}]\,\mathbb{B}^+$

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

poprawność: $w \in L(\mathcal{M}) \qquad \Longleftrightarrow \qquad S \longrightarrow\!\!\!\!\!\rightarrow^*_{\mathcal{G}} w$

Problem słów

Dane:	gramatyka $\mathcal{G}$ 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

1. PSPACE
2. Hierarchia Chomsky'ego
3. Języki kontekstowe

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki
typ 0	częściowo rozstrzygalne	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:

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

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:

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

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

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

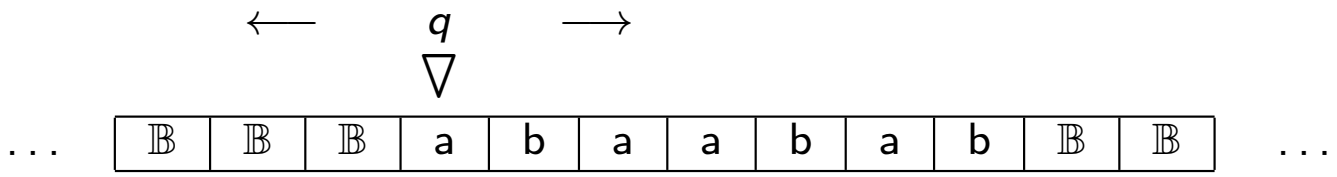
Twierdzenie

Gramatyki kontekstowe $\equiv$ *gramatyki monotoniczne.*

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:

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

(Automat dwukierunkowy, który może pisać po taśmie.)

Twierdzenie

Gramatyki monotoniczne $\equiv$ *niedeterministyczne maszyny liniowo ograniczone.*

(Nie)rozstrzygalność gramatyk kontekstowych

Fakt

Problem słów jest rozstrzygalny dla gramatyk kontekstowych.

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.

funkcja obliczalna:	instancja PCP x	$\longmapsto$	$\mathcal{G}_x$
poprawność:	x ma rozwiązanie	$\Longleftrightarrow$	$L(\mathcal{G}_x) \neq \emptyset$

Pytanie

Czy problem (nie)pustości jest częściowo rozstrzygalny?

Hierarchia Chomsky'ego

typ	języki	automaty	gramatyki	klasa złożoności
typ 0	częściowo rozstrzygalne	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)

Problemy PSPACE-zupełne:

- Problem stopu dla maszyn liniowo ograniczonych.
- Problem słów dla gramatyk kontekstowych.

W następnym odcinku:

?