Obliczalność i złożoność

- czy są problemy, które nie można rozwiązać przy pomocy komputera,
tzn. nie istnieje program (w jakim języku programowania?) rozwiązując ten problem

↳ potrzebny: "Definicja" jakie problemy można rozwiązać na komputerze

↳ Maszyna Turinga
  - bardzo prosta maszyna
  - pracuje na słowach

↳ teza Churcha-Turinga

- Języki formalne
opisują n.p. poprawne (pod względem składni) programy
  • Generowanie wszystkich słów języ
  → Gramatyka
  • Sprawdzanie czy słowa należy do języka → Automat

↳ Hierarchia Chomsky'ego

- Złożoność

Jak szybko można rozwiązać problem?

- niezależnie od maszyny

↳ liczymy (oszacujemy) ilość operacji, któr
wykonuje maszyna Turinga rozwiązująca
problem

Jakie jest najszybsze rozwiązanie?

↳ problemy mają różne poziomy trudnośc

→ klasy P, NP, ...

§1 Wstęp

1.1 Słowa

Def

Alfabet Σ, to skończone nie pusty zbió[r]
Elementy $a \in \Sigma$ nazywają się litery.
Słowo nad Σ, to skończony ciąg liter.
Σ^* oznacza zbiór wszystkich słów nad Σ
λ oznacza puste słowo.

Przykłady:

(i) $\Sigma_1 = \{0, 1\}$
(ii) $\Sigma_2 = \{a, b, c, \ldots\ldots x, y, z\}$
(iii) $\Sigma_3 = \{$ znaki z klawiatury $\}$
(iv) $\Sigma_4 = \Sigma_1 \cup \Sigma_2 \cup \{$ while, if, then, else, ... $\}$

Def

(i) Dla $u, v \in \Sigma^*$ konkatenacja $u \circ v$ dołączy
u i v:

$$(a_1 \ldots a_n) \circ (b_1 \ldots b_m) := a_1 \ldots a_n b_1 \ldots b_m$$

(ii) $|u|$ oznacza długość słowa $u \in \Sigma^*$:

$$|a_1 \ldots a_n| := n$$

Uwaga:

(i) $u \circ (v \circ w) = (u \circ v) \circ w$

(ii) $\lambda \circ u = u = u \circ \lambda$

(ii) $|w \circ v| = |w| + |v|$

Def:

Dla $u, v \in \Sigma^*$ v jest podsłowem u, jeżeli istnie
$x, y \in \Sigma^*$, takie że $u = xvy$.
Jeżeli $x = \lambda$, to v jest prefiksem u.
Jeżeli $y = \lambda$, to v jest sufiksem u.

Przykład:

słowo $u = abbc$ ma 5 prefiksów:

$$\lambda, a, ab, abb, abbc$$

bbc jest sufiksem u (ale nie jest prefiksem)
bb jest podsłowem u.

Indukcja dla słów

Niech $L \subseteq \Sigma^*$.

Z (i) $\lambda \in L$

(ii) $wa \in L$ dla wszystkich $w \in L$ i $a \in \Sigma$

wynika

$$L = \Sigma^*$$

Przykład: Lustrzane słowo

Def $\varrho : \Sigma^* \longrightarrow \Sigma^*$

$\varrho(\lambda) := \lambda$

$\varrho(wa) := a \circ \varrho(w)$

Uwaga: $\varrho(a_1 \ldots a_n) = a_n \ldots a_1$ (dowód?)

Stwierdzenie:

$$\varrho(v \circ w) = \varrho(w) \circ \varrho(v)$$

(i) $w = \lambda$:

$\varrho(v \circ \lambda) = \varrho(v) = \lambda \circ \varrho(v) = \varrho(\lambda) \circ \varrho(v)$

(ii) $w = ua$

Założenie indukcyjne: $\varrho(v \circ u) = \varrho(u) \circ \varrho(v)$

$$\begin{aligned}
\varrho(v \circ (u \circ a)) &= \varrho((v \circ u) \circ a) \\
&= a \circ \varrho(v \circ u) \\
&\overset{z.i.}{=} a \circ (\varrho(u) \circ \varrho(v)) \\
&= (a \circ \varrho(u)) \circ \varrho(v) \\
&= \varrho(ua) \circ \varrho(v)
\end{aligned}$$

1.2 Słowa i liczby naturalne

Liczby są słowami nad $\Sigma = \{0, 1, \dots 9\}$

(lub $\Sigma = \{0, 1$

w ten sposób słowa nad dowolnym alfabetem dają się zidentyfikować z liczbami naturalnymi

do tego: porządek słów

Def $u, v \in \Sigma^*$

- $u <_{lex} v$ wtw. u jest prefiksem v lub istnieją $x, y, y' \in \Sigma^*$, $a, b \in \Sigma$ takie że

$$u = xay \wedge v = xby' \wedge \underline{a < b}$$

- $a < b$ dla $\Sigma = \{a_1, \dots a_n\}$ oznacza że $a = a_i$, $b = a_j$ dla $1 \leq i < j \leq n$

łatwiej: $\Sigma = \{1, \dots n\} =: \Sigma_n$

niestety: $1 < 11 < 111 < \dots\dots < 2$

Def $u, v \in \Sigma_r^*$

$u < v$ wtw.

$|u| < |v|$ lub

$|u| = |v| \wedge u <_{lex} v$

λ	a	b	aa	ab	ba	bb	aaa	aab	...
0	1	2	3	4	5	6	7	8	....

Twierdzenie

$\nu : \Sigma_r^* \longrightarrow \mathbb{N}$

$\nu(\lambda) := 0$

$\nu(wi) := \nu(w) \cdot r + i$

(i) $\nu(a_1 \ldots a_n) = \sum_{k=0}^{n} a_k \cdot r^k$

(ii) ν jest bijekcją

(iii) $u \leq v$ wtw. $\nu(u) \leq \nu(v)$ dla $u, v \in \Sigma^*$

1.3 Języki formalne

Def

Język formalny nad alfabetem Σ jest zbiorem $L \subseteq \Sigma^*$

Przykłady

(i) $\Sigma = \{a, b\}$

$L_1 = \{a, aa, aaa\}$

$L_2 = \{\lambda, a, b, aba\}$

$L_3 = \{w \in \Sigma^* \mid |w| \text{ parzysta}\}$

(ii) $\Sigma = \{0, 1\}$

$L = \{ w \in \Sigma^* \mid v(w) \text{ parzysta} \}$

(iii) $\Sigma = \{ \text{znaki z klawiatury} \}$

$L = \{ w \in \Sigma^* \mid w \text{ jest poprawnym programem C}$

Języki są zbiorami. Dlatego dla L_1 i L_2

$$L_1 \cup L_2, \quad L_1 \cap L_2, \quad L_1 \setminus L_2$$

są zdefiniowane.

<u>Def</u> L_1, L_2 języki nad Σ

$$L_1 \cdot L_2 := \{ u \circ v \mid u \in L_1, u \in L_2 \}$$

<u>Przykłady:</u>

$$L_1 = \{a, aa\}, \quad L_2 = \{b, ab\}$$

(i) $L_1 \cdot L_2 = \{ab, aab, aaab\}$

(ii) $L_2 \cdot L_1 = \{ba, baa, aba, abaa\}$

(iii) $L_1 \cdot L_1 = \{aa, aaa, aaaa\}$

<u>Def</u> L język nad Σ

$$L^0 := \{\lambda\}$$

$$L^{n+1} := L^n \cdot L$$

$$L^* := \bigcup_{n \geq 0} L^n$$

$$L^+ := L^* \setminus \{\lambda\}$$

Przykłady

(i) $L = \{a, b\}$

$L^2 = \{aa, ab, ba, bb\}$

$L^i = \{w \in \{a,b\}^* \mid |w| = i\}$

(ii) $L = \{a, b\}$

$L^* = \{a, b\}^*$

(iii) $L = \{aa, ab, ba, bb\}$

$L^* = \{w \in \{a,b\}^* \mid |w| \text{ parzysta}\}$

1.4 Diagonalizacja

"Niech $\mathcal{F} = \{f_i \mid i \in \mathbb{N}\}$ będzie zbiorem funk

$f_i : \mathbb{N} \to \mathbb{N}$.

Niech $f : \mathbb{N} \to \mathbb{N}$ będzie funkcją, taka że

$f(i) \neq f_i(i)$ dla wszystkich i.

Wtedy $f \notin \mathcal{F}$."

	0	1	2	3	4	....
f_0	$f_0(0)$					
f_1		$f_1(1)$				
f_2			$f_2(2)$			
f_3				$f_3(3)$		
$\vdots$					$f_4(4)$	

Przykład

Nie istnieje funkcja $\varphi : \mathbb{N}^2 \to \mathbb{N}$, taka że dla
każdej funkcji $f : \mathbb{N} \to \mathbb{N}$ istnieje $i \in \mathbb{N}$, taka
że $\forall x \in \mathbb{N} : f(x) = \varphi(i,x)$.

Założenie: φ istnieje.

Niech $f(x) := \varphi(x,x) + 1$.

wtedy $f : \mathbb{N} \to \mathbb{N}$

czyli istnieje $k \in \mathbb{N}$, taka że $f(x) = \varphi(k$

wtedy $\varphi(k,k) = f(k) = \varphi(k,k) + 1$ $\lightning$

Stwierdzenie

Niech M będzie zbiorem wszystkich funk
$f : \mathbb{N} \to \{0,1\}$.

Nie istnieje bijekcja $b : \mathbb{N} \to M$.

Założenie: b istnieje

Niech $f(x) := \begin{cases} 0; & b(x)(x) = 1 \\ 1; & b(x)(x) = 0 \end{cases}$

wtedy $f : \mathbb{N} \to \{0,1\}$, tzn. $f \in M$

czyli istnieje $k \in \mathbb{N} : b(k) = f$

wtedy

$$b(k)(k) = f(k) = \begin{cases} 0; & b(k)(k) = 1 \\ 1; & b(k)(k) = 0 \end{cases} \quad \lightning$$

Wynik

Dla każdego języka programowania P istnieje funkcja, której nie można zreali-zować jako program.

bo każdy program języka P jest słowem nad pewnym alfabecie, czyli istnieje bijek[cja] między $\mathbb{N}$ i programami języka P.

Ale nie istnieje bijekcja między $\mathbb{N}$ i wszys[t]kimi funkcjami.