12

Problemy NP-zupełne

W czasie poprzedniego wykładu rozważaliśmy zależności między klasami złożoności obliczeniowej. Szczególną uwagę zwróciliśmy na dwie klasy złożoności czasowej: klasę P składającą się z problemów rozstrzygalnych w czasie wielomianowym na *deterministycznych* maszynach Turinga, oraz klasę NP składającą się z problemów rozstrzygalnych w czasie wielomianowym na *niedeterministycznych* maszynach Turinga. Klasę NP możemy również określić jako klasę problemów posiadających własność wielomianowej weryfikacji. Oznacza to, że poprawność rozwiązania rozważanego problemu może zostać zweryfikowana w czasie wielomianowym na maszynie deterministycznej.

W klasie NP wyróżniliśmy grupę problemów, których czasowa złożoność obliczeniowa jest ściśle związana ze złożonością obliczeniową całej klasy – tzw. problemy NP-zupełne. Przypomnijmy, problem nazywamy NP-zupełnym jeśli jest w klasie NP oraz każdy problem z klasy NP może być do niego zredukowany w czasie wielomianowym. W czasie bieżącego wykładu poznamy dwa problemy, które będą stanowiły podstawę dowodzenia NP-zupełności innych problemów.

Spełnialność formuł logicznych – SAT

Formułą logiczną nazywamy wyrażenie składające się ze zmiennych oraz operacji logicznych: negacji, koniunkcji oraz alternatywy. Powiemy, że formuła logiczna ϕ jest *spełnialna* jeśli istnieje takie wartościowanie zawartych w niej zmiennych, dla którego cała formuła jest prawdziwa. Przykładowo, formuła

$$\phi \;=\; (\neg x \vee y) \wedge (z \vee \neg y) \wedge (\neg z \vee \neg y)$$

jest prawdziwa dla wartościowania ($x = false$, $y = false$, $z = true$), a więc jest spełnialna.

Pierwszym przykładem problemu NP-zupełnego jest *problem spełnialności* polegający na sprawdzeniu czy dana formuła logiczna jest spełnialna.

$$SAT = \{\phi : \phi \text{ jest spełnialna}\}.$$

Zauważmy, że maszyna deterministyczna może rozwiązać ten problem w czasie wykładniczym testując wszystkie możliwe wartościowania zmiennych. Maszyna niedeterministyczna może natomiast rozwiązać go w czasie wielomianowym wybierając od razu właściwe wartościowanie. Pozostało udowodnić, że każdy problem z klasy NP może być w czasie wielomianowym zredukowany do problemu SAT.

Twierdzenie 12.1 (Cooke-Levin)

Problem SAT jest NP-zupełny. (SAT $\in$ P wtedy i tylko wtedy, gdy P=NP).

Dowód

(1) SAT $\in$ NP.

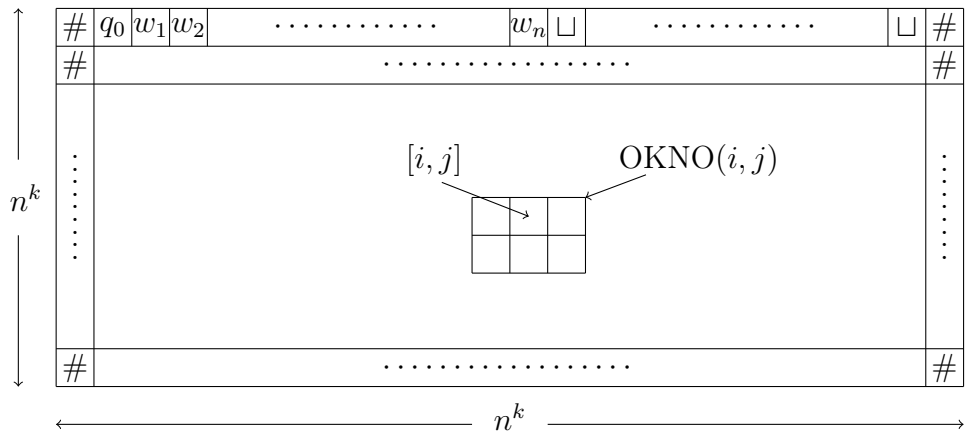
Niedeterministyczna maszyna Turinga może w czasie wielomianowym wybrać wartościowanie zmiennych formuły ϕ i zaakceptować jeśli dla tego wartościowania formuła jest prawdziwa. Równoważnie, dla znanego wartościowania maszyna deterministyczna może w czasie wielomianowym zweryfikować prawdziwość ϕ.

(2) SAT jest NP-trudny.

Musimy udowodnić, że dowolny język $A \in NP$ jest redukowalny do problemu SAT w czasie wielomianowym. Niech M_A będzie niedeterministyczną maszyną Turinga rozstrzygającą język A w czasie n^k dla pewnej stałej $k \in \mathbb{N}$. Dla maszyny M_A oraz słowa $w \in \Sigma^*$ skonstruujemy formułę $\phi_{M_A}(w)$ spełnialną wtedy i tylko wtedy, gdy maszyna M_A akceptuje w.

Tableau maszyny Turinga

Tableau dla maszyny M_A oraz słowa $w = w_1 w_2 \dots w_n$ nazywamy tablicę rozmiaru $n^k \times n^k$, w której wierszach zapisane są konfiguracje ścieżki obliczeń maszyny M_A dla słowa w. Dla uproszczenia zakładamy, że wszystkie konfiguracje M_A mają tę sama długość (n^k) oraz rozpoczynają i kończą się znakiem specjalnym #.
Pierwszy wiersz tableau zawiera konfiguracje początkową maszyny M_A, natomiast każdy kolejny wynika z poprzedniego zgodnie z jej funkcją przejścia.



Rysunek 12.1: Tableau dla maszyny Turinga oraz słowa $w = w_1 w_2 \dots w_n$.

Powiemy, że tableau jest akceptujące jeśli zawiera wiersz odpowiadający konfiguracji akceptującej maszyny M_A. Zauważmy, że każde akceptujące tableau maszyny M_A dla słowa w odpowiada akceptującej ścieżce obliczeń maszyny M_A na w. Zatem maszyna M_A akceptuje słowo w dokładnie wtedy, gdy istnieje akceptujące tableau maszyny M_A dla w.

Redukcja języka A ∈ NP do problemu SAT

Niech $C = Q \cup \Gamma \cup \{\#\}$, gdzie Q jest zbiorem stanów zaś Γ alfabetem taśmy maszyny M_A. Dla każdego $s \in C$ oraz $1 \leqslant i, j \leqslant n^k$ tworzymy zmienną $x_{i,j,s}$. Zmienna $x_{i,j,s}$ ma wartość *true* jeśli w komórce tableau o współrzędnych $[i, j]$ znajduje się znak s. Formuła $\phi_{M_A}(w)$ dla maszyny M_A oraz słowa w ma postać

$$\phi_{M_A}(w) \;=\; \phi_{cell}(w) \wedge \phi_{start}(w) \wedge \phi_{akceptuj}(w) \wedge \phi_{ruch}(w).$$

Poszczególne składniki formuły $\phi_{M_A}(w)$ opiszemy szczegółowo poniżej.

(a) Formuła ϕ_{cell} opisuje zawartość komórek tableau zapewniając, że w każdej z nich może znajdować się dokładnie jeden symbol:

$$\phi_{cell}(w) \;=\; \bigwedge_{1 \leqslant i,j \leqslant n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}) \right) \right]$$

Formuła ϕ_{cell} składa się z koniunkcji formuł opisujących zawartość każdej komórki $[i, j]$ tableau dla $1 \leqslant i, j \leqslant n^k$. Pierwsza część formuły zapewnia, że w każdej komórce znajduje się co najmniej jeden znak. Druga część

zapewnia, że znajduje się w niej co najwyżej jeden znak. Zatem w dowolnym wartościowaniu spełniającym formułę $\phi_{M_A}(w)$ dokładnie jedna zmienna związaną z każdą komórką musi mieć wartość *true*.

(b) Formuła $\phi_{start}(w)$ gwarantuje, że pierwszy wiersz tableau jest konfiguracją początkową maszyny M_A dla słowa w:

$$\begin{aligned}\phi_{start}(w) \;=\; & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \\ & \ldots \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}.\end{aligned}$$

(c) Formuła $\phi_{akceptuj}(w)$ gwarantuje, że w tableau występuje akceptująca konfiguracja maszyny M_A uruchomionej na słowie w:

$$\phi_{akceptuj}(w) \;=\; \bigvee_{1 \leqslant i,j \leqslant n^k} x_{i,j,q_{ACC}}.$$

(d) Formuła $\phi_{ruch}(w)$ gwarantuje, że każdy wiersz tableau jest opisem konfiguracji wynikającej bezpośrednio z konfiguracji opisanej w wierszu poprzedzającym zgodnie z funkcją przejścia maszyny M_A. Przypomnijmy, że konfigurację maszyny M_A możemy opisać za pomocą słowa uq_iv oznaczającego, że M_A jest w stanie q_i, jej taśma zawiera słowo uv, zaś jej głowica znajduje się nad pierwszym znakiem słowa v.

Niech $OKNO(i,j)$ oznacza blok w tableau rozmiaru 2×3:

$$OKNO(i,j) = \Big\{[i,j-1],\ [i,j],\ [i,j+1],\ [i+1,j-1],\ [i+1,j],\ [i+1,j+1]\Big\}.$$

Powiemy, że $OKNO(i,j)$ jest *poprawne* jeśli jego zawartość nie narusza zasady przejścia między kolejnymi konfiguracjami maszyny M_A zgodnie z jej funkcją przejścia δ.

Przykład 12.1

Przypuśćmy, że $\delta(q_1,a) = \{(q_1,b,R)\}$ oraz $\delta(q_1,b) = \{(q_2,c,L),(q_2,a,R)\}$. Przykładami poprawnych okien są wówczas:

a	q_1	b
q_2	a	c

(i)

a	q_1	b
a	a	q_2

(ii)

$\#$	b	a
$\#$	b	a

(iii)

a	a	q_1
a	a	b

(iv)

a	b	a
a	b	q_2

(v)

b	b	b
c	b	b

(vi)

Okna (i) oraz (ii) są w oczywisty sposób zgodne z funkcją przejścia δ, zaś okno (iii) odpowiada brakowi zmian w komórkach nie znajdujących się w bezpośrednim sąsiedztwie głowicy. Okna (iv), (v) oraz (vi) odpowiadają natomiast zmianom zawartości w bezpośrednim sąsiedztwie głowicy i są poprawne pod warunkiem, że odpowiednie znaki znajdują się we właściwych miejscach po ich prawej lub lewej stronie. Jeśli odpowiadające im wiersze tableau nie opisują kolejnych konfiguracji maszyny M_A otrzymanych zgodnie z jej funkcją przejścia, będzie to wykryte w oknach bezpośrednio sąsiadujących z oknami (iv), (v) oraz (vi).

Przykładami niepoprawnych okien mogą być natomiast:

a	b	a
a	a	a

(vii)

a	q_1	b
q_1	a	a

(viii)

b	q_1	b
q_2	b	q_2

(ix)

Okno (vii) jest niepoprawne, ponieważ zmiana zawartości może nastąpić wyłącznie w komórce znajdującej się bezpośrednio pod głowicą. Okno (viii) jest niepoprawne, ponieważ jego zawartość jest niezgodna z funkcją przejścia δ. Okno (ix) jest niepoprawne, ponieważ jego dolny wiersz zawiera dwa symbole stanów kodujące dwie pozycje jednej głowicy.

Podstawą konstrukcji formuły $\phi_{ruch}(w)$ będzie następujące stwierdzenie.

Stwierdzenie 12.2

Jeśli pierwszy wiersz tableau zawiera konfigurację początkową maszyny M_A oraz każde OKNO w tableau jest poprawne, to każdy wiersz w tableau jest konfiguracją M_A poprawnie wynikającą z konfiguracji znajdującej się w wierszu poprzedzającym.

Aby uzasadnić prawdziwość powyższego stwierdzenia rozważmy konfiguracje maszyny M_A opisane przez parę sąsiadujących wierszy tableau. Górny i dolny wiersz każdego okna nie znajdującego się w bezpośrednim sąsiedztwie symbolu stanu (kodującego położenie głowicy maszyny M_A) powinny być identyczne. Zawartość każdego okna sąsiadującego z symbolem stanu maszyny powinna być natomiast zgodna z jej funkcją przejścia. Spełnienie powyższych warunków gwarantuje, że konfiguracja opisana w dolnym wierszu rozważanej pary wynika bezpośrednio z konfiguracji opisanej w górnym wierszu zgodnie z funkcją przejścia maszyny M_A.

Formułę $\phi_{ruch}(w)$ możemy zatem zapisać jako:

$$\phi_{ruch}(w) \;=\; \bigwedge_{1\leqslant i<n^k,\ 1<j<n^k} ,, OKNO(i,j) \text{ jest poprawne}'',$$

gdzie dla symboli a_1, a_2, a_3, a_4, a_5, a_6 formuła „$OKNO(i,j)$ jest poprawne" jest postaci

$$\phi_O \;=\; x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}.$$

Złożoność obliczeniowa redukcji

Ostatnim etapem dowodu NP-zupełności problemu SAT będzie oszacowanie złożoności czasowej skonstruowanej powyżej redukcji. Zgodnie z założeniem maszyna M_A rozstrzyga język w czasie n^k

- Tableau zawiera n^{2k} komórek, dla każdej z nich tworzymy $|C|$ zmiennych, zatem formuła ϕ_{M_A} zawiera $O(n^{2k})$ zmiennych.
- Formuła ϕ_{cell} zawiera element stałego rozmiaru dla każdej komórki tableau, ma więc rozmiar $O(n^{2k})$.
- Formuła ϕ_{start} zawiera element stałego rozmiaru dla każdej komórki w pierwszym wierszu, ma więc rozmiar $O(n^k)$.
- Formuła $\phi_{akceptuj}$ zawiera element stałego rozmiaru dla każdej komórki tableau, ma więc rozmiar $O(n^{2k})$.
- Formuła ϕ_{ruch} zawiera element stałego rozmiaru dla każdej komórki tableau, ma więc rozmiar $O(n^{2k})$.

Podsumowując, cała formuła ϕ_{M_A} ma rozmiar $O(n^{2k})$. Możliwe jest zatem wygenerowanie jej w czasie wielomianowym. ■

Problem spełnialności 3SAT

Problem spełnialności 3SAT, jest modyfikacją problemu SAT, w której postać formuły logicznej jest dość mocno ograniczona. Jednak mimo tych ograniczeń problem 3SAT również okazuje się być NP-zupełny.

Literałem nazywamy zmienną logiczną lub jej zaprzeczenie, np. x lub $\neg y$. *Klauzulą* nazywamy kilka literałów połączonych spójnikiem logicznym $\vee$, np. $(x \vee \neg y \vee z \vee \neg x)$. Powiemy, że formuła logiczna ϕ jest w *koniunktywnej postaci normalnej* (CNF), jeśli składa się z ciągu klauzul połączonych spójnikiem logicznym $\wedge$, np.

$$\phi \;=\; (x \vee \neg z) \wedge (x \vee y \vee \neg z \vee \neg y) \wedge y.$$

Powiemy, że formuła logiczna ϕ jest w postaci 3CNF, jeśli każda klauzula zawiera dokładnie 3 literały.

$$3SAT \;=\; \{\phi :\; \phi \text{ jest spełnialną formułą w postaci 3CNF}\}.$$

Twierdzenie 12.3

Problem 3SAT jest NP-zupełny.

Dowód

Zauważmy, że niedeterministyczna maszyna Turinga może wybrać wartościowanie zmiennych i rozstrzygnąć prawdziwość rozważanej formuły w czasie wielomianowym. Zatem 3SAT $\in$ NP.

Musimy pokazać, że dowolny język $A \in NP$ jest redukowalny w czasie wielomianowym do problemu 3SAT. W tym celu zmodyfikujemy nieco dowód NP-zupełności problemu SAT. Niech M_A będzie maszyną Turinga rozstrzygającą język $A \in NP$, $w \in \Sigma^*$, zaś

$$\phi_{M_A}(w) \;=\; \phi_{cell}(w) \wedge \phi_{start}(w) \wedge \phi_{akceptuj}(w) \wedge \phi_{ruch}(w)$$

będzie formułą skonstruowaną w dowodzie Twierdzenia 12.1. Zauważmy, że:

- Formuły ϕ_{cell} oraz ϕ_{start} są w koniunktywnej postaci normalnej (CNF)
- Formuła $\phi_{akceptuj}$ jest pojedynczą klauzulą (czyli jest w postaci CNF)
- Formuła ϕ_{ruch} jest alternatywą koniunkcji i może być przekształcona w koniunkcję alternatyw (postać CNF) za pomocą praw de Morgana.

Zatem, formuła $\phi_{M_A}(w)$ może być przekształcona do koniunktywnej postaci normalnej. Ponadto, dowolna formuła w postaci CNF może być zamieniona na równoważną jej formułę w postaci 3CNF za pomocą poniższej procedury.

Procedura zamiany formuły w postaci CNF na 3CNF

- Klauzule zawierające mniej niż 3 zmienne rozszerzamy powtarzając jedno- lub dwukrotnie wystąpienie dowolnej z nich.
- Klauzule zawierające więcej niż 3 zmienne dzielimy na klauzule rozmiaru 3 dodając nowe zmienne nie zmieniając jej spełnialności. Formalnie, klauzulę postaci

  $$(a_1 \vee a_2 \vee \ldots \vee a_m)$$

  zamienimy na ciąg klauzul

  $$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee z_3) \wedge \ldots \wedge (\neg z_{m-3} \vee a_{m-1} \vee a_m),$$

  gdzie $z_1, \ldots, z_{m-3}$ nie występują w rozważanej formule.

Podsumowując, dla maszyny Turinga M_A oraz słowa w konstruujemy formułę logiczną $\phi_{M_A}(w)$, która jest prawdziwa wtedy i tylko wtedy, gdy maszyna M_A akceptuje słowo w. Przekształcamy $\phi_{M_A}(w)$ w równoważną jej formułę ϕ'_{M_A} w postaci CNF modyfikując składnik ϕ_{ruch} zgodnie z prawami de Morgana, a następnie w formułę ϕ''_{M_A} w postaci 3CNF za pomocą opisanej powyższej procedury. Ponieważ rozważane przekształcenia mogą być wykonane w czasie wielomianowym, otrzymujemy wielomianową redukcję języka A do problemu spełnialności 3SAT, co kończy dowód. ■

Uwaga

Jeżeli rozmiar klauzul formuły logicznej w koniunktywnej postaci normalnej ograniczymy do dwóch zmiennych, otrzymamy problem 2SAT posiadający rozwiązanie w czasie wielomianowym.