# Úkol č. 3 do předmětu Složitost

Vojtěch Havlena (xhavle03)

1. **Příklad**

Nejprve důkaz, že $MEM_1 \in PSPACE$. Sestrojíme NTS $M$, který má na vstupu $(M_L, w)$ a simuluje $M_L$ na vstupu $w$. Vzhledem k tomu, že LOA je speciální případ nedeterministického TS, platí že $L(M) = MEM_1 \in NSPACE(n)$, A tedy podle Savitchova tvrzení, $MEM_1 \in PSPACE$.

Důkaz PSPACE těžkosti provedeme redukcí z $QBF$. Neformálně ke každé $QBF$ $\phi$ sestrojíme LOA $M_1$, který bude simulovat rekurzivní vyhodnocování procedury $Eval$ (viz. přednáška PSPACE) a přijme právě když formule $\phi$ je pravdivá. Vstupní páska bude reprezentovat zásobník pro simulaci $Eval$. Vzhledem k tomu, že hloubka zanoření je omezena lineárně vzhledem k velikosti $\phi$, velikost pásky bude polynomiální k $|\phi|$ (počet kvantifikátorů plus počet operátorů). Počáteční řetězec na vstupu je tedy nutné zvolit tak, aby jsme obsáhli max. velikost zásobníku.

Více detailněji. Dále uvažuji, že $\phi \equiv Q_1 x_1 \dots Q_m x_m F$ je QBF. Nejprve si připravíme LOA $M_1$, který bude řídit výpočet procedury $Eval$. LOA $M_1$ navrhneme tak, aby pracoval nezávisle na formuli $\phi$. V průběhu výpočtu $M_1$ obsahuje na pásce řetězec ve tvaru

$$\#\langle F\rangle\#(\#f x_1 \dots x_m)^m\#d\#\#x'_1 \dots x'_m\#\langle F'\rangle\#,$$

kde $m$ je počet proměnných ve formuli $\phi$, $x_1 \dots x_m$ je ohodnocení proměnných $x_1 \dots x_m$, $f$ je příznak, který označuje, jak je aktuálně sledovaná proměnná kvantifikována, $d$ je aktuální hloubka zanoření, $\langle F\rangle$ je kód vstupní Booleovské formule a $x'_1 \dots x'_m$ je aktuální ohodnocení proměnných pro které se má vyhodnotit logická funkce $F$.

LOA $M_1$ pracuje potom následovně: Vstupní obsah pásky je

$$\#\langle F\rangle\#\#\langle Q_1\rangle 0^m\# \dots \langle Q_m\rangle 0^m\#\langle m\rangle\#\#0^m\#\langle F\rangle\# \tag{1}$$

(všechny proměnné jsou nastaveny na hodnotu 0 a hloubka na $m$). LOA $M_1$ potom simuluje výpočet LOA $M_F$ (vyhodnocení funkce $F$ podle zadaného přiřazení, popis dále) pro ohodnocení proměnných v hloubce $m$ – za $x'_1 \dots x'_m$ se nahradí ohodnocení z úrovně $m$ a řetězec $\langle F'\rangle$ je obnoven za $\langle F\rangle$ (LOA $M_F$ přepíše řetězec reprezentující formuli, proto je nutné ho obnovit) Podle výsledku $M_F$ se informace propaguje o úroveň níž (řekněme do $i$), kde podle předaného výsledku, aktuální hodnoty $f_i$ a hodnoty $x_i$ provede akce podle procedury $Eval$ (buď se hodnota propaguje do další úrovně nebo je hodnota $x_i$ změněna na další hodnotu, pokud je to ovšem možné, a hodnoty všech proměnných $x_j$, $i < j \leq m$ na všech vyšších hloubkách jsou nastaveny na 0 a celý postup je opakován). Pokud se informace dopropagovala až do první úrovně a procedura $Eval$ je již vyčíslená (rekurzivně jsme již získali odpověď na to, jestli $\phi$ je pravdivá), přijmeme právě tehdy, když získaná hodnota je 1 ($\phi$ je pravdivá). Tedy $M_1$ přijmě právě tehdy, když vstupní formule $\phi$ je pravdivá.

LOA $M_F$ na své vstupní pásce očekává vstupní řetězec ve tvaru $x_1 \dots x_m \# \langle \phi \rangle$, kde $\langle \phi \rangle$ je kód formule $\phi$ a $x_1 \dots x_m$ jsou ohodnocení proměnných $x_1 \dots x_m$. LOA $M_F$ po ukončení výpočtu zapíše na pásku hodnotu $\phi(x_1 \dots x_m)$. Ve zkratce, $M_F$ pracuje tak, že nejprve nahradí všechny výskyty proměnných ve řetězci $\langle \phi \rangle$ za jejich hodnoty. LOA $M_F$ začne procházet vstupní řetězec s formulí a začne vyhodnocovat formuli od nejvnitřnějších podformulí a tyto podformule nahrazuje za jejich výsledné hodnoty. (tedy např. nejprve se vyhodnotí podformule typu $\neg x_i$, $x_i \wedge x_j$ atd. a potom teprve složitější podformule). Časová složitost tohoto přístupu bude velká, ale na druhou stranu je potřeba pouze ta část pásky, kde je zapsán vstup (tedy $M_F$ je opravdu LOA). Činnost $M_F$ je opět nezávislá na vstupní formuli $\phi$.

Hloubka rekurze procedury $Eval$ je $m$, kde $m$ je počet proměnných v $\phi$. Tedy pokud je vstupní řetězec zadán podle (1), stačí pro vyhodnocení pravdivosti QBF $\phi$ pouze prostor pásky, kde je zapsán vstup. Tedy LOA $M_1$ pro libovolnou QBF $\phi$ přijme právě když je pravdivá.

Redukce $R$ tedy pro libovolnou QBF $\phi = Q_1 x_1 \dots Q_m x_m F$ vrátí dvojici $(M_1, w)$, kde

$$w = \#\langle F \rangle \#\#\langle Q_1 \rangle 0^m \# \dots \langle Q_m \rangle 0^m \# \langle m \rangle \#\# 0^m \# \langle F \rangle \#.$$

Velikost řetězce $\langle F \rangle$ je $O(n^2)$, kde $n = |\phi|$ a ve stejném čase může být řetězec sestaven. Vzhledem k tomu, že $m \leq n$, je celková délka $|w| \in O(n^2)$ a tento řetězec může být sestaven v pol. čase. Tedy celá redukce $R$ může být implementována DTS v polynomiálním čase. Navíc platí:

$$\phi \in QBF \Rightarrow R(\phi) \in MEM_1 \quad \text{a} \quad \phi \notin QBF \Rightarrow R(\phi) \notin MEM_1.$$

Tedy $MEM_1$ je $PSPACE$-úplný.

**2. Příklad**

Nejprve dokážeme, že $OPT\_PARTITION \in NPO$.

- Množina vstupních případů problému je množina dvojic $(S, v)$, kde $S$ je množina položek a $v$ je váhová funkce. Deterministický Turingův stroj nejprve zkontroluje, zda na vstupu je správně zformovaná instance problému. Následně ověří, jestli je váhová funkce dobře definována (pro každou položku z $S$ zkontroluje, zda váhová funkce přiřazuje této položce nějakou váhu). Tento TS pracuje v polynomiálním čase. Tedy $I \in P$.
- Uvažujme nějaké $x \in I$. Potom pro každé $A \in F(x)$ platí $A \subseteq S$, tedy $|A| \leq |S|$. Přípustná řešení jsou tedy ohraničena polynomem.
- Opět uvažujme nějaké $x \in I$. Pokud máme nějaké $|A| \leq |S|$, potom lze v polynomiálním čase (vzhledem k $|S|$) rozhodnout, zda $A \subseteq S$ (pro každý prvek z $A$, zkontrolujeme, jestli je i v $S$).

– Cena řešení $c$, která je dána jako rozdíl součtu vah položek $S \setminus A$ a $A$ lze opět vypočítat DTS v polynomiálním čase vzhledem k $|S|$ (pro každý prvek z $S \setminus A$ najdeme odpovídající váhy a sečteme je, to samé pro $A$ a nakonec obě hodnoty odečteme.

V dalším kroku ukážeme, že jazyk asociovaný k tomuto opt. problému $L_{OP}$ je $NP$-úplný. Vzhledem k tomu, že $OPT\_PARTITION \in NPO$, tak $L_{OP} \in NP$. Důkaz, že $L_{OP}$ je $NP$-těžký provedeme redukcí z problému $PARTITION$. Nechť $(T, v)$ je instance problému $PARTITION$. Redukce $R$ pouze vstup transformuje na $((T, v), 0)$. Cena řešení je 0 právě když lze množinu $T$ rozdělit na dvě disjunktní množiny se stejnou váhou. Potom tedy platí, že

$$(T, v) \in PARTITION \Leftrightarrow R((T, v)) \in L_{OP}.$$

Redukci lze zřejmě implementovat pomocí DTS v pol. čase. Tedy $L_{OP}$ je $NP$-úplný jazyk.

Nyní už můžeme přejít k samotnému důkazu neexistence absolutního aproximačního algoritmu. Důkaz povedeme sporem, tedy předpokládáme, že existuje absolutní aproximativní algoritmus $A$ s absolutní chybou $k$. Pomocí algoritmu $A$ vytvoříme polynomiální algoritmus, který pro instanci $OPT\_PARTITION$ nalezene jeho optimální řešení. Tento polynomiální algoritmus nejprve pro vstupní případ $x = (S, v)$ problému $OPT\_PARTITION$ sestrojí vstup $x' = (S, v')$, kde $v'(z) = (k + 1)v(z)$, pro každé $z \in S$ a na vstupu $x'$ odsimuluje výpočet algoritmu $A$. Přípustná řešení jsou pro oba případy stejná, tedy $F(x) = F(x')$ (výběr podmnožin $S$). Avšak cena přípustných řešení problému $x'$ je $(k + 1)$ násobek cen řešení $x$.

Algoritmus $A$ vypočítá $A(x')$. Vzhledem k tomu, že $A$ je absolutní apr. algoritmus s chybou $k$, platí:

$$|c'(A(x')) - OPT(x')| = \left| \left| \sum_{z \in S \setminus A(x')} v'(z) - \sum_{z \in A(x')} v'(z) \right| - OPT(x') \right| \leq k \qquad (2)$$

Vzhledem k tomu, že $v'(z) = (k+1)v(z)$, platí i $OPT(x') = (k+1)OPT(x)$. Tedy úpravou (2) dostáváme

$$\left| \left| \sum_{z \in S \setminus A(x')} (k+1)v(z) - \sum_{z \in A(x')} (k+1)v(z) \right| - (k+1)OPT(x) \right| \leq k \qquad (3)$$

Celou nerovnici můžeme podělit $k + 1$ a protože pracujeme v celých číslech dostáváme následující

$$\left| \left| \sum_{z \in S \setminus A(x')} v(z) - \sum_{z \in A(x')} v(z) \right| - OPT(x) \right| = |c(A(x')) - OPT(x)| = 0.$$

Tedy $A(x')$ je optimálním řešením pro případ $x$. Vzhledem k tomu, že $A$ pracuje s polynomiální složitostí (definice), celý algoritmus pro výpočet opt. řešení pracuje v polynomiálním čase. Tedy $OPT\_PARTITION \in PO$ a vzhledem k tomu, že asociovaný jazyk $L_{OP}$ je $NP$-úplný, dostáváme, že $NP = P$, což je spor s předpokladem. Absolutní apr. algoritmus tedy neexistuje.