Formálne jazyky

Jednosmerné TS alebo konečné automaty

- TS sú robustné voči modifikáciam
- existuje modifikácia, ktorá zmení (zmenší) výpočtovú silu TS?
- áno, modifikácia ale musí ohraničiť výpočtový zdroj Turingovho stroja

  polynomiálny čas trieda P

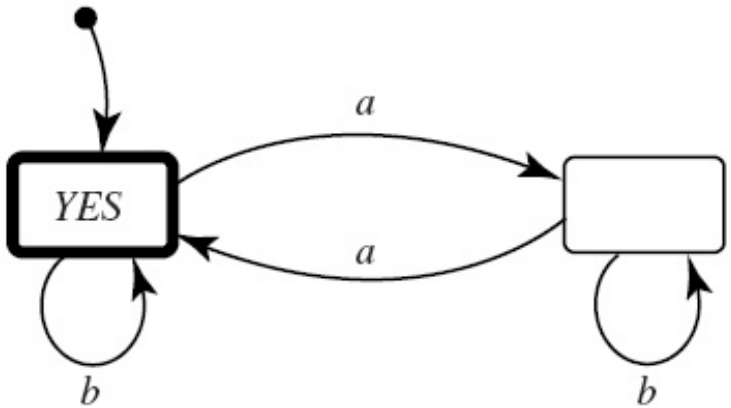
  polynomiálny priestor trieda PSPACE

  (triedy P a PSPACE sú menšie než trieda rozhodnuteľných problémov)

  jeden smer pohybu na páske ohraničenie spočíva v tom, že TS nemá možnosť vrátiť sa k informácii, ktorú už raz prečítal a ani nemá možnosť si o prečítanom úseku uchovať kompletnú informáciu (*riadiaca jednotka má len konečný počet stavov*) = konečné automaty

Konečné automaty

- vstupný reťazec sa číta zľava doprava, symbol po symbole
- prečítaný symbol sa neprepisuje
- výpočet sa zastaví po prečítaní posledného symbolu alebo v situácii, keď prechodový diagram neumožňuje žiadny ďalší krok
- ak sa výpočet zastaví po prečítaní celého vstupu v stave *YES*, znamená to odpoveď „Áno“ (konečný automat akceptuje vstup); ak sa výpočet zastaví v inom stave alebo sa zastaví a nie je prečítaný celý vstup, znamená to odpoveď „Nie“ (automat zamieta vstup)

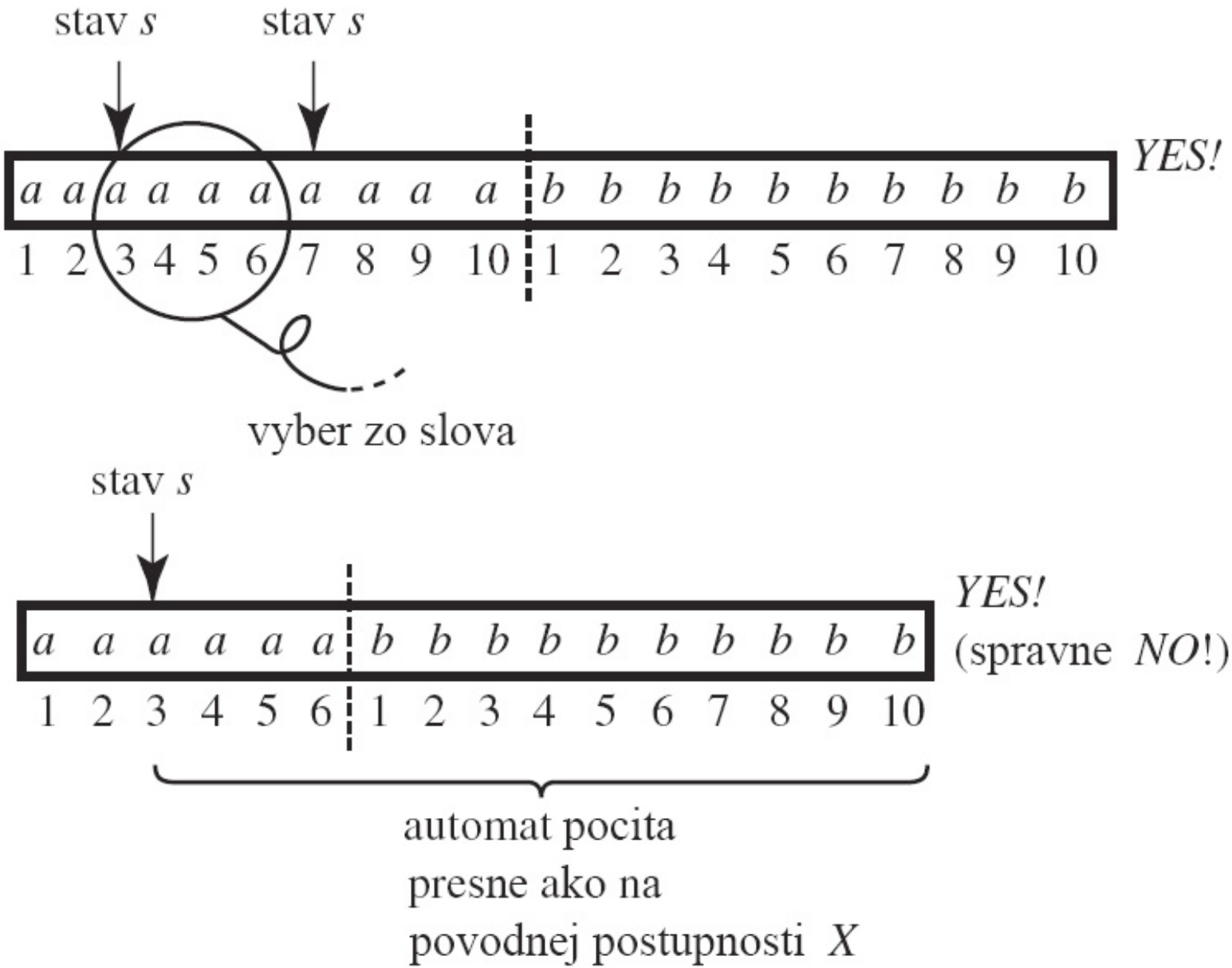


Konečné automaty - dolné odhady

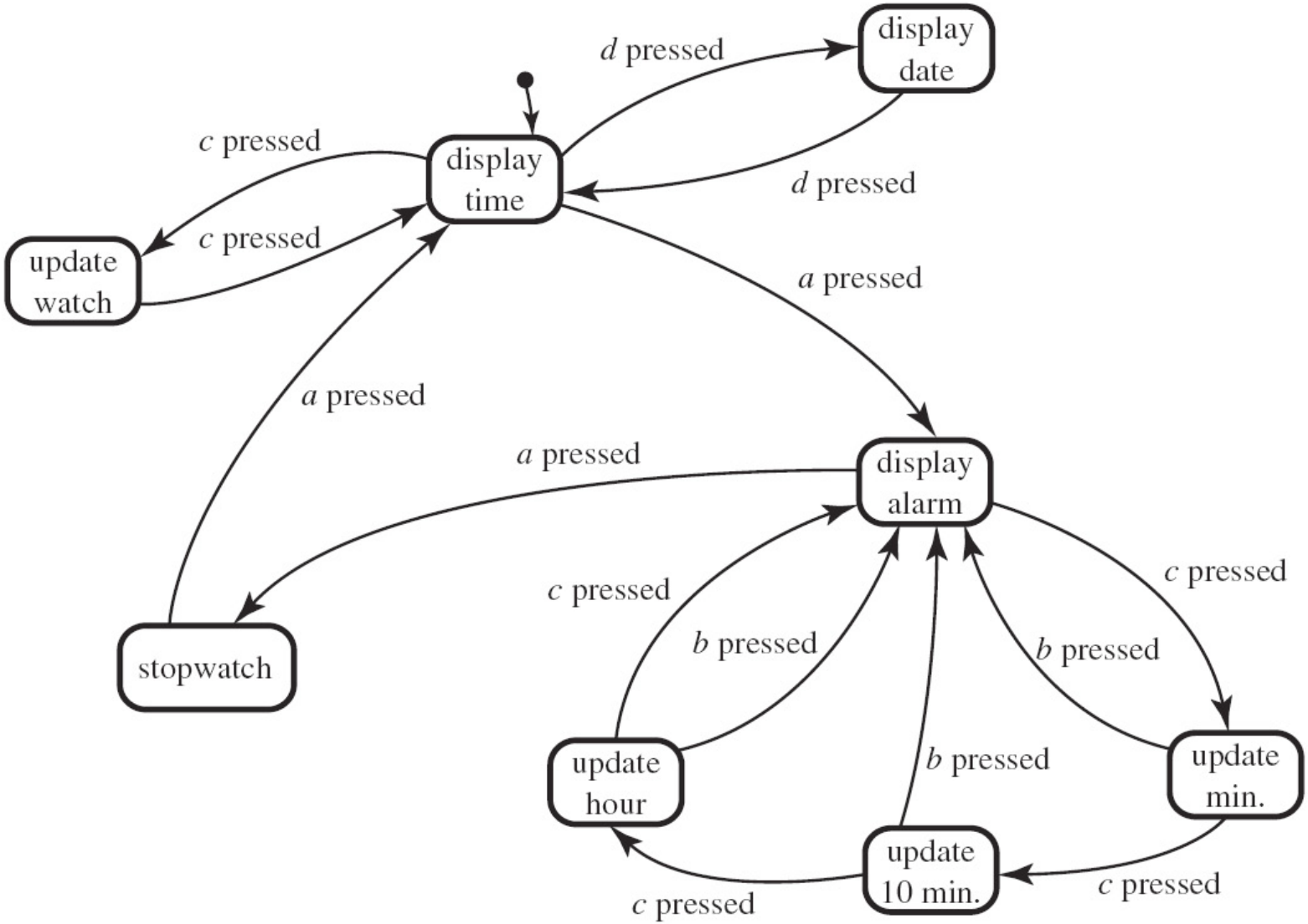
Problém rozhodnúť, či daný reťazec obsahuje rovnaký počet symbolov a, b

Tvrdenie Neexistuje konečný automat, ktorý rieši tento problém.

Dôkaz Sporom.
Predpokladajme, že existuje automat F, ktorý problém rieši. Označme N počet stavov automatu F. Uvážme vstupný reťazec, ktorý obsahuje presne $N+1$ symbolov a, za ktorými nasleduje presne $N+1$ symbolov b. Pri čítaní úvodnej sekvencie a-čok musia byť dve políčka, ktoré automat číta v tom istom stave (*počet políčok je* $N+1$, *počet stavov je* N). Vytvorme nový vstupný reťazec tak, že odstránime všetky symboly medzi týmito dvoma a-čkami (viz obrázok).
Výpočet na novom vstupnom reťazci skončí v tom istom stave a v tej istej pozícii ako výpočet na pôvodnom vstupnom reťazci.
Ak automat akceptuje obidva reťazce dostávame spor (*modifikovaný reťazec nemá požadovanú vlastnosť*). Naopak, ak automat obidva reťazce zamieta – spor (*pôvodný reťazec má požadovanú vlastnosť*).


stav s
stav s
a a a a a a a a a a b b b b b b b b b b
YES!
1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10
vyber zo slova
stav s
a a a a a a b b b b b b b b b b
YES!
(spravne NO!)
1 2 3 4 5 6 1 2 3 4 5 6 7 8 9 10
automat pocita
presne ako na
povodnej postupnosti X

Konečné automaty ako model systému s udalosťami



Konečné automaty - terminológia

pojem jazyka ako ekvivalentu pojmu rozhodovací problém
regularny jazyk
regularne vyrazy

Formálne jazyky

Generatívne výpočtové modely

Fixujme rozhodovací problém P (*resp. jazyk* P_L)

rozhodovanie určiť, či pre daný vstup X je odpoveď „Áno“ alebo „Nie“
(*resp. určiť, či* X *patrí do jazyka* L_X)

generovanie vymenovať všetky vstupy, pre ktoré je odpoveď „Áno“
(*resp. vymenovať všetky slová jazyka* P_L)

Motivácia

návod, ako vytvoriť „správny“ reťazec

formálna gramatika

Formálne gramatiky

príklad

Formálne gramatiky - vlastnosti

Fakt

Trieda jazykov generovaných formálnymi gramatikami je práve trieda rozhodnuteľných problémov.

Formálne gramatiky s obmedzeniami

kontextové gramatiky reťazec na ľavej strane pravidla je nie je dlhší než reťazec na pravej strane pravidla

bezkontextové gramatiky na ľavej strane každého pravidla je práve jeden neterminálny symbol

regulárne gramatiky

Formálne gramatiky - problém syntaktickej analýzy

Pre danú formálnu gramatiku a reťazec rozhodnúť, či je slovo sa gramatikou vygenerovať.

rozhodnúť, či program v programovacom jazyku (definovanom gramatikou) je syntakticky správny

Problém syntaktickej analýzy je

čiastočne rozhodnuteľný pre formálne gramatiky

rozhodnuteľný pre kontextové gramatiky

polynomiálne riešiteľný pre bezkontextové gramatiky

je dôležité, aby sme pre definíciu syntaxe programovacieho jazyka použili čo najjednoduchší typ gramatiky