Prednáška 8 - Zásobníkové automaty, vlastnosti bezkontextových jazykov

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

12.11.2020

Zásobníkové automaty

- Na minulej prednáške sme sa bavili o tzv. **bezkontextových jazkoch**.
- Sú to jazyky, pre ktoré existuje **bezkontextová gramatika**, ktorá ich generuje.
- Taktiež sme si skorej spomínali, že existujú tzv. regulárne jazyky - pre ne existujú regulárne výrazy a konečné automaty.
- **Konečné automaty** sú matematický model **výpočtového zariadenia**, ktoré dokáže akceptovať **regulárne jazyky**.
- Podobne aj **bezkontextové jazyky** majú svoj výpočtový model, ktorý ich vie akceptovať - **zásobníkové automaty (pushdown automata)**.

Zásobníkové automaty

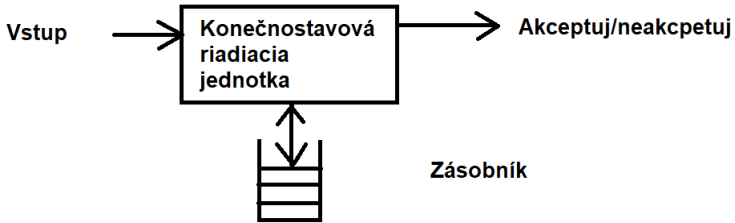
- Zjednodušene povedané, zásobníkové automaty vzniknú tak, že vezmeme ε-NKA a pridáme k nemu "pamäť" vo forme zásobníka.
- T.j. pamäť, do ktorej je možné dáta zapisovať a z ktorej je možné dáta čítať, pričom prístup do nej je realizovaný rovnako, ako do dátového typu zásobník, t.j. podľa princípu Last-In-First-Out.
- Vždy teda vieme prečítať prvú dátovú položku na vrchu zásobníka, rovnako ako prvú musíme vždy odstrániť položku na vrchu zásobníka.
- Keď do pamäte dátovú položku vkladáme, vkladáme ju na vrch zásobníka.

Zásobníkové automaty - neformálny príklad

- Zásobník, do ktorého si zásobníkové automaty vedia ukladať dáta - formou tzv. zásobníkových symbolov, má teoreticky neobmedzenú kapacitu.
- Na rozdiel od konečných automatov si teda zásobníkové automaty (ZA) vedia pamätať nekonečné množstvo informácie (konečné automaty si vedeli pamätať informácie len vďaka svojim stavom, ktorých bol vždy konečný počet).
- Avšak, **rovnako ako KA**, zásobníkové automaty spracúvajú vstup takým spôsobom, že sa **nevedia vrátiť k už prečítanej časti vstupu**.
- V dôsledku existencie pamäte je možné pomocou zásobníkových automatov spracovať aj také jazyky, ktoré sa nedali spracovať konečným automatom, napríklad $L = \{a^n b^n \mid n \geq 0\}$.

Zásobníkové automaty - neformálny príklad

Na obrázku je high-level pohľad na zásobníkový automat:



- Vstup je vstupný reťazec, ktorý sa spracúva
- Konečnostavová riadiaca jednotka predstavuje stavy, v ktorých sa môže ZA nachádzať a prepínanie medzi nimi
- Zásobník je pamäťové médium, kam si ZA ukladá informácie
- Akceptuj/neakceptuj je výstup ZA pre zadaný vstup

Zásobníkové automaty - činnosť

Typická činnosť ZA:

- ZA číta vstupný reťazec symbol-za-symbolom
- Na základe aktuálneho **vstupného symbolu** a aktuálneho **symbolu na vrchu zásobníka** ZA realizuje prechod medzi jeho možnými stavmi.
- Prechody sú teoreticky možné **aj tak**, že sa ignoruje vstupný symbol, t.j. zo vstupu sa akoby prečíta ε a aktuálny vstupný symbol sa môže spracovať v ďalšom stave.

Zásobníkové automaty - činnosť

Jeden prechod ZA:

1. ZA prečíta (spracuje) aktuálny vstupný symbol a posunie sa na ďalší vstupný symbol.

   (v prípade, že sa vstupný symbol ignoruje, tak sa vlastne číta ε)
2. ZA sa prepne z aktuálneho stavu do nového stavu podľa prechodovej funkcie.
3. Symbol z vrchu zásobníka, ktorý bol na prechod použitý sa zo zásobníka odstráni a nahradí sa novým reťazcom symbolov. Teoreticky sa môže nahradiť prázdnym reťazcom (čo de facto znamená len odstránenie symbolu z vrchu zásobníka). Prípadne sa môže nahradiť rovnakým symbolom (čiže de facto sa obsah zásobníka nezmení).

ZA - príklad

Nech je daný jazyk $L_{wwr} = \{ww^R \mid w \in \{0,1\}^*\}$, t.j. palindrómy zo symbolov $0, 1$ párnej dĺžky. Tento jazyk je bezkontextový jazyk, generovaný bezkontextovou gramatikou:

1. $S \to \varepsilon$
2. $S \to 0S0$
3. $S \to 1S1$

Neformálne popíšeme, ako by mohol vyzerať ZA, ktorý dokáže spracovať takýto jazyk.

1. Na začiatku je ZA v počiatočnom stave q_0. V tomto stave ZA číta vstupné symboly a kópiu každého vstupného symbolu vloží do zásobníka.
2. V momente, keď ZA "odhadne", že práve prečítal polovicu zo vstupného reťazca, prepne sa do stavu q_1. To znamená, že ak je na vstupe skutočne reťazec v tvare ww^R, tak sme práve prečítali w a v zásobníku sa nachádza w tak, že jeho posledný symbol je na vrchu a prvý symbol je na dne zásobníka. Čiže od vrchu po spodok sa v zásobníku nachádza práve w^R.
3. V stave q_1 postupne porovnávame čítaný vstupný symbol so symbolom na vrchu zásobníka. Ak sú rovnaké, symbol zo zásobníka odstránime a zároveň pokračujeme v čítaní ďalšieho symbolu.
4. Ak sa nám podarí vyprázdniť zásobník posledným vstupným symbolom, tak sme museli mať na vstupe ww^R a automat bude vstup **akceptovať**.

- Popis činnosti je síce korektný, avšak spolieha sa na prvok náhodnosti - ak je na vstupe slovo z jazyka, t.j. ww^R tak prechod zo stavu q_0 do q_1 musí nastať **práve** po prečítaní polovice vstupného slova, t.j. po prečítaní w.
- Ak by prechod nastal skôr / neskôr, dôjde k zaseknutiu automatu, lebo bude alebo prázdny zásobník, alebo nebudú sedieť vstupný symbol = symbol na vrchu zásobníka.
- Túto situáciu riešime podobne ako pri NKA, ε-NKA. Vstup akceptujeme, ak **existuje** možnosť akceptácie vstupu. To znamená, musíme skúmať všetky možné výpočty a ak aspoň jeden je taký, že vstup akceptujeme, tak potom vstup považujeme za akceptovaný.

Definícia ZA

Formálne definujeme zásobníkový automat ako sedmicu $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:

- Q je konečná množina stavov ZA
- Σ je konečná vstupná abeceda ZA, t.j. abeceda symbolov, z ktorých môže pozostávať vstupný reťazec
- Γ je konečná zásobníková abeceda, t.j. abeceda symbolov, ktoré sa môžu nachádzať v zásobníku
- q_0 je počiatočný stav ZA, $q_0 \in Q$
- Z_0 je počiatočný zásobníkový symbol, $Z_0 \in \Gamma$. Na začiatku činnosti ZA sa v zásobníku nachádza práve len symbol Z_0.
- F sú akceptačné stavy ZA, $F \subseteq Q$
- δ je prechodová funkcia zásobníkového automatu.

Definícia ZA - prechodová funkcia

Prechodová funkcia δ je definovaná tak, že jej vstupom je trojica (q, a, X), kde:

- q je stav ZA, $q \in Q$
- a je alebo vstupný symbol $a \in \Sigma$, alebo $a = \varepsilon$ (čo reprezentuje situáciu,v ktorej sa **ignoruje** vstupný symbol, t.j. nemusí to nutne znamenať, že vstup je prázdny!)
- X je zásobníkový symbol, $X \in \Gamma$
- t.j. uvažujeme situáciu, že ZA je v stave q, na vstupe číta a a z vrchu zásobníka vyberá symbol X

Výstupom prechodovej funkcie je **množina** dvojíc (p, γ):

- p je nový stav, do ktorého sa ZA prepína, $p \in Q$
- γ je reťazec, ktorý je vkladaný do zásobníka (t.j. nahrádza v ňom X)

Ak:

- $\gamma = \varepsilon$, tak sa zo zásobníka len odstráni X
- $\gamma = X$, tak sa zásobník de facto nemení
- $\gamma = YZ$, tak sa v zásobníku X nahradí Z a na vrch sa vloží Y.

ZA - príklad

Zásobníkový automat, ktorého slovný popis je na slajde č. 9, by bol formálne popísaný:

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, Z_0\}$
- q_0 je počiatočný stav
- Z_0 je počiatočný zásobníkový symbol
- $F = \{q_2\}$ bude akceptačný stav
- Prechodová funkcia δ je daná:

ZA - príklad

- $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$, $\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\}$
  Tieto 2 pravidlá sú použité úplne na začiatku činnosti. Na začiatku je v zásobníku len symbol Z_0. Prečítame vstupný symbol (0 alebo 1) a vložíme ho do zásobníka - čo zapíšeme tak, že zo zásobníka vyberáme Z_0 a vkladáme $0Z_0$, resp. $1Z_0$.
- $\delta(q_0, 0, 0) = \{(q_0, 00)\}, \delta(q_0, 0, 1) = \{(q_0, 01)\}$,
  $\delta(q_0, 1, 0) = \{(q_0, 10)\}, \delta(q_0, 1, 1) = \{(q_0, 11)\}$
  Tieto 4 pravidlá znamenajú, že sme v stave q_0 a v podstate presunieme prečítaný symbol na vrch zásobníka. Všimnite si, že sú definované pre všetky možné kombinácie (vstupný symbol,symbol na vrchu zásobníka).

- $\delta(q_0, \varepsilon, 0) = \{(q_1, 0)\}$, $\delta(q_0, \varepsilon, 1) = \{(q_0, 1)\}$, $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$

  Tieto 3 pravidlá umožňujú prechod do stavu q_1 zo stavu q_0 bez čítania vstupného symbolu a bez zmeny obsahu zásobníka.
- $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}, \delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$

  Tieto 2 pravidlá realizujú čítanie vstupného symbolu a jeho porovnanie so symbolom na vrchu zásobníka. Ak sa zhodujú, vstup sa spracuje a symbol sa zo zásobníka odstráni.
- $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$.

  Ak sme korektne prečítali celý vstup, tak existuje situácia, že sme v stave q_1 a na vrchu zásobníka je symbol Z_0. V takom prípade sa prepneme do stavu q_2 a vstup akceptujeme.

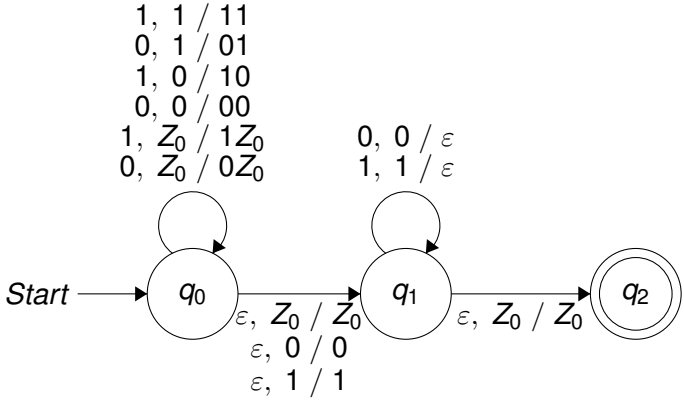
Grafická reprezentácia ZA

Podobne ako pri KA, aj pri ZA máme grafickú reprezentáciu vo forme prechodového diagramu.

- Vrcholy diagramu predstavujú stavy, počiatočný stav je označený šípkou *Start*, akceptačné stavy sú označené 2 kružnicami.
- Ak je v prechodovej funkcii $(p, \alpha) \in \delta(q, a, X)$, tak v automate je šípka zo stavu q do stavu p ohodnotená $a, X/\alpha$, t.j. signalizuje prechod zo stavu q do stavu p ak zo vstupu čítame a, zo zásobníka vyberáme X a vkladáme α.

Grafická reprezentácia ZA - príklad

Pre ZA zo slajdov 13 - 15 je jeho grafická reprezentácia:



Výpočet zásobníkového automatu

- Pri sledovaní toho, ako ZA spracúva vstup, musíme sledovať:
  - Aktuálny stav, v ktorom sa nachádza ZA
  - Aktuálny stav vstupu - t.j. ešte neprečítaná časť a aktuálne čítaný vstupný symbol
  - Aktuálny obsahu zásobníka

  Definujme pojem **konfigurácia**, ktorý bude predstavovať trojicu (q, w, γ), ktorá popisuje aktuálnu situáciu pri spracúvaní vstupného reťazca, kde:
  - q je aktuálny stav, v ktorom sa nachádza ZA
  - w je ešte neprečítaná časť vstupného reťazca
  - γ je aktuálny obsah zásobníka písaný "od vrchu po dno", t.j. prvý symbol γ je symbol na vrchu zásobníka, posledný symbol γ je symbol na dne zásobníka

Výpočet zásobníkového automatu

- Pomocou konfigurácií vieme popísať ako ZA spracúva vstupný reťazec, medzi akými stavmi sa prepína a aký je priebežný obsah zásobníka.
- Pomocou konfigurácií teda popisujeme **výpočet** zásobníkového automatu.
- Každý krok výpočtu automatu predstavuje prechod z jednej konfigurácie do nasledujúcej.
- Prechod medzi konfiguráciami označíme symbolom $\vdash$.

Výpočet zásobníkového automatu

Nech $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je ZA. Nech $\delta(q, a, X)$ obsahuje (p, α). Potom pre všetky reťazce $w \in \Sigma^*$ a $\beta \in \Gamma^*$:

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

Teda prechádzame zo stavu q do stavu p, pričom zo vstupu aw čítame prefix a (a môže byť symbol alebo ε), z vrchu zásobníka vyberáme X a vkladáme doň α. Všetko to, čo bolo pod symbolom X v zásobníku (reťazec β) bude po novom pod symbolmi v α.

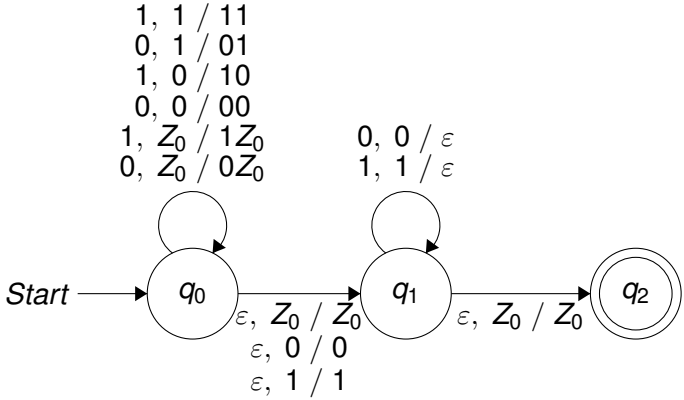
Výpočet zásobníkového automatu

Na popis výpočtu na viacero krokov používame symbol $\vdash^*$ definovaný takto: Nech I je konfigurácia zásobníkového automatu.

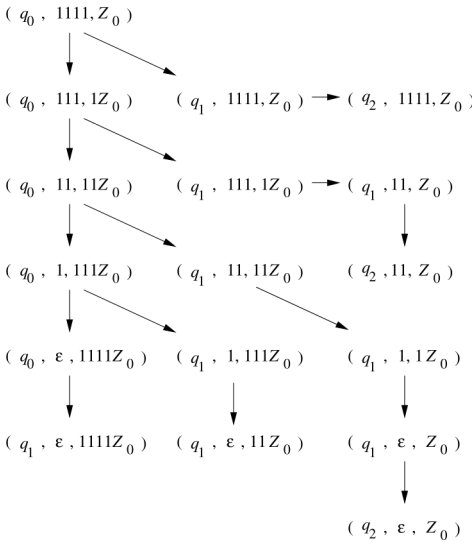
1. (Základný krok:) $I \vdash^* I$ pre všetky konfigurácie I.
2. (Rekurzívny krok:) $I \vdash^* J$, ak existuje konfigurácia K, pre ktorú platí $I \vdash K$ a $K \vdash^* J$.

Príklad ([1])

Uvažujme, ako bude spracovaný reťazec 1111 zásobníkovým automatom:



Príklad ([1])


(q0 , 1111, Z0)
(q0 , 111, 1Z0)
(q1 , 1111, Z0)
(q2 , 1111, Z0)
(q0 , 11, 11Z0)
(q1 , 111, 1Z0)
(q1 , 11, Z0)
(q0 , 1, 111Z0)
(q1 , 11, 11Z0)
(q2 , 11, Z0)
(q0 , ε , 1111Z0)
(q1 , 1, 111Z0)
(q1 , 1, 1Z0)
(q1 , ε , 1111Z0)
(q1 , ε , 11Z0)
(q1 , ε , Z0)
(q2 , ε , Z0)

Príklad ([1])

- Z obrázku je vidieť, že existuje veľa rôznych výpočtových vetiev, avšak len jedna vie skončiť v akceptačnom stave q_2 a zároveň prečítať celý vstup.
- Avšak to postačuje na to, aby sme mohli povedať, že daný ZA akceptuje zadaný reťazec.
- Vo všetkých ostatných vetvách platí, že reťazec nie je akceptovaný, pretože:
  - Automat síce prejde do q_2, ale neprečíta celý vstup
  - Automat síce prečíta celý vstup, ale skončí v stave q_1

Jazyk akceptovaný zásobníkovým automatom

Pri zásobníkových automatoch existujú 2 spôsoby akceptácie reťazcov. V oboch prípadoch je samozrejmé nutnou podmienkou akceptácie vstupu to, aby bol celý prečítaný

- Akceptácia akceptačným stavom - automat prečíta celý vstup a skončí v akceptačnom stave. Obsah zásobníka nie je podstatný.
- Akceptácia prázdnym zásobníkom - automat prečíta celý vstup a skončí s prázdnym zásobníkom. Stav, v ktorom skončí, nie je podstatný.
- Oba spôsoby sú ekvivalentné! Vždy sa dá automat pracujúci jedným spôsobom previesť na automat pracujúci druhým spôsobom a naopak.

Akceptácia akceptačným stavom

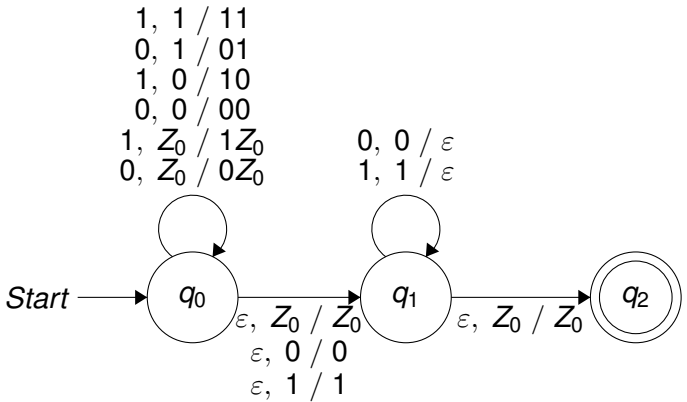
Nech $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je ZA. Potom jazyk $L(P)$ zásobníkového automatu akceptovaný akceptačným stavom je:

$$L(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha)\}$$

kde $q \in F$ a α je ľubovoľný obsah zásobníka.

Príklad

Práve ZA, ktorý sme navrhli pre jazyk $L = \{ww^R \mid w \in \{0,1\}^*\}$ pracuje tak, že akceptuje reťazce akceptačným stavom:



Akceptácia prázdnym zásobníkom stavom

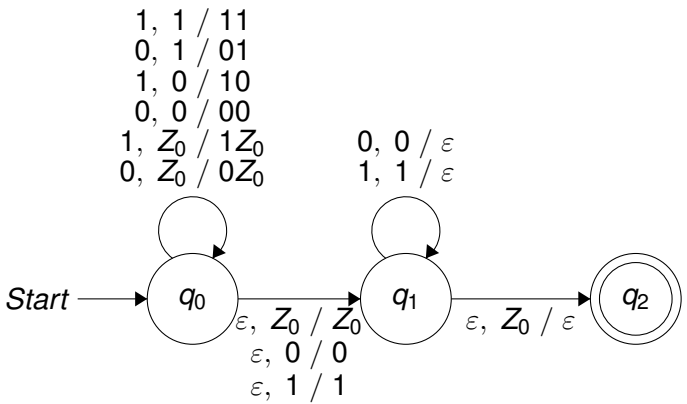
Nech $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je ZA. Potom jazyk $N(P)$ zásobníkového automatu akceptovaný prázdnym zásobníkom je:

$$N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$$

kde q je ľubovoľný stav.

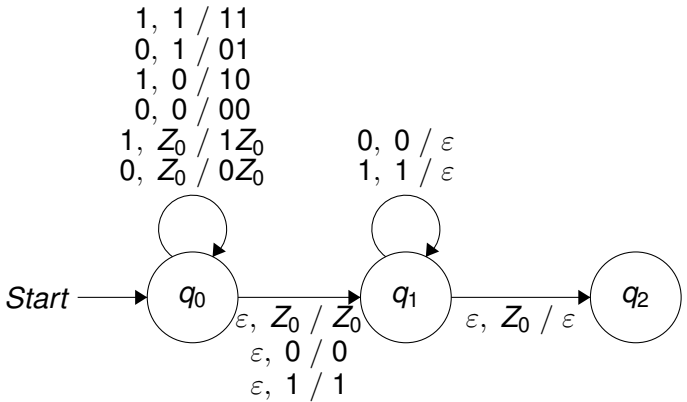
Príklad

Verzia ZA, ktorý akceptuje jazyk $L = \{ww^R \mid w \in \{0,1\}^*\}$ pomocou prázdneho zásobníka:



Príklad

Rovnako dobrá by bola aj verzia, kde nie je akceptačný stav, keďže pri akceptácii prázdnym zásobníkom je irelevantný.



Vzťah ZA a BezkG

- Zásobníkové automaty zohrávajú pri akceptácii bezkontextových jazykov rovnakú úlohu ako konečné automaty pri akceptácii regulárnych jazykov.
- Dá sa ukázať, že:
  - Každý bezkontextový jazyk je akceptovateľný nejakým zásobníkovým automatom.
  - Každý jazyk nejakého zásobníkového automatu je zároveň bezkontextový jazyk.
- Čo vlastne znamená:
  - Každý jazyk, pre ktorý existuje bezkontextová gramatika, ktorá ho generuje, je akceptovateľný nejakým zásobníkovým automatom.
  - Každý jazyk nejakého zásobníkového automatu je zároveň generovateľný nejakou bezkontextovou gramatikou.

Rozhodovacie problémy a bezkontextové jazyky

- Pri regulárnych jazykoch sme si spomínali 3 dôležité rozhodovacie problémy:
  - Či je regulárny jazyk prázdny, t.j. $L = \emptyset$ (emptiness problem)
  - Či pre nejaký reťazec w a jazyk L platí, že $w \in L$ (membership problem)
  - Či pre 2 jazyky L_1, L_2 platí $L_1 = L_2$ (equivalence problem)
- Vo všetkých prípadoch sme mali vždy k dispozícii reprezentáciu jazyka ako konečného automatu alebo regulárneho výrazu.
- Tie isté problémy môžeme riešiť aj pri bezkontextových jazykoch, s tým rozdielom, že reprezentácie bezkontextových jazykov sú alebo zásobníkový automat, alebo bezkontextová gramatika.

Problém $L = \emptyset$?

- Pre bezkontextové jazyky (podobne ako pre regulárne) platí, že **existuje algoritmus**, ktorý dokáže zistiť, či $L = \emptyset$, t.j. táto úloha je rozhodnuteľný problém.
- My si ukážeme algoritmus, ktorý to dokáže zistiť, ak je bezkontextový jazyk daný bezkontextovou gramatikou.
- V prípade, že by bol jazyk daný zásobníkovým automatom, bolo by potrebné najprv automat konvertovať na bezkontextovú gramatiku (čo sme si síce neukazovali, ale v praxi existuje algoritmus, ktorý to dokáže).

Algoritmus riešenia $L = \emptyset$

- Algoritmus, ktorý vie rozhodnúť, či je bezkontextový jazyk popísaný bezkontextovou gramatikou prázdny alebo nie, hľadá v gramatike tzv. **generujúce symboly**.
- Generujúce symboly gramatiky sú také symboly, pre ktoré platí, že sa z nich **dá odvodiť** nejaký reťazec terminálov, t.j. ak X je symbol gramatiky, tak potom musí existovať reťazec $w \in T^*$, že $X \Rightarrow^* w$.
- Pre terminálne symboly **vždy platí**, že vedia generovať samé seba, t.j. $a \Rightarrow^* a$ pre $a \in T$.
- Ak nájdeme generujúce symboly gramatiky, tak potom sa stačí pozrieť, či do nich patrí aj **počiatočný neterminál gramatiky**. Ak **áno**, tak to znamená, že sa z neho **dá odvodiť** nejaký reťazec terminálov, a teda **generovaný jazyk nie je prázdny**.
- Ak počiatočný neterminál gramatiky **nepatrí** medzi generujúce symboly, tak potom gramatika **určite negeneruje žiadne reťazce** a jazyk **je prázdny**.

Príklad

V gramatike

- $S \to AB \mid a$
- $A \to b$

sú generujúce symboly $\{a, b, S, A\}$, pretože pre ne platí, že sa z nich dajú odvodiť reťazce terminálov, napríklad

- $S \Rightarrow a$
- $A \Rightarrow b$
- $a \Rightarrow^* a$
- $b \Rightarrow^* b$

ale pre B neexistuje reťazec, ktorý by sa z neho dal odvodiť. Zároveň vidíme, že S **je generujúci symbol**, a teda generovaný jazyk určite **nie je prázdny**.

Algoritmus hľadania generujúcich symbolov

Algoritmus hľadania generujúcich symbolov, je rekurzívny a vyzerá nasledovne:

1. (Základný krok:) Každý terminálny symbol je určite generujúci, lebo na 0 krokov generuje sám seba.
2. (Rekurzívny krok:) Nech v pravidlách je pravidlo $A \to \alpha$, kde α je reťazec terminálov a neterminálov. Navyše, nech o **každom symbole** v α vieme, že je generujúci. Potom je aj A generujúci symbol. Navyše, ak je pravá strana pravidla ε, tak daný neterminál je taktiež generujúci, lebo aj ε je reťazec nad terminálmi.

Príklad

Nech gramatika $G = (V, T, P, S)$, kde $V = \{S, A, B\}$, $T = \{a, b\}$, S je počiatočný neterminál a pravidlá:

- $S \to AB$
- $A \to aBa \mid \varepsilon$
- $B \to bAb$

Pomocou algoritmu budeme hľadať generujúce symboly

1. Vždy platí, že terminály sú generujúce symboly, t.j. generujúce symboly sú $\{a, b\}$.
2. Z pravidla $A \to \varepsilon$ vidíme, že aj A je generujúci symbol, t.j. generujúce symboly sú $\{A, a, b\}$.
3. Keď teraz vieme, že $\{A, a, b\}$ sú určite generujúce symboly, tak v pravidle $B \to bAb$ sú na pravej strane všetky symboly generujúce. Teda aj B je generujúci symbol, čiže generujúce symboly sú určite $\{A, B, a, b\}$.
4. Keď teraz vieme, že $\{A, B, a, b\}$ sú určite generujúce symboly, tak v pravidle $S \to AB$ sú na pravej strane všetky symboly generujúce. Teda aj S je generujúci symbol, čiže generujúce symboly sú určite $\{S, A, B, a, b\}$.

Keďže do množiny generujúcich symbolov patrí aj počiatočný neterminál S, tak jazyk $L(G)$ určite **nie je prázdny**.

Príklad č. 2

Nech gramatika $G = (V, T, P, S)$, kde $V = \{S, A, B\}$, $T = \{a, b\}$, S je počiatočný neterminál a pravidlá:

- $S \to AB$
- $A \to aABa$
- $B \to bAb \mid b$

Pomocou algoritmu budeme hľadať generujúce symboly

1. Vždy platí, že terminály sú generujúce symboly, t.j. generujúce symboly sú $\{a, b\}$.
2. Keď vieme, že $\{a, b\}$ sú určite generujúce symboly, tak v pravidle $B \to b$ sú na pravej strane všetky symboly generujúce. Teda aj B je generujúci symbol, čiže generujúce symboly sú určite $\{B, a, b\}$.
3. Tu algoritmus končí, keďže pre neterminály S, A sú v pravidlách len také pravé strany, kde sa nachádza aspoň 1 symbol, ktorý nie je generujúci.
4. Teda množina generujúcich symbolov tejto gramatiky je $\{B, a, b\}$

Keďže do množiny generujúcich symbolov **nepatrí** počiatočný neterminál S, tak jazyk $L(G)$ je určite **prázdny** a táto gramatika negeneruje žiadne reťazce. Teda $L(G) = \emptyset$.

Problém $w \in L$?

- Pre bezkontextové jazyky (podobne ako pre regulárne) platí, že **existuje algoritmus**, ktorý dokáže zistiť, či $w \in L$, t.j. táto úloha je rozhodnuteľný problém.
- V prípade, že je bezkontextový jazyk daný zásobníkovým automatom, tak stačí dať na vstup ZA reťazec w a spracovať ho. Ak ZA reťazec w **akceptuje**, tak musí platiť, že $w \in L$, keďže L je práve jazyk akceptovaný ZA.
- V prípade, že je bezkontextový jazyk daný bezkontextovou gramatikou, existujú algoritmy (budete sa ich učiť na predmete Automaty a formálne jazyky), ktoré dokážu pre bezkontextovú gramatiku G a reťazec w zistiť, či $w \in L(G)$ (známy je napr. CYK algoritmus).

Problém $L_1 = L_2$?

- Pre bezkontextové jazyky (**na rozdiel od regulárnych**) **neplatí**, že **existuje algoritmus**, ktorý dokáže zistiť, či 2 bezkontextové jazyky L_1,L_2 popisujú ten istý jazyk, t.j. či $L_1 = L_2$.
- Rozhodovací problém ekvivalencie 2 bezkontextových jazykov je tzv. **nerozhodnuteľný problém**, t.j. dá sa ukázať, že **neexistuje** algoritmus, ktorý by pre 2 bezkontextové gramatiky, alebo pre 2 zásobníkové automaty, dokázal v konečnom čase dať odpoveď na otázku, či popisujú ten istý jazyk.

Problém $L_1 = L_2$?

- Problém ekvivalencie 2 bezkontextových jazykov je jedným z dôležitých príkladov **nerozhodnuteľných** problémov, t.j. problémv, pre ktoré **neexistuje** algoritmus, ktorý by ich vedel riešiť.
- Porovnajte si to so situáciu, kde ekvivalenciu 2 regulárnych jazykov sme riešiť **vedeli** pomocou algoritmu ekvivalencie stavov konečných automatov.

Iné nerozhodnuteľné problémy

Existuje niekoľko ďalších rozhodovacích problémov na bezkontextových jazykoch, ktoré sú nerozhodnuteľné, napríklad:

- Je prienik 2 bezkontextových jazykov prázdny jazyk? Inými slovami, ak sú dané 2 bezkontextové gramatiky, existuje taký reťazec, ktorý je generovaný oboma gramatikami? Alebo ak sú dané 2 zásobníkové automaty, existuje reťazec, ktorý je akceptovaný oboma automatmi?
- Je daný bezkontextový jazyk rovný Σ^*, kde Σ je jeho abeceda? Inými slovami, ak je jazyk daný bezkontextovou gramatikou, generuje gramatika úplne všetky možné reťazce? Ak je jazyk daný zásobníkovým automatom, akceptuje zásobníkový automat úplne všetky možné vstupné reťazce?

Použitá literatúra

1 Hopcroft, Motwani, Ullman - Introduction to Automata Theory, Languages and Computations, 3rd Ed.