Nedeterministický syntaktický analyzátor.

Thursday, May 30, 2013 8:40 AM

Při syntaktické analýze konstruujeme derivační strom. Podle toho, jak je konstruován derivační strom věty, rozlišujeme dvě základní metody syntaktické analýzy:

1. metoda shora dolů

- derivační strom konstruujeme od kořene k listům a zleva doprava (provádíme levou derivaci)

2. metoda zdola nahoru

- postupujeme od listů směrem ke kořeni, ale také zleva doprava (provádíme pravou derivaci)

K syntaktické analýze se využívají zásobníkové automaty (ZA), které jsou obecně nedeterministické (nepoužitelné pro SA). Pro konstrukci SA lze použít buď:

- Deterministickou simulaci nedeterministického ZA = algoritmus syntaktické analýzy s návraty.
- Zdokonalit konstrukci ZA tak, aby byl pro určitou třídu BKG deterministický (pohled do zásobníku nebo dále do vstupního řetězce).

Obecný popis nedeterminismu a determinismu

Základem **nedeterminismu** je tedy vždy problém **výběr správného pravidla**, ať už analýzou shora dolů nebo zdola nahoru. Pokud se nemá analyzátor podle čeho rozhodnout, prostě **prochází prostor všech řešení** buď do šířky nebo do hloubky (backtracking) a hledá to správné řešení. Tato metoda je tedy značně **neefektivní**, protože v nejhorším případě může projít všechny možnosti a nenajít žádné správné řešení, tedy správnou množinu pravidel, jejichž expanzí/redukcí lze dosáhnout požadovaného výsledku.

V případě, že chceme analyzovat vstup **deterministicky**, musíme analyzátor **zdokonalit**. Ten musí mít přesnou informaci o tom, jaké pravidlo gramatiky v danou chvíli použít. Automat může využít informaci o **dosud provedené částečné derivaci** a také o **vstupu**, který ještě nebyl zpracován. Zásobníkovému automatu, který je abstraktním modelem syntaktického analyzátoru, tedy připravíme rozkladovou tabulku (lookup table), ve které bude určeno, jaké pravidlo má analyzátor použít podle vstupu a stavu zásobníku.

Metoda shora dolů

Postup z přednášek:

A.

Konstrukce ZA, který je modelem syntaktické analýzy metodou shora dolů.

$\mathcal{P} = (\{q\}, T, N \cup T, \delta, S, \emptyset)$, kde δ je definováno takto:

1. $\delta(q, e, A) = \{(q, \alpha) : A \to \alpha \in P\}$ pro $\forall A \in N$,
2. $\delta(q, a, a) = \{(q, e)\}$ pro $\forall a \in T$.

Operaci 1. nazýváme expanzí (nahradí na vrcholu zásobníku a tím i ve větné formě neterminální symbol některou jeho pravou stranou).

Operaci 2. nazýváme srovnáním (čteného vstupního symbolu a symbolu z vrcholu zásobníku).

Tento ZA má vrchol zásobníku vždy vlevo.

Př. Zapsat $\mathcal{P}$ pro G[E] (na tabuli)

$\mathcal{P} = (\{q\}, \{(,),+,*,a\}, \{E,T,F,(,),+,*,a\}, \delta, q, E, \emptyset)$
$\delta(q, e, E) = \{(q, E+T), (q, T)\}$
$\delta(q, e, T) = \{(q, T*F), (q, F)\}$
$\delta(q, e, F) = \{(q, (E)), (q, a)\}$
$\delta(q,a',a') = \{(q, e)\}$ pro $\forall a' \in \{(,),+,*,a\}$

Např. zpracování věty a+a
Tady je vrchol zásobníku
$(q, a+a, E) \vdash (q, a+a, E + T) \vdash (q, a+a, T + T)$ to jsou expanze
$\vdash (q, a+a, F + T) \vdash (q, a+a, a + T)$ teď provedeme 2krát srovnání
$\vdash (q, +a, + T) \vdash (q, a, T)$ a opět expandujeme
$\vdash (q, a, F) \vdash (q, a, a)$ a naposledy srovnáme $\vdash (q, e, e)$

Zásobník je po přečtení vstupního řetězce prázdný, takže řetězec byl akceptován

Derivační strom konstruujeme od kořene (ohodnoceného startovním symbolem) dolů k listům, zleva doprava podle levé derivace. Jedná se o zásobníkový automat LL. Počáteční konfigurace automatu se dá popsat uspořádanou trojicí **(q, w, alfa)**, kde:

q =vnitřní stav
w = dosud nezpracovaní část vstupu
alfa = obsah zásobníku
Na počátku práce je automat v konfiguraci **(q_0, w, z_0)**, napr. (q, abaaab, s).

Pokud jen generujeme větu v gramatice, můžeme v případě více pravidel se stejnou levou stranou náhodně vybírat. Naším úkolem však bývá spíše analýza již existující věty. Zde již náhoda nepřipadá v úvahu, protože posloupnost pravidel pro levou derivaci již nemusí být jednoznačná. Potřebujeme automat, který tuto analýzu provádí, a tento automat musí mít možnost jednoznačně vybírat mezi pravidly to správné.

Existují dva postupy analýzy (nedeterministická a deterministická):

- **analýza s návratem** (nedeterministická)
  - postupně zkoušíme vhodná pravidla. Nejdřív první, pokračujeme dále ve výpočtu, a když se ukáže, že pravidlo nevyhovuje (dostaneme se do slepé uličky), vrátíme se zpátky a vyzkoušíme druhé pravidlo atd. Tato metoda je sice účinná. ale zbytečně pomalá.
  - Rekurzivní sestup.

- **deterministická analýza**
  - při výběru pravidla se řídíme dalšími informacemi. Může to být *pohled do budoucnosti*, kdy se díváme dále do vstupní posloupnosti symbolů a řídíme se tím, co později dostaneme na vstupu. Nebo například kontrolujeme obsah zásobníku (nestačí nám pouze vidět ten symbol, který ze zásobníku vyjímáme, ale i další, které jsou pod ním).
  - Prediktivní LL parser

Metoda zdola nahoru

Postup z přednášek:

Konstrukce ZA, který je modelem syntaktické analýzy metodou zdola nahoru.

$P = (\{q, r\}, T, N \cup T \cup \{\#\}, \delta, q, \#, \{r\})$, kde δ je definováno takto:

1. $\delta(q, a, e) = \{ (q, a) \}$ pro $\forall a \in T$,
2. $\delta(q, e, \alpha) = \{ (q, A) : A \to \alpha \in P \}$,
3. $\delta(q, e, \#S) = \{ (r, e) \}$.

Operaci 1. nazýváme přesun (přesun vstupního symbolu na vrchol zásobníku).

Operaci 2.nazýváme redukce (náhrada pravé strany pravidla na vrcholu zásobníku a tím i ve větné formě stranou levou).

Operace 3. je přijetí.

Tento ZA má vrchol zásobníku vpravo.
Konfiguraci budeme zapisovat ve tvaru: (stav, zásobník, vstup).
Zřetězením stavu zásobníku se zbytkem vstupu pak uvidíme jednotlivé větné formy

Př. Zapsat $\mathcal{P}$ pro G[E] (na tabuli)

- $\mathcal{P} = (\{q,r\}, \{(,),+,*,a\}, \{\#,E,T,F,(,),+,*,a\}, \delta, q, \#, r)$

  $\delta(q, a', e) = \{ (q, a') \}$ pro $\forall a \in T$,
  $\delta(q, e, E+T) = \{ (q, E) \}$
  $\delta(q, e, T) = \{ (q, E) \}$
  $\delta(q, e, T*F) = \{ (q, T) \}$
  …
  $\delta(q, e, \#E) = \{ (r, e) \}$

Např. zpracování věty a+a

Vrchol

$(q, \#, a+a) \vdash (q, \#a, +a) \vdash (q, \#F, +a) \vdash (q, \#T, +a)$
$\vdash (q, \#E, +a) \vdash (q, \#E+, a) \vdash (q, \#E+a, e) \vdash (q, \#E+F, e)$
$\vdash (q, \#E+T, e) \vdash (q, \#E, e) \vdash (r, e, e)$

ZA konstruované dle A. i B. jsou obecně nedeterministické (nepoužitelné pro SA). Pro konstrukci SA lze použít buď:

a) Deterministickou simulaci nedeterministického ZA = algoritmus syntaktické analýzy s návraty.
b) Zdokonalit konstrukci ZA tak, aby byl pro určitou třídu BKG deterministický.

Pozn.: Obsah zásobníku zřetězený se zbytkem vstupu je větnou formou.

- Konstruujeme derivační strom zdola od listů nahoru ke kořeni, přičemž postupujeme zleva doprava.

- Stejně jako u první metody i zde budeme používat lineární rozklad, tentokrát pro pravou derivaci - je to proces nalezení pravého rozkladu věty (LR gramatika).
- I zde musíme rozhodovat, která pravidla chceme použít. Tentokrát však nejde o pravidla se stejnou levou stranou (pro stejný neterminál), ale rozhodujeme se mezi pravidly, která mají podobnou pravou stranu a jsou proto použitelná pro tentýž podřetězec větné formy.

Řešíme to podobně jako u předchozí metody:

- **analýza s návratem** (nedeterministicky)
  - Vybereme ve větné formě jeden podřetězec (jako první vybíráme ten, který začíná nejvíc nalevo, je co nejdelší a je shodný s pravou stranou některého pravidla), přepíšeme neterminálem na pravé straně pravidla a pokračujeme v konstrukci derivačního stromu.
  - Pokud zjistíme, že tento krok nevede k úspěchu, vyzkoušíme jiný podřetězec atd (backtracking). Tato metoda je příliš časově náročná.
- **deterministická analýza (LALR, SLR)**
  - Využíváme další informace získané při překladu, např. obsah nepřečtené části vstupního kódu nebo obsah zásobníku.

ZA konstruované výše uvedenými metodami jsou obecně nedeterministické (nepoužitelné pro SA). Pro konstrukci SA lze použít buď:

a) Neterministickou simulaci nedeterministického ZA = algoritmus syntaktické analýzy s návraty.
b) Zdokonalit konstrukci ZA tak, aby byl pro určitou třídu BKG deterministický.

Pozn.: Obsah zásobníku zřetězený se zbytkem vstupu je větnou formou.

From <https://d.docs.live.net/e3534876709763a3/Dokumenty/ZCU/Statnice/Statnice.docx>