METODY TRANSLACJI 2004

Folie z wykładu

notatka nr 2

Nie zawsze Bottom-up shift-reduce Parsing jest możliwy

- Gramatyka:

  1. $S \to BAb$
  2. $S \to CAc$
  3. $A \to BA$
  4. $A \to a$
  5. $B \to a$
  6. $C \to a$

- Język: $aa^+b + aa^+c$.
- Automat ze stosem dla lewostronnego wyprowadzenia:

  - wrzuć wszystkie a na stos,
  - po przeczytaniu b (lub c) podaj 1,5 (odpowiednio 2,6),
  - użyj a ze stosu aby n razy podać 3, 5
  - podaj 4

Niemożność odtworzenia wyprowadzenia prawostronnego

Teza: żaden deterministyczny automat ze stosem nie zrekonstruuje wyprowadzenia prawostronnego dla tego języka

Idea:

prawostronne wyprowadzenie dla $a^{n+2}b$: 55^n43^n1
prawostronne wyprowadzenie dla $a^{n+2}c$: 65^n43^n2

dowód

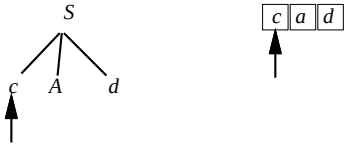
- jesli w trakcie czytania lter a byłby jakiś output, to można byłoby złośliwie dobrać c lub b na końcu
- liczba liter a's musi być zapamiętana przez zapis na stosie
- aby dać output 5^n trzeba wykorzystać stos (zdjąć a)
- ponieważ n zostało zapomniane, nie ma jak wygenerować outputu 3^n!

Parsing z backtracking

- Gramatyka:
  $S \rightarrow cAd$
  $A \rightarrow ab \mid a$
- input do parsowania: *cad*

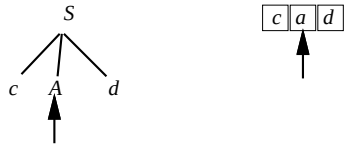
Przebieg parsingu

1. Krok: wybierz $S \rightarrow cAd$ jako pierwszą produkcję (jednoznacznie!)



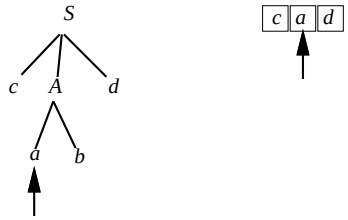
Przebieg parsingu

2. Krok: porównaj pierwszy liść i pierwszy symbol inputu
shift



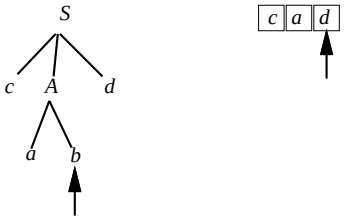
Przebieg parsingu

3. Krok: wybierz produkcję $A \rightarrow ab$, aby zastąpić A (nieprawidłowy wybór).



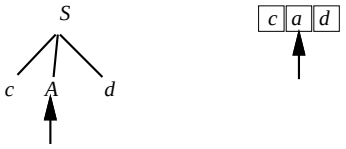
Przebieg parsingu

4. Krok: porównaj kolejny liść i kolejny symbol inputu
zgadza się, shift



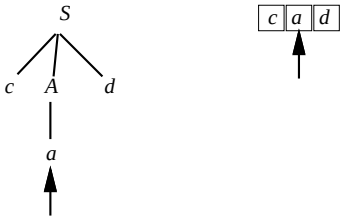
Przebieg parsingu

5. Krok: porównaj kolejny liść i kolejny symbol inputu
Nie zgadza się.
Cofamy ostatni wybór.



Przebieg parsingu

6. Krok: wybierz następną możliwą produkcję dla zastąpienia A: $A \rightarrow a$ (tym razem będzie ok).



Przebieg parsingu

7. Krok: porównaj ...
8. Krok: porównaj ...
9. Krok: akceptuj (brak dalszych liści i koniec inputu)

Unikanie backtrackingu

Przykład: `if ... then ... else`
`if ... then ...`

Ogólnie: $A \to \alpha\beta \,|\, \alpha\gamma$.

Backtracking niezbędny, bo po przeczytaniu α nie wiadomo, czy potem występuje β czy γ.

Lewostronna faktoryzacja

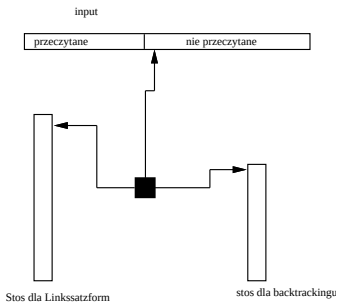
zamień $A \to \alpha\beta \,|\, \alpha\gamma$ na

$A \to \alpha A'$

$A' \to \beta \,|\, \gamma$

Implementacja parsera top-down z backtrackingiem

Automat z dwoma stosami, operacje `push` i `pop` na każdym stosie
Uwaga: tak samo silny model jak TM, czas obliczeń? (lewostronna rekursja!)



Konfiguracja parsera:

- lewy stos+ nieprzeczytana część inputu = Linkssatzform.
- prawy stos - lista produkcji odpowiadająca lewostronnemu wyprowadzeniu
- Konfiguracja: (s, i, β, α),
  s=stan (n =normal, b =backtracking, t =termination)
  i = pozycja w inpucie
  β = zawartość lewego stosu
  α = zawartość prawego stosu.

Kroki parsera

(a) rozszerzenie drzewa wyprowadzenia :

$$(n, i, \beta A, \alpha) \vdash (n, i, \beta\gamma_1, \alpha A_1)$$

gdzie $A \to \gamma_1$ pierwsza produkcja dla A,
(A_1 = ID tej produkcji).

(b) match :

$$(n, i, \beta a, \alpha) \vdash (n, i+1, \beta, \alpha a)$$

jeśli i-ty symbol inputu to a.

(c) koniec :

$$(n, i, \$, \alpha) \vdash (t, i, \$, \alpha)$$

jeśli i-ty znak inputu to znacznik końca.
$= dno stosu.

Kroki parsera

(c') wygenerowanie outputu :

$$(t, i, \$, \alpha A_j) \vdash (t, i, \$, \alpha)$$

z outputem A_j,

$$(t, i, \$, \alpha a) \vdash (t, i, \$, \alpha)$$

bez outputu, jeśli a to znak terminalny,

(d) niezgodność :

$$(n, i, \beta a, \alpha) \vdash (b, i, \beta a, \alpha)$$

jeśli i-ty znak inputu to nie a

Kroki parsera

(e) backtracking na znaku terminalnym :

$$(b, i, \beta a, \alpha a) \vdash (b, i-1, \beta a, \alpha)$$

(f) backtracking ze zmianą produkcji :

- wybierz następną produkcję dla A:

$$(b, i, \beta\gamma_j, \alpha A_j) \vdash (n, i, \beta\gamma_{j+1}, \alpha A_{j+1})$$

- halt, jeśli aktualna konfiguracja to $(b, 1, \beta\gamma_j, \alpha A_j)$, A to znak startowy, nie ma $j+1$-wszej produkcji dla A,
- w przeciwnym przypadku :

$$(b, i, \beta\gamma_j, \alpha A_j) \vdash (b, i, \beta A, \alpha)$$

Złożoność backtrackingu

gramatyka: $S \to aSS|\varepsilon$.

oznaczenia : $X(n) =$ liczba wypr. lewostronnych a^n,
$Y(n) =$ liczba wypr. lewostronnych zgodnych z a^n

własności :

$$\begin{aligned} X(0) &= 1 \\ X(n) &= \sum_{i=0}^{n-1} X(i) \cdot X(n-1-i) \\ Y(0) &= 2 \\ Y(n) &= Y(n-1) + \sum_{i=0}^{n-1} X(i) \cdot Y(n-1-i) \end{aligned}$$

wniosek : $X(n) \geq 2^{n-1}$ i stąd $Y(n) \geq 2^n$

Próba sparsowania $a^n b$ to eksponencjalnie wiele kroków.

Granica górna dla backtrackingu

Niech G gramatyka bezkontekstowa bez lewostronnej rekursji,
wtedy: istnieje stała c, taka że

$$A \stackrel{i}{\Rightarrow}_l wB\alpha$$

($wB\alpha$ wyprowadzona z A w i krokach wyprowadzenia lewostronnego), to

$$i \leq c^{|w|+2}$$

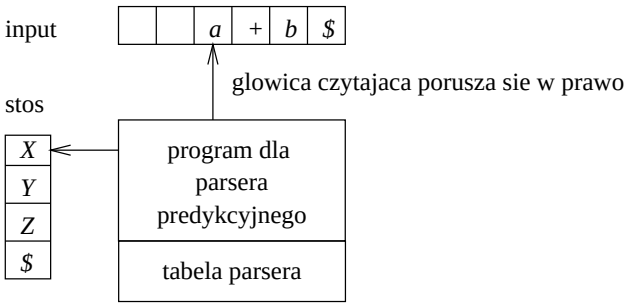
Eliminacja lewostronnej rekursji

jak w dowodzie normalnej postaci Greibacha dla gramatyk bezkontekstowych,

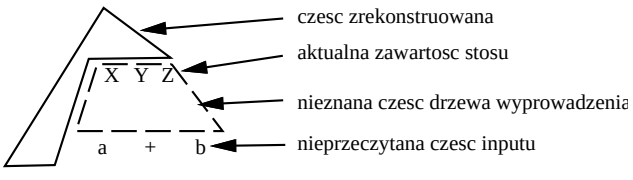
$$A_i \Rightarrow A_j$$

tylko dla $j > i$.

Parsery predykcyjne top-down



Konfiguracja parsera:



Stan rekonstruckji drzewa wyprowadzenia:

Sposób działania parsera predykcyjnego

a- następna czytana litera
X- symbol na wierzchu stosu

1.) $X = a = \$$ $\Rightarrow$ akceptuj
2.) $X = a \neq \$$ $\Rightarrow$ usuń X ze stosu, przesuń głowicę 1 pozycję na prawo
3.) X nieterminal
szukaj $M[X,a]$ w tabeli
1. $M[X,a]$ puste: `Syntax error`
2. $M[X,a]$ zawiera produkcję $X \to \alpha$,
zastąp X przez α na stosie, głowica stoi.

Konstrukcja tabeli

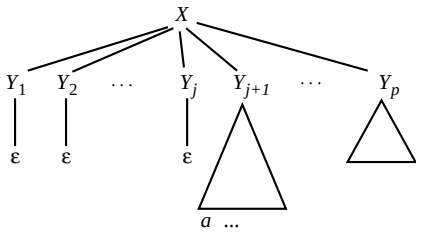
FIRST(α) = $\{ x \mid x$ symbol terminalny, $\exists\beta : \alpha \stackrel{*}{\Rightarrow} x\beta \}$
FOLLOW(A) = $\{ x \mid x$ znak terminalny, $\exists\alpha,\beta : S \stackrel{*}{\Rightarrow} \alpha A x \beta \}$
(α- dowowlne, A- nieterminal)

Obliczanie funkcji FIRST

1. X – symbol terminalny, to FIRST(X) = $\{X\}$
2. X – symbol nieterminalny, to wykonujemy następujące operacje w pętli, aż nic się nie zmienia:
   (a) $X \to a\alpha$, a symbol terminalny, to
   FIRST(X) := FIRST(X)$\cup$ $\{a\}$
   (b) $X \to \varepsilon$, to
   FIRST(X) := FIRST(X) $\cup\{\varepsilon\}$
   (c) $X \to Y_1 \ldots Y_p$ to ...

Obliczanie funkcji FIRST

przypadek: $X \to Y_1 \ldots Y_p$
jeśli $Y_1 \ldots Y_j$ nieterminalne i
$\forall i < j : \varepsilon \in$ FIRST(Y_i), to
FIRST(X):= FIRST(X) $\cup$ FIRST(Y_{j+1})
(gdy $j = p$ to FIRST(X):= FIRST(X) $\cup\{\varepsilon\}$)



FIRST dla słów

- FIRST(X) zdefiniowany, gdy X pojedynczy symbol,
- FIRST$(X\alpha)$ = FIRST(X), gdy $\varepsilon \notin$ FIRST(X),
- FIRST$(X\alpha)$ = FIRST$(X) \cup$ FIRST(α) w przeciwnym przypadku.

Obliczanie FOLLOW

1. S symbol startowy, to \$ $\in$ FOLLOW(S)

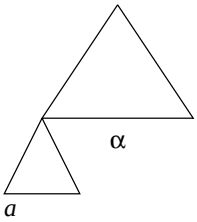
powtarzaj kroki 2 i 3 aż nic się nie zmienia

2. $A \to \alpha B \beta, \beta \neq \varepsilon$, to:
   FOLLOW(B) := FOLLOW(B) $\cup$ FIRST(β)
3. $A \to \alpha B$ lub
   $A \to \alpha B \beta$ i $\varepsilon \in$ FIRST(β):
   FOLLOW(B) := FOLLOW(B) $\cup$ FOLLOW(A)

Konstrukcja tabeli parsera

dla każdej produkcji $A \to \alpha$:

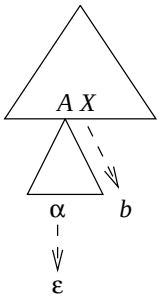
- $\forall a \in \Sigma$: jeśli $a \in$ FIRST(α), to wpisz $A \to \alpha$ do $M[A, a]$



Konstrukcja tabeli parsera

dla każdej produkcji $A \to \alpha$:

- jeśli $\varepsilon \in$ FIRST(α), to wpisz $A \to \alpha$ in $M[A, b]$
  dla każdego $b \in$ FOLLOW(A).



Problemy

- może się zdarzyć, że tabela zawiera więcej niż jeden wpis w określonym miejscu
- wtedy parser bezużyteczny! Zmienić gramatykę