METODY TRANSLACJI 2004

Folie z wykładu

notatka nr 3

Parsery bottom-up

- LR(0)
- LR(1)
- LR(k)
- sprytne uproszczenia

Parsery LR(0)

Notacja: LR(0)

- input czytany z **l**ewej na prawą,
- p**R**awostronne wyprowadzenie jest rekonstruowane,
- czytamy, **0** znaków inputu przed podjęciem decyzji

Tylko gramatyki prefixowe

$$w \in L_G \quad \Leftrightarrow \quad \text{żaden prefix } w \text{ należy do } L_G$$

(przy braku tej własności – po rozpoznaniu prefixu parser nie wiedziałby czy zaprzestać, bo nie wie, czy jest na końcu inputu)

nie jest to istotne ograniczenie:

L język bezkontekstowy $\Rightarrow$
$L' = \{w\$ \mid w \in L\}$ jest bezkontekstowy z własnością prefixów

Handle

- $S \overset{*}{\Rightarrow}_r SyAx \rightarrow y\alpha x$ jest wyprowadzeniem prawostronnym, to α jest handle

Poszukiwanie handle

sytuacja: α na stosie, x po prawej stronie głowicy. tj. $S \stackrel{*}{\Rightarrow}_r \alpha x$.
Kroki parsera:

- Redukcja, jeśli handle na szczycie stosu
- Shift w przeciwnym wypadku

Handle z αx to:

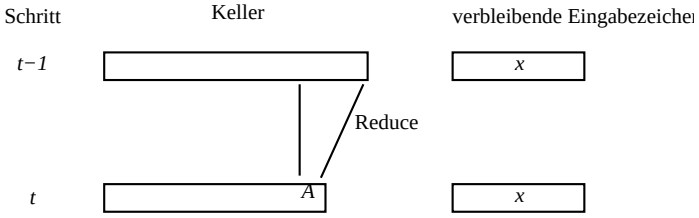
- suffix α, albo
- ciąg składający się częściowo z α i częściowo z x,
- podciąg x.

textbfLemat Handle nie leży w środku stosu.

Miejsca handle

Dowód t pierwszy moment gdy handle leży w środku stosu. rozważamy krok $t-1$:

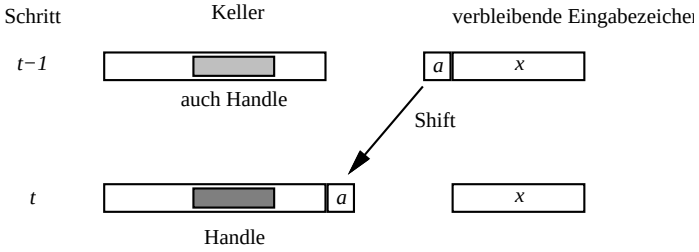
Przypadek 1:



znak A musi jako nieterminal należeć do handle! (gdyby, nie, to symbol A przeszedłby niezmieniony do chwili $t+1$, a to znaczyłoby że krok wyprowadzenia prawostronnego z chwili $t+1$ do chwili t zignorowałby A)

Miejsca handle

Przypadek 2:



Element

Jeśli $A \to \alpha\beta$ jest produkcją,
to $A \to \alpha \cdot \beta$ jest **elementem** (α, β mogą być puste)

Intuicje:
x to nieprzeczytana część inputu, α to zawartość stosu

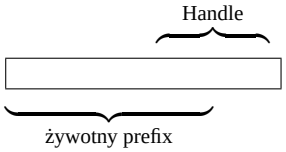
$$S \stackrel{*}{\Rightarrow}_r \alpha x$$

Handle β znajduje się częściowo w α i częściowo w x.
kropka odziela części i odpowiada kropce w $A \to \alpha \cdot \beta$

Żywotny prefix

niech $S \stackrel{*}{\Rightarrow}_r \alpha x$,
β - handle dla αx.

Wtedy prefix y ciągu αx jest **żywotny**, jeśli nie sięga za β.



Ważne elementy

Element $A \to \alpha \cdot \beta$ jest *ważny* dla żywotnego prefixu γ, jeśli istnieje wyprowadzenie prawostronne typu

$$S \stackrel{*}{\Rightarrow}_r \delta A w \Rightarrow_r \delta\alpha\beta w$$

gdzie $\gamma = \delta\alpha$ ($\alpha\beta$ to handle dla $\delta\alpha\beta w$).

Zastosowania ważnych elementów

Dla handle (i tylko dla handle) mamy element z punktem **na pierwszej z prawej pozycji** (tj. rightmost)!

Takie elementy nazywają się **zupełne**.

Rozpoznawanie handle = znalezienie ważnych elementów dla każdego żywotnego prefixa.

Automat skończony dla obliczania ważnych elementów

Stany: elementy i stan początkowy q_0,
automat niedeterministyczny

Funkcja przejścia:

- $\delta(q_0, \varepsilon) = \{S \to \cdot\alpha \mid S \to \alpha \text{ jest produkcją}\}$
- $\delta(A \to \alpha \cdot X\beta, X)$ zawiera $A \to \alpha X \cdot \beta$
  dla każdego X (terminal lub nieterminal)

  Uzasadnienie:
  niech $A \to \alpha \cdot X\beta$ będzie ważny dla γ.
  Tzn., istnieje $S \overset{*}{\Rightarrow}_r \delta Aw \Rightarrow_r \underbrace{\delta\alpha}_{\gamma} X\beta w$

  Wtedy:
  $S \overset{*}{\Rightarrow}_r \delta Aw \Rightarrow_r \underbrace{\delta\alpha X}_{\gamma X} \beta w$
  i element $A \to \alpha X \cdot \beta$ jest ważny dla γX.

Automat skończony dla obliczania ważnych elementów

Funkcja przejścia, cd:

- $\delta(A \to \alpha \cdot B\beta, \varepsilon)$ zawiera $B \to \cdot\eta \mid B \to \eta$ produkcja

  Uzasadnienie:
  niech $A \to \alpha \cdot B\beta$ ważny element dla γ,
  tj.
  $S \overset{*}{\Rightarrow}_r \delta Aw \Rightarrow_r \underbrace{\delta\alpha}_{\gamma} B\beta w$.

  wtedy też
  $S \overset{*}{\Rightarrow}_r \delta Aw \Rightarrow_r \delta\alpha B\beta w \overset{*}{\Rightarrow}_r \delta\alpha B\beta' w \Rightarrow_r \underbrace{\delta\alpha}_{\gamma} \eta\beta' w$.

  Tzn. $B \to \cdot\eta$ jest ważny dla γ.

Zamiana na automat deterministyczny

Konstrukcja potęgowa - ale “leniwa”, konstruujemy tylko te zbiory, które potrzebujemy

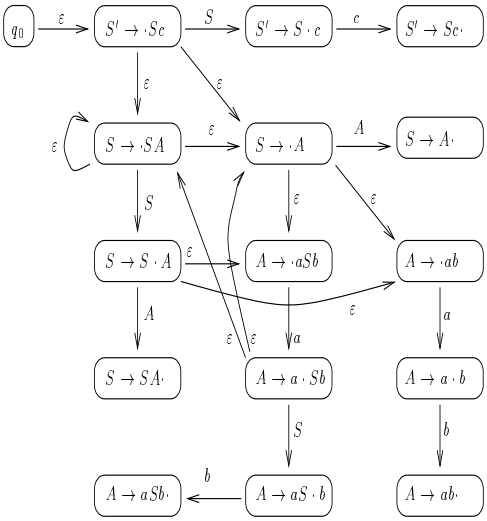
przykład: Gramatyka:

$\overline{S' \to Sc}$

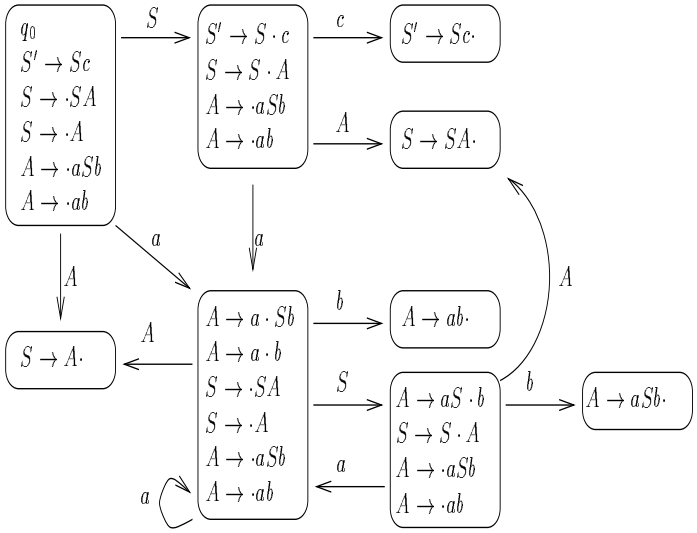
$S \to SA \mid A$

$A \to aSb \mid ab$

Automat niedeterministyczny



Automat deterministyczny



Twierdzenie

$\delta(q_0, \gamma)$ (stan po wczytaniu γ) zawiera

$$A \to \alpha \cdot \beta$$

wtedy i tylko wtedy gdy

$$A \to \alpha \cdot \beta$$

jest ważny dla γ.

($\Rightarrow$) oczywiste (na podstawie definicji).
($\Leftarrow$)
Trzeba pokazać, że ważny element jest osiągalny.
Niech $A \to \alpha \cdot \beta$ będzie ważny dla γ, tj.

$$S \overset{*}{\Rightarrow}_r \gamma_1 A w \Rightarrow_r \underbrace{\gamma_1 \alpha}_{\gamma} \beta w$$

Pokazujemy, że $\delta(q_0, \gamma_1)$ zawiera $A \to \cdot\alpha\beta$.

Wtedy kilka kroków (przesunięcie kropki) pokazuje, że $\delta(q_0, \gamma)$ zawiera element $A \to \alpha \cdot \beta$.

Indukcja po długości wyprowadzenia.

Wyprowadzenie $S \stackrel{*}{\Rightarrow}_r \gamma_1 A w$ przedstawiamy jako

$$S \stackrel{*}{\Rightarrow}_r \gamma_2 B x \Rightarrow_r \gamma_2\gamma_3 A\gamma_4 x \stackrel{*}{\Rightarrow}_r \underbrace{\gamma_2\gamma_3}_{\gamma_1} \underbrace{A}\ \underbrace{yx}_{w}$$

Z powodu wyprowadzenia
$S \stackrel{*}{\Rightarrow}_r \gamma_2 B x \Rightarrow_r \gamma_2\gamma_3 A\gamma_4 x$ element $[B \to \cdot\gamma_3 A\gamma_4]$ jest ważny dla γ_2.

Z indukcji $\delta(q_0, \gamma_2)$ zawiera także $[B \to \cdot\gamma_3 A\gamma_4]$.
Wtedy: $\delta(q_0, \gamma_2\gamma_3)$ zawiera $B \to \gamma_3 \cdot A\gamma_4$.
Po jednym kroku: $\delta(q_0, \gamma_2\gamma_3) = \delta(q_0, \gamma_1)$ zawiera $A \to \cdot\alpha\beta$.

Gramatyki LR(0)

Gramatyka bezkontekstowa nazywa się LR(0), jeśli:

1) Symbol początkowy S nie występuje po prawej stronie produkcji.

2) Dla każdego żywotnego prefixu γ mamy: jeśli $A \to \alpha\cdot$ jest ważny dla γ, wtedy

   – żaden inny element $C \to \delta\cdot$ nie jest ważny dla γ.
   – żaden inny element z symbolem terminalnym po prawej stronie kropki nie jest ważny dla γ.

(pomijamy zagadnienia, jakie występują gdy $\alpha = \varepsilon$)

Uwagi do definicji

definicja dopasowana do budowy parsera:

warunek 1) łatwy do osiągnięcia:
nowy symbol startowy: S', jedyna produkcja dla S' to $S' \to S$.

warunek 2.1) służy do unikania konfliktów przy parsingu. np. dwa ważne elementy $A \to \alpha\cdot$, $C \to \delta\cdot$ dla jednego γ wskazywałyby na dwie różne redukcje.

warunek 2.2) podobnie: $A \to \alpha\cdot$ i $B \to \beta\cdot c\delta$ dawałyby konflikt typu Shift-Reduce.

Testowanie na własność LR(0)

- konstruuj deterministyczny automat skończony, który oblicza ważne elementy,
- sprawdź czy żaden stan mie przeczy własnościom LR(0).

Konstrukcja parsera LR(0)

Niech G będzie gramatyką LR(0).

Twierdzenie Można skonstruować automat deterministyczny ze stosem rozpoznający L_G.

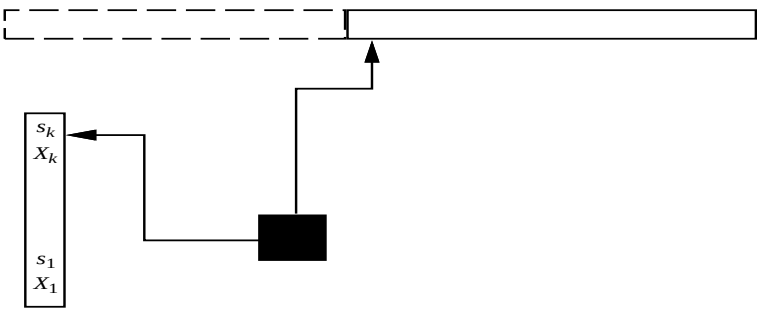
(ten automat nazywamy **LR(0)–parserem**.

Konstrukcja parsera LR(0)

akceptowanie parser akceptuje z pustym stosem.

zawartość stosu: $X_1 s_1 X_2 s_2 \ldots X_k s_k$, gdzie

- $X_1 X_2 \ldots K_k$ + nieprzeczytana część inputu = "Rechtssatzform"
- $X_1 \ldots X_k$ tworzy żywotny prefix,
- s_i jest stanem automatu, który oblicza ważne elementy, po przeczytaniu $X_1 \ldots X_i$.



Konstrukcja parsera LR(0)

Krok parsera:

Przypadek 1: Redukcja – s_k zawiera pełny element $(A \to X_{k-i} \dots X_k)$

W stosie znajduje się handle $X_{k-i} \dots X_k$

- usuń $X_{k-i} s_{k-i} \dots X_k s_k$ ze stosu
- przeczytaj s_{k-i-1}
- wpisz A na stos
- wpisz na stos stan $\delta(s_{k-i-1}, A)$ automatu dla $X_1 \dots X_{k-i-1} A$.

Przypadek 2: Shift – s_k nie zawiera pełnego elementu. Brak handle na stosie. Dalej wypełniamy:

- przeczytaj następny znak inputu i wpisz na stos jako X_{k+1},
- oblicz $s_{k+1} = \delta(s_k, X_{k+1})$ i wpisz na stos.

Poprawność parsera

Twierdzenie LR(0)-parser prawidłowo pracuje dla gramatyk LR(0).

- jeśli istnieje wyprowadzenie prawostronne, to jest ono rekonstruowane krok po kroku (brak niedeterminizmu!)
  Istotnie, jeśli $A \to \alpha\cdot$ i $B \to \gamma\cdot\rho$ jest ważny dla prefixu ω, wtedy pierwszy znak ρ to nieterminal. Wtedy element $B \to \gamma\cdot\rho$ nie może być użyty do Shift-u. Więc parser **musi** robić co trzeba.

Moc LR(0)

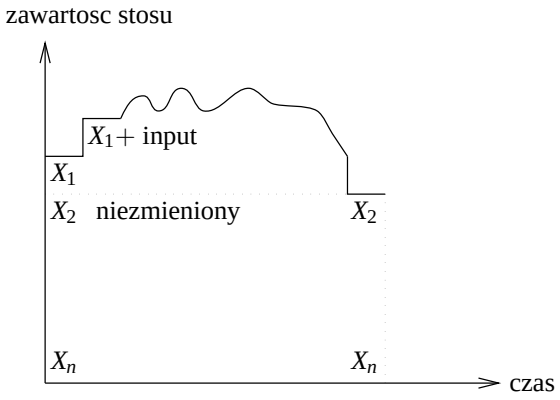
Twierdzenie
Jeśli L jest językiem bezkontekstowym, dla którego istnieje automat deterministyczny ze stosem, który akceptuje z pustym stosem, to

L ma gramatykę LR(0).

- wszystko co można zrobić, można zrobić za pomocą LR(0)
- nikt nie mówi, że łatwo skonstruować gramatykę,
  nikt nie mówi, że automat ma mało stanów

Przypomnienie - równoważność automatów ze stosem i CFL

Najważniejszy trick przy budowie gramatyki: stany postaci $[qXp]$.
Znaczenie: „wpisujemy X na stos w stanie q, usuwamy X przy stanie p“.



Przypomnienie - równoważność automatów ze stosem i CFL

Dla przejścia $(q,X) \to (q_1, X_1X_2 \ldots X_n)$ automatu ze stosem mamy produkcję:
$[q,X,q'] \to [q_1X_1q_2][q_2X_2q_3] \ldots [q_nX_nq']$
($q_1, q_2, \ldots, q_{n-1}$ są dowolne).

Konstrukcja gramatyki

niech Q zbiór stanów automatu M,
δ - funkcja przejścia M.

Stany nieterminalne:

- $[qXp]$ $\quad p,q \in Q$, gdzie X symbol ze stosu
- A_{qaY} $\quad q \in Q$, a to symbol inputu, Y symbol ze stosu

A_{qaY} gwarantować mają, że G_M jest LR(0).
Produkcje:

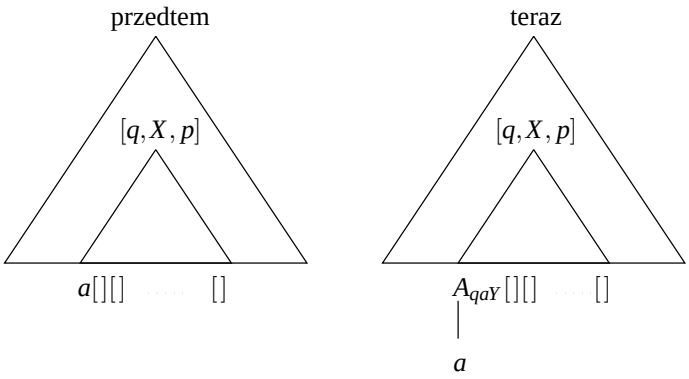
(1) $S \to [q_0Z_0p]$ $\quad$ dla każdego $p \in Q$

(2) $[qYp] \to A_{qaY}$,
jeśli $\delta(q,a,Y) = (p,\varepsilon)$

(3) $[qYq_{k+1}] \to A_{qaY}[q_1X_1q_2][q_2X_2q_3]\cdots[q_kX_kq_{k+1}]$,
jeśli $\delta(q,a,Y) = (q_1,X_1\ldots X_k)$ $\forall q_2,\ldots,q_k \in Q$

(4) $A_{qaY} \to a \quad \forall q,a,Y$

Szkic dowodu



Podstawowa własność

Ciąg kroków M, w trakcie których M czyta input,
odpowiada
ciągowi prawostronnych wyprowadzeń przy pomocy produkcji typu $A_{qaY} \to a$
(w odbiciu zwierciadlanym)

Powód: $[qYp] \to \underbrace{A_{qaY}}_{\text{później}} \underbrace{[qX_1p_1]\cdots[p_kX_{k+1}p]}_{\text{zostaje wcześniej zastąpione przez terminale}}$

- Zakładamy, że gramatyka nie ma zbędnych symboli.
- Dla $[qXp]$ niech w_{qXp} będzie ciąg terminali t.że: $[qXp] \stackrel{*}{\Rightarrow}_r w_{qXp}$

- $h(\alpha)$ to ciąg jaki można wyprowadzić z α:

  $h(A_{qaY}) = a$

  $h([qXp]) = w_{qXp}$

  $h(XY) = h(X)h(Y)$

- Funkcja N: $N(\gamma)$ jest liczbą kroków automatu, które odpowiadają wyprowadzeniu $\gamma \overset{*}{\Rightarrow}_r h(\gamma)$. Tj.

  $N(A_{qaY}) = 1$

  $N([qXp]) =$ liczba kroków dla w_{qXp}

- Funkcja m (ślad wyprowadzenia):

  $m(A_{qaY}) = (qaY)$

  $m([qXp]) =$ (ciąg indeksów A dla wyprowadzenia z w_{qXp})^R

  $m(XY) = m(X)m(Y)$

Podstawowa własność zdefiniowanej gramatyki

Jeśli $X \to \beta$ i $Y \to \delta$ są produkcjami postaci (2) oraz (3), wtedy δ nie jest **właściwym** podciągiem β.

Wynika to z tego, że A_{qaY} (pierwszy znak w β) koduje długość β.

Kierunek dowodu

Aby wykazać własność LR(0) musimy pokazać że jeśli $B \to \beta\cdot$ jest ważny dla żywotnego prefixu γ, to:

- żaden element $A_{qaX} \to \cdot a$ nie jest ważny dla γ,
- żaden element $C \to \alpha\cdot$ nie jest ważny dla γ.

Lemat Niech J = zbiór ważnych elementów dla żywotnego prefixu γ.
Jeśli $B \to \beta\cdot$ jest w J, to żaden $A_{qaY} \to \cdot a$ nie należy do J.

Dowód Lematu

Przypadek 1: $B \to \beta\cdot$ jest postaci $S \to [q_0 Z_0 p]\cdot$.

Wtedy $\gamma = [q_0 Z_0 p]$.

Jeśli $A_{qaY} \to \cdot a$ jest w J, to
$S \overset{*}{\Rightarrow}_r \gamma A_{qaY} y$.
Ale γ istnieje tylko po pierwszym kroku wyprowadzenia.

Dowód Lematu

Przypadek 2: $B \to \beta\cdot$ jest postaci $[qY p] \to A_{qaY}[][]\cdots[]\cdot$,

Wtedy $\gamma = \gamma'\beta$.
Załóżmy, że mamy wyprowadzenia:
$S \overset{*}{\Rightarrow}_r \gamma'\beta y$
$S \overset{*}{\Rightarrow}_r \gamma'\beta A_{qaY} y'$.
β może tylko być otrzymane z produkcji $B \to \beta$.
W momencie powstania β nie może A_{qaY} stać po prawej stronie B (prawostronne wyprowadzenie!).
Ale wtedy w następnym kroku używany ostatni nieterminal z β!

Dowód Lematu

Przypadek 3: $B \to \beta\cdot$ jest postaci $A_{pbZ} \to b\cdot$

- γA_{qaY} jest żywotnym prefixem, bo $A_{qaY} \to \cdot a$ jest ważny dla γ
- z drugiej strony $\gamma = \gamma' b$, gdyż $A_{pbZ} \to b\cdot$ jest ważny dla γ. Więc $\gamma A_{qaY} = \gamma' b A_{qaY}$ to żywotny prefix.

Dowód Lematu

- jeśli $b \neq \varepsilon$, to mielibyśmy terminal na lewo od nieterminala. Dla tej gramatyki to niemożliwe. Więc: $b = \varepsilon$.

- Rozważamy żywotne prefixy γA_{qaY}, $\gamma A_{p\varepsilon Z}$.

  Pokażemy, że automat nie będzie deterministyczny. Rozpatrujemy:
  $h(\gamma A_{qaY}) = h(\gamma)a$ i $h(\gamma A_{p\varepsilon Z}) = h(\gamma)$

  Rozważamy $N(\gamma)+1$ kroków M na $h(\gamma)a$:

– pierwsze $N(\gamma)+1$ kroków M na $h(\gamma)a = h(\gamma A_{p\varepsilon Z})a$ to $m(\gamma)m(A_{p\varepsilon Z}) = m(\gamma)(p,\varepsilon,Z)$.

– Z drugiej strony ponieważ $h(\gamma)a = h(\gamma A_{qaY})$, jest $N(\gamma)+1$ kroków w $m(\gamma A_{qaY}) = m(\gamma)m(A_{qaY}) = m(\gamma)(q,a,Y)$

Część 2 dowodu własności LR(0)

Lemat

Niech J będzie zbiorem ważnych elementów dla γ.

Jeśli $B \to \beta\cdot \in J$, to żaden inny element $C \to \alpha\cdot$ nie jest w J.

Dowód

Przypadek 1: $B \to \beta\cdot$, $C \to \alpha\cdot$ nie są elementami typu $A_{qaY} \to a\cdot$

- $\beta = A_{\ldots}[]\ldots[]$ – wtedy $A_{\ldots}$ wyznacza β
- $B \to \beta$ ma formę $S \to [q_0 Z_0 p]$ - łatwe ...jak poprzednio

Dowód

Przypadek 2: $B \to \beta\cdot$, $C \to \alpha\cdot$ są postaci $A_{qaY} \to a\cdot$.

Podprzypadki:

a) $\alpha = \beta$:

rozważamy $N(\gamma)+1$ kroków na $h(\gamma)a$

Wtedy $h(\gamma A_{q\alpha Y}) = h(\gamma)\alpha$ i $h(\gamma A_{q'\beta Y'}) = h(\gamma)\beta = h(\gamma)\alpha$

1. na $h(\gamma A_{q\alpha Y})$ pierwsze $N(\gamma)+1$ kroków to:
   $m(\gamma)m(A_{q\alpha Y}) = m(\gamma)(q\alpha Y)$
2. dla $h(\gamma A_{q'\beta Y'})$ pierwszych $N(\gamma)+1$ kroków to:
   $m(\gamma)m(A_{q'\beta Y'}) = m(\gamma)(q'\beta\ Y')$

niedeterminizm!

Dowód

b) $\alpha = \varepsilon$:

Rozważmy $N(\gamma)+1$ kroków na $h(\gamma)\beta$.

Rozważamy inputy:
$h(\gamma A_{q\varepsilon Y}) = h(\gamma)$ i $h(\gamma A_{q'\beta Y'}) = h(\gamma)\beta$

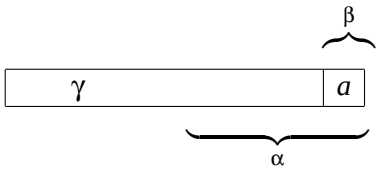
Dla $h(\gamma)\beta$:
z powodu $\gamma A_{q\varepsilon Y}$ obliczenie to $m(\gamma)(q\varepsilon Y)$.

z powodu $\gamma A_{q'\beta Y'}$ obliczenie to $m(\gamma)(q'\beta Y')$.

Dla tego samego inputu dwa różne obliczenia.

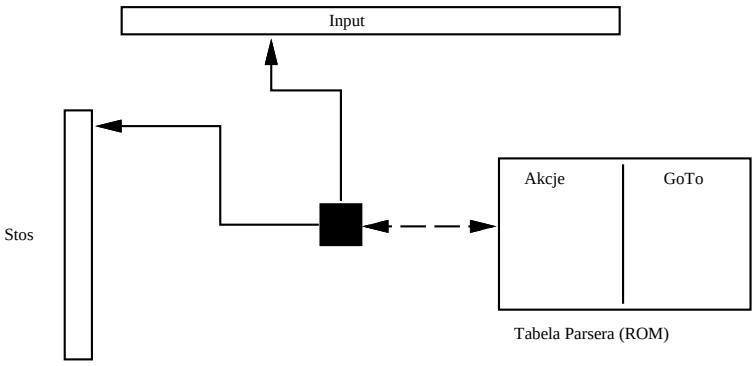
c) $\alpha = \beta = \varepsilon$ ....

Przypadek 3: $B \to \beta\cdot$ ma formę $A_{qaY} \to a\cdot$, ale $C \to \alpha\cdot$ nie.



Mielibyśmy terminale w α, co nie może się zdarzyć.

Inne parsery



Inne parsery

- Na stosie: $s_0 X_1 s_1 X_2 s \cdots X_m s_m$
- Konfiguracja: $(\underbrace{s_0 X_1 s_1 X_2 s \cdots X_m s_m}_{\text{stos}}, \underbrace{a_i a_{i+1} \cdots a_n \$}_{\substack{\text{nie przeczytana część} \\ \text{inputu}}})$
- Operacje

  ACTION(s_m, a), poprzez Look–up w tabeli parsera:

  1. `shift` s : wpisz a na stos, dodatkowo s
  2. `redukuj`: przez $A \to \beta$
  3. `akceptuj`
  4. `error`

  Nowa konfiguracja:

dla 1.) $(s_0 X_1 s_1 X_2 s_2 \cdots X_m s_m a_i s, a_{i+1} \cdots a_n \$)$
$s = \text{GOTO}(s_m, a_i)$,

dla 2.) $(s_0 X_1 s_1 X_2 s_2 \cdots X_{m-r} s_{m-r} A s, a_i \cdots a_n \$)$
$s = \text{GOTO}(s_{m-r}, A)$, gdzie $r =$ długość β

Simple–LR–Parsing Table (SLR)

Jeśli LR(0) dla danej gramatyki nie funktionuje, SLR pomaga uniknąć konfliktów

Idea: użyć następny symbol inputu do decyzji.

Przykład: mamy handle β, oraz γ,
nie wiadomo czy użyć $A \to \beta$ czy $B \to \gamma$
Czytamy następny znak c.
Jeśli np. $c \in \text{FOLLOW}(A)$ i $c \notin \text{FOLLOW}(B)$, to tylko $A \to \beta$ wchodzi w grę.

Definicja parsera SLR

Jak LR(0), ale używamy informacje:

- Jeśli $A \to \alpha\cdot$ należy do s_m, wtedy

  ACTION(s_m, a)= `zredukuj przy pomocy` $A \to \alpha$, jeśli $a \in$ FOLLOW(A).

- Jeśli $A \to \alpha \cdot a\beta$ jest w s_m, wtedy ACTION(s_m, a)= `shift`

Nie chodzi tylko o wczesne wykrywanie błędów!
Tabela SLR nie może zawierać żadnych konfliktów.
ACTION(s_m, a) musi być jednoznacznie zdefiniowana.

Przykład

$E' \to E, \quad E \to E + T \,|\, T, \quad T \to T * F \,|\, F, F \to (E) \,|\, \text{id}$

- nie jest LR(0):
  rozważ $E' \Rightarrow E$ oraz $E' \Rightarrow E \to E + T$.
  Dla prefixa E następujące elementy są ważne:
  $E' \to E\cdot, E \to E \cdot + T$.
  Konflikt typu shift-reduce.

- dla SLR: jak wygląda ACTION dla stanu
  $I = \{E' \to E\cdot, E \to E \cdot + T\}$?

  ACTION($I_1, \$$) = `akceptuj`

  ACTION($I_1, +$) = `shift`

  Unikamy konfliktu, bo FOLLOW(E') nie zawiera "+".

SLR nie zawsze wystarcza

Gramatyka:
$S \to L = R \,|\, R$
$L \to *R \,|\, id$
$R \to L$
Jest jednoznaczna!
Stan $I_2 = \{S \to L\cdot = R, R \to L\cdot\}$ opisuje ważne elementy dla prefixu L.

- zauważmy, że „=“ $\in$ FOLLOW(R)
  (bo $S \Rightarrow_r L = R \Rightarrow_r *R = R$)

- ACTION$(I_2, =)$ = `shift`, z powodu elementu $S \to L\cdot = R$

  ACTION$(I_2, =)$ = `reduce`, ponieważ $=$ należy do FOLLOW(R)

Konflikt! SLR-parser nie działa.

Kanoniczne tabele LR

Intensywniej używamy następnego znaku.

LR(1)–element jest postaci [LR(0)–element, symbol inputu].
Element $[A, \to \alpha\cdot\beta, a]$ nazywa się *ważny* dla prefixu γ, jeśli istnieje wyprowadzenie prawostronne takie że:

$$S \stackrel{*}{\Rightarrow}_r \delta A w \Rightarrow_r \delta\alpha\beta w$$

gdzie $\gamma = \delta\alpha$ i a jest pierwszym terminalem w (lub $w = \varepsilon$ oraz $a =$ \$).
Więc $a \in$ FOLLOW(A).
Uwaga: a **nie musi być następnym znakiem inputu!**

Przykład

Gramatyka: $S \to BB,\ B \to aB \mid b,$ tak więc $L = a^*ba^*b$

- $S \Rightarrow_r BB \Rightarrow_r BaB \Rightarrow_r aBab \Rightarrow_r aaBab \Rightarrow_r \underbrace{aaa}Bab,$
  więc $[B \to a\cdot B, a]$ jest ważny dla aaa.
- $S \Rightarrow_r BB \Rightarrow_r BaB \Rightarrow_r \underbrace{Baa}B,$
  więc $[B \to a\cdot B, \$]$ jest ważny dla Baa.
- Czy $[B \to a\cdot B, a]$ jest ważny dla Baa? Jeśli tak, to

  $S \stackrel{*}{\Rightarrow}_r BaBa... \Rightarrow_r BaaBa... \stackrel{*}{\Rightarrow}_r a\ldots aba\ldots aba\ldots.$
  Ale dla słów z L może drugie b stać tylko na końcu. Więc $[B \to a\cdot B, a]$ nie jest ważny dla Baa.

Idea obliczenia ważnych LR(1)–elementów

- niech $[A \to \alpha \cdot B, a]$ ważny LR(1)–element dla $\delta\alpha$, oraz

$$S \stackrel{*}{\Rightarrow}_r \delta A a x \Rightarrow_r \delta\alpha B a x \stackrel{B\to\eta}{\Rightarrow}_r \delta\alpha\eta a x$$

  Wtedy $[B \to \cdot\eta, a]$ jest ważny dla $\delta\alpha$.

Idea obliczenia ważnych LR(1)–elementów

- Ogólnie:
  niech $[A \to \alpha \cdot B\beta, a]$ ważny LR(1)–element dla γ,

$$S \stackrel{*}{\Rightarrow}_r \delta A a x \Rightarrow_r \delta\alpha B\beta a x \stackrel{*}{\Rightarrow}_r \delta\alpha B\beta' a x \Rightarrow_r \underbrace{\delta\alpha}_{\gamma}\eta\beta' a x,$$

  wtedy $[B \to \cdot\eta, y]$ jest ważny dla γ, gdzie y to pierwszy symbol $\beta' a x$. (β' zawiera tylko symbole terminalne.)

  Co z y?

  Przypadek 1: $\beta' = \varepsilon$, wtedy $y = a$

  Przypadek 2: $\beta' \neq \varepsilon$, wtedy

$$y \in \text{FIRST}(\beta' a x) = \text{FIRST}(\beta' a) \in \underbrace{\text{FIRST}(\beta a)}_{\text{już znane}}$$

Idea obliczenia ważnych LR(1)–elementów

- niech $[A \to \alpha \cdot X\beta, a]$ ważny dla γ.
  ponieważ $S \stackrel{*}{\Rightarrow}_r \delta A a x \Rightarrow_r \underbrace{\delta\alpha X}_{\gamma'}\beta a x$
  jest $[A \to \alpha \cdot X\beta, a]$ ważny dla γ'.

Procedura obliczania elementów LR(1)

PROCEDURE CLOSURE(I); (tranzytywne domknięcie)
REPEAT
$\forall$ elementu $[A \to \alpha \cdot B\beta, a] \in I$, produkcji $B \to \gamma$,
$b \in$ FIRST(βa) oraz $[B \to \cdot\gamma, b] \notin I$:
$I := I \cup [B \to \cdot\gamma, b]$
UNTIL nie ma zmian
output I

Procedura obliczania elementów LR(1)

PROCEDURE GOTO(I, X); (X to symbol z inputu.)
$I := \{[A \to \alpha X \cdot \beta, a] \mid [A \to \alpha \cdot X\beta, a] \in I\}$
output I

MAIN PART;
$C :=$ CLOSURE($\{[S' \to \cdot S] \mid S' \to S$ w gramatyce$\}$)
REPEAT dla każdego $I \in C, X$ t.że GOTO(I, X) $\neq \emptyset$
$C := C \cup \{$ CLOSURE(GOTO(I, X))$\}$
UNTIL nie ma zmian w C.

Sposób działania

tabela parsingu steruje parserem. Niech a - aktualny symbol z tasmy, I – symbol na wierzchu stosu. (I to zbiór ważnych elementów LR(1)).

- Jeśli $[A \to \alpha \cdot a\beta, b] \in I$, wykonaj Shift:
  - głowica na taśmie przesuwa się na prawo,
  - dwa nowe symbole na stos:
    a oraz CLOSURE(GOTO(I, a)).
- Jeśli $[A \to \alpha\cdot, a] \in I$, wtedy ACTION($I, a$) = reduce przy pomocy $A \to \alpha$
- Jeśli $[S' \to S\cdot, \$] \in I$, wtedy ACTION($I, S$) = akceptuj.

Gramatyki LR(1)

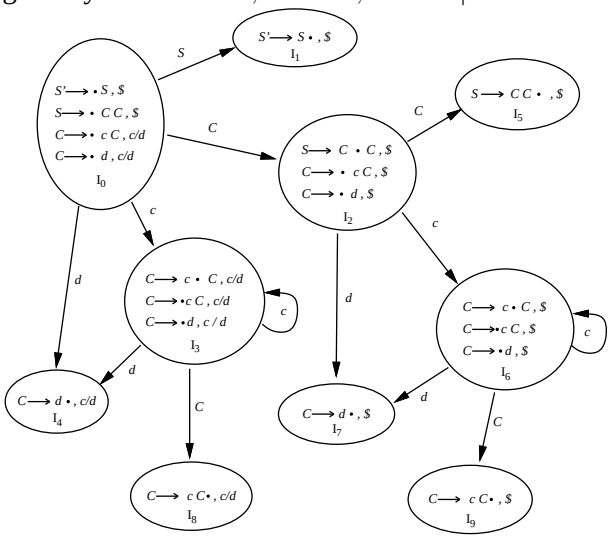
Gramatyka nazywa się LR(1), jeśli żadne konflikty nie występują w ACTION(). Typy konfliktów:

$$\left.\begin{array}{l}[A \to \alpha\cdot, b]\\ [A \to \beta\cdot, b]\end{array}\right\}$$ konflikt reduce–reduce

$$\left.\begin{array}{l}[A \to \alpha\cdot a\beta, c]\\ [B \to \beta\cdot, a]\end{array}\right\}$$ konflikt shift–reduce

Konstrukcja tabeli

gramatyka: $S' \to S, S \to CC, C \to cC|d$



Kanoniczna tabela parsingu

stan	ACTION			GOTO	
	c	*d*	$	*S*	*C*
0	s3	s4		1	2
1			ACC		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

- Każda gramatyka SLR jest LR(1).
  (ale LR(1)- parser może okazać się większy - niepotrzebnie)
- Nie każda LR(1) jest SLR.
- Praktycznie LR(1)- parsery są często wystarczające, choć nie każda gramatyka dCFL jest LR(1).
- LR(1)-tabela jest często za duża dla praktycznych zastosowań.

  $\Rightarrow$ Motywacja dla gramatyk i parserów pomiędzy SLR oraz LR(1).