Opracowanie do egzaminu
Teoria kompilacji i kompilatory 2020

Spis treści

Opracowanie na podstawie:

- Alfred V. Aho, Jeffrey Ullman, Monica S. Lam, Sethi Ravi Kompilatory - Reguły, Metody, Narzędzia, PWN 2019
- Wykłady z przedmiotu Teoria kompilacji i kompilatorów - w mniejszym stopniu

1 Gramatyki LL(k)

Pytanie

Czym jest funkcja $PRFX_k$?

Definicja

Dla pewnego języka $L(G)$ funkcja $PRFX_k(\alpha)$ określa zbiór prefiksów (łańcuchów terminalnych) symboli podstawowych o długości k, lub mniejszej, wyprowadzalnych z pewnego α i jest zdefiniowana następująco:

$$PRFX_k(\alpha) = \{x : (\alpha \overset{*,l}{\Longrightarrow} x\beta \wedge |x| = k) \vee (\alpha \overset{*,l}{\Longrightarrow} x \wedge |x| < k)\}$$

gdzie $x \in V^*$

Zwróćmy szczególnie uwagę na drugi człon definicji. Jeśli jesteśmy w stanie wyprowadzić słowo, które składa się tylko z terminali (koniec wyprowadzenia), to wówczas może ono mieć długość mniejszą od k, ale również należy do zbioru $PRFX_k$.

Pytanie

Czym jest funkcja $FOLLOW_k$?

Definicja

Dla pewnego języka $L(G)$ funkcja $FOLLOW_k(\beta)$ określa zbiór możliwych następników (kontynuacji) β i jest zdefiniowana następująco:

$$FOLLOW_k(\beta) = \{\omega : s \overset{*}{\Rightarrow} \alpha\beta\gamma \wedge \omega \in PRFX_k(\gamma)\}$$

Pytanie

Podaj definicję gramatyki LL(k).

Definicja

Jeżeli gramatyka G jest bezkontekstowa oraz dla każdych dwóch wywodów zachodzi:

$$S \overset{*,l}{\Longrightarrow} \omega A \alpha \overset{l}{\Longrightarrow} \omega \beta \alpha \overset{*,l}{\Longrightarrow} \omega x$$

$$S \overset{*,l}{\Longrightarrow} \omega A \alpha \overset{l}{\Longrightarrow} \omega \gamma \alpha \overset{*,l}{\Longrightarrow} \omega y$$

to gramatykę nazywamy gramatyką $LL(k)$ gdy

$$\text{jeżeli} \quad PRFX_k(x) = PRFX_k(y) \quad \text{to} \quad \beta = \gamma$$

gdzie $w, x, y \in V^*$, $\alpha, \beta, \gamma \in (N \cup V)^*$, $A \in N$.

Równoważnie możemy powiedzieć, że:

Gramatyka $LL(k)$ w wywodzie lewostronnym ma własność taką, że zawsze k pierwszych symboli słowa końcowego w sposób jednoznaczny określa regułę (produkcję) jaką należy zastosować.

Pytanie

Podaj definicję gramatyki LL(1).

W oparciu o pojęcia $PRFX_k$ i $FOLLOW_k$ możliwe jest także uproszczone zdefiniowanie gramatyki $LL(1)$.

Definicja

Bezkontekstowa gramatyka G jest $LL(1)$ wtedy i tylko wtedy, gdy $\forall A \to \alpha_1 | \alpha_2 | ... | \alpha_n$ zachodzi, że

1. $PRFX_1(\alpha_1) ... PRFX_1(\alpha_n)$ są parami rozłączne;
2. jeżeli $\alpha_i \overset{*}{\Rightarrow} \varepsilon$, to $PRFX_1(\alpha_j) \cap FOLLOW_1(A) = \emptyset$ dla $1 \leq j \leq n, i \neq j$.

Pytanie

Co możemy powiedzieć o iloczynie $PRFX_k(\beta\alpha) \cap PRFX_k(\gamma\alpha)$ w gramatykach LL(k) ?

Twierdzenie

Jeżeli $G =< N, V, P, S >$ jest gramatyką bezkontekstową, to G jest gramatyką $LL(k)$ wtedy i tylko wtedy, gdy dla każdej pary produkcji $(A \to \beta, A \to \gamma) \in P$ oraz dla $wA\alpha$ takiego że $S \overset{*,l}{\Longrightarrow} wA\alpha$, gdzie $w \in V^*$, $A \in N$ oraz $\alpha \in (N \cup V)^*$, zachodzi że:

$$PRFX_k(\beta\alpha) \cap PRFX_k(\gamma\alpha) = \emptyset$$

Pytanie

A tutaj takie jakieś pytanka ze slajdów.

Zakładając, że G jest gramatyką bezkontekstową pytamy:

- czy G jest $LL(k)$ dla danego k?
- czy G jest $LL(k)$ dla dowolnego k, tzn. czy $\exists k \in N : G$ jest $LL(k)$?
- czy $\exists G' : L(G') = L(G) \wedge G'$ jest $LL(k)$?

Pytania 2 i 3 są nierozstrzygalne.

Pytanie

Tutaj coś o gramatykach LL(k) związanego z determinizmem...

Twierdzenie

Istnieje język L' bezkontekstowy i zdeterminowany, taki że $\neg\exists G \in G_{LL(k)} : L' = L(G)$

Twierdzenie

$\exists L' \in L_{LL(k)} : \neg\exists G \in G_{LL(k-1)} : L' = L(G)$.

Można także zapisać: $G_{LL(k)} \not\supseteq G_{LL(k-1)}$ oraz $L_{LL(k)} \not\supseteq L_{LL(k-1)}$.

Pytanie

Co powoduje usunięcie ϵ produkcji dla gramatyki $G_{LL(k)}$?

Usunięcie ε-produkcji powoduje że $G_{LL(k)}$ staje się $G_{LL(k+1)}$.

Twierdzenie

Jeżeli gramatyka G jest $LL(k)$ bez ε-produkcji i $k \geq 2$, to istnieje G' będąca $LL(k-1)$ taka, że $L(G) = L(G')$.

2 Parsery LL(k)

Pytanie

Czym jest konfiguracja parsera LL(k) ?

$$(x, X\alpha, \Pi)$$

gdzie x – część wejścia, $X\alpha$ – stos z symbolem na wierzchołku (X) i resztą stosu (α), Π – ciąg numerów produkcji.

3 Parsery LR

Pytanie

Czy są prefiksy żywotne ?

Prefiksy/przedrostki żywotne (viable prefix)

- Prefiksy prawostronnych form zdaniowych, które mogą wystąpić na stosie analizatora redukującego (shift-reduce parser) nazywają się *prefiksami żywotnymi*.
- Prefiks żywotny to prefiks prawostronnej formy zdaniowej, który kończy się przed prawym końcem skrajnego prawego uchwytu tej formy zdaniowej.

Jednak nie każdy prefiks prawostronnej formy zdaniowej może pojawić się na stosie, gdyż parser nie może wykonać przesunięcia sięgającego poza uchwyt. Na przykład przyjmijmy, że

$$E \underset{rm}{\overset{*}{\Rightarrow}} F * \mathbf{id} \underset{rm}{\Rightarrow} (E) * \mathbf{id}$$

Wówczas w różnych momentach podczas analizy stos będzie przechowywać (, (E i wreszcie (E), ale nie może zawierać (E)*, gdyż (E) jest uchwytem, który parser musi zredukować do F przed przesunięciem $*$.

W powyższym przykładzie mamy dwie prawostronne formy zdaniowe. (E) jest prefiksem żywotnym, bo może pojawić się na stosie i jest prefiksem prawostronnej formy zdaniowej. $(E)*$ też jest prefiksem prawostronnej formy zdaniowej, ale nie może pojawić się na stosie dlatego nie jest prefiksem żywotnym.

Pytanie

Czym są sytuacje z jądra i sytuacje spoza jądra ?

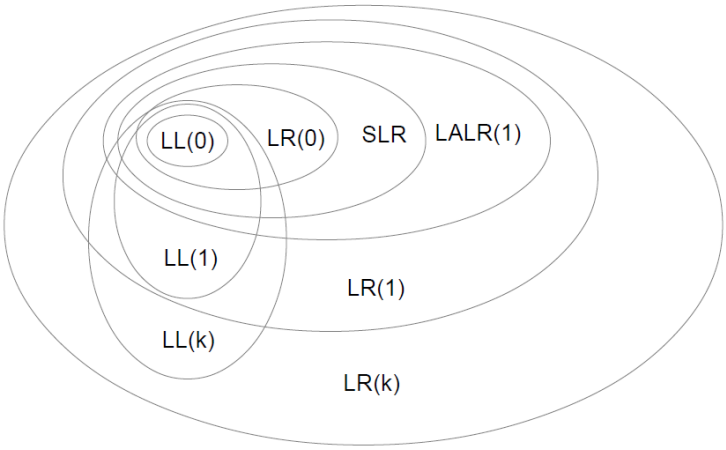
Podział zbioru sytuacji

- Jądro (kernel) zbioru sytuacji – wszystkie sytuacje, w których kropka nie jest na lewym krańcu oraz sytuacja startowa S' → •S
- Sytuacje spoza jądra (nonkernel) – wszystkie sytuacje, w których kropka jest na lewym krańcu poza sytuacją S' → •S

Pytanie

Jaka jest zależność pomiędzy gramatykami LL i LR ?

Zależność między gramatyki



Pytanie

Czym są sytuacje możliwe ?

Sytuacje możliwe

- Sytuacja $A \rightarrow \beta_1 \bullet \beta_2$ jest możliwa dla prefiksu żywotnego $\alpha\beta_1$ jeśli istnieje wyprowadzenie prawostronne $S' \Rightarrow^* \alpha Aw \Rightarrow^* \alpha\beta_1 \bullet \beta_2 w$
- Na ogół sytuacja jest możliwa dla wielu różnych prefiksów żywotnych

A tak mniej formalnie... Jak mamy DAS, który opisuje nam parser SLR, np. taki:

DAS

	E	T	F	+	*	(	)	id
> I0	I1	I2	I3			I4		I5
I1				I6				
I2					I7			
I3								
I4	I8	I2	I3			I4		I5
I5								
I6		I9	I3			I4		I5
I7			I10			I4		I5
I8				I6			I11	
I9					I7			
I10								
I11								

Przykład

- Gramatyka (wzbogacona):
  E' → E
  E → E + T | T
  T → T * F | F
  F → (E) | id
- Rozważmy prefiks żywotny E+T* - po jego przeczytaniu DAS będzie w stanie I7 zawierającym sytuacje: T→T*•F, F→•(E), F→•id (które są jedynymi możliwymi sytuacjami dla E+T*)

4 Wstęp do Canonical-LR i LALR

Pytanie

Z czego korzystają parsery CLR oraz LALR, z czego nie korzystały parsery SLR ?

Korzystają z jednego symbolu podglądanego na wejściu.

Pytanie

Z czego korzysta metoda kanonicznego LR ?

Metoda ta wykorzystuje symbole podglądane oraz wielki zbiór sytuacji, nazywany sytuacjami $LR(1)$.

Pytanie

Z czego korzysta metoda LALR ?

Metoda bazuje na zbiorze sytuacji $LR(0)$ i zawiera znacznie mniej stanów niż typowe parsery bazujące na sytuacjach $LR(1)$.

Dzięki starannemu dołączeniu "podglądania" do sytuacji $LR(0)$ możemy obsłużyć przy użyciu metody LALR znacznie więcej gramatyk niż w przypadku metody SLR, ale budowane tablice analizy nie są większe od tablic SLR. LALR jest najlepsza metoda do zastosowania w większości sytuacji.

Pytanie

Na co pozwala zastosowanie parsera Canonical LR ?

Możliwe jest przenoszenie w stanie większej ilości informacji, które to informacje pozwolą na wyeliminowanie niektórych spośród takich nieprawidłowych redukcji $A \to \alpha$. Przez podział stanów, jeśli to konieczne, możemy doprowadzić do układu, w którym każdy stan parsera LR wskazuje dokładnie, jaki symbol wejściowy może następować po uchwycie α, dla którego istnieje możliwa redukcja do A.

Czyli mówiąc mniej formalnie: może wystąpić sytuacja w której możliwa jest do zastosowania redukcja, która nie powinna być zastosowana. Parser CLR pozwala na podejrzenie jednego symbolu w przód i zdecydowanie czy redukcja jest rzeczywiście potrzebna.

Pytanie

Co jest dodatkowo przechowywaną informacją w stanach ? Czym są sytuacje LR(1) ?

Ta dodatkowa informacja jest dołączana do stanu przez przedefiniowanie sytuacji, aby uwzględniały one symbol terminalny jako drugi komponent. Ogólna forma sytuacji przybiera postać $[A \xrightarrow{\alpha} \cdot\beta, a]$, gdzie $A \xrightarrow{\alpha} \beta$ jest produkcja, natomiast a jest albo terminalem, albo prawym znacznikiem końcowym \$. Taki obiekt nazywamy sytuacja $LR(1)$.

Pytanie

Które sytuacje LR(1) są istotne z puntu widzenia procesu parsingu ?

Podgląd nie ma żadnego wpływu na sytuacje w postaci $[A \to \alpha \cdot \beta, a]$, gdy β

nie jest ϵ, ale sytuacja w postaci $[A \to \alpha\cdot, a]$ wywołuje redukcje według produkcji $A \to \alpha$ tylko wtedy, gdy kolejnym symbolem na wejściu jest a. Dokonujemy zatem redukcji według $A \to \alpha$ tylko dla tych

symboli wejściowych a, dla których $[A \to \alpha\cdot, a]$ jest sytuacja $LR(1)$ dopuszczalną dla stanu znajdującego się na wierzchołku stosu.

Pytanie

Jak skonstruować zbiór sytuacji LR(1) ?

Metoda budowy kolekcji zbiorów dopuszczalnych sytuacji $LR(1)$ jest zasadniczo identyczna jak budowa kanonicznej kolekcji zbiorów sytuacji $LR(0)$. Musimy jedynie zmodyfikować procedury $CLOSURE$ i $GOTO$.

Domknięcie(I)

```
function domkniecie(I)
begin
   repeat
      for każdej sytuacji [A → α•Bβ, a] z I,
         każdej produkcji B → γ z G',
         każdego terminala b z FIRST(βa) do
         dodaj [B → •γ, b] do zbioru I;
   until nie można dodać nowych sytuacji do I
   return I
end
```

Przejście(I,X)

```
function GOTO(I,X)
begin
   niech J będzie zbiorem sytuacji [A → αX•β,a]
   takich, że [A → α•Xβ,a] jest w I
   return domknięcie(J)
end
```

Pytanie

Jaki jest ogólny sposób na utworzenie tablicy metodą LALR ?

LALR(1)

- Dla DAS LR(1) istnieje wiele stanów, które mają taki sam pierwszy komponent sytuacji (sytuację LR(0)), a różnią się tylko podglądanym symbolem.
- Algorytm LALR(1) – bazuje na identyfikacji takich stanów i połączeniu ich symboli podglądanych.
  W przypadku sytuacji postaci [A→γ•,{...}], zbiór {…} jest na ogół mniejszy niż zbiór Follow

Konstrukcja DAS LALR(1) na bazie DAS LR(1)

- Identyfikujemy w DAS LR(1) wszystkie stany, które mają takie samo jądro i łączymy symbole podglądane w ramach poszczególnych sytuacji LR(0). Czyli sytuacja LALR(1) ma jako pierwszą składową sytuację LR(0), a jako drugą zbiór symboli podglądanych.

Pytanie

Jak w ogólności przebiega algorytm parsingu metodą LALR ?

Algorytm parsingu LALR(1)

- Algorytm LALR(1) jest taki sam jak w przypadku algorytmu LR(1)
- Gramatyka jest LALR(1) jeżeli tak określone reguły są jednoznaczne.
- Możliwe jest powstanie konfliktów w LALR(1), których nie było w LR(1) – konflikty reduce-reduce

5 Gramatyki operatorowe

Pytanie

Czym jest gramatyka operatorowa ?

Gramatyka operatorowa

- Gramatyka operatorowa właściwości:
  - Gramatyka bezkontekstowa, w której w żadnej prawej stronie produkcji nie ma dwóch sąsiadujących z sobą symboli nieterminalnych
  - Żadna prawa strona produkcji nie jest pusta
- Przykład
  - E → -E | (E) | E O E | id
  - O → + | - | * | / | ^
  - E → -E | (E) | E + E | E – E | E * E | E / E | E ^ E | id

Pytanie

Czym jest operator, a czym operand w gramatykach operatorowych ?

Gramatyka operatorowa

- W gramatykach operatorowych przyjmuje się następujące nazewnictwo:
  - operator – określa symbol terminalny
  - operand – określa symbol nieterminalny

Pytanie

Jakie relacje pierwszeństwa wyróżnia się w gramatykach operatorowych ?

Relacje pierwszeństwa (precedence relations)

- Dla par symboli terminalnych określa się trzy rozłączne relacje pierwszeństwa:

relacja	znaczenie
a <• b	a ma niższy priorytet niż b
a $\doteq$ b	a ma taki sam priorytet jak b
a •> b	a ma wyższy priorytet niż b

Pytanie

Czym jest gramatyka z pierwszeństwem operatorów ?

Gramatyka z pierwszeństwem operatorów

- Gramatyka G=(V, T, P,S) $\in$ GBK jest gramatyką z pierwszeństwem operatorów <=>
  - G – jest gramatyką operatorową
  - Dla każdej pary terminali (operatorów) zachodzi co najwyżej jedna z relacji: <• , $\doteq$, •>

Relacje pierwszeństwa

- Mimo podobieństwa relacji pierwszeństwa do relacji arytmetycznych: <, =, >
  mają one zupełnie inne własności
- Na przykład:
  - Dla tego samego języka może równocześnie zachodzić: a <• b i a •> b
  - Dla pewnej pary terminali a i b może nie zachodzić żadna relacja pierwszeństwa

Pytanie

Jakie znasz dwie podstawowe metody ustalenia relacji pierwszeństwa dla operatorów ?

Ustalenie relacji pierwszeństwa

- Istnieją dwie podstawowe metody określania relacji pierwszeństwa dla operatorów:
  - Metoda intuicyjna – oparta na tradycyjnych pojęciach łączności i priorytetach operatorów
  - Metoda oparta na tworzeniu gramatyki jednoznacznej dla języka

Pytanie

Jak znajdujemy uchwyt w metodzie pierwszeństwa operatorów ?

Przykład

	id	+	*	$
id		•>	•>	•>
+	<•	•>	<•	•>
*	<•	•>	•>	•>
$	<•	<•	<•	

- id + id * id + id
- $ <• id •> + <• id •> * <• id •> + <• id •>$
- Procedura znalezienia uchwytu:
  - Przeglądanie łańcucha od lewego krańca do pierwszego •>
  - Przeglądanie w kierunku lewego krańca do symbolu <•
  - Uchwyt zawiera wszystko pomiędzy symbolami <• i •> **włączając w to wszystkie zawarte lub otaczające nieterminale**