Wprowadzenie

Języki formalne i techniki translacji - Wykład 1

Maciek Gębala

8 października 2019

Maciek Gebala

Vprowadzenie

Literatura

- A.V. Aho, R. Sethi, J.D. Ullman, Kompilatory. Reguly, metody i narzędzia, WNT, Warszawa 2002, (ISBN: 83-204-2656-1)
- J.E. Hopcroft, J.D. Ullman, Wprowadzenie do teorii automatów, języków i obliczeń, WNT, Warszawa 1994 (ISBN 83-01-11298-0)
- T.A. Sudkamp, Languages and Machines, Pearson, 2006, (ISBN: 978-81-317-1475-1)

Maciek Gębal

Wprowadzeni

Proces translacji

Kompilator

Program czytający kod napisany w języku źródłowym i tłumaczący go na równoważny kod w języku wynikowym.

 $\begin{array}{ccccc} \operatorname{Program} & & \operatorname{Program} \\ \operatorname{\acute{z}r\acute{o}dlowy} & \longrightarrow & \operatorname{Kompilator} & \longrightarrow & \operatorname{wynikowy} \\ \downarrow & & \downarrow \\ & \operatorname{Komunikaty} \circ \operatorname{blędach} & & \end{array}$

Maciek Gebala

Wprowadzenie

Kompilacja typu analiza-synteza

- Analiza rozłożenie programu na części składowe i stworzenie jego pośredniej reprezentacji.
- Synteza przekształcenie reprezentacji pośredniej na program wynikowy.

Notatki
Notatki
Notatki
Notatki

Przykłady narzędzi dokonujących analizy	Notatki
Edytory strukturalneFormatory kodu programu	
Kontrolery statyczneInterpretery	
,	
Maciek Gębala Wprowadzenie	
Przykłady programów przypominających kompilatory	Notatki
a Formatowy tokaty	
Formatory tekstuKompilatory do układów scalonych	
Interpretery zapytań	
Maciek Gębala Wprowadzenie	
	Notatki
Kontekst kompilatora	Notatki
Kontekst kompilatora Szkieletowy program źródłowy	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor	
Kontekst kompilatora Szkieletowy program źródłowy	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy	
Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki	
Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator Biblioteki Bezwzględny kod maszynowy	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki Bezwzględny kod maszynowy	
Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator Bezwzględny kod maszynowy Wyrowadzenie Fazy kompilatora	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki Bezwzględny kod maszynowy Marcek Captala Wprowadzenie Fazy kompilatora	
Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator Bezwzględny kod maszynowy Worwadzenie Fazy kompilatora Program źródłowy Analizator leksykalny	
Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator Bezwzględny kod maszynowy Worowschenie Fazy kompilatora Program źródłowy Analizator leksykalny Analizator semantyczny Analizator semantyczny Obsługa	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki Bezwzględny kod maszynowy Tazy kompilatora Program źródłowy Analizator leksykalny Analizator składniowy Zarządzanie Analizator składniowy Zarządzanie Analizator kodu pośredniego Generator kodu pośredniego	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki Bezwzględny kod maszynowy Tomora Coptala Wypowedzenie Fazy kompilatora Program źródłowy Analizator leksykalny Analizator składniowy Zarządzanie ← Analizator semantyczny → Obsługa blędów Symboli ← Generator kodu pośredniego Optymalizacja kodu	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program źródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator ← Biblioteki Bezwzględny kod maszynowy Macoek Optilatora Program źródłowy Analizator leksykalny Analizator semantyczny Analizator semantyczny Optymalizacja kodu Generator kodu pośredniego Optymalizacja kodu Generator kodu Generator kodu	
Kontekst kompilatora Szkieletowy program źródłowy Preprocesor Program zźródłowy Kompilator Program w asemblerze Assembler Przemieszczalny kod maszynowy Konsolidator Biblioteki Bezwzględny kod maszynowy Konsolidator Biblioteki Bezwzględny kod maszynowy Konsolidator Assembler Przemieszczalny kod maszynowy Konsolidator Assembler Program źródłowy Analizator leksykalny Analizator leksykalny Zarządzanie Analizator semantyczny Obsługa blędów Symboli Generator kodu pośredniego Optymalizacja kodu Generator kodu	

Fazy kompilatora Notatki Analiza leksykalna Ciąg znaków składający się na program źródłowy jest przekształcany w ciąg tokenów (symboli leksykalnych). Grupowanie symboli leksykalnych programu źródłowego w wyrażenia gramatyczne (tworzenie drzewa wyprowadzenia). Analiza semantyczna Kontrola programu pod względem poprawności semantycznej (np. kontrola typów) i zbieranie informacji do generowania kodu. Fazy kompilatora Notatki Zarządzanie tablicą symboli Zapamiętywanie identyfikatorów używanych w programie źródłowym i zbieranie informacji o różnych atrybutach tych identyfikatorów. Wykrywanie i zgłaszanie błędów Obsługa błędów w taki sposób aby nie przerywać kompilacji po pierwszym błędzie. Błędy wykrywane są w fazie analizy. Fazy kompilatora Notatki Generowanie kodu pośredniego Reprezentacja programu dla pewnej abstrakcyjnej maszyny (łatwa do utworzenia i tłumaczenia na program wynikowy). Optymalizacja kodu Poprawienie kodu pośredniego w taki sposób aby kod maszynowy działał szybciej. Generowanie kodu wynikowego Generowanie kodu wynikowego, najczęściej w asemblerze. Grupowanie faz Notatki Fazy zależne przede wszystkim od języka źródłowego i praktycznie niezależne od języka wynikowego. Zwykle składa się z analizatora leksykalnego, składniowego i semantycznego oraz tablicy symboli, generatora kodu pośredniego i obsługi błędów. Tył kompilatora Fazy zależne od maszyny docelowej a niezależne od języka źródłowego. Zwykle składa się z optymalizacji i generowania kodu z tablicą symboli i obsługą błędów.

Języki formalne Notatki Podstawowe definicje Alfabet (Σ) – skończony zbiór symboli. Słowo – skończony ciąg symboli z alfabetu. Język – zbiór słów nad danym alfabetem (∅ - język pusty). Deterministyczny Automat Skończony (DFA) Notatki Deterministyczny automat skończony to uporządkowana piątka $(Q, \Sigma, \delta, q_0, F)$ gdzie Q – skończony zbiór stanów, Σ – skończony alfabet wejściowy, δ – funkcja przejścia postaci $extbf{Q} imes extbf{Q} o extbf{Q},$ q0 - stan początkowy, $F \subseteq Q$ – zbiór stanów akceptujących. Notacja δ możemy rozszerzyć do $\hat{\delta}: Q \times \Sigma^* \to Q$ zgodną z definicją rekurencyjną $\hat{\delta}(q,\varepsilon) = q$ i $\hat{\delta}(q,wa) = \delta(\hat{\delta}(q,w),a)$. Automat akceptuje słowo w jeśli $\hat{\delta}(q_0, w) \in F$. Przykład Notatki $\textit{M} = (\{\textit{q}_0, \textit{q}_1, \textit{q}_2, \textit{q}_3\}, \{0, 1\}, \delta, \textit{q}_0, \{\textit{q}_0\})$ 0 **q**₀ **q**₂ q_1 q_1 $q_3 q_0$ q_2 q_0 q_3 Rysunek i analiza na tablicy Niedeterministyczny Automat Skończony (NFA) Notatki Istnieje zero, jedno lub więcej przejś

symbolu wejściowym.

Automat niedeterministyczny akcept odpowiadający mu ciąg przejść ze s akceptującego.

ć ze stanu przy tym samym		 	 	
$\Gamma o 2^Q$		 	 	
tuje słowo w jeżeli istnieje		 	 	
tanu początkowego do stanu	J	 	 	
Wprowadzenie		 	 	

Przykład

$$\begin{split} M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\}) \\ & \frac{\delta \quad | \quad 0 \quad 1}{q_0 \quad \{q_0, q_3\} \quad \{q_0, q_1\}} \\ q_1 \quad \emptyset \quad \{q_2\} \\ q_2 \quad \{q_2\} \quad \{q_2\} \\ q_3 \quad \{q_4\} \quad \emptyset \\ q_4 \quad \{q_4\} \quad \{q_4\} \end{split}$$

Rysunek i analiza na tablicy



Niedeterministyczny Automat Skończony z ε-ruchami (NFA_{ε})

Dopuszczamy przejścia między stanami bez symboli wejściowych.

$$\delta: \mathbf{Q} \times (\mathbf{\Sigma} \cup \{\varepsilon\}) \rightarrow \mathbf{2}^{\mathbf{Q}}$$

Wyrażenia regularne (RE)

Niech L, L_1, L_2 – języki nad alfabetem Σ . Wtedy

$$L_1L_2 = \{ xy : x \in L_1 \land y \in L_2 \}$$
 (złożenie, konkatenacja)

$$L^0 = \{\varepsilon\}$$

$$L^{i+1} = LL^i \quad dla i > 0$$

$$L^{+1} = LL^{i}$$
 dla $i > 0$

$$L^{*} = \bigcup_{i=1}^{\infty} L^{i}$$
 (domknięcie Kleene'ego) $\left(L^{+} = \bigcup_{i=1}^{\infty} L^{i}\right)$

Definicja

- $\begin{tabular}{ll} \hline \bullet & Wyrażenia regularne \emptyset, ε, a reprezentują odpowiednio język \\ \hline \hline \end{tabular}$ pusty, $\{\varepsilon\}$, $\{a\}$ dla $a \in \Sigma$.
- 3 Jeżeli r i s są wyrażeniami regularnymi reprezentującymi odpowiednio języki R i S to
 - (r+s) reprezentuje język $R \cup S$, (rs) reprezentuje język RS,

 - r* reprezentuje język R*.

Przykład

$$(0+1)^*(00+11)(0+1)^*$$

$$(\varepsilon+0+00)[(1+11)(0+00)]^*(\varepsilon+1+11)$$

otatki
otatki
otatki
otatki