Własności języków regularnych. Analiza leksykalna

Języki formalne i techniki translacji - Wykład 3

Maciek Gębala

22 października 2019

Notatki

Własności języków regularnych

Lemat o pompowaniu

Niech L będzie językiem regularnym. Wtedy istnieje stała n t.że jeśli z jest dowolnym słowem z L oraz $|z| \geqslant n$, to z możemy przedstawić w postaci $z = uvw$, gdzie $|uv| \leqslant n$ i $|v| \geqslant 1$ oraz $uv^i w$ należy do L dla każdego $i \geqslant 0$.

n jest nie większe niż liczba stanów najmniejszego DFA akceptującego L.

Dowód

Na tablicy.

Notatki

Wykorzystanie Lematu o pompowaniu

Dowodzenie że L nie jest regularny

1. Załóż, że L jest regularny i istnieje odpowiednie n.
2. Wybierz słowo z zgodnie z lematem (jego długość musi zależeć od n).
3. Pokaż, że dla każdego podziału z zgodnego z lematem istnieje i takie, że $uv^i w \notin L$.

Przykład

$$L = \{ 0^{i^2} \ : \ i \in N \}$$

Załóżmy, że L jest regularny i weźmy n z Lematu o pompowaniu.
Weźmy $z = 0^{n^2}$.
$z = uvw$ i $|uv| \leqslant n$ oraz $|v| \geqslant 1$. Weźmy $i = 2$. Wtedy mamy

$$n^2 < |uv^2w| = |uvw| + |v| \leqslant n^2 + n < (n+1)^2,$$

czyli $uv^2w \notin L$. L nie jest regularny.

Notatki

Własności języków regularnych

Lemat. Klasa języków regularnych jest zamknięta na operację sumy, dopełnienia, przecięcia, złożenia i domknięcia Kleene'ego.

Dowód

Suma, złożenie i domknięcie Kleene'ego: z definicji RE.
Dopełnienie: jeśli L akceptowany przez DFA $M = (Q, \Sigma, \delta, q_0, F)$ to $\overline{L}$ akceptowany przez $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$.
Przecięcie: Jeśli L_1 i L_2 akceptowane przez odpowiednie DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ i $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, to $L_1 \cap L_2$ akceptowany przez $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$, gdzie $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$.
Suma: $M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), (Q_1 \times Q_2) \setminus ((Q_1 \setminus F_1) \times (Q_2 \setminus F_2)))$.

Własności języków regularnych

Lemat. Zbiór słów akceptowanych przez DFA M o n stanach jest
1. niepusty $\iff$ M akceptuje słowo o długości mniejszej niż n;
2. nieskończony jeśli M akceptuje słowo o długości l, dla $n \leqslant l < 2n$.

Lemat. Istnieje algorytm rozstrzygający czy dwa automaty skończone są równoważne (akceptują te same języki).

Dowód

Weźmy M_1 i M_2 DFA akceptujące odpowiednio języki L_1 i L_2. Jeśli $L_1 \neq L_2$ to $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ niepusty.

Notatki

Analiza leksykalna

Rozbicie ciągu znaków wejściowych na symbole leksykalne (wyrazy posiadające określone znaczenie). Ciąg symboli leksykalnych stanowi wejście dla analizatora składniowego.

Podstawowe pojęcia

- Symbol leksykalny (token)
- Leksem (symbol leksykalny może mieć wiele leksemów)
- Wzorzec

Notatki

Przykład symboli leksykalnych

```
double sqr(double x)
{
    return x*x;
}
```

Token	Leksem
Identyfikator_typu	double
Identyfikator	sqr
'('	(
Identyfikator	x
')'	)
'{'	{
KW_return	return
Operator_binarny	*
';'	;
'}'	}

Notatki

Zapis wzorca

- Wzorce zapisujemy jako wyrażenia regularne.
- Składnia wyrażeń rozszerzona aby umożliwić zwięzły zapis.
- W opisie przez wyrażenia regularne używamy następujących priorytetów: gwiazdka Kleene'go, złożenie, suma.

Przykład

Symbol leksykalny	Wyrażenie regularne
Identyfikator	`[a-zA-Z_][a-zA-Z0-9]*`
'('	`\(`
'{'	`\{`
Operator_binarny	`\*`
KW_return	`return`

Notatki

Implementacja analizatora leksykalnego

- Wykorzystanie generatorów analizatorów leksykalnych (np. LEX, FLEX).
- Napisanie analizatora bezpośrednio w jakimś języku programowania.

Złożoność pamięciowa i czasowa automatów skończonych

Automat	Pamięć	Czas
DFA	$O(2^{\|r\|})$	$O(\|x\|)$
NFA	$O(\|r\|)$	$O(\|x\| \cdot \|r\|)$

gdzie $|r|$ - długość wyrażenia regularnego, $|x|$ - długość łańcucha wejściowego.

Jednak implementacja DFA jest dużo łatwiejsza, a wielkość zmniejsza się w trakcie minimalizacji.

Notatki

Koncepcja FLEX-a

- Generowanie kodu analizatora na podstawie zadanej specyfikacji.
- Domyślnie analizator jest w języku C.
- Wygenerowany kod źródłowy kompilujemy jako samodzielny program lub moduł programu.
- `yylex()` – funkcja wygenerowana przez LEX-a odpowiedzialna za działanie leksera (można ją wykorzystać w innej aplikacji).

`scan.l` → flex → `scan.c` → gcc → `scan[.exe]`

Notatki

Specyfikacja pliku źródłowego

Specyfikacja składa się z 3 części:

- Sekcja definicji.
- Sekcja reguł przetwarzania, gdzie reguła składa się z dwóch części
  - Wzorca (wyrażenia regularnego)
  - Operacji (zapisanej w C)
- Sekcja podprogramów.

Notatki

Podstawowe reguły działania

- Niedopasowane znaki są przepisywane na wyjście.
- Można definiować operacje puste (wzorzec bez reguły przetwarzania).
- Znaki specjalne poprzedzamy znakiem \.
- Wzorce zawierające spacje ujmujemy w cudzysłów podwójny.

Notatki

Przykład

```
1  %{
2  #include<stdio.h>
3  int yywrap();
4  int yylex();
5  int NL=0;
6  %}
7  %%
8  ^[[:blank:]]*register[[:blank:]]+       ;
9  long[[:blank:]]+int     printf("long");
10 unsigned[[:blank:]]+int printf("unsigned␣int");
11 signed[[:blank:]]+int   printf("int");
12 \n                      { printf("\n"); NL++; }
13 %%
14 int yywrap() {
15     printf("---\n\%d\n",NL);
16     return 1;
17 }
18 int main() {
19     return yylex();
20 }
```


Notatki

Wyrażenia regularne w FLEX-ie

Wyrażenie	Opis
`^x`	Wzorzec od początku linii
`x$`	Wzorzec do końca linii
`xy`	Konkatenacja wzorców
`x|y`	alternatywa wzorców
`x*`	domknięcie zwrotne
`x+`	domkniecie dodatnie
`x?`	opcjonalność (występuje 0 lub 1 raz)
`x{3}`	trzykrotne powtórzenie wzorca
`x{2,4}`	od dwóch do czterech powtórzeń
`x{2,}`	co najmniej dwa powtórzenia
`(x|y)z`	nawiasy wyrażają priorytet
`[a-z]`	klasa znaków, jeden znak ze zbioru od a do z
`[^a-z]`	dowolny znak spoza klasy
`.`	dowolny znak (ale nie \n)

Notatki

Zmienne wbudowane

- `yytext` – wskaźnik na ostatnio rozpoznany (dopasowany) leksem;
- `yyleng` – długość dopasowanego leksema;

Notatki

Zasady dopasowywania

Co zrobi LEX jeśli tekst może się dopasować do kilku wzorców?

Niejednoznaczność w LEX-ie rozstrzygana jest według 2 zasad:

1. Zasada najdłuższego dopasowania.
2. Zasada wcześniejszego dopasowania.

Notatki