

Języki formalne i metody kompilacji

Laboratorium nr 7

Stefan Sokołowski

Zadanie 1:

Napisać rekursywne parsery zstępujące, **na razie bez budowania drzew wyvodu**, dla gramatyk:

(a) $\langle X \rangle ::= a \mid b\langle X \rangle c$

(b) $\langle X \rangle ::= a \mid b\langle Y \rangle$
 $\langle Y \rangle ::= c \mid \langle X \rangle d \mid e\langle Y \rangle$

— por. omawiane na wykładzie parsery zstępujące¹.

Wskazówka:

Każdemu nieterminalowi ma odpowiadać jedna funkcja rekursywna, której ciało ma być napisane zgodnie z kształtem produkcji z tego nieterminalu:

- wystąpieniu *terminalu* po prawej stronie produkcji ma odpowiadać sprawdzenie, czy na wejściu jest dokładnie ten terminal i jeśli tak, wczytanie następnego „na zapas”;
- wystąpieniu *nieterminalu* po prawej stronie produkcji ma odpowiadać wywołanie odpowiadającej mu funkcji oraz sprawdzenie, czy to wywołanie zakończyło się sukcesem;
- w wypadku *udanego* rozpoznania całej produkcji sygnalizujemy sukces (`return TRUE;`);
- w wypadku *nieudanego* rozpoznania produkcji podejmujemy następujące działania:
 - jeśli błąd nastąpił po wczytaniu jakiegoś terminalu, to zatrzymujemy wykonanie programu i sygnalizujemy błąd „twardy” (`bład(...);`),
 - jeśli błąd nastąpił, zanim jakiś terminal został wczytany, to przechodzimy do następnej produkcji dla tego samego nieterminalu (za znakiem „`|`” w gramatyce),
 - jeśli błąd nastąpił, zanim jakiś terminal został wczytany, i nie ma już żadnej następnej produkcji, to sygnalizujemy błąd „miękki” (`return FALSE;`) ale nie zatrzymujemy wykonania programu.

Zadanie 2:

Napisać rekursywne parsery zstępujące, **na razie bez budowania drzew wyvodu**, dla gramatyk:

(a) $\langle X \rangle ::= a \mid \langle X \rangle b \langle X \rangle c$

(b) $\langle S \rangle ::= a \mid \langle T \rangle b$
 $\langle T \rangle ::= c \mid \langle S \rangle d$

— por. zad. 1(a) i 1(b) z Lab. 6²

Wskazówka:

Zacząć trzeba od zamiany lewostronnej rekursji na iteracje; dalej jak w zad. 1. „Gwiazdkowanym” napisom w gramatyce powinny odpowiadać pętle w funkcjach parsingu.

¹ iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm/Slajdy/07a.pdf str. 5–7.

² iis.ans-elblag.pl/~stefan/Dydaktyka/JezForm/Slajdy/06c.pdf.

Zadanie 3 (domowe na punkty):

Napisać rekursywny parser zstępujący, **na razie bez budowania drzew wyvodu**, dla gramatyki

```
⟨S⟩ ::= a | ( ⟨S⟩ ⟨S⟩ )
```

Na przykład napis `((aa)((aa)a))` jest programem w sensie tej gramatyki.

Uwaga:

Czas na przysłanie (na `s.sokolowski@ans-elblag.pl`): 2 tygodnie od zajęć 7 — potem już nie przyjmuję rozwiązań. Liczba punktów zależy od oryginalności i elegancji rozwiązania. W mailu proszę koniecznie podać swoje **nazwisko, numer zajęć i numer zadania**.

Zadanie 4 (domowe specjalne):

Napisać rekursywny parser zstępujący, **na razie bez budowania drzew wyvodu**, dla gramatyki

```
⟨program⟩ ::= ⟨deklaracje⟩ ⟨instrukcje⟩  
⟨deklaracje⟩ ::= d | ⟨deklaracje⟩ d  
⟨instrukcje⟩ ::= ⟨komenda⟩ | ⟨instrukcje⟩ ⟨komenda⟩  
⟨komenda⟩ ::= k | ( ⟨program⟩ )
```

Na przykład napis `dddkk(dkk)k` jest programem w sensie tej gramatyki.

Uwaga:

Pierwszej osobie, która przysła mi (na `s.sokolowski@ans-elblag.pl`) poprawne rozwiązanie tego zadania, przysługują punkty do zaliczenia laboratorium/ćwiczeń. Liczba tych punktów zależy od oryginalności i elegancji rozwiązania. W mailu proszę koniecznie podać swoje **nazwisko, numer zajęć i numer zadania**.
