Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Zwięzły opis narzędzi wraz konkretnymi przykładami, próba porównania i rankingu

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Do stworzenia parsera nie jest konieczne używanie zewnętrznych narzędzi – można posłużyć się algorytmem tworzenia funkcji rekurencyjnych dla obliczania atrybutów gramatyki L-atrybutywnej bazującej na gramatyce LL(1) podczas parsingu.

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Jest to jednak rozwiązanie problematyczne gdyż jest podatne na błędy, ma ograniczenia w stosunku do gramatyki, utrzymanie kodu jest trudne a wykrywanie błędów w ciągach wejściowych jest utrudnione.

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Narzędzia do tworzenia analizatorów leksykalnych

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

JLex:
A Lexical Analyzer Generator for Java
(http://www.cs.princeton.edu/~appel/modern/java/JLex/)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Postać pliku konfiguracyjnego:

kod użytkownika

%%

dyrektywy JLex

%%

reguły

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

- kod użytkownika – bezpośrednio kopiowany na wyjście
- dyrektywy JLex – pozwalają wykonać odpowiedni kod w żądanym miejscu
- reguły – reguła składa się z opcjonalnej listy stanów, wyrażenia regularnego i akcji

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Przykładowe dyrektywy JLex'a

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

%{
<code>
%}

Pozwala skopiować kod do klasy:

class Yylex {
... <code> ...
}

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

%init{
<code>
%init}

Pozwala skopiwoać kod do konstruktora klasy:

class Yylex {
Yylex () {
... <code> ...
}
}

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

%eof{
<code>
%eof}

Pozwala skopiować kod na koniec pliku wynikowego.

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

%integer
%intwrap

Powoduje że tokeny zwracają odpowiednio wartości całkowite i obiekty typu Number
(domyślnie zwracają obiekty klasy Yytoken)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Reguły JLex'a

<expression> { <action> }

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

```
%%

DIGIT=[0-9]
LETTER=[a-zA-Z]
WHITESPACE=[ \t\n]      // spacja, tabulator, nowa linia

// Włącza liczenie linii
%line

%%

{LETTER}({LETTER}|{DIGIT}*) {System.out.println(yyline+1 + ":
  Identyfikator " + yytext());}
{DIGIT}+                    {System.out.println(yyline+1 + ": Liczba");}
"="                        {System.out.println(yyline+1 + ": Przypisanie");}
"=="                        {System.out.println(yyline+1 + ": Porównanie");}
{WHITESPACE}*                   { }
.                  {System.out.println(yyline+1 + ": zły znak");}
```

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Cechy:

- napisany w Javie dla Javy
- dostępny kod źródłowy pod licencją Open Source
- opis gramatyki podobny do Lexa
- współpraca z generatorem parserów CUP
- wynik: klasa Yylex – w pełni funkcjonalny skaner
- uboga dokumentacja

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

JFlex:

The Fast Scanner Generator for Java

(http://jflex.de/)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Oparty na JLex, zawiera większą w porównaniu do niego funkcjonalność (np. Liczenie kolumn).

Możliwości:

- Zgodny składniowo z JLex
- Pełne wsparcie dla Unikodu
- Szybciej niż JLex generuje skanery
- Generuje skanery szybciej działające niż JLex (zgodnie z informacjami ze strony producenta)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

- Generacja różnego rodzaju skanerów (szybsze/mniejsze itp)
- Współpraca z generatorami parserów LALR:
  - CUP
  - BYacc
  - ANTLR
- Licencja GPL
- Podświetlenie składni w Vim i XEmacs
- Łatwa integracja z programem ANT (odpowiednik MAKE dla Javy)
- Rozbudowana dokumentacja

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Jax

http://linux4u.jinr.ru/usoft/WWW/www_blackdown.org/kbs/jax.html

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Cechy:

- Nie wspiera Unikodu
- Nie pozwala na definiowanie makr
- Składnia podobna do Lex'a

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Przykład:

http://linux4u.jinr.ru/usoft/WWW/www_blackdown.org/kbs/htmlsplit.lex

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Ranking możliwości:

- Jax – najprostrzy – nie wspiera np. Makr i Unikod'u
- JLex – funkcjonalnie podobny do Lex'a
- JFlex – podobny do Flexa – w porównaniu z JLex ma trochę większe możliwości

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Generatory Parserów

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

ANTLR

ANother Tool for Language Recognition

(http://www.antlr.org/)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

http://www-users.mat.uni.torun.pl/~tmpd/lgm/ANTLR.pdf

- twórcą **ANTLR** jest Terence Parr, który prace nad nim zapoczątkował w roku 1989.
- opisy gramatyk są do siebie podobne – przetwarzanie języków typu **LL(k)** dla wszystkich wariantów gramatyk
- akceptacja trzech rodzajów gramatyk: parserów, lekserów oraz drzew parserów
- parsing metodą top-down
- domyślnie ANTLR generuje lekser i parser w Javie, a plik z gramatyką ma rozszerzenie .g

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Gramatyki:
podklasy Lexer, Parser lub TreeParser

```
class ExprParser extends Parser;

expr:   mexpr ((PLUS|MINUS) mexpr)*
    ;

mexpr
    :   atom (STAR atom)*
    ;

atom:   INT
    |   LPAREN expr RPAREN
    ;
```

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Definicja operatorów i whitespace'ów w Lexerze

```
class ExprLexer extends Lexer;

options {
    k=2; // needed for newline junk
    charVocabulary='\u0000'..'\u007F'; // allow ascii
}

LPAREN: '(' ;
RPAREN: ')' ;
PLUS  : '+' ;
MINUS : '-' ;
STAR  : '*' ;
INT   : ('0'..'9')+ ;
WS    : ( ' '
        | '\r' '\n'
        | '\n'
        | '\t'
        )
        {$setType(Token.SKIP);}
      ;
```

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

JavaCC

Java Compiler Compiler

(https://javacc.dev.java.net/)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

CECHY:

- Generacja parserów top-down
- podobieństwo JavaCC do lex/flex
- Specifikacja leksykalna i gramatyczna zapisywana do jednego pliku
- Preprocesor JJTree do budowania drzew
- Tworzenie dokumentachi na podstawie gramatki
- Obsługa Unicode
- Dobre raportowanie błędów oraz debuggging

- Sporo dokumentacji i przykładów:
  http://www.engr.mun.ca/~theo/JavaCC-FAQ/javacc-screen-faq.pdf

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

ZASTOSOWANIE:
Tworzenie parserów dla: RTF, Visual Basic, Python, pliki Rational Rose mdl , XML, XML DTDs, HTML, C, C++, Java, JavaScript, Oberon, SQL, VHDL, VRML, ASN1, nagłówki e-maili itd.

NIE OBSŁUGUJE:

- automatycznej budowy drzew -> JJTree lub JTB
- nie buduje tablic symboli -> parsery oparte na JavaCC
- brak generacji języków wyjściowych -> wyjście dla stringów z drzewa

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

CUP Parser Generator for Java

Java(tm) Based Constructor of Useful Parsers
(http://www.cs.princeton.edu/~appel/modern/java/CUP/)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

- popularny generator parserów dla Javy
- pełni tą samą rolę co YACC, podobna składnia, zawiera wiele funkcji (w odróżnieniu: napisany w Javie)
- licencja open source
- współpracuje z wieloma generatorami skanerów

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

PRZYKŁAD – GRAMATYKA WYR. ARYTM.

```
expr_list ::= expr_list expr_part | expr_part
expr_part ::= expr ';'
expr      ::= expr '+' expr | expr '-' expr | expr '*' expr
              | expr '/' expr | expr '%' expr | '(' expr ')'
             | '-' expr | number
```

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Specyfikacja w CUP

```
import java_cup.runtime.*;

/* Preliminaries to set up and use the scanner.  */
init with {: scanner.init();            :};
scan with {: return scanner.next_token(); :};

/* Terminals (tokens returned by the scanner). */
terminal          SEMI, PLUS, MINUS, TIMES, DIVIDE, MOD;
terminal          UMINUS, LPAREN, RPAREN;
terminal Integer  NUMBER;

/* Non terminals */
non terminal          expr_list, expr_part;
non terminal Integer  expr, term, factor;

/* Precedences */
precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE, MOD;
precedence left UMINUS;
```

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

Specyfikacja w CUP (bez akcji)

```
/* The grammar */
expr_list ::= expr_list expr_part |
              expr_part;
expr_part ::= expr SEMI;
expr      ::= expr PLUS expr
            | expr MINUS expr
            | expr TIMES expr
            | expr DIVIDE expr
            | expr MOD expr
                | MINUS expr %prec UMINUS
            | LPAREN expr RPAREN
                | NUMBER
                ;
```

Akcje:

```
expr_part ::= expr:e
                {: System.out.println("= " + e); :}
              SEMI
                ;
expr      ::= expr:e1 PLUS expr:e2
                {: RESULT = new Integer(e1.intValue() + e2.intValue()); :}
                |
              expr:e1 MINUS expr:e2
              {: RESULT = new Integer(e1.intValue() - e2.intValue()); :}
```

.....itd.

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

INNE ?

http://catalog.compilertools.net/java.html

- BYACC/JAVA - rozbudowa Berkeley v 1.8 YACC-compatible generatorów parserów
- COCO/JAVA - Java version of CoCo – skaner i generator parserów .
- GENERIC INTERPRETER - pisanie "w locie", podgląd strumienia podczas rozbudowy gramatyki
- JACCIE - zabawka edukacyjna do wizualizacji technik kompilacji
- JAX - generacja skanerów z wyrażeń regularnych, brak makr.
- JAY- wersja BSD Yacc, generująca kod w Javie .
- JB - system do generacji parserów przy pomocy Gnu Bison do Javy
- JELL - parsers gramatyk LL(1)
- JTB - Java Tree Builder, używany z generatorami parserów JavaCC
- LOLO - wyciąga symbole z sekwencji znaków ASCII lub Unicode; nie oparty na automacie skończonym
- OOPS - object-oriented parser generator zaimlementowany w Javie, sprawdza gram. W EBNF czy jest LL(1), gen. parser
- PAT - Package COM.stevesoft.pat – kompilacja wyr. regularnych dla Perla
- SABLE CC. - object-oriented framework. Generacja kompilatorów i interpretatorów w Javie
- itd. ;)

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

CIEKAWE LINKI:

- http://en.wikipedia.org/wiki/Compiler-compiler
- http://mindprod.com/jgloss/parser.html
- http://www.swtech.com/java/parser/

Wybrane narzędzia do tworzenia analizatorów leksykalnych i składniowych w Javie

DZIĘKUJEMY
ZA
UWAGĘ.

Przemysław Kantyka
Patryk Orzechowski

PYTANIA ?