Tim's Logo Interpreter

Informatyka Stosowana III. rok

11.10.2011r.

1 Opis programu

Program `Tim's Logo Interpreter` jest implementacją interpretera podzbioru języka `Logo`. Inspirowany jest `UCBLogo`, który jest uznawany za standard de facto składni `Logo`. Wszędzie gdzie jest to możliwe (oraz pozwala na to zastosowany sposób implementacji) zachowana jest zgodność z `UCBLogo`. Mimo, że `UCBLogo` jest programem o otwartym kodzie źródłowym `Tim's Logo Interpreter` kończy swoje podobieństwo doń na składni i sposobie interpretacji poleceń - implementacja jest autorska i wykorzystuje zupełnie inne mechanizmy oraz struktury danych.

2 Gramatyka

Język `Logo` został oryginalnie zaprojektowany jako język funkcyjny szeroko czerpiący z `Lisp`-u. Zbiór tokenów `Logo` jest niewielki, wobec czego budowa skanera jest prosta. Jednak fakt, że `Logo` praktycznie nie posiada słów kluczowych powoduje, że nie jest możliwe napisanie parsera, który będzie tworzyć drzewo syntaktyczne wprowadzonych poleceń bez ich uprzedniej interpretacji. Wobec tego `Tim's Logo Interpreter` korzysta ze skanera, oraz prostego quasi-parsera, który steruje działaniem interpretera.

2.1 Zbiór tokenów skanera

Wyrażenie regularne	Nazwa Tokena	Opis
`[\[]`	`OPEN_LIST`	nawias otwierający listę
`[\]]`	`CLOSE_LIST`	nawias zamykający listę
`[0-9]+("."[0-9]+)?\b`	`NUMBER`	wartość - liczba
`["][0-9a-zA-Z"."]+`	`WORD`	wartość - słowo
`[:][0-9a-zA-Z"."]+`	`THING`	wartość - zmienna
`[a-zA-Z"."]+`	`IDENTIFIER`	identyfikator polecenia

Należy zauważyć, że tokeny `NUMBER`, `WORD`, `THING` odpowiadają za dostarczanie wartości wypełniających wejścia poleceń. Podobnie działają listy (zamknięte

pomiędzy `[` oraz `]`), jednak należy zauważyć (co nie jest realizowane w skanerze), że ich wnętrze - pomimo możliwości dostarczania przez skaner całego zbioru tokenów - jest interpretowane jako zbiór wartości.

2.2 Gramatyka parsera

```
%start expressions

%%

expressions
        : many EOF
        ;

many
        : single many
        | single
        ;

single
        : OPEN_LIST
        | CLOSE_LIST
        | WORD
        | THING
        | IDENTIFIER
        | NUMBER
        ;
```

Kod parsera w notacji BNF jest trywialny - ma on za zadanie dostarczać odpowiednie dane interpreterowi. Zasadniczo możliwym jest napędzanie interpretera bezpośrednio wyjściem skanera - jednak w implementacji zastosowano ten prosty parser dla wygody wywoływania odpowiednich metod interpretera.

3 Opis struktury logicznej programu

Program `Tim's Logo Interpreter` został napisany z wykorzystaniem paradygmatu obiektowego. Interfejs użytkownika obsługiwany jest przez:

- klasę `Console` - dostarcza metod standardowego wejścia i wyjścia
- klasę `TurtleConsole` - dostarcza medod sterowania grafiką żółwia

Oprócz generowanego parsera program wykorzystuje własne mechanizmy:

- klasę `Interpreter` - silnik interpretera
- klasę `IntepretersStack` - stos wywołań interpretera
- klasę `FunctionContainer` - kontener zdefiniowanych poleceń

- klasę `FunctionExecutor` - moduł wykonywania poleceń
- klasę `VariableContainer` - kontener zdefiniowanych zmiennych

Interpreter napędzany jest kolejnymi tokenami dostarczanymi przez skaner/-parser. Posiada on dwa stany (`ST_INTERPRET`, `ST_ENTER_LIST`), w zależności od stanu wykonuje odpowiednie operacje na dostarczanych tokenach. Dla stanu `ST_ENTER_LIST` wszystkie tokeny traktowane są jak wartość (słowo) oraz dopisywane do aktualnie wprowadzanej listy. Stan `ST_INTERPRET` jest stanem startowym i w stanie tym na tokenach wykonywane są odpowiadające im operacje:

- `NUMBER`, `THING`, `WORD` - dla tych tokenów wykonywana jest próba wprowadzenia wartości na wejście wierzchołka stosu wywołań poleceń
- `OPEN_LIST`, `CLOSE_LIST` - aktualizują stan interpretera
- `IDENTIFIER` - odkłada na stos wywołanie polecenia

Kiedy do polecenia na wierzchołku stosu wywołań zostanie dostarczona wymagana ilość wejść następuje ściągnięcie polecenia ze stosu oraz jego wykonanie przez `FunctionExecutor`. Wykonane polecenie może również dostarczyć wartości, która to dalej jest przekazywana na wejście uaktualnionego wierzchołka stosu wywołań.

4 Informacje o stosowanych pakietach zewnętrznych i programach narzędziowych

Program `Tim's Logo Interpreter` napisany został w języku `JavaScript` i stosuje nowy element standardu `HTML5 - canvas`. Wykorzystuje również generator skanerów/parserów `Jison`, będący implementacją pakietu `bison` dla języka `JavaScript`. Kolejnym wykorzystanym modułem jest biblioteka do obsługi klas w języku `JavaScript - jsoo.js` napisana przez Dariusza Dziuka.

Stosowanym edytorem był `SublimeText 2`, system kontroli wersji `SVN`, oraz przeglądarki `Google Chrome` oraz `Mozilla Firefox` w najnowszych wersjach stabilnych.

5 Instrukcja obsługi

Aby uruchomić `Tim's Logo Interpreter` należy otworzyć plik `index.html` z głównego katalogu projektu (lub też wejść na stronę `http://www.tymon.miropa.com.pl/Logo`) w przeglądarce obsługującej standard języka `ECMAScript`. Po uruchomieniu programu można wywołać komendę `help`, która zwraca listę zdefiniowanych poleceń. Po starcie program jest gotowy do interpretacji wprowadzanych wyrażeń.

6 Przykładowe programy

```
repeat 36 [ right 10 repeat 360 [forward 1 right 1]]
```

7 Możliwe rozszerzenia programu

Program `Tim's Logo Interpreter` został tak zaprojektowany, aby możliwym stało się zaimplementowanie większości mechanizmów języka `Logo`, znanych z `UCBLogo`. Stosunkowo łatwymi do dodania są:

- możliwość definiowania funkcji użytkownika
- wartości logiczne