ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ
Εργασία 1
2022-2023
Κανονικές Εκφράσεις και FLEX / BISON Συντακτική Ανάλυση

Μισαηλίδης Σάββας (ics21166)

1. An Agent Rule Language

Το συγκεκριμένο απόσπασμα αναφέρεται στην μετατροπή της αρχικής γραμματικής, σε γραμματική LL καθώς και την εύρεση των FIRST/FOLLOW.

1. Γραμματική LL

Παρατηρώντας την γραμματική, υπήρχε πρόβλημα με το GuardedActions, λόγω ότι παρουσιάζει αριστερή αναδρομή. Επομένως, με την βοήθεια της απαλοιφής αριστερής αναδρομής έχουμε την παρακάτω γραμματική:

```
Agent ::= “name” “[“ GuardedActions “]”
GuardedActions ::= GuardedAct GuardedActions'
| “mem-clear” “;” GuardedActions'
GuardedActions' ::= GuardedAct GuardedActions'
| ε
GuardedAct ::= “detector” “Rel” “T_NUM” “=>” Action “;”
| “true” “=>” Action “;”
Action ::= “forward” “T_NUM”
| “turn” “left” “T_NUM”
| “turn” “right” “T_NUM”
```

Όπου και έχει προστεθεί το GuardedActions’, απαλείφοντας έτσι το πρόβλημα της αναδρομής.

2. **FIRST/FOLLOW**

   Παρακάτω απεικονίζονται τα FIRST/FOLLOW της αλλαγμένης γραμματικής LL:

   FIRST(Agent) = {"name”}
   FIRST(GuardedActions) = {“mem-clear”, “detector”, “true”}
   FIRST(GuardedActions’) = {"detector”, “true”, ε}
   FIRST(GuardedAct) = {“detector”, “true”}
   FIRST(Action) = {“forward”, “turn”}

   FOLLOW(Agent) = {EOF}
   FOLLOW(GuardedActions) = {“]”}
   FOLLOW(GuardedActions’) = {“]”}
   FOLLOW(GuardedAct) = {“;”, “]”}
   FOLLOW(Action) = {“;”}

3. **Παραδείγματα κώδικα**

   Αφότου λοιπόν μετατράπηκε η γραμματική σε LL και βρέθηκαν τα FIRST/FOLLOW, με την χρήση της Αναδρομικής κατάβασης παράχθηκε ο παρακάτω κώδικας όπου απεικονίζονται οι συναρτήσεις agent και guardedActions (έχουν υλοποιηθεί και οι υπόλοιπες). Με βάση τα first/follow που έχουν βρεθεί στο προηγούμενο βήμα, η σειρά των εντολών ήταν εύκολη να προστεθεί. Για παράδειγμα, στην συνάρτηση agent, όπως και στην γραμματική μας, πρώτα γίνεται match στον “name”, έπειτα ανοίγεται η παρένθεση “[“. Στην συνέχεια, καλείται η συνάρτηση guardedActions, μέσω της οποίας είτε γίνεται match στο “mem-clear” ή καλείται η guardedAct και αμέσως μετά η guardedActionsPrime (guardedActions’) κλπ. Στο τέλος, αφότου έχει επιστρέψει η εκτέλεση του προγράμματος στην συνάρτηση agent, προστίθεται η κλειστή παρένθεση “]”, όπου και μετά τερματίζει το πρόγραμμα εμφανίζοντας είτε μήνυμα success! Ή μήνυμα λάθους σύνταξης.

```
newline \n|\x0A|\x0D|\x0A
ws ([ \t\r]+
WORD ^[a-z0-9]+
NUMBERS [0-9]+
REL [><=]

%%
{newline} { line++;}
{WORD} {return(TK_NAME);}
"[" {return(TK_OPENBRACKET);}
"]" {return(TK_CLOSEBRACKET);}
"mem-clear" {return(TK_MEM_CLEAR);}
";" {return(TK_SEMICOLON);}
"left-laser" {return(TK_LEFT_LASER);}
"right-laser" {return(TK_RIGHT_LASER);}
"front-sonar" {return(TK_FRONT_SONAR);}
{REL} {return(TK_REL);}
{NUMBERS} {return(TK_NUM);}
"->" {return(TK_ARROW);}
"turn left" {return(TK_LEFT);}
"turn right" {return(TK_RIGHT);}
"forward" {return(TK_FORWARD);}
"true" {return(TK_TRUE);}
%%
```

```
void agent(){
    match(TK_NAME);
    match(TK_OPENBRACKET);
    guardedActions();
    match(TK_CLOSEBRACKET);
}

void guardedActions(){
    if(token != TK_MEM_CLEAR) {
        guardedAct();
        guardedActionsPrime();

    }
    else if(token == TK_MEM_CLEAR) {

        match(TK_MEM_CLEAR);
        match(TK_SEMICOLON);
        guardedActionsPrime();
    }
    else {
      error_syntax();
    }
}
```

2. AgentSpeak

Το συγκεκριμένο απόσπασμα αναφέρεται στο κομμάτι της δημιουργίας του συντακτικού αναλυτή agentSpeak.

Παρατηρώντας την δοσμένη γραμματική, δεν υπάρχει κάποιο πρόβλημα ώστε να μην είναι LL(1), επομένως προχώρησα απευθείας στην σύνταξη του κώδικα και πιο συγκεκριμένα στην συγγραφή του agentSpeak.y και agentSpeak.i.

1) agentSpeak.y

παρακάτω απεικονίζονται στιγμυότυπα του κώδικα. Πιο συγκεκριμένα, τα tokens και τους Bison κανόνες, οι οποίοι λόγω ότι η γραμματική ήταν ήδη LL(1) απλά έγινε μετατροπή.

```
%define parse.error verbose

%token T_VAR
%token T_ATOM
%token T_NUMBER
%token T_COMP_OP
%token T_ARROW "<-"
%token T_NOT "not"
%token T_TRUE "true"

%token '.'
%token '('
%token ')'
%token ':'
%token ';'
%token '&'
%token '?'
%token '!'
%token '|'
%token ','

%left '+' '-'
%left '&'
%left ','
%left '.'

%nonassoc '='
%nonassoc '>'
%nonassoc '<'
```

```
agent : beliefs plans;

beliefs : beliefs belief | /*empty*/;

belief : predicate '.' ;

predicate : T_ATOM '(' terms ')';

plans : plans plan | /*empty*/;

plan : trig_event ':' context T_ARROW body '.';

trig_event : '+' predicate | '-' predicate | '+' goal | '-' goal;

context : T_TRUE | cliterals;

cliterals : literal | literal '&' cliterals;

literal : predicate | T_NOT '(' predicate ')' | boolExpr;

goal : '!' predicate | '?' predicate;

body : T_TRUE | actions;

actions : action | action ';' actions;

action : predicate | goal | belief_update;

belief_update : '+' predicate | '-' predicate;

terms : term | term ',' terms;

term : T_VAR | T_ATOM | T_NUMBER | T_ATOM '(' terms ')';

boolExpr : boolE | boolExpr '|' boolE;

boolE : boolarg relOp boolarg;

boolarg : T_NUMBER | T_VAR;

relOp : T_COMP_OP;
```

2) agentSpeak.l

παρακάτω απεικονίζονται στιγμυότυπα του κώδικα. Πιο συγκεκριμένα, τα regex και τους FLEX returns, οι οποίοι λόγω ότι η γραμματική ήταν ήδη LL(1) απλά έγινε μετατροπή.

```
ws [ \t]+
newline \n|\x0A|\x0D\x0A


nzdigit [1-9]
digit (0|{nzdigit})
digits {digit}+
floats  {digits}(\.{digits})
capital [A-Z]
smallcase [a-z]
letter [A-Za-z]
variable {capital}({letter}|{digit}|_)*
atom {smallcase}({letter}|{digit}|_)*
```

```
%%

{ws} {/* do nothing */}

{digits} {return T_NUMBER;}
{floats} {return T_NUMBER;}
{atom} {
    if (strcmp(yytext, "true") == 0) {
        return T_TRUE;
    } else if (strcmp(yytext, "not") == 0) {
        return T_NOT;
    } else {
        return T_ATOM;
    }
}

{variable} {return T_VAR;}
"<-" {return T_ARROW;}
"_" {return T_VAR;}
">"|"<"|"="|">="|"=<" {return T_COMP_OP;}

"." {return '.';}
"(" {return '(';}
")" {return ')';}
":" {return ':';}
"+" {return '+';}
"-" {return '-';}
"&" {return '&';}
"!" {return '!';}
"?" {return '?';}
";" {return ';';}
"," {return ',';}
"|" {return '|';}
```