

Ocampiler

Autores : Felipe Assad , Jorge Chagas, Thiago

Data : 30/05/2019

Universidade Federal Fluminense

Objetivos desta apresentação

- Apresentar as modificações no lexer, parser e em Pi.ml

O que foi feito

- Implementamos as modificações nas declarações e nas regras para o parser
- Os novos tokens no lexer
- As novas estruturas em Pi.ml

O que foi feito

No Lexer adição dos novos tokens:

```
| '*'           { TIMESORPOINTER }  
| "="          { BIND }  
| "let"        { LET }  
| "var"        { VAR }  
| "cns"        { CNS }  
| "in"         { IN }  
| "&"          { ADDRESS }  
| ",,"        { COMMA }
```

O que foi feito

Alteramos o parser para aceitar declaration e variable

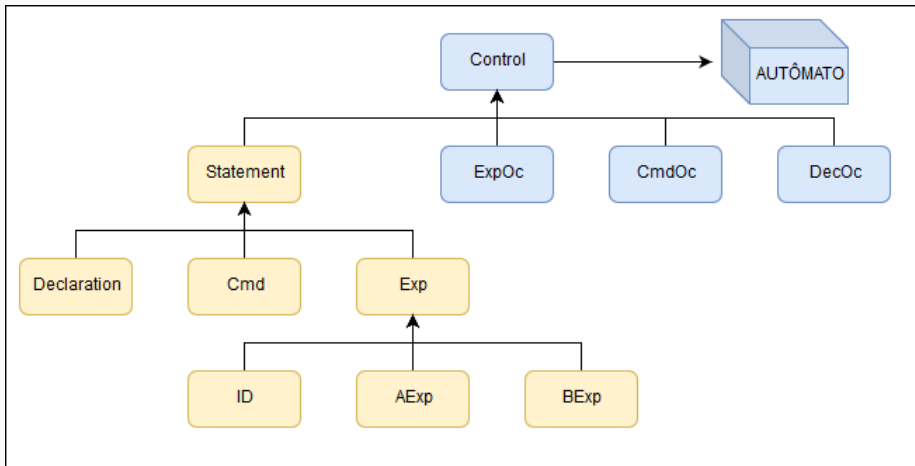
```
statement:
  expression { Pi.Exp($1)}
  | command   {Pi.Cmd($1)}
  | declaration {Pi.Dec($1)}
;
declaration:
  | VAR ID BIND expression {Pi.Bind(Pi.Id($2), Pi.Ref($4)) }
  | CNS ID BIND expression {Pi.Bind(Pi.Id($2), $4) }
  | LPAREN declaration RPAREN { $2 }
;
command:
  LOOP expression DO command END { Pi.Loop(($2), $4)}
  | IF expression THEN command ELSE command END { Pi.Cond(($2), $4, $6)}
  | IF expression THEN command END { Pi.Cond(($2), $4, Pi.Nop)}
  | ID ASSIGN expression { Pi.Assign(Pi.Id($1), $3) }
  | command command { Pi.CSeq($1, $2) }
  | LET declaration IN command {Pi.Blk($2, $4)}
  | LET declaration IN command END {Pi.Blk($2, $4)}
  | LPAREN command RPAREN { $2 }
;
expression:
  arithmeticExpression { Pi.AExp( $1) }
  | booleanExpression { Pi.BExp( $1) }
  | variable { $1 }
  | LPAREN expression RPAREN { $2 }
```

O que foi feito

Alteramos o parser para aceitar declaration e variable

```
variable:
    ID                                { Pi.Id( $1) }
    | TIMESORPOINTER ID              { Pi.ValRef(Pi.Id($2))}
    | ADDRESS ID                     { Pi.DeRef(Pi.Id($2))}
    | LPAREN variable RPAREN         { $2 }
;
arithmeticExpression:
    NUMBER                           { Pi.Num($1) }
    | arithmeticExpression PLUS arithmeticExpression { Pi.Sum(Pi.AExp($1), Pi.AExp($3) ) }
    | arithmeticExpression PLUS variable           { Pi.Sum(Pi.AExp($1), $3 ) }
    | variable PLUS arithmeticExpression          { Pi.Sum($1, Pi.AExp($3) ) }
    | variable PLUS variable                       { Pi.Sum($1, $3 ) }
```

O que foi feito



O que foi feito

Pi.ml:

```
and statement =
  | Exp of expression
  | Cmd of command
  | Dec of declaration

and decOC =
  | OPREF
  | OPBLKDEC
  | OPBLKCMD
  | OPBIND

and control =
  | Statement of statement
  | ExpOc of expOc
  | CmdOc of cmdOc
  | DecOc of decOC;;

and expression =
  | AExp of arithmeticExpression
  | BExp of booleanExpression
  | Id of string
  | Ref of expression
  | DeRef of expression
  | ValRef of expression

and command =
  | Loop of expression * command
  | CSeq of command * command
  | Nop
  | Assign of expression * expression
  | Cond of expression * command * command
  | Blk of declaration * command

and declaration =
  | Bind of expression * expression
```


O que não foi feito e porque

- As modificações no Autômato

- Podemos ter declarações como:
- `let cns z = &x`
- `let cns z = *x`
- Definir um outro operador pra ponteiro

Avaliação da evolução do trabalho

- Implementar um parser para a linguagem Imp-1 estendendo Imp-0 com declarações de variáveis e constantes. (OK)
- Implementar IR-mark1: (i) Interpreting Automata com ambientes, (ii) declarações de variáveis e constantes.
- Implementar um compilador de Imp-1 para IR-mark1.