Language analyser

November 25, 2016

1 Intro

The goal is to add the for statement : $< forstat >::= for\ \ ident := < exp > step < exp > until < exp > do < statlist >$

In the archive, you will find a model of a full analyser for the following grammar :

```
<statlist>   ::= <stat> | <stat> ; <statlist>
<stat>     ::= ...| <forstat> | <affectstat>
<affectstat> ::= ident := <exp>
```

In the archive, you have

- a Makefile (to use in order to compile/build all files)
- lexer.mll
- parser.mly
- prog.ml which calls the lexer on a file (”exStatlist.c”)
- lang.ml which defines the regular fonctions (backpatch, currentquad, current register...)
- exStatlist.c an example of correct program.

To test this analyser, execute the Makefile (make) and then execute prog: ./prog of files you build.

2 Expected outputs

- Single assignement

```
x:=2+3;
```

```
L1:   +,2,3,R0
L2:   :=,R0,nil,x
```

- Sequence

```
x:=2+3;
y:=x*3;
a:=y;
```

```
L1:   +,2,3,R0
L2:   :=,R0,nil,x
L3:   *,x,3,R1
L4:   :=,R1,nil,y
L5:   :=,y,nil,a
```

- For loop

```
a:=4;
b:=7;
res := 1;
for x := 1 step 1 until 3 do
        res := res*a;
endloop;
```

```
L1:   :=,4,nil,a
L2:   :=,7,nil,b
L3:   :=,1,nil,res
L4:   :=,1,nil,x
L5:   >?,x,3,L10
L6:   *,res,a,R0
L7:   :=,R0,nil,res
L8:   +,x,1,x
L9:   goto,nil,nil,L5
```

3 cut

In order to analyse the for statement, it is necessary to introduce a ”cut”. Here is the new rules you need to implement:

```
<forstat> ::= <forpart1> do <statlist> endloop
<forpart1> ::= for ident := <exp> step <exp> until <exp>
```