# Compilation and Program Analysis (#8) : Functions: syntax, semantics, code generation

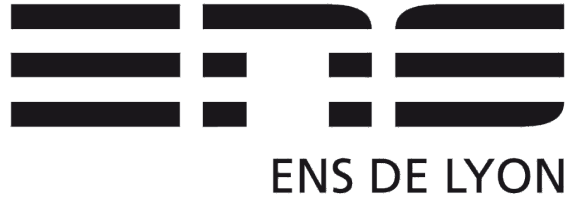
Laure Gonnord

http://laure.gonnord.org/pro/teaching/capM1.html

Laure.Gonnord@ens-lyon.fr

Master 1, ENS de Lyon

nov 2017

# Big picture

So far :

- All variables were global.
- No function call.

Inspiration : Y. Lakhnesh, UGA (first part), N. Louvet, Lyon1 (archi part), C. Alias (code gen part).

# Concrete syntax 1/2

- blocks are like before :

```
block
  : stat*    #statList
  ;
stat_block
  : OBRACE block CBRACE
  | stat
;
```

- procedures declaration :

```
declproc:
  : PROC ID IS stat
;
```

- variable decl can be local.

# Concrete syntax 2/2

And now there will two new kinds of statements :

```
stat
  : assignment
  | if_stat
  | while_stat
  | log
  | CALL ID
  | BEGIN declvar* declproc* block_stat END
  ;
```

- We can declare local procedures inside local procedures.

On board : add new concrete syntax for **functions**.

# Abstract syntax

WLOG, we will only consider programs with procedures :

$$
\begin{array}{lll}
S & \in & \textbf{Stm} \\
S & ::= & x := a \mid \text{skip} \mid S_1; S_2 \mid \\
 & & \text{if } b \text{ then } S_1 \text{ else } S_2 \\
 & & \text{while } b \text{ do } S \text{ od} \mid \text{begin } D_V \; D_P; S \text{ end} \mid \text{call } p \\
D_V & ::= & \text{var } x := a; \; D_V \mid \epsilon \\
D_P & ::= & \text{proc } p \text{ is } S; \; D_P \mid \epsilon
\end{array}
$$

# Operational semantics for local variables

Variable declaration : we invent $\rightarrow_D$ a semantic for declarations, and the following definitions :

- $DV(D_V)$ is the set of variables declared in $D_V$
- $\sigma'[X \mapsto \sigma] = \lambda x.$ if $x \in X$ then $\sigma(x)$ else $\sigma'(x)$.

Rules :

$$\frac{(D_V, \sigma[x \mapsto \mathcal{A}[a]\sigma]) \rightarrow_D \sigma'}{(\text{var } x := a; D_V, \sigma) \rightarrow_D \sigma'}$$

$$(\varepsilon, \sigma) \rightarrow_D \sigma$$

# Rules for blocs + Example

Be careful to restore everything after the local bloc :

$$\frac{(D_V, \sigma) \rightarrow_D \sigma' \qquad (S, \sigma') \rightarrow \sigma"}{(\text{begin } D_V; S \text{ end}, \sigma) \rightarrow_D \sigma"[DV(D_V) \mapsto \sigma]}$$

Seq is unchanged :

$$\frac{(S_1, \sigma) \rightarrow \sigma' \ (S_2, \sigma') \rightarrow \sigma''}{\big((S_1; S_2), \sigma\big) \rightarrow \sigma''}$$

Execute the semantics :

```
begin var y := 1;
      (x := 1;
      begin var x := 2; y := x+1 end;
      x := y+x)
end
```

# Dynamic versus Static binding, an example

What will be the behavior of call q ?

```
begin   var x := 0;
        proc p is x := x * 2;
        proc q is call p;
        begin var x := 5;
              proc p is x := x + 1;
              call q; y := x;
        end;
end
```

It depends !

# Dynamic versus Static binding, ex 3/4

Dynamic binding for variables and static for procedures :

```
begin   var x := 0;
        proc p is x := x * 2;
        proc q is call p;
        begin var x := 5;
              proc p is x := x + 1;
              call p; y := x;
        end;
end
```

```
begin   var x := 0;
        proc p is x := x * 2;
        proc q is call p;
        begin var x := 5;
              proc p is x := x + 1;
              x := x + 1; y := x;
        end;
end
```

# Dynamic versus Static binding, ex 4/4

Static for variables and procedures :

```
begin   var x := 0;
        proc p is x := x * 2;
        proc q is call p;
        begin var x := 5;
              proc p is x := x + 1;
              call p; y := x;
        end;
end
```

```
begin   var x := 0;
        proc p is x := x * 2;
        proc q is call p;
        begin var x := 5;
              proc p is x := x + 1;
              x := x * 2; y := x;
        end;
end
```

# Semantics : dynamic links 1/2

How ?

- States : variable $\rightarrow$ int.
- Environment : procedure name $\rightarrow$ Stm.
- Configuration : $(Env_P \times Stm \times State) \cup State$

The dynamic link for procedures is made by calling the *current* value at the call :

$$\frac{(env, env(p), \sigma) \rightarrow \sigma'}{(env, \textsf{call } p, \sigma) \rightarrow \sigma'}$$

# Semantics : dynamic links 2/2

Thus, environments are kept for the sequence :

$$\frac{(env, S_1, \sigma) \to \sigma' \qquad (env, S_2, \sigma') \to \sigma"}{(env, (S_1; S_2), \sigma) \to \sigma'}$$

but they are updated with a local declaration :

$$\frac{(D_V, \sigma) \to_D \sigma' \quad (\texttt{upd}(env, D_P), S, \sigma') \to \sigma"}{(env, \texttt{begin } D_V D_P; S \texttt{ end }, \sigma) \to \sigma"[\texttt{DV}(D_V) \mapsto \sigma]}$$

with

- $\texttt{upd}(env, \varepsilon) = env$
- $\texttt{upd}(env, \texttt{proc } p \texttt{ is S}; D_P) = \texttt{upd}(env[p \mapsto S], D_P)$

▶ Ex : test on the example !

# Static links for procedures

We need to store an environnement while defining a procedure :

- Environment : procedure name $\rightarrow Stm \times Env_P$
- Configuration : $(Env_P \times Stm \times State) \cup State$

Now update is modified :

- $\texttt{upd}(env, \varepsilon) = env$
- $\texttt{upd}(env, \texttt{proc } p \texttt{ is } \texttt{S}; D_P) = \texttt{upd}(env[p \mapsto (S, env)], D_P)$.

```
begin   var x := 2;
        proc p is x := 0;
        proc q is begin x := 1; (proc p is call p); call p end;
        call q
end
```

# Static links for procedures 2/2

Two possibilities for the call, let $env(p) = (S, env')$ in :

$$\frac{(env', S, \sigma) \to \sigma'}{(env, \texttt{call } p, \sigma) \to \sigma'}$$

or

$$\frac{(env'[p \mapsto [(S, env')], S, \sigma) \to \sigma'}{(env, \texttt{call } p, \sigma) \to \sigma'}$$

# Static link for variables and procedures 1/3

Config = $(env_V, env_P, S, sto)$ or $(env_V, sto)$ with :

- $env_V : x \mapsto address$ "symbol table".
- $env_P : p \mapsto (env_V, env_P, S)$ to store values during variable/proc declaration
- $sto : address \mapsto \mathbb{Z}$ (old $\sigma(x) = sto(env_V(x))$).
- $new(sto)$ gives a new address.

# Static link for variables and procedures 2/3

Variable declaration :

- $(\varepsilon, env_V, sto) \rightarrow_D (env_V, sto)$
- $$\frac{(D_V, env_V[x \mapsto nc], sto[nc \mapsto v]) \rightarrow_D (env'_V, sto')}{((\texttt{var}\ \ x := a; D_V), env_V, sto) \rightarrow_D (env'_V, sto')}$$

  with $v = \mathcal{A}[a](sto \circ env_V)$, $nc = new(sto)$ such that $x \rightarrow nc \rightarrow v$

# Static link for variables and procedures 3/3

$$(env_V, env_P, x := a, sto) \rightarrow (env_V, sto[nc \mapsto v])$$

with $v = \mathcal{A}[a](sto \circ env_V)$ and $nc = env_V(x)$

$$\frac{(env'_V, env'_P, S, sto) \rightarrow (env'_V, sto')}{(env_V, env_P, \texttt{call } p, sto) \rightarrow (env_V, sto')}$$

or

$$\frac{(env'_V, env'_P[p \mapsto (env'_V, env'_P, S)], S, sto) \rightarrow (env'_V, sto')}{(env_V, env_P, \texttt{call } p, sto) \rightarrow (env_V, sto')}$$

where $env_P(p) = (env'_V, env'_P, S)$.