Blocks, Procedures, Parameters
Natural Semantics Using the Environment-Store Model
3 Blocks, Procedures, Parameters
3.1 Natural Semantics Using the Environment-Store Model

Environment-store model

• Definition 6.1 (blackboard).

• A moment’s thought 6.2 (oneself).

• Definition 6.3 (blackboard).

• Example 6.5 (blackboard).

• A moment’s thought 6.6 (oneself).

• State as the composition of store and environment (blackboard).
Updates

• Environment update (blackboard).

• A moment’s thought 6.4 (blackboard).
3 Blocks, Procedures, Parameters

3.1 Natural Semantics Using the Environment-Store Model

Semantics of statements

• Specification (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (Table 6.3 without procedure environments and except for the block rule).

• A moment’s thought 6.9 (blackboard): variables have constant identity.
Exercises

• Find the natural semantics derivation tree for the swap program

\[(z := x ; x := y) ; y := z]\]

in the variable environment \(\{x \mapsto 0, y \mapsto 1, z \mapsto 2, \text{next} \mapsto 3\}\) and the initial store \(\{0 \mapsto 5, 1 \mapsto 7, 2 \mapsto 0\}\) (blackboard).

• Find the natural semantics derivation tree for the factorial program

\[p := 1 ; \quad \textbf{while} \quad \neg(x = 1) \quad \textbf{do} \quad (p := p \times x ; x := x - 1)\]

in the variable environment \(\{x \mapsto 0, p \mapsto 1, \text{next} \mapsto 2\}\) and the initial store \(\{0 \mapsto 2, 1 \mapsto 0\}\) (oneself).
Natural Semantics of Blocks
Variable declarations and blocks

• Syntactic categories (blackboard).

• Abstract syntax (blackboard).

• Discussions (blackboard):
  – The difference between assignments and a variable declaration;
  – Empty declarations.
3.2 Natural Semantics of Blocks

Semantics of variable declarations

• The new mapping (blackboard).

• Specification (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (Table 6.2).

• Discussion (blackboard): why has the semantics of declarations a structure different from that of composition statements?
3.2 Natural Semantics of Blocks

Semantics of block statements

- The block rule (Table 6.3, without procedure environments).
- A moment’s thought 6.9 revisited (oneself): is the principle still valid?
- A moment’s thought 6.10 (blackboard).
Exercise

• Find the natural semantics derivation tree for the “blocked” swap program

\[
\text{begin var } z := x; \ x := y ; \ y := z \text{ end}
\]

in the variable environment \(\{x \mapsto 0, y \mapsto 1, \text{next} \mapsto 2\}\) and the initial store \(\{0 \mapsto 5, 1 \mapsto 7\}\) (blackboard).
Exercise

• Consider the “blocked” factorial program

```
begin
  var p := 1; var z := x;
  while ¬(z = 1) do (p := p * z ; z := z - 1)
end
```

(oneself):

– Find the natural semantics derivation tree for it in the variable environment \{ x \mapsto 0, \text{next} \mapsto 1 \} and the initial store \{ 0 \mapsto 2 \};

– What is the final value of \( x \)?

– What would change if \( z \) were replaced with \( x \) throughout the program?
Natural Semantics of Procedures
Procedure names, declarations and calls

• Syntactic categories (blackboard).

• Abstract syntax (blackboard).
Scope rules

- Informal definitions (blackboard):
  - Static scopes – areas in the code;
  - Dynamic scopes – areas in the call stack.
- Example in Fig. 6.2 (blackboard).
- Example languages for each scoping (oneself).
- A moment’s thought 6.12 (oneself).
3 Blocks, Procedures, Parameters
3.3 Natural Semantics of Procedures

Semantics of fully dynamic scoping

• Nature of procedure environments (blackboard).

• Semantics of procedure declarations (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (Table 6.4).

• Semantics of statements (blackboard):
  – Adding of a procedure environment;
  – The block rule (Table 6.3);
  – The call rule (Table 6.5).
Exercises

• A moment’s thought 7.3 (oneself).

• Find the natural semantics of the program

begin
  var x := 0;
  proc p is call q;
  proc q is
    if x <= 1 then (x := x + 1 ; call q) else skip;
    begin var x := 1; call p end
end

in the empty initial environments and store (blackboard).

• Problem 6.13 (oneself).
Domain equations

- Nature of procedure environments under static scoping (blackboard):
  - Try of defining them as in the textbook;
  - Discussion on domain equations.

- Alternative: lists of environment pairs (blackboard).
Semantics of fully static scoping

• Semantics of procedure declarations (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (analogous to Table 6.8).

• Semantics of statements (blackboard):
  – Adding of scope environment;
  – The block rule (analogous to Table 6.3);
  – The call rule (analogous to Table 6.9).
Subtleties

- A moment’s thought 6.15 (oneself).

- Could we define semantics of procedure declarations similarly to variable declarations, i.e., with fusing new bindings to the existing environment (oneself)?
Exercises

• Find the natural semantics of the program

begin
  var x := 0;
  proc p is call q;
  proc q is
    if x <= 1 then (x := x + 1 ; call q) else skip;
    begin var x := 1; call p end
end

in the empty initial environments and store (blackboard).

• Problem 6.16 (oneself).
3 Blocks, Procedures, Parameters
3.4 Natural Semantics of Parameters
Procedures with a parameter

• Abstract syntax (blackboard).

• Terminology (blackboard):
  – Formal parameter;
  – Actual parameter.
3 Blocks, Procedures, Parameters
3.4 Natural Semantics of Parameters

Parameter mechanisms

• Informal definitions (blackboard):
  – Call-by-reference – reference to the actual parameter is passed;
  – Call-by-value – value of the actual parameter is computed and passed;
  – Call-by-name – the actual parameter is passed textually.
Semantics of procedure declarations

• Nature of procedure environments (blackboard).

• Semantics of procedure declarations (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (analogous to Table 7.1).
Semantics of call-by-reference

- Formal semantics (blackboard):
  - The call rule (analogous to Tables 7.2 and 7.3).

- The consequences (blackboard):
  - Aliasing – update of the formal parameter affects the actual parameter;
  - Must the actual parameter be a variable? Why? What is call-by-need?
Exercises

• Find the natural semantics derivation tree for the program

\[
\text{begin} \\
\text{proc } p(\text{var } x) \text{ is } x := x + 1; \text{ call } p(y) \\
\text{end}
\]

in the initial environment \{x \mapsto 0, y \mapsto 1, \text{next} \mapsto 2\} and store \{0 \mapsto 0, 1 \mapsto 0\}.

• Problem 7.2 for the initial environment \{z \mapsto 0, \text{next} \mapsto 1\} and store \{0 \mapsto 0\} (blackboard, oneself).
Semantics of call-by-value

• Formal semantics (blackboard):
  – The call rule (analogous to Table 7.4).

• The consequences (blackboard):
  – No aliasing;
  – Perhaps some computation before the procedure can start.
Exercises

• Find the natural semantics derivation tree for the program

begin

proc \texttt{p(val x)} \texttt{is} x := x + 1;

\texttt{call p(y)}

end

in the initial environment \{x \mapsto 0, y \mapsto 1, \text{next} \mapsto 2\} and store \{0 \mapsto 0, 1 \mapsto 0\} (blackboard).

• Find the natural semantics derivation tree for the program in Fig. 7.1 in the initial environment \{z \mapsto 0, \text{next} \mapsto 1\} and store \{0 \mapsto 0\} (blackboard, oneself).
Substitution revisited

- Example in Fig. 7.3 (blackboard):
  - Local $x$ is not replaced;
  - Local $y$ is renamed.

- A moment’s thought 7.15 (oneself).

- Definition (blackboard):
  - Substitution in procedure declarations (corrected Def. 7.13);
  - Substitution on statements (corrected Table 7.7).
Issues to think of

• Could we define substitution separately for variable declarations and use it in the definition of substitution in block statements (oneself)?

• Why must the variables in the l.h.s. of assignment statements also be replaced by substitutions (oneself)?

• Could we allow assigning to the formal parameter (oneself)?

• Perhaps we should provide three kinds of assignment just like there are three kinds of parameter passing. How (oneself)?
Exercise

• Provided $S$ is

\begin{verbatim}
begin
  var  y := 3;
  var  z := 2;
  proc q(name x) is y := x + 2;
    z := (z + x) * y ; call q(z)
end

find $S[x \mapsto y + 4]$ by definition (oneself).
\end{verbatim}
3 Blocks, Procedures, Parameters
3.4 Natural Semantics of Parameters

Semantics of call-by-name

• Scope issues (blackboard):
  – Substituting local expressions into a global procedure?
  – Substituting the body of a global procedure into a local context?

• Formal semantics (blackboard):
  – The call rule (analogous to Table 7.6).
  – How to do it correctly?

• Consequences (blackboard):
  – Lazy evaluation;
  – Repeated evaluation (Jensen’s device).
Exercise

• Find the natural semantics derivation tree of the program

begin

proc q(name x) is y := x + 2;
  z := (z + x) * y ; call q(z)
end

for the initial environment \{x \mapsto 0, y \mapsto 1, z \mapsto 2, next \mapsto 3\} and store \{0 \mapsto 6, 1 \mapsto 3, 2 \mapsto 2\} (blackboard).
Exercises

• Find the natural semantics derivation tree for the program

```
begin
  var a := 73287;
  proc p(name x) is skip;
  call p(a * a + 1000000)
end
```

in the empty initial environments and store (oneself).

• Build the natural semantics derivation tree from left to right for the program in Fig. 7.2 until the call to the procedure `findsum` (inclusive), in the empty initial environment and store (oneself).
Structural Semantics of Blocks and Procedures
Evaluation contexts

- Abstract syntax from the textbook (blackboard).
Dynamic scoping

• Structural semantics (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (analogous to Tables 10.1 and 10.3).
Exercises

• Find the structural semantics derivation sequence of the program

\[
\text{begin} \\
\quad \text{var } x := 0; \\
\quad \text{proc } p \text{ is call } q; \\
\quad \text{proc } q \text{ is} \\
\quad \quad \text{if } x \leq 1 \text{ then } (x := x + 1 ; \text{call } q) \text{ else skip; } \\
\quad \text{begin var } x := 1; \text{ call } p \text{ end} \\
\text{end}
\]

in the empty initial environment and store, under dynamic scoping (blackboard).

• Find the structural semantics derivation sequence of the program in Fig. 6.2 in the empty initial environments and store and dynamic scoping (oneself).
Static scoping

• Structural semantics (blackboard):
  – Configurations and terminal configurations;
  – Transitions;
  – Axioms and rules (analogous to Tables 10.1 and 10.2).
Exercises

• Find the structural semantics derivation sequence of the program

begin
  var x := 0;
  proc p is call q;
  proc q is
    if x <= 1 then (x := x + 1 ; call q) else skip;
  begin var x := 1; call p end
end

in the empty initial environment and store, under static scoping (blackboard).

• Find the structural semantics derivation sequence of the program in Fig. 6.2

in the empty initial environments and store, under static scoping (oneself).
Questions for thinking

- Are evaluation contexts really necessary (oneself)?

- Can’t we keep syntactic changes smaller by retaining the statements category and adding a minimalistic definition

\[ C ::= S \mid C \ ; \ S \mid \text{active } C \ \text{end} \]

for evaluation context (blackboard)?