6. Explicit Specifications

This part focuses on techniques for writing explicit specifications for processes.

The outline of this part:

- The structure of explicit specification
- Assignment statements
- Sequence of statements
- Conditional statements
- The block statement
- While statements
- Method invocation
The structure of explicit specification

The general form of a process specification:

```
process ProcessName(input) output
ext ExternalVariables
pre PreCondition
post PostCondition
decom LowerLevelModuleName
explicit ExplicitSpecification
comment InformalExplanation
end_process
```
A general structure of an explicit specification of a process is:

```
explicit
VariableDeclarations;
Statement
```

The VariableDeclarations gives the declarations of local variables within this process and their format is the same as declarations of local store variables in a module. Statement indicates an operation, which can be a simple statement or a compound statement.
Assignment statement

\[ v := \text{PE} \]

Where \( v \) is a variable and \( \text{PE} \) an expression.

Example:

1. \( x := x + \text{fact}(y) + \text{obj.m1}(y) \)
2. \( x := \text{get}({x: \text{account\_file} | (\exists y: \text{account\_file} | x.\text{balance} > y.\text{balance})}) \)
Sequence of statements

S_1;
S_2;
...
S_n;

Each $S_i$ (i=1..n) is a statement.
Conditional statements

Two forms are available:

(1) if B1 then S
(2) if B2 then S1 else S2

Examples:
(1) if x > 10 then x := 20
(2) if x > 10 then x := x + 1 else x := x - 1
The block statement

begin
  S
end

where S can be a simple statement or a compound statement.
Example:

if \( x > 1 \) then
begin
    S_1;
    S_2;
    S_3;
end
else
begin
    S_4;
    S_5;
end;
While statement

while B do
  S

Example: an explicit specification for the process to yield the factorial for a natural number:

\[ f = 1; \]
\[ n = 10; \]
\[ \text{while } n > 1 \text{ do} \]
\[ \text{begin} \]
  \[ f = f \times n; \]
  \[ n = n - 1; \]
\[ \text{end}; \]
Method invocation

obj.m1()

Where obj is an object of some class and m1 is its method.

This method invocation can be used either as an independent statement or as part of an expression used in another statement.
The implicit specification for the process

Check_Password is:
process Check_Password(card_id: nat, sel: bool, pass: nat)
    account1: Account | pr_meg: string | account2: Account
ext rd #account_file: set of Account;
post sel = false and
    (exists![x: account_file] | x.account_no = card_id and
     x.password = pass and account1 = x) or
sel = true and (exists![x: account_file] | x.account_no = card_id and
     x.password = pass and account2 = x) or
not (exists![x: account_file] | x.account_no = card_id and
     x.password = pass) and pr_meg = "Reenter your password or
     insert the correct card"
comment
    If sel is false and the input card_id and pass are correct with respect to the
    exiting information in account_file, the account information is passed to the
    output account1. If sel is true and the input card_id and pass are correct, the
    account information is passed to the output account2. However, if neither the
    card_id nor pass is correct, a prompt message pr_meg is given.
end_process;
An explicit specification is:

process Check_Password(card_id: nat, sel: bool, pass: nat)                                                              
    account1: Account | pr_meg: string | account2: Account                                                                 
ext rd #account_file: set of Account;
explicit                                                                                                              
begin                                                                                                                  
    account1 := get({x: account | x.account_no = card_id                                                                   
                      and x.password = pass});                                                                    
    if account1 = nil                                                                                                       
then pr_meg = "Reenter your pass or insert the correct card"                                                              
else if sel = true                                                                                                        
    then begin account2 := account1;                                                                                      
        account1 := nil;                                                                                                   
        account1 := nil;                                                                                                   
    end                                                                                                                    
else account2 := nil;                                                                                                     
end                                                                                                                       
comment                                                                                                                  
...                                                                                                                        
end_process;
Exercise 6

Write explicit specifications for the processes of the ATM given in chapter 4.

a. Receive_Command
b. Withdraw
c. Show_Balance