BL Program

PROGRAM identifier IS

BEGIN

END identifier

Math Model for Program

- What pieces of information do we need to keep track of for a BL program?
The Math Model

- math subtype PROGRAM is (  
  name: IDENTIFIER  
  context: CONTEXT  
  body: STATEMENT  
  )  
  exemplar p  
  constraint  
  root (p.body).kind = BLOCK

New Instruction

INSTRUCTION identifier IS

END identifier
Math Model Continued...

- What pieces of information do we need to keep track of for a new BL instruction?

Math Model Continued...

- `math subtype CONTEXT is finite set of ( name: IDENTIFIER body: STATEMENT )

exemplar c

c constraint

1. c is a (partial) function
2. for each (name, body) pair in c, name is not one of primitives nor the empty string, and body is a BLOCK
**Program Component**

- **Type**
  - Program_Kernel is modeled by PROGRAM

- **Initial Value**
  - IS_INITIAL_PROGRAM (self) ≡
    - self.name = empty_string and
    - self.context = empty_set and
  - IS_INITIAL_STATEMENT (self.body)

---

**Program SteerClear**

Draw a picture of this BL program's abstract view:

```
PROGRAM SteerClear IS
  INSTRUCTION StepAside IS
    IF random THEN
      turnright
    ELSE
      turnleft
    END IF
  move
END StepAside
  INSTRUCTION TurnAround IS
    turnright
    turnright
  END TurnAround
BEGIN
  WHILE true DO
    IF next-is-empty THEN
      skip
    ELSE
      IF next-is-wall THEN
        TurnAround
      ELSE
        StepAside
      END IF
    END IF
  END WHILE
END SteerClear
```
Program SteerClear (cont’d)

p =

Program Continued...

- Operations
  - p.Swap_Name (name)
  - p.Swap_Body (statement)
  - p.Add_To_Context (name, statement)
  - p.Remove_From_Context (name, name_copy, statement)
  - p.Remove_Any_From_Context (name, statement)
  - p.Is_In_Context (name)
  - p.Size_Of_Context ()
**Swap_Name**

What effect will the following statements have on $p$, the program object for program SteerClear?

object Text name;
name = "Timid";
p.Swap_Name (name);

name = "SteerClear"

**Swap_Body**

What effect will the following statements have on object $p$ from the previous slide?

object Statement b, i, c;
object Text m = "move";
object Integer test = RANDOM;
c.Compose_Call (m);
b.Add_To_Block (0, c);
i.Compose_If (test, b);
b.Add_to_Block (0, i);
p.Swap_Body (b);
**Context Operations**

What result is produced by these statements if \( p \) has its original value?

\[
p.\text{Size\_Of\_Context}() \ ?
\]

\[
p.\text{Is\_In\_Context}(\text{"StepAside"}) \ ?
\]

**Context Operations (cont’d)**

What result is produced by these statements if \( p \) has the value shown on the previous slide?

\[
\text{object Text n; object Statement b; p.\text{Remove\_From\_Context} (\text{"TurnAround"}, n, b) \ ?}
\]

\[
n=\text{"TurnAround"} \quad b =
\]
**Context Operations (cont’d)**

What result is produced by these statements if p and b have the values shown on the previous slide?

```plaintext
n = "TurnBack";
p.Add_To_Context (n, b) ?

n="" b = BLOCK
```

---

**An Operation on Program**

What would it mean to *demobilize* a program?
Statements in BL Program

PROGRAM identifier IS
  BEGIN
  END identifier

INSTRUCTION identifier IS
  END identifier

State the Problem

global_procedure Demobilize ( 
  alters Program& p 
); 
/*!

  ensures 
  p.name = #p.name and 
  p.context = 
    CONTEXT_DEMOBILIZE (#p.context) and 
  p.body = DEMOBILIZE (#p.body) 
  */
State the Problem Continued...

math definition CONTEXT_DEMOBILIZE (c: CONTEXT): CONTEXT satisfies
if c = empty_set then
  CONTEXT_DEMOBILIZE (c) = c
else
  there exists n: IDENTIFIER, s: STATEMENT, rest_of_context: CONTEXT
  (c = {(n, s)} union rest_of_context and
  CONTEXT_DEMOBILIZE(c) =
  {(n, DEMOBILIZE(s))} union
  CONTEXT_DEMOBILIZE (rest_of_context))

Implementation

procedure_body Demobilize (alters Program& p)
{
  object Program tmp;
  object Text name;
  object Statement body;
  p.Swap_Body (body);
  Demobilize (body);
  tmp.Swap_Body (body);
  while (p.Size_Of_Context () > 0)
    {p.Remove_Any_From_Context (name, body);
     Demobilize (body);
     tmp.Add_To_Context (name, body);
    }
  p.Swap_Name (name);
  tmp.Swap_Name (name);
  p &= tmp;
}