CS477 Formal Software Development Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

Slides based in part on previous lectures by Mahesh Vishwanathan, and by Gul Agha

February 22, 2013

Modificaton of data from Last Time

data was

```
type_synonym data = "int"
```

Now data is

```
datatype data = DN "int" | DR "real"
```

Tagged disjoint union of int and real

Revised Lifting Constants, Operators

Need to lift constants, variables, boolean and arithmetic operators to functions over states:

- Constants:

```
definition Data :: "data ⇒ exp" where
 "Data d ≡ λ s. d"
definition N :: "int ⇒ exp" where "N n ≡ λ s. DN n"
definition Real :: "real ⇒ exp" where
 "Real r ≡ λ s. DR r"

definition is_int_b :: "exp ⇒ bool_exp" where
 "is_int_b x ≡ λ s. (∃ n. x s = DN n)"

definition is_real_b :: "exp ⇒ bool_exp" where
 "is_real_b x ≡ λ s. (∃ r. x s = DR r)"
```


Revised Lifting Constants, Operators

- Arithmetic operations do type checking and coercion
  Before:

```
definition plus_e :: "exp ⇒ exp ⇒ exp"
 (infixl "[+]" 150) where
"(p [+] q) ≡ λ s. (p s + (q s))"
```

Now:

```
definition plus_e :: "exp ⇒ exp ⇒ exp"
 (infixl "[+]" 150) where
"(p [+] q) ≡
 λ s. (case p s of DN n ⇒
                 (case q s of DN m ⇒ DN(n + m)
                            | DR y ⇒ DR((real n) + y))
               | DR x ⇒
                 (case q s of DN m ⇒ DR(x + real m)
                            | DR y ⇒ DR(x + y)))"
```


HOL Type for Deep Part of Embedding

```
datatype command =
   AssignCom "var_name" "exp"          (infix "::=" 110)
 | SeqCom "command" "command"          (infixl ";" 109)
 | CondCom "bool_exp" "command" "command"
       ("IF _/ THEN _/ ELSE _/ FI" [120,120,120]60)
 | WhileCom "bool_exp" "command"
       ("WHILE _/ DO _/ OD" [120,120]60)
```


Defining Hoare Logic Rules

```
inductive valid :: "bool_exp ⇒command ⇒bool_exp ⇒bool"
("{{_}}_{{_}}" [120,120,120]60)where
AssignmentAxiom:
"{{(P[x⇐e])}}(x::=e) {{P}}" |
SequenceRule:
"⟦{{P}}C {{Q}}; {{Q}}C' {{R}}⟧
⟹{{P}}(C;C'){{R}}" |
RuleOfConsequence:
"⟦|⊨(P [⟶] P') ; {{P'}}C{{Q'}}; |⊨(Q' [⟶] Q) ⟧
⟹{{P}}C{{Q}}" |
IfThenElseRule:
"⟦{{(P [∧] B)}}C{{Q}}; {{(P[∧]([¬]B))}}C'{{Q}}⟧
⟹{{P}}(IF B THEN C ELSE C' FI){{Q}}" |
WhileRule:
"⟦{{(P [∧] B)}}C{{P}}⟧
⟹{{P}}(WHILE B DO C OD){{(P [∧] ([¬]B))}}"
```

DEMO

Annotated Simple Imperative Language

- We will give verification conditions for an annotated version of our simple imperative language
- Add a presumed invariant to each while loop

$\langle command \rangle$ ::= $\langle variable \rangle := \langle term \rangle$
| $\langle command \rangle; \ldots ; \langle command \rangle$
| *if* $\langle statement \rangle$ *then* $\langle command \rangle$ *else* $\langle command \rangle$
| *while* $\langle statement \rangle$ *inv* $\langle statement \rangle$ *do* $\langle command \rangle$

Hoare Logic for Annotated Programs

Assingment Rule

$$\frac{}{\{\!|P[e/x]|\!\}\ x\ :=\ e\ \{\!|P|\!\}}$$

Rule of Consequence

$$\frac{P \Rightarrow P' \quad \{\!|P'|\!\}\ C\ \{\!|Q'|\!\} \quad Q' \Rightarrow Q}{\{\!|P|\!\}\ C\ \{\!|Q|\!\}}$$

Sequencing Rule

$$\frac{\{\!|P|\!\}\ C_1\ \{\!|Q|\!\} \quad \{\!|Q|\!\}\ C_2\ \{\!|R|\!\}}{\{\!|P|\!\}\ C_1;\ C_2\ \{\!|R|\!\}}$$

If Then Else Rule

$$\frac{\{\!|P \wedge B|\!\}\ C_1\ \{\!|Q|\!\} \quad \{\!|P \wedge \neg B|\!\}\ C_2\ \{\!|Q|\!\}}{\{\!|P|\!\}\ \mathit{if}\ B\ \mathit{then}\ C_1\ \mathit{else}\ C-2\ \{\!|Q|\!\}}$$

While Rule

$$\frac{\{\!|P \wedge B|\!\}\ C\ \{\!|P|\!\}}{\{\!|P|\!\}\ \mathit{while}\ B\ \mathit{inv}\ P\ \mathit{do}\ C\ \{\!|P \wedge \neg B|\!\}}$$