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

March 12, 2014
DEMO
Algorithm for Proving Hoare Triples?

- Have seen in Isabelle that much of proving a Hoare triple is routine
- Will this always work?
- Why not automate the whole process?
  - Can’t (always) calculate needed loop invariants
  - Can’t (always) prove implications (side-conditions) in Rule of Consequence application
- Can we automate all but this?
- Yes! But how?
  1. Annotate all `while` loops with needed invariants
  2. Use routine to “roll back” post-condition to *weakest precondition*, gathering side-conditions as we go
- 2 called *verification condition generation*
Give verification conditions for an annotated version of our simple imperative language

Add a presumed invariant to each while loop

\[
\langle \text{command} \rangle ::= \langle \text{variable} \rangle := \langle \text{term} \rangle \\
| \langle \text{command} \rangle ; \ldots ; \langle \text{command} \rangle \\
| \text{if } \langle \text{statement} \rangle \text{ then } \langle \text{command} \rangle \text{ else } \langle \text{command} \rangle \\
| \text{while } \langle \text{statement} \rangle \text{ inv } \langle \text{statement} \rangle \text{ do } \langle \text{command} \rangle
\]

Example: \[
\text{while } y < n \text{ inv } x = y \ast y \\
\text{do} \\
x := (2 \ast y) + 1; \\
y := y + 1 \\
\text{od}
\]
datatype 'data annotated_command = 
    AnnAssignCom "var_name" "'data exp"
        (infix "=" 110)
    | AnnSeqCom "'data annotated_command"
        "'data annotated_command"
        (infixl ";;" 109)
    | AnnCondCom "'data bool_exp"
        "'data annotated_command"
        "'data annotated_command"
        ("If _/ Then _/ Else _/ Fi" [70,70,70]70)
    | AnnWhileCom "'data bool_exp" "'data annotated_command"
        ("While _/ Inv _/ Do _/ Od" [70,70]70)
Hoare Logic for Annotated Programs

Assingment Rule

\[ \{ P[e/x] \} x := e \{ P \} \]

Rule of Consequence

\[ P \Rightarrow P' \quad \{ P' \} C \{ Q' \} \quad Q' \Rightarrow Q \]

\[ \{ P \} C \{ Q \} \]

Sequencing Rule

\[ \{ P \} C_1 \{ Q \} \quad \{ Q \} C_2 \{ R \} \]

\[ \{ P \} C_1; \ C_2 \{ R \} \]

If Then Else Rule

\[ \{ P \land B \} C_1 \{ Q \} \quad \{ P \land \neg B \} C_2 \{ Q \} \]

\[ \{ P \} \text{ if } B \text{ then } C_1 \text{ else } C_2 \text{ if } B \text{ then } C \{ Q \} \]

While Rule

\[ \{ P \land B \} C \{ P \} \]

\[ \{ P \} \text{ while } B \text{ inv } P \text{ do } C \{ P \land \neg B \} \]
Defining Hoare Logic Rules

```plaintext
inductive ann_valid :: "'data bool_exp ⇒
'data annotated_command ⇒ 'data bool_exp ⇒ bool"
("{ _,_ }" [60,60,60]60)where
AnnAssignmentAxiom:"{(P[x:=e])}(x:=e) {P}" |
AnnSequenceRule:
"[(P=C {Q}; {Q=C'} {R})]⇒[(P(C;;C'){R}]" |
AnnRuleOfConsequence:
"[(|=(P [→] P') ; {P'}C{Q'} ; |=(Q' [→] Q))]
⇒{P}C{Q}" |
AnnIfThenElseRule:
"[{(P [∧ B)C{Q}; {(P[∧]([¬ B)))C'}{Q}]
⇒{(P)(If B Then C Else C' Fi){Q}" |
AnnWhileRule:
"[{(P [∧ B)C{P}]}
⇒{(P)(While B Inv P Do C Od){P [∧] ([¬ B])}]"
```

Elsa L Gunter ()
CS477 Formal Software Development Method
Relation Between Two Languages

- Hoare Logic for Simple Imperative Programs and Hoare Logic to Annotated Programs almost the same
- What is the precise relationship?
- First need precise relation between the two languages

Definition

\[
\begin{align*}
\text{strip}(v := e) & = v := e \\
\text{strip}(C_1 ; C_2) & = \text{strip}(C_1) ; \text{strip}(C_2) \\
\text{strip}(\text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi}) & = \\
& \quad \text{if } B \text{ then } \text{strip}(C_1) \text{ else } \text{strip}(C_2) \text{ fi} \\
\text{strip}(\text{while } B \text{ inv } P \text{ do } C \text{ od}) & = \text{while } B \text{ do } \text{strip}(C) \text{ od}
\end{align*}
\]

- We recursively remove all invariant annotations from all `while` loops
Relation Between Two Hoare Logics

Theorem

For all pre- and post-conditions $P$ and $Q$, and annotated programs $C$, if

$\{P\} \ C \ {Q}\$,

then

$\{P\} \ \text{strip}(C) \ \{Q\}$.

Proof.

(Sketch) Use rule induction on proof of $\{P\} \ C \ {Q}\$; in case of While Rule, erase invariant
Theorem

For all pre- and post-conditions $P$ and $Q$, and unannotated programs $C$, if

$\{P\} \ C \ \{Q\}$,

then there exists an annotated program $S$ such that

$C = \text{strip}(S)$ and $\{|P|\} \ S \ \{|Q|\}$.

Proof.

(Sketch) Use rule induction on proof of $\{P\} \ C \ \{Q\}$; in case of While Rule, add invariant from precondition as invariant to command.
Question: Given post-condition $Q$, and annotated program $C$, what is the most general pre-condition $P$ such that $\{P\} C \{Q\}$?

Answer: Weakest Precondition

Definition

\[
\begin{align*}
wp (x := e) Q &= Q[x \leftarrow e] \\
wp (C_1; C_2) Q &= wp C_1 (wp C_2 Q) \\
wp (\text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi}) Q &= \\
&\quad (B \land (wp C_1 Q)) \lor ((\neg B) \land (wp C_2 Q)) \\
wp (\text{while } B \text{ inv } P \text{ do } C \text{ od}) Q &= P
\end{align*}
\]

Assumes, without verifying, that $P$ is the correct invariant
Weakest in weakest precondition means any other valid precondition implies it:

Theorem

For all annotated programs $C$, and pre- and post-conditions $P$ and $Q$, if $\{\mid P\mid\} C \{\mid Q\mid\}$ then $P \Rightarrow \text{wp } C Q$.

- Proof somewhat complicated
- Uses induction on the structure of $C$
- In each case, want to assert triple proof must have used rule for that construct (e.g. Sequence Rule for sequences)
- Can't because of Rule Of Consequence
- Must induct on proof (rule induction) - in each case
- Uses:

Lemma

$\forall C P Q. (P \Rightarrow Q) \Rightarrow (\text{wp } C P \Rightarrow \text{wp } C Q)$
What About Precondition?

Question: Do we have $\{\text{wp } C \ Q\} \ C \ {\|} \ Q\}$?

Answer: Not always - need to check while-loop side-conditions – verification conditions

Question: How to calculate verification conditions?

Definition

\[
\begin{align*}
\text{vcg} \ (x := e) \ Q &= \text{true} \\
\text{vcg} \ (C_1; C_2) \ Q &= (\text{vcg} \ C_1 \ (\text{wp} \ C_2 \ Q)) \land (\text{vcg} \ C_2 \ Q) \\
\text{vcg} \ (\text{if } B \ \text{then } C_1 \ \text{else } C_2 \ \text{fi}) \ Q &= (\text{vcg} \ C_1 \ Q) \land (\text{vcg} \ C_2 \ Q) \\
\text{vcg} \ (\text{while } B \ \text{inv } P \ \text{do } C \ \text{od}) \ Q &= \\
&= ((P \land B) \Rightarrow (\text{wp} \ C \ P)) \land (\text{vcg} \ C \ P) \land ((P \land (\neg B)) \Rightarrow Q)
\end{align*}
\]
**Verification Condition Guarantees wp Precondition**

**Theorem**

\[ \text{vcg } C \quad Q \Rightarrow \{ wp \ C \quad Q \} \quad C \quad \{ Q \} \]

**Proof.**

**(Sketch)**

- Induct on structure of \( C \)
- For each case, wind back as we did in specific examples:
  - Assignment: \( wp \ C \quad Q \) exactly what is needed for Assignment Axiom
  - Sequence: Follows from inductive hypotheses, all elim, and modus ponens
  - If\_Then\_Else: Need to use Precondition Strengthening with each branch of conditional; \( wp \) and inductive hypotheses give the needed side conditions
  - While: Need to use Postcondition Weakening, While Rule and Precondition Strengthening
Verification Condition Guarantees \( \wp \) Precondition

**Corollary**

\[
((P \Rightarrow \wp C Q) \land (\text{vcg} C Q)) \Rightarrow \{P\} C \{Q\}
\]

This amounts to a method for proving Hoare triple \( \{P\} C \{Q\} \):

1. Annotate program with loop invariants (reduces to showing \( \{P\} C \{Q\} \))
2. Calculate \( \wp C Q \) and \( \text{vcg} C Q \) (automated)
3. Prove \( P \Rightarrow \wp C Q \) and \( \text{vcg} C Q \)

Basic outline of interaction with Boogie: Human does 1, Boogie does 2, Z3 / Simplify / Isabelle + human / ... does 3

For more information