Theoretical Computer Science

Week1: Hoare Logic for Verification of Properties of Algorithms

1.3.2018

Ana Sokolova

Verification of Propositions about Algorithms

- Hoare Logic: Calculus for proving propositions about algorithms and programs, program verification [C.A.R. Hoare, 1969]

- Static propositions over states (valuations of variables), that the algorithm (the program) can have at particular locations, e.g.
  … {level < max} level := level +1;… {0 < i ∧ i < 10} a[i] := 42;…;…

- The propositions must be provable for all executions of the algorithm.
  Contrary to dynamic testing: The algorithm is executed for given inputs.

Verification of Propositions about Algorithms

- Structural inference rules enable further logical conclusions
  {level+1 ≤ max} level := level + 1; **{level ≤ max}**
  due to assignment inference rule

- Program verification may prove that
  - a proposition about states holds at a particular program location
  - an invariant holds before and after the execution of a program block
  - an algorithm computes the required output for every allowed input
  e.g. $\{a, b \in N\}$ Eucledean Algorithm $\{x = gcd(a, b)\}$
  - a loop terminates

- An algorithm and the corresponding propositions are constructed together

Preview of Concepts

- Propositions characterise states of an execution
- We will write algorithms in pseudo code
- Applications of structural inference rules
- Loop invariants
- Chain inferences of already verified properties
- Proofs of loop termination

Preview Example: Verification of the Euclidean Algorithm

Precondition: $x, y \in N$, let G be the greatest common divisor (gcd) of x and y

Postcondition: $a = G$

Algorithm with

```
    a := x; b:= y;
while a ≠ b do
    if a > b :

        a := a - b;
    else

        b := b - a;
```

{Proposition over variables}:

$\{INV: G = gcd(a,b) \land a>0 \land b>0\}$

$\{INV \land a \neq b\}$

$\{G = gcd(a,b) \land a>0 \land b>0 \land a>b\}$

$\Rightarrow \{ G = gcd(a-b,b) \land a-b>0 \land b>0\}$

$\{INV\}$

$\{G = gcd(a,b) \land a>0 \land b>0 \land b>a\}$

$\Rightarrow \{G = gcd(a,b-a) \land a>0 \land b-a>0\}$

$\{INV \land a=b\} \Rightarrow \{a = G\}$

Termination

Notation for instructions

Instruction type	Notation	Example
Sequence	Instruction1; Instruction2	a := x; b := y
Assignment	Variable := Expression	a := x

Alternative	if Condition :
	    Instruction1
	else
	    Instruction2
		falls a > b :
		    a := a-b
		sonst
		    b := b-a

Conditional statement	if Condition :
	    Instruction
		falls a > b :
		    a := a-b

Subroutine	Sub()	gcd()

Loop	while Condition do
	    Instruction
		solange a ≠ b do
		    falls a > b : ...

Pre- and Postconditions of Instructions

{P} A1 {Q} A2 {R}

precondition of A1

postcondition of A1
precondition of A2

postcondition A2

- To verify an algorithm, we need to prove a triple for every instruction A
  {P} A {Q}
  If the proposition P holds before the execution of the instruction A, then Q holds after the execution of A, given that A terminates

- The propositions can be composed according to the structure of A
  For every type of instruction, one inference rule

- A specification provides a pre- and postcondition for the whole algorithm
  {Precondition} Algorithm {Postcondition}

Assignment Inference Rule

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

Substitution - x is substituted by e

In order to prove that the proposition P holds for x after the assignment, one must prove that the same statement P holds for e before the assignment!

Sequence Inference Rule

$$\frac{\{P\}\ A1\ \{Q\} \qquad \{Q\}\ A2\ \{R\}}{\{P\}\ A1;A2\ \{R\}}$$

If {P} A1 {Q} and {Q} A2 {R} are correct triples, then also {P} A1;A2 {R} is a correct triple!

Consequence Inference Rules

$$\frac{\{P\}\ A\ \{R\} \qquad R \Rightarrow Q}{\{P\}\ A\ \{Q\}}$$

Postcondition weakening

$$\frac{P \Rightarrow R \qquad \{R\}\ A\ \{Q\}}{\{P\}\ A\ \{Q\}}$$

Precondition strengthening

Alternative Inference Rule

$$\frac{\{P \wedge C\}\ A1\ \{Q\} \qquad \{P \wedge \neg C\}\ A2\ \{Q\}}{\{P\}\ \text{If } C\text{: } A1 \text{ else } A2\ \{Q\}}$$

From the common precondition P both branches lead to the same postcondition Q!

Conditional Inference Rule

$$\frac{\{P \wedge C\} \quad A1\ \{Q\} \qquad \{P \wedge \neg C\} \Rightarrow \{Q\}}{\{P\}\ \text{If } C\text{: } A1\ \{Q\}}$$

From the common precondition P both the instruction and the implication lead to the same postcondition Q!

Call Inference Rule

{P} Sub() {Q}

The subroutine Sub has no parameters and produces no output. Its effect on global variables is specified with the preconditionon P and the postcondition Q. Then this triple holds!

Due to no parameters and output, the use of this rule is limited.

Loop Inference Rule

$$\frac{\{INV \wedge C\}\ L\ \{INV\}}{\{INV\}\ \text{while}\ C\ \text{do}\ L\ \{INV \wedge \neg C\}}$$

INV is a loop invariant, i.e., it holds:
- before the loop,
- before and after any execution of L and
- after the loop

Loop termination

- The termination of a loop while C do L
  must be proven separately
  1. Find an integer expression E over the loop variables
  and show that every iteration of L reduces the value of E
  2. Show that E is bounded from below, e.g. that $E \geq 0$ is a consequence of the loop invariant.
  one may also take another bound (not just 0), E may also increase with every loop itertion and be bounded from above!

- Nontermination can be proven by showing
  1. that $R \wedge C$ is a pre- and postcondition of L
  2. that there exists an input for which $R \wedge C$ holds before the loop
  R may characterise a particular state in which the loop does not terminate

- There exist loops for which one can not decide if they terminate or not.

Exercise on Invariants

- There are b black and w white balls in a pot and $b + w > 0$ ($b \geq 0$, $w \geq 0$)

  while there are at least 2 balls in the pot
      take two arbitrary balls out of the pot
      if they have the same color:
          throw both away
          add a new black ball to the pot
      else
          return the white ball to the pot and
          throw the black ball away

- What is the color of the last ball that remains in the pot?

- Find invariants that answer this question!