

CS 6110 – Formal Methods in System Design | Spring 2015
Feb-18

Lecture 7 Verification Conditions II

Zvonimir Rakamarić
University of Utah

Last Time

- ▶ Simple command language
- ▶ Basic verification condition generation
- ▶ Weakest preconditions

Simple Command Language

$x := E$

`havoc` x

`assert` P

`assume` P

$S ; T$ [sequential composition]

$S \square T$ [choice statement]

Weakest Preconditions Cookbook

- ▶ $\text{wp}(x := E, Q) = Q[E / x]$
- ▶ $\text{wp}(\text{havoc } x, Q) = (\forall x. Q)$
- ▶ $\text{wp}(\text{assert } P, Q) = P \wedge Q$
- ▶ $\text{wp}(\text{assume } P, Q) = P \rightarrow Q$
- ▶ $\text{wp}(S ; T, Q) = \text{wp}(S, \text{wp}(T, Q))$
- ▶ $\text{wp}(S \square T, Q) = \text{wp}(S, Q) \wedge \text{wp}(T, Q)$

Checking Correctness with wp

```

{true}
wp(x := 1, x + 2 = 3) = 1 + 2 = 3 ∧ true
x := 1;
wp(y := x + 2, y = 3) = x + 2 = 3 ∧ true
y := x + 2;
wp(assert y = 3, true) = y = 3 ∧ true
assert y = 3;
{true}

```

Check: $\text{true} \rightarrow 1 + 2 = 3 \wedge \text{true}$

Absolute Value Example

```

if (x >= 0) {
  abs_x := x;
} else {
  abs_x := -x;
}
assert abs_x >= 0;

```

This Time

- ▶ If statements
- ▶ Design by contract + Dafny
- ▶ Procedures

Structured if Statement

- ▶ Just a “syntactic sugar”:
if E then S else T
gets desugared into
(assume E ; S) □ (assume ¬E ; T)

Absolute Value Example

```
if (x >= 0) {  
    abs_x := x;  
} else {  
    abs_x := -x;  
}  
assert abs_x >= 0;
```

Design by Contract

- ▶ Also called **assume-guarantee reasoning**
- ▶ Developers annotate software components with **contracts** (formal specifications)
 - ▶ Document developer's intent
 - ▶ Complex system verification broken down into compositional verification of each component
- ▶ Typical contracts
 - ▶ Annotations on procedure boundaries
 - ▶ Preconditions
 - ▶ Postconditions
 - ▶ Annotations on loop boundaries
 - ▶ Loop invariants

Design by Contract cont.

- ▶ First used in Eiffel [Bertrand Meyer]
- ▶ Native support:
 - ▶ Eiffel, Racket, SPARK Ada, Spec#, Dafny,...
- ▶ Third-party support:
 - ▶ Code Contracts project for .NET
 - ▶ Java Modeling Language
 - ▶ Contracts for Python
 - ▶ contracts.ruby
 - ▶ ...
- ▶ Runtime or static checking of contracts

Code Contracts Example

```
static int BinarySearch(int[] array, int value)
{
    Contract.Requires(array != null);
    ...
}
```

Spec# Example

```

static int BinarySearch(int[] a, int key)
requires forall{int i in (0: a.Length), int j in
    (i: a.Length); a[i] <= a[j]};
ensures 0 <= result ==> a[result] == key;
ensures result < 0 ==> forall{int i in (0:
    a.Length); a[i] != key};
{
    ...
}

```

Java Modeling Language (JML) Example

```

class BankingExample {
    public static final int MAX_BAL = 1000;
    private int balance;
    //@ invariant balance >= 0 && balance <= MAX_BAL;

    //@ ensures balance == 0;
    public BankingExample() { this.balance = 0; }

    //@ requires 0 < amount && amount+balance < MAX_BAL;
    //@ ensures balance == \old(balance) + amount;
    public void credit(int amount) {
        this.balance += amount;
    }
}

```

Assume-Guarantee Reasoning

- ▶ Example

 - foo() {...}

 - bar() {...foo();...}

- ▶ How to verify/check bar?

Assume-Guarantee Reasoning cont.

- ▶ Solution 1

 - ▶ Inline foo

- ▶ Solution 2

 - ▶ Write contract/specification P of foo

 - ▶ Assume P when checking bar

 - bar() {...assume P;...}

 - ▶ Guarantee P when checking foo

 - foo() {...assert P;}

- ▶ Pros/cons?

Procedure

- ▶ Procedure is a complex user-defined command
 - ▶ `procedure M(x,y,z) returns (r,s,t)`
 - ▶ `requires P`
 - ▶ `ensures Q`
 - ▶ `{S}`
- ▶ `requires` is a **precondition**
 - ▶ Predicate P has to hold at procedure entry
- ▶ `ensures` is a **postcondition**
 - ▶ Predicate Q has to hold at procedure exit
- ▶ S is procedure body (command)
- ▶ Note: assume procedures have no side-effects

Procedure Example

```
procedure abs(x) returns (abs_x)
requires -1000 < x && x < 1000
ensures abs_x >= 0
{
    if (x >= 0) {
        abs_x := x;
    } else {
        abs_x := -x;
    }
}
```

Desugaring Procedure Call

▶ procedure $M(x,y,z)$ returns (r,s,t)
 requires P
 ensures Q
 $\{S\}$

▶ call $a,b,c := M(E,F,G)$
 desugared into:

$x' := E; y' := F; z' := G;$

assert P' ;

assume Q' ;

$a := r'; b := s'; c := t';$

where:

• x',y',z',r',s',t' are fresh variables

• P' is P with x',y',z' for x,y,z

• Q' is Q with x',y',z',r',s',t' for x,y,z,r,s,t

Desugaring Call Example

```
procedure abs(x) returns (abs_x)
```

```
requires -1000 < x && x < 1000
```

```
ensures abs_x >= 0
```

```
{
  if (x >= 0) {
    abs_x := x;
  } else {
    abs_x := -x;
  }
}
```

```
call a := abs(b);
```

```
assert a >= 0;
```

Desugaring Call Example

Desugaring Procedure Implementation

- ▶ procedure $M(x,y,z)$ returns (r,s,t)
requires P
ensures Q
{ S }
- ▶ Implementation is correct if this is correct:
assume P ;
 S ;
assert Q ;

Desugaring Implementation Example

```
procedure abs(x) returns (abs_x)
requires -1000 < x && x < 1000
ensures abs_x >= 0
{
  if (x >= 0) {
    abs_x := x;
  } else {
    abs_x := -x;
  }
}
```

Desugaring Implementation Example