Software Architecture

Lecture 11

Program Correctness

Néstor Cataño

Innopolis University

Spring 2016

Programmer's Major Concerns

1. Mathematical Correctness Concern
   – Whether the program defines a proper relation between an **initial state** and a **final state**
2. Engineering concerns about efficiency
   – They are only defined in relation to an implementation

Correctness

- We are interested in mechanisms that, when started in an initial state, will end up in a final state which, as a rule, depends on the choice of the initial state

Deterministic Mechanisms

- Result depends on the choice of the input

Non-Deterministic Mechanisms

- **Choice of the input** will produce **one** of the possible outputs
- The input only fixes the **class** of possible outputs

The Idea

- We would like to know the set of initial states such that the use of the mechanism will result in a final state satisfying a so-called post-condition

The Idea

- For example, we would like to know the set of initial states such that executing `x:= x + 5` will result in a final state satisfying the post-condition $x \geq 13$

The Weakest PreCondition

- The condition that characterizes the set of all initial states such that the use of the mechanism will leave the system in a state satisfying the post-condition is called the **Weakest PreCondition** for that post-condition

The Weakest PreCondition

- We call it **weakest** because the weaker a condition the more states satisfy it and we are aim here at characterizing **all** possible starting states that lead to a desired state

Weaker Conditions

- `c1` is weaker than `c2` if `c2` implies `c1`
  - `x > 0` is weaker than `x > 5`

Correctness Formula

- Let `prog` be program code.
  A **correctness formula** is
  an expression of the form

```
{P} prog {Q}
```

"Any execution of `prog` starting in a state where `P` holds, will terminate in a state where `Q` holds".

Correctness Formula

- `{P} prog {Q}`
  - `prog` is program code
  - `P` is a PreCondition
  - `Q` is a post-condition

```
{x ≥ 7} x:= x + 5 {x ≥ 13}
```

Correctness Formula

- Is this formula correct ?

```
{x ≥ 10} x:= x + 5 {x ≥ 13}
```

Correctness Formula

- Is this formula correct ?

$$\{x \geq 10\}\ x := x + 5\ \{x \geq 13\}$$

- The formula holds whenever $x \geq 10$ is true before $x := x + 5$ is executed, the condition $x \geq 13$ holds after

Correctness Formula

- When is this formula correct ?

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

Correctness Formula

- When is this formula correct ?

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

- It is correct if an only if

$$P \implies x \geq 8$$

Correctness Formula

- When is this formula correct ?

```
{P} x:= x + 5 {Q}
```

Correctness Formula

- When is this formula correct ?

```
{P} x:= x + 5 {Q}
```

- the formula is correct when

```
P ==> Q[x \ x + 5]
```

Weakest Precondition

- What is the weakest precondition **P** making the following formula true ?

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

Weakest Precondition

- What is the weakest precondition **P** making the following formula true ?

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

$$(x \geq 13)[x \setminus x + 5]$$

Weakest Precondition

- The condition characterizing all the initial states so that executing a program `prog` will result in a final state satisfying a post-condition `Q`, will be called the **weakest PreCondition**, denoted by

`WP``(prog,Q)`

A Remark

- We are sometimes not interested in finding the exact form of `WP(prog,Q)` but would be content with a stronger condition `C`, that is, a condition for which

```
C ==> WP(prog,Q)
```

Weakest Precondition Calculus

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

Weakest Precondition Calculus

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

$$(x \geq 13)[x \setminus x + 5]$$

Weakest Precondition Calculus

$\{P\}\ x := x + 5\ \{x \geq 13\}$

$(x \geq 13)[x \setminus x + 5]$

$x + 5 \geq 13$

Weakest Precondition Calculus

$\{P\}\ x := x + 5\ \{x \geq 13\}$

$(x \geq 13)[x \backslash x + 5]$

$x + 5 \geq 13$

$x \geq 8$

Key Remark

- $x \geq 8$ is the weakest condition P such that the following correctness formula holds

  $$\{P\}\ x := x + 5\ \{x \geq 13\}$$

- In particular
  - $x \geq 10 \implies x \geq 8$
  - $x \geq 15 \implies x \geq 8$

Results On WP

Results On WP

1. `WP(prog,false)= false`
2. `Q1 ==> Q2` then
   `WP(prog,Q1)==> WP(prog,Q2)`
3. `WP(prog,Q)` <u>`and`</u> `WP(prog,R)`
   `= WP(prog,Q` <u>`and`</u> `R)`
4. `WP(prog,Q)` <u>`or`</u> `WP(prog,R)`
   `==> WP(prog,Q` <u>`or`</u> `R)`

WP Calculus → Assignment Rule

```
WP(x:=E,Q) = Q[x\E]
```

WP Calculus → Assignment Rule

```
WP(x:=E,Q) = Q[x\E]
```

```
{P} x:= x + 5 {x ≥ 13}
```

WP Calculus → Assignment Rule

```
WP(x:=E,Q) = Q[x\E]
```

```
{P} x:= x + 5 {x ≥ 13}
```

```
WP(x:= x + 5,x ≥ 13) = (x≥13)[x\x+5]
```

WP Calculus → Assignment Rule

$$WP(x:=E,Q) = Q[x\backslash E]$$

$$\{P\}\ x := x + 5\ \{x \geq 13\}$$

$$WP(x := x + 5, x \geq 13) = (x \geq 13)[x\backslash x+5]$$

$$x \geq 8$$

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

```
{P} x:=z+1; y:=x+y {y > 5}
```

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

```
{P} x:=z+1; y:=x+y {y > 5}
```

```
WP(x:=z+1; y:=x+y, y > 5)
```

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

```
{P} x:=z+1; y:=x+y {y > 5}
```

```
WP(x:=z+1; y:=x+y, y > 5)
```

```
WP(x:=z+1, WP(y:=x+y, y > 5))
```

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

```
{P} x:=z+1; y:=x+y {y > 5}
```

```
WP(x:=z+1; y:=x+y, y > 5)
```

```
WP(x:=z+1, x+y > 5)
```

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

```
{P} x:=z+1; y:=x+y {y > 5}
```

```
WP(x:=z+1; y:=x+y, y > 5)
```

```
z+1+y > 5
```

WP Calculus→ Composition Rule

```
WP(S;T,Q) = WP(S,WP(T,Q))
```

```
{P} x:=z+1; y:=x+y {y > 5}
```

```
WP(x:=z+1; y:=x+y, y > 5)
```

```
z+y > 4
```

WP Calculus → Implication Rule

```
WP(if (C) S else T, Q) =
```

WP Calculus → Implication Rule

```
WP(if (C) S else T, Q) =
```

```
 C => WP(S,Q) &&
!C => WP(T,Q)
```

WP Calculus → Implication Rule

```
WP(if (C) S, Q)
```

WP Calculus → Implication Rule

```
WP(if (C) S, Q)
```

```
C => WP(S,Q)
```

WP Calculus

```
{P} if(x>y) z:=x else z:=y {z>0}
```

WP Calculus

```
{P} if(x>y) z:=x else z:=y {z>0}
```

```
x>y => WP(z:=x,z>0) &&
x≤y => WP(z:=y,z>0)
```

WP Calculus

```
{P} if(x>y) z:=x else z:=y {z>0}
```

```
x>y => x>0 &&
x≤y => y>0
```

WP Calculus → Implication Rule

```
WP(skip, Q) = Q
```

WP Calculus → Abort Rule

```
WP(abort, Q) = false
```

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

1. The Loop-Invariant holds initially

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

1. The Loop-Invariant holds initially

```
{true} B {I}
```

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

1. The Loop-Invariant holds initially

```
WP(B,I) = true
```

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

2. Program `prog` maintains the Loop-Invariant `I` provided that the guard `C` holds as well

$$\{I \wedge C\}\ prog\ \{I\}$$

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

2. Program `prog` maintains the Loop-Invariant `I` provided that the guard `C` holds as well

$$(I \wedge C) \Rightarrow WP(prog, I)$$

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Loop-Invariant I
Variant V
```

3. The `Variant` is strictly decreased by the execution of `prog` provided that the invariant `I` and the guard `C` hold

$$\{I \land C\}\ prog\ \{0 \leq V < V_0\}$$

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

Post-condition **Q**
Invariant **I**
Variant **V**

3. The Va…y the exe…t the inva…d

Remark!

V_0 is **V** before executing **prog**

$\{I \wedge C\}$ prog $\{0 \leq V < V_0\}$

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Invariant I
Variant V
```

3. The `Variant` is strictly decreased by the execution of `prog` provided that the invariant `I` and the guard `C` hold

$$(I \wedge C) \Rightarrow WP(prog, 0 \leq V < V_0)$$

When a while loop is Correct?

```
B
while( C ) {
   prog
}
```

```
Post-condition Q
Invariant I
Variant V
```

4. The post-condition holds after the loop ends

$$(I \wedge \neg C) \Rightarrow Q$$

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant **$zy+x = M$**

Post-condition
$zy+x = M \wedge x < y$

Variant **x**

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant $zy+x = M$

Post-condition
$zy+x = M \wedge x < y$

Variant x

1. WP(B,I) = true

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant $zy+x = M$

Post-condition
$zy+x = M \wedge x < y$

Variant x

1. $WP(x:=M;\ y:=N;\ z:=0,\ zy+x=M) = true$

Is this Program Correct?

```
WP(x:=M; y:=N; z:=0,zy+x=M) =
```

Is this Program Correct?

```
WP(x:=M; y:=N; z:=0,zy+x=M) =
```

```
WP(x:=M; y:=N,0y+x=M) =
```

Is this Program Correct?

```
WP(x:=M; y:=N; z:=0,zy+x=M) =
```

```
WP(x:=M; y:=N,0y+x=M) =
```

```
WP(x:=M,0N+x=M) =
```

Is this Program Correct?

```
WP(x:=M; y:=N; z:=0,zy+x=M) =
```

```
WP(x:=M; y:=N,0y+x=M) =
```

```
WP(x:=M,0N+x=M) =
```

```
0N+M=M =
```

Is this Program Correct?

```
WP(x:=M; y:=N; z:=0,zy+x=M) =
```

```
WP(x:=M; y:=N,0y+x=M) =
```

```
WP(x:=M,0N+x=M) =
```

```
0N+M=M =
```

```
true
```

Is Is this Program Correct?Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant $zy+x = M$

Post-condition
$zy+x = M \wedge x < y$

Variant x

2. $(I \wedge C) \Rightarrow WP(prog, I)$

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant zy+x = M

Post-condition
 zy+x = M ∧ x < y

Variant x

2. (zy+x=M) ∧ (x≥y) =>
    WP(z:= z+1; x:= x-y, zy+x=M)

Is this Program Correct?

$$(zy+x=M) \wedge (x \geq y) \Rightarrow$$
$$WP(z:=z+1;\ x:=x-y,\ zy+x=M)$$

Is this Program Correct?

```
(zy+x=M) ∧ (x≥y) =>
      WP(z:= z+1; x:= x-y, zy+x=M)
```

```
WP(z:= z+1; x:= x-y, zy+x=M)=
```

Is this Program Correct?

```
(zy+x=M) ∧ (x≥y) =>
      WP(z:= z+1; x:= x-y, zy+x=M)
```

```
WP(z:= z+1; x:= x-y, zy+x=M)=
```

```
WP(z:= z+1, zy+x-y=M) =
```

Is this Program Correct?

```
(zy+x=M) ∧ (x≥y) =>
      WP(z:= z+1; x:= x-y, zy+x=M)
```

```
WP(z:= z+1; x:= x-y, zy+x=M)=
```

```
WP(z:= z+1, zy+x-y=M) =
```

```
(z+1)y+x-y=M
```

Is this Program Correct?

```
(zy+x=M) ∧ (x≥y) =>
      WP(z:= z+1; x:= x-y, zy+x=M)
```

```
WP(z:= z+1; x:= x-y, zy+x=M)=
```

```
WP(z:= z+1, zy+x-y=M) =
```

```
(z+1)y+x-y=M
```

```
zy+x=M
```

Is this Program Correct?

$$(zy+x=M) \wedge (x \geq y) \Rightarrow$$
$$zy+x=M$$

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant $zy+x = M$

Post-condition $zy+x = M \wedge x < y$

Variant x

3. $(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$

Is this Program Correct?

$$(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$$

Is this Program Correct?

$(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$

$(z_0 y + x_0 = M) \wedge (x_0 \geq y) \Rightarrow WP(prog_0, 0 \leq X < X_0)$

Is this Program Correct?

$(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$

$(z_0 y + x_0 = M) \wedge (x_0 \geq y) \Rightarrow WP(prog_0, 0 \leq X < X_0)$

$WP(z := z_0 + 1;\ x := x_0 - y, 0 \leq X < X_0) =$

Is this Program Correct?

$(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$

$(z_0 y + x_0 = M) \wedge (x_0 \geq y) \Rightarrow WP(prog_0, 0 \leq x < x_0)$

$WP(z := z_0 + 1;\ x := x_0 - y, 0 \leq x < x_0) =$

$WP(z := z_0 + 1, 0 \leq x_0 - y < x_0) =$

Is this Program Correct?

$(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$

$(z_0 y + x_0 = M) \wedge (x_0 \geq y) \Rightarrow WP(prog_0, 0 \leq x < x_0)$

$WP(z := z_0 + 1;\ x := x_0 - y, 0 \leq x < x_0) =$

$WP(z := z_0 + 1, 0 \leq x_0 - y < x_0) =$

$0 \leq x_0 - y < x_0$

Is this Program Correct?

$$(I_0 \wedge C_0) \Rightarrow WP(prog_0, 0 \leq V < V_0)$$

$$(z_0 y + x_0 = M) \wedge (x_0 \geq y) \Rightarrow 0 \leq x_0 - y < x_0$$

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant **zy+x = M**

Post-condition
zy+x = M ∧ x < y

Variant **x**

4.(I ∧ ¬C) => Q

Is this Program Correct?

```
x:= M;
y:= N;
z:= 0;
while( x≥y ) {
  z:= z + 1;
  x:= x - y;
}
```

Invariant $zy+x = M$

Post-condition
$zy+x = M \wedge x < y$

Variant x

4. $(zy+x = M) \wedge \neg(x \geq y) \Rightarrow$
$zy+x = M \wedge x < y$

What does A(x,y) calculate?

```
function A(x, y)
  if x = 0 return y
  while y ≠ 0 {
   if x > y
     x := x - y
   else
     y := y - x
  }
  return x
```

Example

- `x = 6, y = 9`
- `A(x,y) = ?`

Example

- `x = 6, y = 9`
- `A(6,9) = ?`

Example

- `x = 6, y = 9`
- `A(6,9) =`
- `A(6,3)`

Example

- `x = 6, y = 9`
- `A(6,9) =`
- `A(6,3) =`
- `A(3,3)`

Example

- `x = 6, y = 9`
- `A(6,9) =`
- `A(6,3) =`
- `A(3,3) = 3`

The Greatest Common Divisor
Euclid's Algorithm

```
function GCD(x, y){
  while !(x = y) {
   if x > y
      x := x - y
   else
      y := y - x
  }
  return x
}
```