

CS 6110 – Formal Methods in System Design | Spring 2015
Mar-2

Lecture 8 Loops and Loop Invariants

Zvonimir Rakamarić
University of Utah

slides acknowledgements: Z. Manna, R. Leino


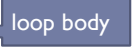
Last Time

- ▶ Design by contract
- ▶ Procedures

This Time

- ▶ Loops
- ▶ Loop Invariants

While Loop

```
while E   
do  
  S   
end
```

- ▶ Loop body S executed as long as loop condition E holds

Desugar While Loop by Unrolling N Times

```

while E do S end =
if E {
  S;
  if E {
    S;
    if E {
      S;
      if E {assume false;} // blocks execution
    }
  }
}
}

```

Example

```

i := 0;
while i < 2 do i := i + 1 end

i := 0;
if i < 2 {
  i := i + 1;
  if i < 2 {
    i := i + 1;
    if i < 2 {
      i := i + 1;
      if i < 2 {assume false;} // blocks execution
    }
  }
}
}

```

First Issue with Unrolling

```

i := 0;
while i < 4 do i := i + 1 end

i := 0;
if i < 4 {
  i := i + 1;
  if i < 4 {
    i := i + 1;
    if i < 4 {
      i := i + 1;
      if i < 4 {assume false;} // blocks execution
    }
  }
}
}

```

Second Issue with Unrolling

```

i := 0;
while i < n do i := i + 1 end

i := 0;
if i < n {
  i := i + 1;
  if i < n {
    i := i + 1;
    if i < n {
      i := i + 1;
      if i < n {assume false;} // blocks execution
    }
  }
}
}

```

While Loop with Invariant

```

while E
  invariant J
do
  S
end

```

Annotations:

- loop condition (points to E)
- loop invariant (points to J)
- loop body (points to S)

- ▶ Loop body S executed as long as loop condition E holds
- ▶ Loop invariant J must hold on every iteration
 - ▶ J must hold initially and is evaluated before E
 - ▶ J must hold even on final iteration when E is false
 - ▶ Provided by a user or inferred automatically

Desugaring While Loop Using Invariant

- ▶ while E invariant J do S end

```

assert J;
havoc x; assume J;
(
  assume E; S; assert J; assume false
□
  assume ¬E
)

```

Annotations:

- check that the loop invariant holds initially (points to `assert J;`)
- jump to an arbitrary iteration of the loop (points to `assume J;`)
- where x denotes the assignment targets of S (points to `havoc x;`)
- check that the loop invariant is maintained by the loop body (points to `assert J;`)
- exit the loop (points to `assume ¬E`)

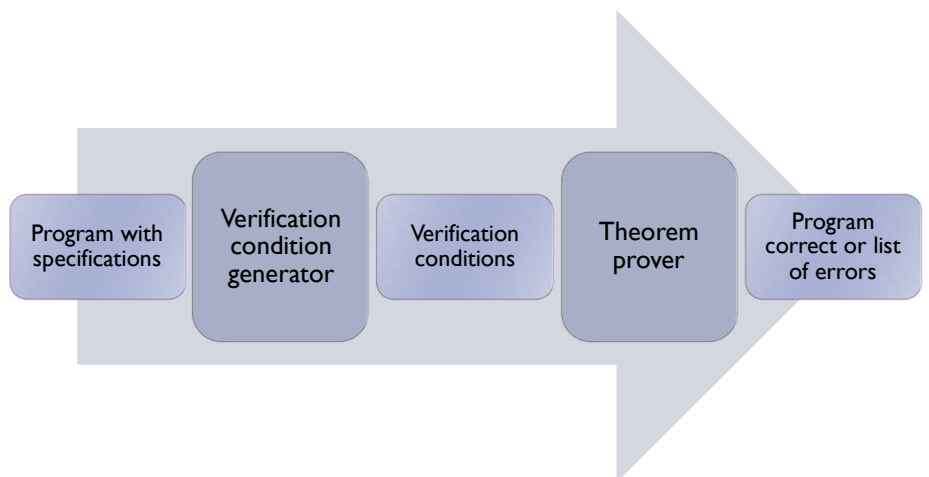
Weakest Precondition of While

- ▶ $\text{wp}(\text{while } E \text{ invariant } J \text{ do } S \text{ end}, Q) =$

Dafny

- ▶ Simple “verifying compiler”
 - ▶ Proves procedure contracts statically for all possible inputs
 - ▶ Uses theory of weakest preconditions
- ▶ Input
 - ▶ Annotated program written in simple imperative language
 - ▶ Preconditions
 - ▶ Postconditions
 - ▶ Loop invariants
- ▶ Output
 - ▶ Correct or list of failed annotations

Dafny Architecture



Next Time

- ▶ Program correctness: strategies