Lecture 10
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 condition

loop body

- 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
```

loop condition

loop invariant

loop body

- 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

check that the loop invariant holds initially

assert J;

jump to an arbitrary iteration of the loop

havoc x; assume J;

where x denotes the assignment targets of S

(

assume E; S; assert J; assume false

□

assume ¬E

)

check that the loop invariant is maintained by the loop body

exit the loop

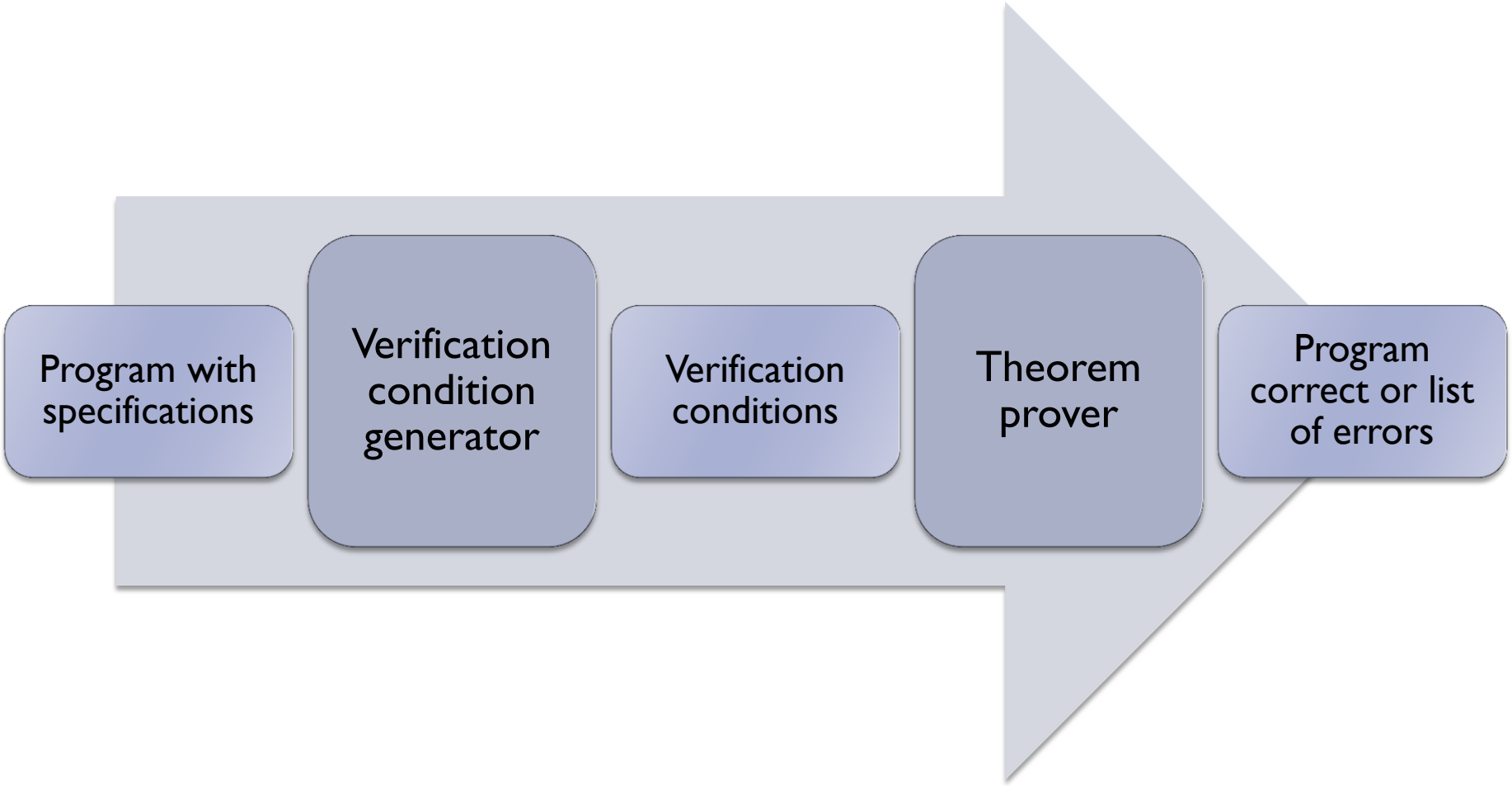
Weakest Precondition of While

- wp(while E invariant J do S 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