Lecture 8

Loops and Loop Invariants

Zvonimir Rakamarić
University of Utah

slides acknowledgements: Z. Manna, R. Leino

Announcements

- Homework 2 due at midnight today
- Project proposals due on Wednesday

Project Proposal

- At most 1 page per team
  - Title
  - Team members (at most 2)
  - Related work
  - Proposed work with basic milestones
- Single-columned, single-spaced, 11pt font, 1 inch margins
- I will accept only **PDF** files

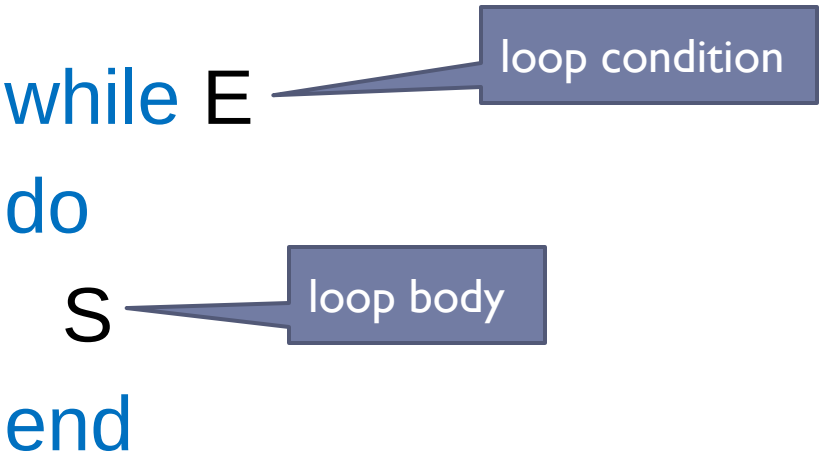
Last Time

- Design by contract
- Procedures

This Time

- Loops
- Loop Invariants
- PiVC verifying compiler with examples

While Loop



- 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
```

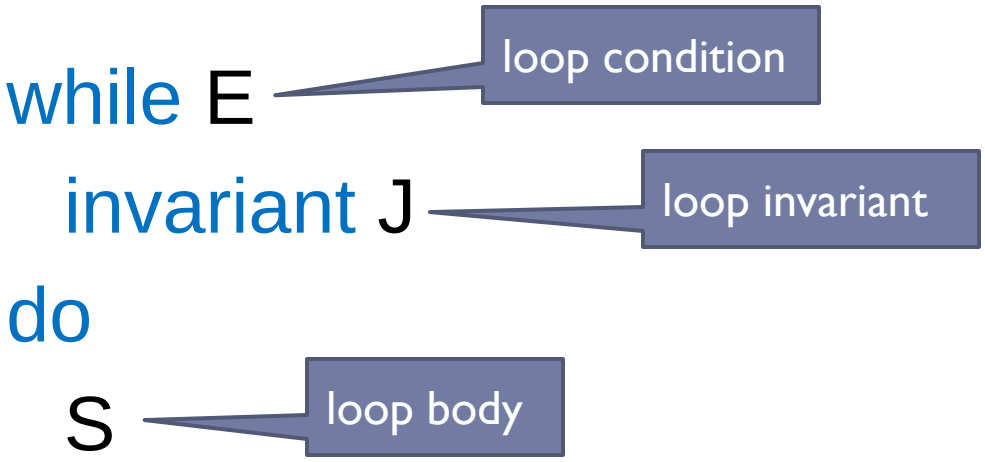
```
i := 0;
if i < 4 {
   i := i + 1;
   if i < 4 {
      i := i + 1;
      if i < 4 {
         i := i + 1;
         if i < 4 {assume false;} // blocks execution
      }
   }
}
```

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
      }
   }
}
```

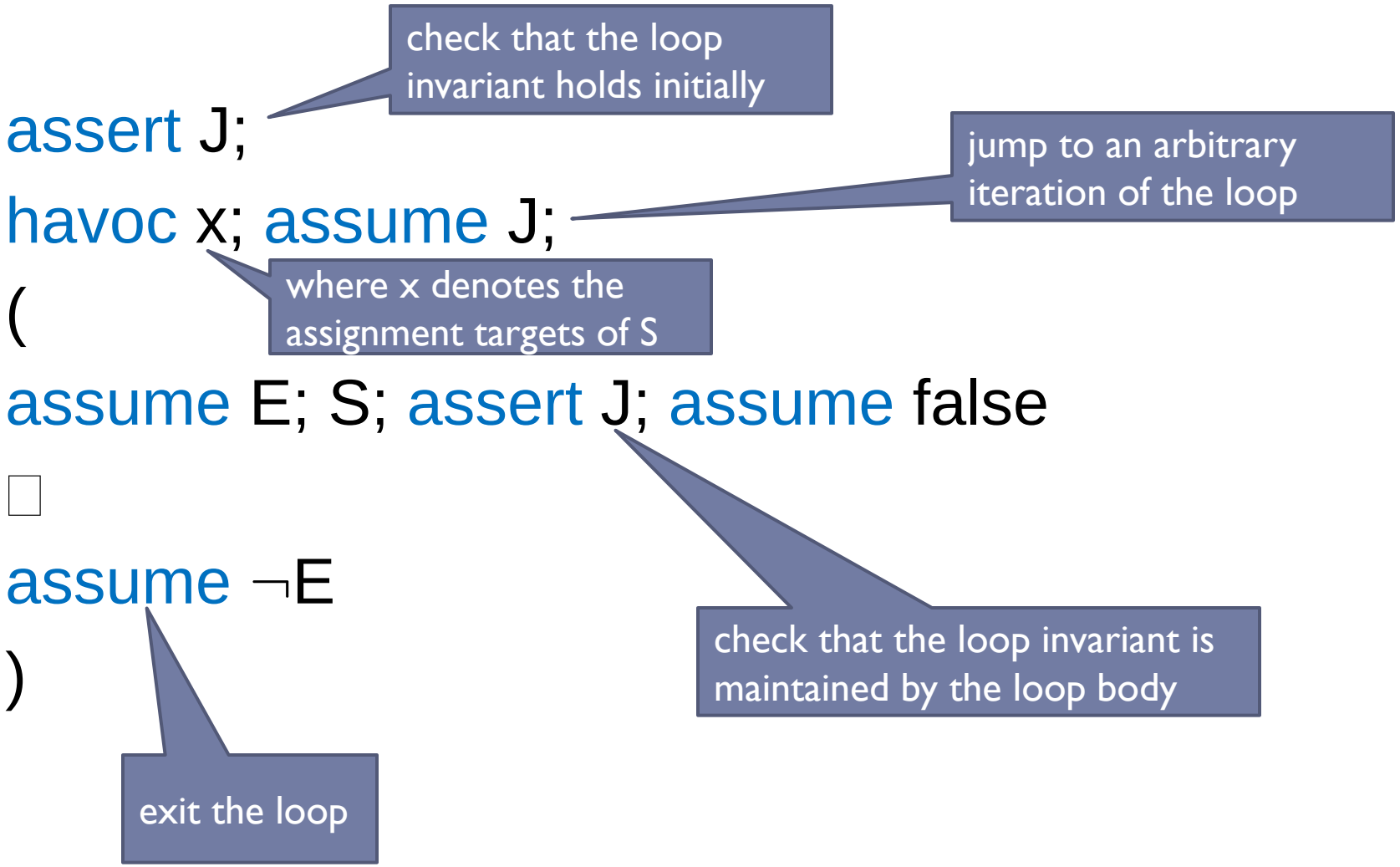
While Loop with Invariant



- 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



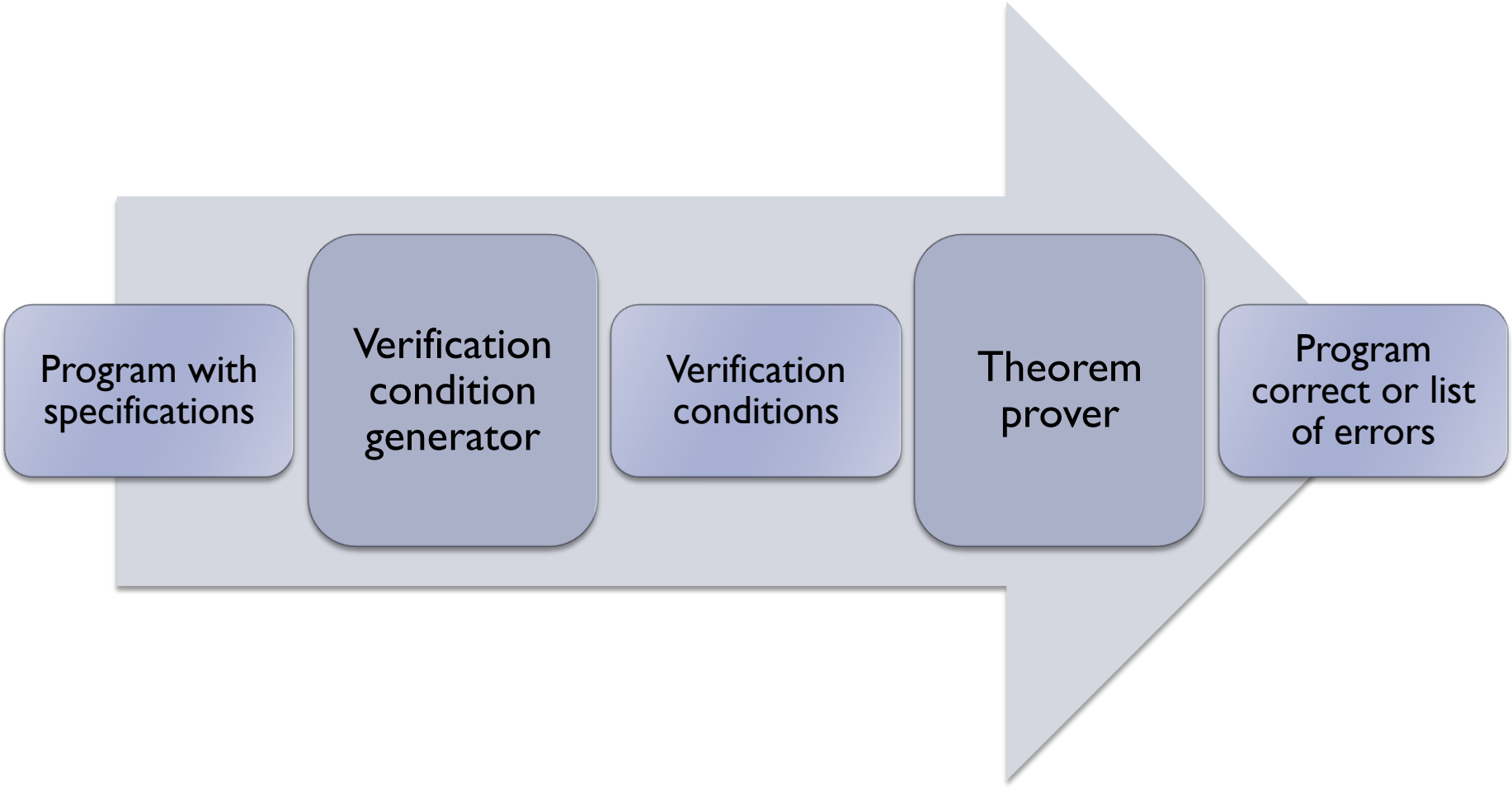
Weakest Precondition of While

- wp(while E invariant J do S end, Q) =

PiVC

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

PiVC Architecture



Pi Language

- Types
  - Basic types: `int`, `bool`
  - Array types: `int[]`, `bool[]`
- Constants
  - Integer constants: `-2`, `-1`, `0`, `1`, `2`...
  - Boolean constants: `true`, `false`
- Variable declaration
  - `int x;` `int[] a;`
- Assignment
  - `a := b;` `a[i] := b[j];`
- Array size
  - `|a|`

Functions in Pi

```
@pre x > 0 && y > 0
@post rv < 0
int foo(int x, int y, int[] a) {
 int z;
 z := -5;
 return z;
}
```

While Loop in Pi

```
i := 0;
while
   @L: 0 <= i && i <= n
   (i < n)
{
   a[i] := i;
   i := i + 1;
}
```

For Loop in Pi

```
for
   @L: 0 <= i && i <= n
   (int i := 0; i < n; i := i + 1)
{
   a[i] := i;
}
```

Quantifiers in Pi

```
exists x. (0 <= x && x <= 10 &&
  a[x] = 0)

forall x. (0 <= x && x <= 10 ->
  a[x] = 0)
```

Example: Absolute Value

- Signature:

  `int abs(int n)`

- Specification:
  - Returns absolute value of `n`

Example: Swap

- Signature:

```
int[] swap(int[] a, int i, int j)
```

- Specification:
  - Swaps elements of array `a` at indices `i` and `j` and returns the updated array

(Dumb) Example: Multiply by 2

```
int Multiply2(int n) {
 int r := 0;
 for
  (int i := 0; i < n; i := i + 1)
 {
  r := r + 2;
 }
 return r;
}
```

- Specification:
  - Given a non-negative integer `n`, function `Multiply2` multiplies it by 2

Example: Initialize Array

- Signature:

```
int[] InitializeArray(int[] a, int e)
```

- Specification:
  - Initializes elements of array `a` to `e` and returns the updated array

Example: Linear Search I

- Signature:

  ```
  bool LinearSearch(int[] a, int e)
  ```

- Specification:
  - Returns `true` if `e` is found in array `a`, otherwise returns `false`

Example: Linear Search II

- Signature:

```
bool LinearSearch(int[] a, int l,
    int u, int e)
```

- Specification:
  - Returns `true` if `e` is found in array `a` between `l` and `u`, otherwise returns `false`

Next Time

- Program correctness: strategies