Formal Methods and Specification (LS 2021)
Lecture 10: Automatic Synthesis of Loop Invariants

Stefan Ratschan

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

Evropský sociální fond Praha & EU: Investujeme do vaší budoucnosti

This Lecture

Verification conditions: automatic check by decision procedures

This Lecture

Verification conditions: automatic check by decision procedures

But: loop invariants needed

This Lecture

Verification conditions: automatic check by decision procedures

But: loop invariants needed

Synthesis of loop invariants is an active research area

This Lecture

Verification conditions: automatic check by decision procedures

But: loop invariants needed

Synthesis of loop invariants is an active research area

This lecture: some basic techniques

Weakest Preconditions

Spec: Input:
Output: O

Program P

Weakest Preconditions

Spec: Input:
Output: O

Program P

Question: For which condition I is program P correct?

Weakest Preconditions

Spec: Input: Output: O Program P

Question: For which condition I is program P correct? $\perp$!

Weakest Preconditions

Spec: Input: Output: O Program P

Question: For which condition I is program P correct? $\perp$! weakest one?

Weakest Preconditions

Spec: Input: Output: O Program P

Question: For which condition I is program P correct? $\perp$! weakest one?

Equivalent question:

Which **assume** at the beginning of P ensures correctness of $P; @O$?

Weakest Preconditions

Spec: Input:
Output: O

Program P

Question: For which condition I is program P correct? $\perp$! weakest one?

Equivalent question:
Which **assume** at the beginning of P ensures correctness of $P; @O$?

Example:

assume $I(x, y, z)$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
$@\ z \geq 0$

Weakest Preconditions

Spec: Input:
Output: O

Program P

Weakest Preconditions

Spec: Input: Output: O

Program P

A formula I is a *precondition* of a program P and a formula O iff

- every free variable of I is a program variable of P and
- the program **assume** $I; P; @O$ is correct

Weakest Preconditions

Spec: Input:
Output: O

Program P

A formula I is a *precondition* of a program P and a formula O iff

- every free variable of I is a program variable of P and
- the program **assume** $I; P; @O$ is correct

Every precondition I of a program P and a formula O is
a *weakest precondition* of P and O iff
for every precondition I' of P and O, $\models I' \Rightarrow I$.

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $z \geq 0$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Result:

$$\forall x_1, y_1, z_1 \,.\, [y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0?$$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
$@\ z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Result:

$$\forall x_1, y_1, z_1 \,.\, [y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0?$$

Simplification:

$$\forall x_1, y_1 \,.\, [y_1 = x \wedge x_1 = -10] \Rightarrow x_1 + y_1 \geq 0$$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
$@\ z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Result:

$$\forall x_1, y_1, z_1 \,.\, [y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0?$$

Simplification:

$$\forall x_1, y_1 \,.\, [y_1 = x \wedge x_1 = -10] \Rightarrow x_1 + y_1 \geq 0$$
$$\forall x_1 \,.\, y_1 = x \Rightarrow -10 + y_1 \geq 0$$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Result:

$$\forall x_1, y_1, z_1 \,.\, [y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0?$$

Simplification:

$\forall x_1, y_1 \,.\, [y_1 = x \wedge x_1 = -10] \Rightarrow x_1 + y_1 \geq 0$
$\forall x_1 \,.\, y_1 = x \Rightarrow -10 + y_1 \geq 0$
$-10 + x \geq 0$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
$@\ z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Result:

$$\forall x_1, y_1, z_1 \,.\, [y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0?$$

Simplification:

$\forall x_1, y_1 \,.\, [y_1 = x \wedge x_1 = -10] \Rightarrow x_1 + y_1 \geq 0$
$\forall x_1 \,.\, y_1 = x \Rightarrow -10 + y_1 \geq 0$
$-10 + x \geq 0$, $x \geq 10$

Computation of Weakest Preconditions: Example

$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
$@\ z \geq 0$

Verification condition:

$$[y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0$$

Result:

$$\forall x_1, y_1, z_1 \,.\, [y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1] \Rightarrow z_1 \geq 0?$$

Simplification:

$\forall x_1, y_1 \,.\, [y_1 = x \wedge x_1 = -10] \Rightarrow x_1 + y_1 \geq 0$
$\forall x_1 \,.\, y_1 = x \Rightarrow -10 + y_1 \geq 0$
$-10 + x \geq 0$, $x \geq 10$

Check:

assume $x \geq 10$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
$@\ z \geq 0$

Computation of Weakest Preconditions

Spec:

Input:		program P without control structures
Output:	O	

Computation of Weakest Preconditions

Spec: Input: program P without control structures
Output: O

Weakest precondition in quantified form:

$$\forall \vec{v} \,.\, VC(P; @O)$$

where $\vec{v}$ is a tuple of the auxiliary variables of $VC(P; @O)$.

Computation of Weakest Preconditions

Spec:

Input:		program P without control structures
Output:	O	

Weakest precondition in quantified form:

$$\forall \vec{v} \;.\; VC(P; @O)$$

where $\vec{v}$ is a tuple of the auxiliary variables of $VC(P; @O)$.

Here: VC must use the original variable names
for variables corresponding to initial values.

Computation of Weakest Preconditions

Spec:

Input:		program P without control structures
Output:	O	

Weakest precondition in quantified form:

$$\forall \vec{v} \,.\, VC(P; @O)$$

where $\vec{v}$ is a tuple of the auxiliary variables of $VC(P; @O)$.

Here: VC must use the original variable names
for variables corresponding to initial values.

Problem: $\forall \vec{v}$

Computation of Weakest Preconditions

Spec:

Input:		program P without control structures
Output:	O	

Weakest precondition in quantified form:

$$\forall \vec{v} \,.\, VC(P; @O)$$

where $\vec{v}$ is a tuple of the auxiliary variables of $VC(P; @O)$.

Here: VC must use the original variable names
for variables corresponding to initial values.

Problem: $\forall \vec{v}$

Quantifiers corresponding to variables introduced by assignments
can be easily eliminated

Computation of Weakest Preconditions

Spec: Input: program P without control structures

Output: O

Weakest precondition in quantified form:

$$\forall \vec{v} \,.\, VC(P; @O)$$

where $\vec{v}$ is a tuple of the auxiliary variables of $VC(P; @O)$.

Here: VC must use the original variable names
for variables corresponding to initial values.

Problem: $\forall \vec{v}$

Quantifiers corresponding to variables introduced by assignments
can be easily eliminated

Quantifiers for ariables to user input or procedure calls not!

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Example:

input x
`@` $x^2 + a \geq 1$

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Example:

input x
@ $x^2 + a \geq 1$

Weakest precondition

$$\forall x \,.\, x^2 + a \geq 1$$

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Example:

input x
@ $x^2 + a \geq 1$

Weakest precondition

$$\forall x \,.\, x^2 + a \geq 1$$

$$\forall x \,.\, x^2 \geq 1 - a$$

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Example:

input x
@ $x^2 + a \geq 1$

Weakest precondition

$$\forall x \,.\, x^2 + a \geq 1$$

$$\forall x \,.\, x^2 \geq 1 - a$$

$$0 \geq 1 - a$$

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Example:

input x
@ $x^2 + a \geq 1$

Weakest precondition

$$\forall x \,.\, x^2 + a \geq 1$$

$$\forall x \,.\, x^2 \geq 1 - a$$

$$0 \geq 1 - a, a \geq 1$$

Quantifier Elimination

- Given: formula ϕ
- Find: formula ϕ' s.t. $\models \phi \Leftrightarrow \phi'$, but ϕ' is quantifier free

Example:

input x
@ $x^2 + a \geq 1$

Weakest precondition

$$\forall x \,.\, x^2 + a \geq 1$$

$$\forall x \,.\, x^2 \geq 1 - a$$

$$0 \geq 1 - a, a \geq 1$$

In general: very difficult problem

Strongest Postconditions

Spec: Input: I Output:

Program P

Strongest Postconditions

Spec: Input: I Output:

Program P

Dual question:

For which condition O is program **assume** $I; P; @O$ correct?

Strongest Postconditions

Spec: Input: I Output: Program P

Dual question:

For which condition O is program **assume** $I; P; @O$ correct?

Strongest Postconditions

Spec: Input: I Output:

Program P

Dual question:

For which condition O is program **assume** $I; P; @O$ correct?

$\top$?

Strongest Postconditions

Spec: Input: I Output: Program P

Dual question:

For which condition O is program **assume** $I; P; @O$ correct?

$\top$? strongest one?

Strongest Postconditions

Spec: Input: I Output: Program P

Dual question:

For which condition O is program **assume** $I; P; @O$ correct?

$\top$? strongest one?

A formula O is a *postcondition* of a program P and a formula I iff

- every free variable of O is a program variable of P, and
- the program **assume** $I; P; @O$ is correct

Strongest Postconditions

Spec: Input: I Output:

Program P

Dual question:

For which condition O is program **assume** $I; P; @O$ correct?

$\top$? strongest one?

A formula O is a *postcondition* of a program P and a formula I iff

- every free variable of O is a program variable of P, and
- the program **assume** $I; P; @O$ is correct

Every postcondition O of a program P and a formula I is a *strongest postcondition* of P and I iff for every postcondition O' of P and I, $\models O \Rightarrow O'$.

Computation of Strongest Postconditions: Example

assume $x \geq 10$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $O(x, y, z)$

Computation of Strongest Postconditions: Example

assume $x \geq 10$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $O(x, y, z)$

We can execute the program before the assertion iff

$$x \geq 10 \wedge y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1$$

is satisfiable (viz symbolic execution).

Computation of Strongest Postconditions: Example

assume $x \geq 10$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $O(x, y, z)$

We can execute the program before the assertion iff

$$x \geq 10 \land y_1 = x \land x_1 = -10 \land z_1 = x_1 + y_1$$

is satisfiable (viz symbolic execution). O has to hold on the resulting x_1, y_1, z_1, that is for the x_1, y_1, z_1, for which

$$\exists x, y, z \, . \, [x \geq 10 \land y_1 = x \land x_1 = -10 \land z_1 = x_1 + y_1]$$

Computation of Strongest Postconditions: Example

assume $x \geq 10$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $O(x, y, z)$

We can execute the program before the assertion iff

$$x \geq 10 \wedge y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1$$

is satisfiable (viz symbolic execution). O has to hold on the resulting x_1, y_1, z_1, that is for the x_1, y_1, z_1, for which

$$\exists x, y, z \;.\; [x \geq 10 \wedge y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1]$$

QE:

$$\exists x, y, z \;.\; [x \geq 10 \wedge y_1 = x \wedge y_1 \geq 10 \wedge x_1 = -10 \wedge z_1 = x_1 + y_1]$$

Computation of Strongest Postconditions: Example

assume $x \geq 10$
$y \leftarrow x$
$x \leftarrow -10$
$z \leftarrow x + y$
@ $O(x, y, z)$

We can execute the program before the assertion iff

$$x \geq 10 \wedge y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1$$

is satisfiable (viz symbolic execution). O has to hold on the resulting x_1, y_1, z_1, that is for the x_1, y_1, z_1, for which

$$\exists x, y, z \,.\, [x \geq 10 \wedge y_1 = x \wedge x_1 = -10 \wedge z_1 = x_1 + y_1]$$

QE:

$$\exists x, y, z \,.\, [x \geq 10 \wedge y_1 = x \wedge y_1 \geq 10 \wedge x_1 = -10 \wedge z_1 = x_1 + y_1]$$

$$y_1 \geq 10 \wedge x_1 = -10 \wedge z_1 = x_1 + y_1$$

Computation of Strongest Postconditions

Spec: Input: I program P without control structures
Output:

Computation of Strongest Postconditions

Spec: **Input:** I program P without control structures
Output:

Strongest postcondition in quantified form, with indexed variables:

$$\exists \vec{v} \,.\, F_{pre}(\textbf{assume}\ I; P)$$

where $\vec{v}$ is a tuple of the variables in $F_{pre}(\textbf{assume}\ I; P)$
not corresponding to a final value.

Computation of Strongest Postconditions

Spec: Input: I program P without control structures
Output:

Strongest postcondition in quantified form, with indexed variables:

$$\exists \vec{v} \,.\, F_{pre}(\textbf{assume } I; P)$$

where $\vec{v}$ is a tuple of the variables in $F_{pre}(\textbf{assume } I; P)$
not corresponding to a final value.

In general, no simple way of elimination quantifiers.

Computation of Strongest Postconditions

Spec: **Input:** I program P without control structures
Output:

Strongest postcondition in quantified form, with indexed variables:

$$\exists \vec{v} \, . \, F_{pre}(\textbf{assume } I; P)$$

where $\vec{v}$ is a tuple of the variables in $F_{pre}(\textbf{assume } I; P)$
not corresponding to a final value.

In general, no simple way of elimination quantifiers.

To relate to program variables, rename variables.

Finding Loop Invariants: Example

Specification:

- Input: array a
- Output: r s.t. $r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7\right]$

Finding Loop Invariants: Example

Specification:

- Input: array a
- Output: r s.t. $r \Leftrightarrow \left[\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7\right]$

Program:

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ ???
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\bot$
2	$\bot$
3	$\bot$
4	$\top$

Invariant involved in three verification conditions:

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ ???
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\bot$
2	$\bot$
3	$\bot$
4	$\top$

Invariant involved in three verification conditions:

- holds in first loop iteration

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ ???
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\bot$
2	$\bot$
3	$\bot$
4	$\top$

Invariant involved in three verification conditions:

- holds in first loop iteration
- if it holds, and the loop is re-entered, then it must hold again

Loop Invariant for Example

$r \leftarrow \perp$
for $i \leftarrow 1$ **to** n **do**
  @ ???
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\perp$
2	$\perp$
3	$\perp$
4	$\top$

Invariant involved in three verification conditions:

- holds in first loop iteration
- if it holds, and the loop is re-entered, then it must hold again
- if it holds, and the loop is left, then assertion after the loop must hold

Loop Invariant for Example

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ ???
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\bot$
2	$\bot$
3	$\bot$
4	$\top$

Invariant involved in three verification conditions:

- holds in first loop iteration
- if it holds, and the loop is re-entered, then it must hold again
- if it holds, and the loop is left, then assertion after the loop must hold

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Path:

$r \leftarrow \bot$
assume $i = 1$
@ ???

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Path:

$r \leftarrow \bot$
assume $i = 1$
@ ???

strongest postcondition

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ ???
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Path:

$r \leftarrow \bot$
assume $i = 1$
@ $i = 1 \wedge \neg r$

strongest postcondition

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $i = 1 \wedge \neg r$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $i = 1 \wedge \neg r$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Paths (without unnecessary assumes):

assume $i = 1 \wedge \neg r$
assume $a[i] = 7$
$r \leftarrow \top$
$i \leftarrow i + 1$
@ ???

assume $i = 1 \wedge \neg r$
assume $a[i] \neq 7$
$i \leftarrow i + 1$
@ ???

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ $i = 1 \wedge \neg r$
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Paths (without unnecessary assumes):

assume $i = 1 \wedge \neg r$
assume $a[i] = 7$
$r \leftarrow \top$
$i \leftarrow i + 1$
@ $a[1] = 7 \wedge r \wedge i = 2$

assume $i = 1 \wedge \neg r$
assume $a[i] \neq 7$
$i \leftarrow i + 1$
@ $a[1] \neq 7 \wedge \neg r \wedge i = 2$

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $i = 1 \wedge \neg r$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Paths (without unnecessary assumes):

assume $i = 1 \wedge \neg r$
assume $a[i] = 7$
$r \leftarrow \top$
$i \leftarrow i + 1$
@ $a[1] = 7 \wedge r \wedge i = 2$

assume $i = 1 \wedge \neg r$
assume $a[i] \neq 7$
$i \leftarrow i + 1$
@ $a[1] \neq 7 \wedge \neg r \wedge i = 2$

assume $i = 1 \wedge \neg r$
if $a[i] = 7$ **then** $r \leftarrow \top$
@ $[a[1] = 7 \wedge r \wedge i = 2]$??? $[a[1] \neq 7 \wedge \neg r \wedge i = 2]$

Loop Invariant Computation: Forward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $i = 1 \land \neg r$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \land a[k] = 7]$
return r

Basic Paths (without unnecessary assumes):

assume $i = 1 \land \neg r$
assume $a[i] = 7$
$r \leftarrow \top$
$i \leftarrow i + 1$
@ $a[1] = 7 \land r \land i = 2$

assume $i = 1 \land \neg r$
assume $a[i] \neq 7$
$i \leftarrow i + 1$
@ $a[1] \neq 7 \land \neg r \land i = 2$

assume $i = 1 \land \neg r$
if $a[i] = 7$ **then** $r \leftarrow \top$
@ $[a[1] = 7 \land r \land i = 2] \lor [a[1] \neq 7 \land \neg r \land i = 2]$

Forward Computation: Extended Assertion

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $i = 1 \wedge \neg r$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Forward Computation: Extended Assertion

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $[i = 1 \wedge \neg r] \vee [a[1] = 7 \wedge r \wedge i = 2] \vee [a[1] \neq 7 \wedge \neg r \wedge i = 2]$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Forward Computation: Merging of Branches

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ $[i = 1 \wedge \neg r] \vee [a[1] = 7 \wedge r \wedge i = 2] \vee [a[1] \neq 7 \wedge \neg r \wedge i = 2]$
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Forward Computation: Merging of Branches

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ $[i = 1 \wedge \neg r] \vee [i = 2 \wedge [r \Leftrightarrow a[1] = 7]]$
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Forward Computation: Merging of Branches

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ $[i = 1 \wedge \neg r] \vee [i = 2 \wedge [r \Leftrightarrow a[1] = 7]]$
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic Paths:

assume $i = 2 \wedge [r \Leftrightarrow a[1] = 7]$
assume $a[i] = 7$
$r \leftarrow \top$
$i \leftarrow i + 1$
@ ???

assume $i = 2 \wedge [r \Leftrightarrow a[1] = 7]$
assume $a[i] \neq 7$
$i \leftarrow i + 1$
@ ???

assume $i = 2 \wedge [r \Leftrightarrow a[1] = 7]$
if $a[i] = 7$ **then** $r \leftarrow \top$
@ $i = 3 \wedge r \Leftrightarrow [a[1] = 7 \vee a[2] = 7]$

Forward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**

$$\texttt{@} \quad \begin{array}{l}[i = 1 \wedge \neg r] \vee \\ [i = 2 \wedge r \Leftrightarrow a[1] = 7] \vee \\ [i = 3 \wedge r \Leftrightarrow [a[1] = 7 \vee a[2] = 7]]\end{array}$$

  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Forward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**

$$\begin{array}{ll} & [i = 1 \wedge \neg r] \vee \\ \texttt{@} & [i = 2 \wedge r \Leftrightarrow a[1] = 7] \vee \\ & [i = 3 \wedge r \Leftrightarrow [a[1] = 7 \vee a[2] = 7]] \end{array}$$

if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

How to finish the process?

Forward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**

$$
\begin{array}{ll}
 & [i = 1 \wedge \neg r] \vee \\
@ & [i = 2 \wedge r \Leftrightarrow a[1] = 7] \vee \\
 & [i = 3 \wedge r \Leftrightarrow [a[1] = 7 \vee a[2] = 7]]
\end{array}
$$

  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

How to finish the process?

Fixpoint! $\top$?

Forward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**

$$
\begin{array}{ll}
 & [i = 1 \wedge \neg r] \vee \\
@ & [i = 2 \wedge r \Leftrightarrow a[1] = 7] \vee \\
 & [i = 3 \wedge r \Leftrightarrow [a[1] = 7 \vee a[2] = 7]]
\end{array}
$$

  if $a[i] = 7$ **then** $r \leftarrow \top$

@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

How to finish the process?

Fixpoint! $\top$?

After leaving the loop, the final assertion must hold!

Forward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**

$$@ \quad \begin{array}{l} [i = 1 \wedge \neg r] \vee \\ [i = 2 \wedge r \Leftrightarrow a[1] = 7] \vee \\ [i = 3 \wedge r \Leftrightarrow [a[1] = 7 \vee a[2] = 7]] \end{array}$$

    if $a[i] = 7$ **then** $r \leftarrow \top$

@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

How to finish the process?

Fixpoint! $\top$?

After leaving the loop, the final assertion must hold!

Each part for a specific i,

$$r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i - 1 \wedge a[k] = 7]$$

Checking the Invariant

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Checking the Invariant

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\bot$
2	$\bot$
3	$\bot$
4	$\top$

Checking the Invariant

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

1			4
4	3	7	6

i	r
1	$\bot$
2	$\bot$
3	$\bot$
4	$\top$

Paths through loop:

assume $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
assume $a[i] = 7$
$r \leftarrow \top$
$i \leftarrow i+1$
assume $i \leq n$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$

assume $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
assume $a[i] \neq 7$
$i \leftarrow i+1$
assume $i \leq n$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$

Checking the Invariant

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  @ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
  if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Checking the Invariant

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k]=7]$
    if $a[i]=7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k]=7]$
return r

Checking the Invariant

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    @ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
    if $a[i] = 7$ **then** $r \leftarrow \top$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Final basic paths:

assume $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
assume $a[i] = 7$
$r \leftarrow \top$
assume $i = n$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$

assume $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq i-1 \wedge a[k] = 7]$
assume $a[i] \neq 7$
assume $i = n$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ ???
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ ???
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ ???
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Start with weakest precondition,
    ensuring that the result is correct when leaving the loop

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  if $a[i] = 7$ **then** $r \leftarrow \top$
  @ ???
@ $r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Start with weakest precondition,
  ensuring that the result is correct when leaving the loop

assume $i = n$
@ $r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]$

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ ???
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Start with weakest precondition,
    ensuring that the result is correct when leaving the loop

assume $i = n$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$

Verification condition:

$$i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$$

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \land a[k] = 7]]$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \land a[k] = 7]$
return r

Basic paths in loop, again ignoring initial assume:
$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \land a[k] = 7]]$

Loop Invariant Computation: Backward Direction

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  if $a[i] = 7$ **then** $r \leftarrow \top$
  @ $i = n \Rightarrow [r \Leftrightarrow [\exists k\ .\ 1 \leq k \leq n \wedge a[k] = 7]]$
@ $r \Leftrightarrow [\exists k\ .\ 1 \leq k \leq n \wedge a[k] = 7]$
return r

Basic paths in loop, again ignoring initial assume:
$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k\ .\ 1 \leq k \leq n \wedge a[k] = 7]]$

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] \neq 7$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k\ .\ 1 \leq k \leq n \wedge a[k] = 7]]$

Weakest Precondition: **if**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]]$

Weakest Precondition: **if**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

$$\forall i_1, r \,.\, \left[\begin{array}{r} [i_1 = i + 1 \wedge i_1 \leq n \wedge a[i_1] = 7 \wedge r \wedge i_1 = n] \Rightarrow \qquad\qquad \\ r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7] \end{array} \right]$$

Weakest Precondition: **if**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

$$\forall i_1, r \,.\, \left[\begin{array}{r} [i_1 = i + 1 \wedge i_1 \leq n \wedge a[i_1] = 7 \wedge r \wedge i_1 = n] \Rightarrow \qquad\qquad\qquad \\ r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7] \end{array} \right]$$

after QE:

$$[i + 1 \leq n \wedge a[i + 1] = 7 \wedge i + 1 = n] \Rightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$$

Weakest Precondition: **if**

$i \leftarrow i+1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

$$\forall i_1, r \,.\, \left[\begin{array}{r} [i_1 = i+1 \wedge i_1 \leq n \wedge a[i_1] = 7 \wedge r \wedge i_1 = n] \Rightarrow \qquad\qquad \\ r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7] \end{array} \right]$$

after QE:

$$[i+1 \leq n \wedge a[i+1] = 7 \wedge i+1 = n] \Rightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$$

equivalent to $\top$

Weakest Precondition: **if**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] = 7$
$r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

$$\forall i_1, r \,.\, \left[\begin{array}{r} [i_1 = i + 1 \wedge i_1 \leq n \wedge a[i_1] = 7 \wedge r \wedge i_1 = n] \Rightarrow \qquad\qquad\qquad \\ r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7] \end{array} \right]$$

after QE:

$$[i + 1 \leq n \wedge a[i + 1] = 7 \wedge i + 1 = n] \Rightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$$

equivalent to $\top$

Intuition: after execution of the **if** branch,
the result is correct, independently of the initial state

Weakest Precondition: **else**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] \neq 7$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

Weakest Precondition: **else**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] \neq 7$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]]$

$$\forall i_1 \, . \, \left[\begin{array}{r} [i_1 = i + 1 \wedge i_1 \leq n \wedge a[i_1] \neq 7 \wedge i_1 = n] \Rightarrow \quad\quad\quad \\ r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7] \end{array} \right]$$

Weakest Precondition: **else**

$i \leftarrow i + 1$
assume $i \leq n$
assume $a[i] \neq 7$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

$$\forall i_1 \,.\, \left[\begin{array}{r} [i_1 = i + 1 \wedge i_1 \leq n \wedge a[i_1] \neq 7 \wedge i_1 = n] \Rightarrow \qquad\qquad \\ r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7] \end{array} \right]$$

QE:

$$[a[n] \neq 7 \wedge i + 1 = n] \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$$

Backward Computation: Merging of Branches

assume $\top \wedge [a[n] \neq 7 \wedge i+1 = n] \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
$i \leftarrow i+1$
if $a[i] = 7$ **then** $r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

Backward Computation: Merging of Branches

assume $\top \wedge [a[n] \neq 7 \wedge i + 1 = n] \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
$i \leftarrow i + 1$
if $a[i] = 7$ **then** $r \leftarrow \top$
@ $i = n \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$

In loop:

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ $\begin{array}{rcl} i = n & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \wedge \\ [a[n] \neq 7 \wedge i + 1 = n] & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7] \end{array}$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Backward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  if $a[i] = 7$ **then** $r \leftarrow \top$

$$@ \begin{array}{rcl} i = n & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \wedge \\ [a[n] \neq 7 \wedge i + 1 = n] & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \end{array}$$

@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Backward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$

$$@ \begin{array}{rcl} i = n & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \wedge \\ [a[n] \neq 7 \wedge i + 1 = n] & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \end{array}$$

@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Again: fixpoint needed

Backward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  if $a[i] = 7$ **then** $r \leftarrow \top$
  @ $\begin{array}{rcl} i = n & \Rightarrow & [r \Leftrightarrow [\exists k \; . \; 1 \leq k \leq n \wedge a[k] = 7]] \wedge \\ [a[n] \neq 7 \wedge i + 1 = n] & \Rightarrow & [r \Leftrightarrow [\exists k \; . \; 1 \leq k \leq n \wedge a[k] = 7]] \end{array}$
@ $r \Leftrightarrow [\exists k \; . \; 1 \leq k \leq n \wedge a[k] = 7]$
return r

Again: fixpoint needed

$$[\forall k \in \{i+1, \ldots, n\} \; . \; a[k] \neq 7] \Rightarrow [r \Leftrightarrow [\exists k \; . \; 1 \leq k \leq n \wedge a[k] = 7]]$$

Backward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
  if $a[i] = 7$ **then** $r \leftarrow \top$

$$@ \begin{array}{rcl} i = n & \Rightarrow & [r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]] \wedge \\ [a[n] \neq 7 \wedge i + 1 = n] & \Rightarrow & [r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]] \end{array}$$

@ $r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Again: fixpoint needed

$$[\forall k \in \{i + 1, \ldots, n\} \, . \, a[k] \neq 7] \Rightarrow [r \Leftrightarrow [\exists k \, . \, 1 \leq k \leq n \wedge a[k] = 7]]$$

Also holds initially

Backward Computation: Generalization

$r \leftarrow \bot$
for $i \leftarrow 1$ **to** n **do**
    if $a[i] = 7$ **then** $r \leftarrow \top$
    @ $\begin{array}{rcl} i = n & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \wedge \\ [a[n] \neq 7 \wedge i + 1 = n] & \Rightarrow & [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]] \end{array}$
@ $r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]$
return r

Again: fixpoint needed

$$[\forall k \in \{i+1, \dots, n\} \,.\, a[k] \neq 7] \Rightarrow [r \Leftrightarrow [\exists k \,.\, 1 \leq k \leq n \wedge a[k] = 7]]$$

Also holds initially, so this is another, different loop invariant!

Summary: Computation of Loop Invariants

Forward computation:

iterative add (using disjunction) strongest postconditions

strongest possible assertion after $0, 1, 2, \dots$ loop iterations

Summary: Computation of Loop Invariants

Forward computation:

iterative add (using disjunction) strongest postconditions

strongest possible assertion after $0, 1, 2, \ldots$ loop iterations

Backward computation:

iteratively add (using conjunction) weakest preconditions

ensures correct result after $0, 1, 2, \ldots$ loop iterations

Summary: Computation of Loop Invariants

Forward computation:
iterative add (using disjunction) strongest postconditions
strongest possible assertion after $0, 1, 2, \ldots$ loop iterations

Backward computation:
iteratively add (using conjunction) weakest preconditions
ensures correct result after $0, 1, 2, \ldots$ loop iterations

Needs: generalization step

Summary: Computation of Loop Invariants

Forward computation:
iteratively add (using disjunction) strongest postconditions
strongest possible assertion after $0, 1, 2, \dots$ loop iterations

Backward computation:
iteratively add (using conjunction) weakest preconditions
ensures correct result after $0, 1, 2, \dots$ loop iterations

Needs: generalization step

Current research

Summary: Computation of Loop Invariants

Forward computation:
iteratively add (using disjunction) strongest postconditions
strongest possible assertion after $0, 1, 2, \ldots$ loop iterations

Backward computation:
iteratively add (using conjunction) weakest preconditions
ensures correct result after $0, 1, 2, \ldots$ loop iterations

Needs: generalization step

Current research:

- often does not insist on strongest/weakest
  (interpolation $\approx$ approximate QE)
- trade in generalization for this

Summary: Computation of Loop Invariants

Forward computation:
iterative add (using disjunction) strongest postconditions
strongest possible assertion after $0, 1, 2, \ldots$ loop iterations

Backward computation:
iteratively add (using conjunction) weakest preconditions
ensures correct result after $0, 1, 2, \ldots$ loop iterations

Needs: generalization step

Current research:

- often does not insist on strongest/weakest
  (interpolation $\approx$ approximate QE)
- trade in generalization for this

Software verification competition:
`https://sv-comp.sosy-lab.org/2021/`

Examples of Industrial Tools

- http://www.absint.com/astree/
- Mathworks Polyspace
- https://www.imandra.ai

Examples of Industrial Tools

- `http://www.absint.com/astree/`
- Mathworks Polyspace
- `https://www.imandra.ai`

Finite state systems: viz MI-TES

Conclusion

Computation of loop invariants is difficult to automatize

Conclusion

Computation of loop invariants is difficult to automatize

Industrial use in specific applications

Conclusion

Computation of loop invariants is difficult to automatize

Industrial use in specific applications

Huge progress each year

Literature I

Aaron Bradley and Zohar Manna. *The calculus of computation*. Springer, 2007.