Weakest Precondition Calculus

19CSE205 : PROGRAM REASONING

Dr. Swaminathan J

Assistant Professor

Department of Computer Science and Engineering

AMRITA VISHWA VIDYAPEETHAM

Jul - Dec 2020

Contents

Can proof constructions be automated?

In the last lecture we saw the process of carrying out program proofs.

$\mathcal{V}$

Source/Model annotated with specification $\longrightarrow$ **Verification system** $\longrightarrow$

- YES, if able to prove.
- NO, if able to disprove.
- NO RESPONSE

So, how does the verfication system construct proofs? Is it possible to automate the proof construction?

Can proof constructions be automated?

In the last lecture we saw the process of carrying out program proofs.

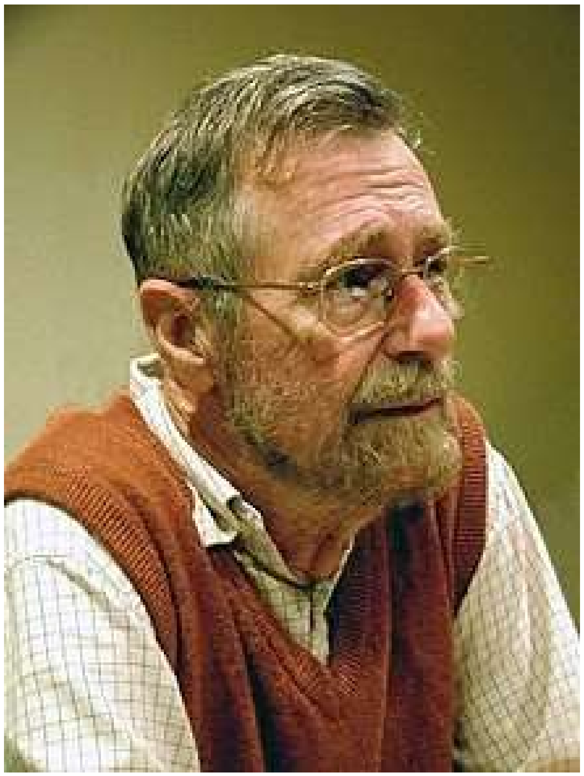
$$\text{Source/Model annotated with specification} \longrightarrow \boxed{\text{Verification system}}^{V} \longrightarrow \begin{cases} \text{- YES, if able to prove.} \\ \text{- NO, if able to disprove.} \\ \text{- NO RESPONSE} \end{cases}$$

So, how does the verfication system construct proofs? Is it possible to automate the proof construction?

- Dikjstra first provided a procedural approach to realize this objective in the form of Weakest Precondition calculus (topic of this lecture).
- This was followed by several improvisation and new techniques such as model checking, design by contract, satisfiability modulo theories, etc.
- The verification process demands lot of computing power. With the advancements in hardware speed, several tools are now available to construct proofs in an automated way.

Edsger W. Dijkstra

Dijkstra has made phenomenal contributions to computer science. He was one of the reasons for computer science to become a separate discipline. He received the ACM Turing Award in the year 1972.



A subset of his works you will study in B.Tech

- Formal specification and verification
  - Weakest precondition calculus
- Concurrency control
  - Banker's algorithm to prevent deadlock
  - Semaphore: A synchronization mechanism
  - Dining Philosophers Problem
- Algorithms
  - Single source shortest path algorithm

Must read: https://en.wikipedia.org/wiki/Edsger_W._Dijkstra.

Weakest precondition calculus

Weakest precondition calculus is a deductive system, proposed by Dijkstra, that provides an algorithmic solution to perform symbolic execution on program statements in the backward direction in order to deduce the predicate that will guarantee a given postcondition.

Weakest precondition calculus

Weakest precondition calculus is a deductive system, proposed by Dijkstra, that provides an algorithmic solution to perform symbolic execution on program statements in the backward direction in order to deduce the predicate that will guarantee a given postcondition.

We call the deduced predicate as the weakest precondition. The weakest precondition P for a statement S and a postcondition Q is written as $P = wp(S, Q)$.

Weakest precondition calculus

Weakest precondition calculus is a deductive system, proposed by Dijkstra, that provides an algorithmic solution to perform symbolic execution on program statements in the backward direction in order to deduce the predicate that will guarantee a given postcondition.

We call the deduced predicate as the weakest precondition. The weakest precondition P for a statement S and a postcondition Q is written as
$P = wp(S,Q)$.

Predicate	An expression that evaluates to either true or false.
Postcondition	A predicate that evaluates to true after execution a statement/block.
Precondition	A predicate which when true before execution of a statement/block ensures postcondion is true after execution of that statement/block.
Symbolic execution	An analysis technique that uses symbolic values to variables in an attempt to identify different execution paths that a program takes.
Deductive system	A system that uses axioms and rules of inference to prove theorems.

1. Skip

A skip statement refers to a blank statement.

- A skip statement does not change the program state.
- More specifically, it doesn't affect the postcondition.

Consider the example below

P	
S	skip
Q	$x > 0$

1. Skip

A skip statement refers to a blank statement.

- A skip statement does not change the program state.
- More specifically, it doesn't affect the postcondition.

Consider the example below

P	$x > 0$	
S	skip	$wp(skip, Q) = Q$
Q	$x > 0$	

A skip statement refers to a blank statement.

- A skip statement does not change the program state.
- More specifically, it doesn't affect the postcondition.

Consider the example below

P	$x > 0$	
S	skip	$wp(skip, Q) = Q$
Q	$x > 0$	

$$wp(skip, x>0) = x>0$$

Each statement can be viewed as a predicate transformer that turns a precondition to a postcondition. wp(S,Q) does the reverse transformation.

2. Assignment

A assignment statement is of the form value = expression.

- An assignment statement changes the program state.
- It results in the variable on the left hand side to change.

Consider the example below

P	
S	$\boxed{x = y + 5}$
Q	$x > 7$

A assignment statement is of the form value = expression.

- An assignment statement changes the program state.
- It results in the variable on the left hand side to change.

Consider the example below

P	$y > 2$	
S	$\boxed{x = y + 5}$	$wp(x{=}E, Q) = Q[x \leftarrow E] = P$
Q	$x > 7$	

2. Assignment

A assignment statement is of the form value = expression.

- An assignment statement changes the program state.
- It results in the variable on the left hand side to change.

Consider the example below

P	$y > 2$	
S	$\boxed{x = y + 5}$	$wp(x{=}E, Q) = Q[x \leftarrow E] = P$
Q	$x > 7$	

$$wp(x{=}y{+}5, x{>}7) = y{+}5{>}7 = y{>}2$$

A postcondition can have many preconditions. For the above example $y>2$, $y>34$, $y>100$ will all ensure the postcondition $x>7$. Among them $y>2$ is the least constraining condition and hence it is the weakest precondition.

A sequence denotes a block of statements.

- A sequence usually results in change of state more than once.

P

S_1 $\boxed{y = z * 2}$

Q

S_2 $\boxed{x = y + 5}$

R $x > 7$

3. Sequence

A sequence denotes a block of statements.

- A sequence usually results in change of state more than once.

P	
S_1	$y = z * 2$
Q	$y > 2$
S_2	$x = y + 5$
R	$x > 7$

$$wp(S_2,R) = wp(x{=}E_2,R) = R[x \leftarrow E_2] \quad = Q$$

A sequence denotes a block of statements.

- A sequence usually results in change of state more than once.

P	$z > 1$
S_1	$y = z * 2$
Q	$y > 2$
S_2	$x = y + 5$
R	$x > 7$

$$wp(S_1;S_2,R) = wp(S_1,wp(S_2,R))$$
$$= wp(y=E_1,Q)$$
$$= Q[y\leftarrow E_1] \quad = P$$

$$wp(S_2,R) = wp(x=E_2,R)$$
$$= R[x\leftarrow E_2] \quad = Q$$

$$wp(y=z*2;x=y+5,x>7) = wp(y=z*2,wp(x=y+5,x>7)) = wp(y=z*2,y+5>7)$$
$$= wp(y=z*2,y>2) = z*2>2 = z>1$$

A sequence denotes a block of statements.

- A sequence usually results in change of state more than once.

P	$z > 1$
S_1	$y = z * 2$
Q	$y > 2$
S_2	$x = y + 5$
R	$x > 7$

$$\begin{aligned} wp(S_1;S_2,R) &= wp(S_1,wp(S_2,R)) \\ &= wp(y=E_1,Q) \\ &= Q[y \leftarrow E_1] \quad = P \end{aligned}$$

$$\begin{aligned} wp(S_2,R) &= wp(x=E_2,R) \\ &= R[x \leftarrow E_2] \quad = Q \end{aligned}$$

$$wp(y=z*2;x=y+5,x>7) = wp(y=z*2,wp(x=y+5,x>7)) = wp(y=z*2,y+5>7)$$
$$= wp(y=z*2,y>2) = z*2>2 = z>1$$

Generalization: $wp(S_1;..;S_n,Q) = wp(S_1;..;S_{n-1},P_{n-1}) \ldots = wp(S_1,P_1) = P$
where $P_n = R$, $P_{i-1} = wp(S_i,P_i)$ and $P_0 = P$

A prelude to conditional branching

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   It doesn't rain $\Rightarrow$ I will shop

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND / OR?

   It doesn't rain $\Rightarrow$ I will shop

Generalizing, if B then S_1 else S_2

A prelude to conditional branching

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop

Generalizing, if B then S_1 else S_2

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$

A prelude to conditional branching

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop
2. It rains $\wedge$ I play chess

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$

A prelude to conditional branching

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop

2. It rains $\wedge$ I play chess

   AND / OR?

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$

A prelude to conditional branching

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop
2. It rains $\wedge$ I play chess

   OR

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop
2. It rains $\wedge$ I play chess

   OR

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$
2. $B \wedge S_1 \vee \neg B \wedge S_2$

A prelude to conditional branching

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop

2. It rains $\wedge$ I play chess

   OR

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$
2. $B \wedge S_1 \vee \neg B \wedge S_2$

wp (if B then S_1 else S_2,Q)

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop

2. It rains $\wedge$ I play chess

   OR

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$
2. $B \wedge S_1 \vee \neg B \wedge S_2$

wp (if B then S_1 else S_2,Q)

$= B \Rightarrow wp(S_1,Q) \wedge \neg B \Rightarrow wp(S_2,Q)$

$= B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

A conditional is a control point that results in alternate execution paths. This has some subtle issues. We will do it in two steps.

Example: If it rains then I will play chess else I will shop.

1. It rains $\Rightarrow$ I will play chess

   AND

   It doesn't rain $\Rightarrow$ I will shop

2. It rains $\wedge$ I play chess

   OR

   Its doesn't rain $\wedge$ I shop

Generalizing, if B then S_1 else S_2

1. $B \Rightarrow S_1 \wedge \neg B \Rightarrow S_2$
2. $B \wedge S_1 \vee \neg B \wedge S_2$

wp (if B then S_1 else S_2,Q)

$= B \Rightarrow wp(S_1,Q) \wedge \neg B \Rightarrow wp(S_2,Q)$

$= B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

B does not change state. Hence, wp(B,Q) is not required.

We will use: $wp(\text{if } B \text{ then } S_1 \text{ else } S_2) = B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1 $\quad$ if $y < 0$ then

$\qquad x = y + 1$

else

S_2 $\qquad x = y - 1$

Q $\quad x > 0$

4. Conditional branching

We will use: wp(if B then S_1 else S_2) $= B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1
```
if y < 0 then
    x = y + 1
else
    x = y − 1
```
S_2

Q $\quad x > 0$

wp(if $y<0$ then $x=y+1$ else $x=y-1$, $x>0$)

We will use: $wp(\text{if } B \text{ then } S_1 \text{ else } S_2) = B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1
S_2

```
if y < 0 then
    x = y + 1
else
    x = y − 1
```

Q $\quad x > 0$

$wp(\text{if } y<0 \text{ then } x=y+1 \text{ else } x=y-1, x>0)$

$= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)$

We will use: $wp(\text{if } B \text{ then } S_1 \text{ else } S_2) = B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1 $\quad$ if $y < 0$ then
$\quad\quad x = y + 1$

S_2 $\quad$ else
$\quad\quad x = y - 1$

Q $\quad x > 0$

$$wp(\text{if } y<0 \text{ then } x=y+1 \text{ else } x=y-1, x>0)$$

$$= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)$$

$$= y<0 \wedge y+1>0 \quad \vee \quad y\geq 0 \wedge y-1>0$$

4. Conditional branching

We will use: wp(if B then S_1 else S_2) $= B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1
```
if y < 0 then
    x = y + 1
else
    x = y − 1
```
S_2

Q $\quad x > 0$

$$wp(\text{if } y<0 \text{ then } x=y+1 \text{ else } x=y-1, x>0)$$
$$= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)$$
$$= y<0 \wedge y+1>0 \quad \vee \quad y\geq 0 \wedge y-1>0$$
$$= y<0 \wedge y> -1 \quad \vee \quad y\geq 0 \wedge y>1$$

We will use: $wp(\text{if } B \text{ then } S_1 \text{ else } S_2) = B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P	$y > 1$
S_1	if $y < 0$ then $x = y + 1$
S_2	else $x = y - 1$
Q	$x > 0$

$$wp(\text{if } y<0 \text{ then } x=y+1 \text{ else } x=y-1, x>0)$$

$$= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)$$

$$= y<0 \wedge y+1>0 \quad \vee \quad y\geq 0 \wedge y-1>0$$

$$= y<0 \wedge y> -1 \quad \vee \quad y\geq 0 \wedge y>1$$

$$= \text{FALSE} \quad \vee \quad y>1$$

4. Conditional branching

We will use: $wp(\text{if } B \text{ then } S_1 \text{ else } S_2) = B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1
if $y < 0$ then
    $x = y + 1$
S_2
else
    $x = y - 1$

Q $\quad x > 0$

$wp(\text{if } y<0 \text{ then } x=y+1 \text{ else } x=y-1, x>0)$

$= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)$

$= y<0 \wedge y+1>0 \quad \vee \quad y\geq 0 \wedge y-1>0$

$= y<0 \wedge y>-1 \quad \vee \quad y\geq 0 \wedge y>1$

$= \text{FALSE} \quad \vee \quad y>1$

$= y>1 \quad = P$

AMRITA VISHWA VIDYAPEETHAM DEEMED TO BE UNIVERSITY

We will use: wp(if B then S_1 else S_2) = B $\wedge$ wp(S_1,Q) $\vee \neg$B $\wedge$ wp(S_2,Q)

P $\quad y > 1$

S_1
```
if y < 0 then
    x = y + 1
else
    x = y − 1
```
S_2

Q $\quad x > 0$

What if we had if..then without the else part?

```
if y < 0 then
    x = y + 1
```

wp(if $y<0$ then $x=y+1$ else $x=y-1$,$x>0$)

$= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)$

$= y<0 \wedge y+1>0 \quad \vee \quad y\geq 0 \wedge y-1>0$

$= y<0 \wedge y> -1 \quad \vee \quad y\geq 0 \wedge y>1$

$=$ FALSE $\quad \vee \quad y>1$

$= y>1 \quad = P$

We will use: wp(if B then S_1 else S_2) = $B \wedge wp(S_1,Q) \vee \neg B \wedge wp(S_2,Q)$

P $\quad y > 1$

S_1 if $y < 0$ then
$\quad x = y + 1$

S_2 else
$\quad x = y - 1$

Q $\quad x > 0$

What if we had if..then without the else part?

As good as $\longrightarrow$

if $y < 0$ then
$\quad x = y + 1$
else
$\quad x = x$

$$
\begin{aligned}
&wp(\text{if } y<0 \text{ then } x=y+1 \text{ else } x=y-1, x>0)\\
&= y<0 \wedge wp(x=y+1, x>0) \vee \neg(y<0) \wedge wp(x=y-1, x>0)\\
&= y<0 \wedge y+1>0 \vee y\geq 0 \wedge y-1>0\\
&= y<0 \wedge y>-1 \vee y\geq 0 \wedge y>1\\
&= \text{FALSE} \vee y>1\\
&= y>1 \quad = P
\end{aligned}
$$

Proving theorems on program correctness

To conclude, by combining skip, assignment, sequence and conditional branching statements we can construct a wide variety of programs.

- We can hence deduce weakest precondition P for a program given Q.

Steps to prove program correctness

- Let S be the program.
  Let P′ be the input condition.
  Let Q′ be the output condition.

Theorem: Does $P' \Rightarrow Q'$?

Proof: Let postcondition $Q = Q'$

1. Obtain $P = wp(S,Q)$. i.e. $P \Rightarrow Q$
2. Check $P' \Rightarrow P$
   Therefore $P' \Rightarrow P \Rightarrow Q = Q'$

Recall previous example:

Let S: y = z * 2; x = y + 5;

Let Q′: x > 7

1. Given P′: z < 0
   We obtained P: z > 1
   Does $z < 0 \Rightarrow z > 1$? No
2. Given P′: z > 5
   We obtained P: z > 1
   Does $z > 5 \Rightarrow z > 1$? Yes

Note: We will cover loops later as it involves some intricacies.