Weakest Precondition Calculus

COMP2600 — Formal Methods for Software Engineering

Rajeev Goré

Australian National University

Semester 2, 2016

(Most lecture slides due to Ranald Clouston)

Hoare Logic = Syntax and (Semantics or Calculus)

Syntax	*Semantics*	*WP Calculus*
CPL predicates	*CPL semantics*	*No ND*!
variables $x, y, z \cdots$ arithmetic expressions $1+2, x < y, \cdots$ and predicates built from them	states map variables and expressions to values and predicates to true/false	rules of arithmetic e.g $1+2=3$, $2^2=4$, $6/3=2$ etc
programming language $:=$; `if.then.` `while`	as usual	one rule for each (seen rule for `:=`, `;`, `if.then.else.`, `prestr`, `postwk`)
Hoare Triple $\{P\}$ `S` $\{Q\}$	if pre-state satisfies P and `S` terminates then post-state satisfies Q	*proofs* (much harder!)

Edsger W. Dijkstra (1930 – 2002)

The originator of this week's material (in 1976), and a good source of quotes:

> Program testing can be quite effective for showing the presence of bugs, but is hopelessly inadequate for showing their absence.

> The question of whether Machines Can Think... [is] about as relevant as the question of whether Submarines Can Swim.

He also had some pretty uncompromising views on how introductory computer science should be taught:

> In order to drive home the message that this introductory programming course is primarily a course in formal mathematics, we see to it that the programming language in question has not been implemented on campus so that students are protected from the temptation to test their programs.

Introduction

Dijkstra's **Weakest Precondition Calculus** is another technique for proving properties of imperative programs.

Hoare Logic presents ***logic*** problems:

- Given a precondition P, code fragment S and postcondition Q, is $\{P\}S\{Q\}$ true?

WP is about evaluating a ***function***:

- Given a code fragment S and postcondition Q, find the *unique* P which is the weakest precondition for S and Q.

The wp Function

If S is a code fragment and Q is an assertion about states, then the ***weakest precondition*** for S with respect to Q is an assertion that is true for *precisely* those initial states from which:

- S *must terminate*, and
- executing S must produce *a state satisfying* Q.

That is, the weakest precondition P is a ***function*** of S and Q:

$$P = wp(S, Q)$$

(wp is sometimes called a *predicate transformer*, and the Weakest Precondition Calculus is sometimes called *Predicate Transformer Semantics*.)

Relationship with Hoare logic

Hoare Logic is ***relational***:

- For each Q, there are *many* P such that $\{P\}S\{Q\}$.
- For each P, there are *many* Q such that $\{P\}S\{Q\}$.

WP is ***functional***:

- For each Q there is *exactly one* assertion $wp(S,Q)$.

WP does *respect* Hoare Logic: $\{wp(S,Q)\}\ S\{Q\}$ is true.

Relationship with Hoare logic ctd.

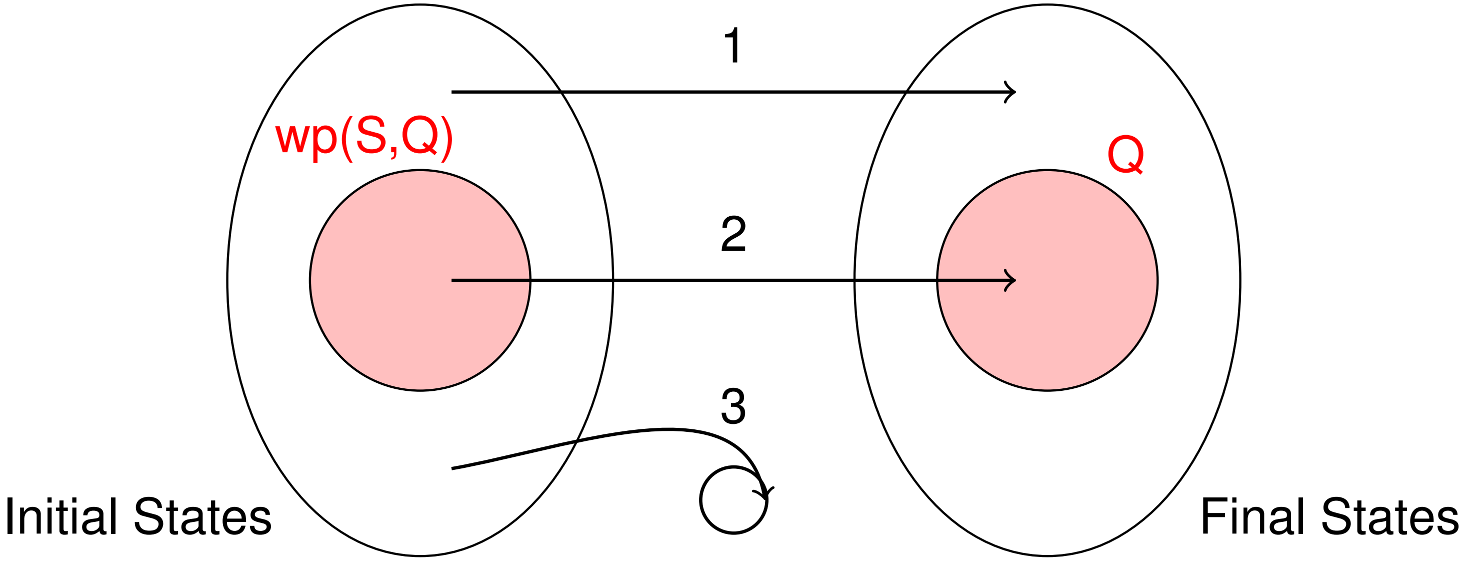
Hoare Logic is about ***partial correctness***:

- We ***don't*** care about termination.

WP is about ***total correctness***:

- We ***do*** care about termination.

A Diagrammatic View of Weakest Preconditions



Each arc shows the code S acting on a different initial state:

- Arc 1 produces a final state not satisfying Q ✗
- Arc 2 produces a final state satisfying Q ✓
- Arc 3 gets into a loop and produces no final state ✗

Intuition Example 1

Consider code `x:=x+1` and postcondition $(x > 0)$

One valid precondition is $(x > 0)$, so in Hoare Logic the following is true:

$$\{x > 0\}\ \texttt{x:=x+1}\ \{x > 0\}$$

Another valid precondition is $(x > -1)$, so:

$$\{x > -1\}\ \texttt{x:=x+1}\ \{x > 0\}$$

$(x > -1)$ is *weaker* than $(x > 0)$ (…because $(x > 0) \Rightarrow (x > -1)$)

In fact, $(x > -1)$ is the *weakest* precondition:

$$wp(\texttt{x:=x+1},\ x > 0) \quad \equiv \quad (x > -1)$$

Intuition Example 2

Consider code `a:=a+1; b:=b-1` and postcondition $a \times b = 0$

A very strong precondition is $(a = -1) \wedge (b = 1)$:

$$\{(a = -1) \wedge (b = 1)\}\texttt{a:=a+1; b:=b-1}\{a \times b = 0\}$$

A *weaker* precondition is $a = -1$.

The *weakest* precondition is $(a = -1) \vee (b = 1)$

$$wp(\texttt{a:=a+1; b:=b-1}, a \times b = 0) \quad \equiv \quad (a = -1) \vee (b = 1)$$

Intuition Example 3

The assignment axiom of Hoare Logic gives us (for any postcondition Q):

$$\{Q(y+z)\}\ \texttt{x:=y+z}\ \{Q(x)\}$$

Intuitive justification:

- Let v be the value arrived at by computing `y+z`.
- If $Q(y+z)$ is true initially, then so is $Q(v)$.
- Since the variable `x` has value v after the assignment (and nothing else changes in the state), It must be that $Q(x)$ holds after that assignment.

Intuition Example 3 ctd.

In fact $Q(y+z)$ is the ***weakest precondition.***

Intuitive justification:

- Let v be the value arrived at by computing `x:=y+z`.
- If $Q(x)$ is true after the assignment, so is $Q(v)$.
- If $Q(v)$ is true after the assignment, then it must also be true *before* the assignment, because x does not appear in $Q(v)$ and nothing else has changed.
- Thus, $Q(y+z)$ was true initially.
- So (combined with previous slide), $Q(x)$ holds after the assignment ***if and only if*** $Q(y+z)$ held before it.

Weakest Precondition of Assignment (Rule 1/4)

We have argued that the Assignment axiom of Hoare Logic is designed to give the “best” — i.e. the *weakest* — precondition:

$$\{Q(e)\}\ \texttt{x:=e}\ \{Q(x)\}$$

Therefore we should expect that the rule for Assignment in the weakest precondition calculus corresponds closely:

$$wp(\texttt{x:=e},\, Q(x)) \ \equiv\ Q(e)$$

Assignment Examples

$$\begin{aligned} wp(\texttt{x:=y+3},\ (x > 3)) &\equiv y+3 > 3 && \text{(substitute } y+3 \text{ for } x\text{)} \\ &\equiv y > 0 && \text{(simplify)} \end{aligned}$$

$$\begin{aligned} wp(\texttt{n:=n+1},\ (n > 5)) &\equiv n+1 > 5 && \text{(substitute } n+1 \text{ for } n\text{)} \\ &\equiv n > 4 && \text{(simplify)} \end{aligned}$$

Weakest Preconditions for Sequences (Rule 2/4)

As in Hoare Logic, we expect the rule for sequencing to ***compose*** the effect of the consecutive statements. The rule is:

$$wp(S_1;S_2,\ \ Q) \ \equiv\ wp(S_1,\ \ wp(S_2,\ \ Q))$$

Example:

$$\begin{aligned}
&wp(\texttt{x:=x+2;}\ \ \texttt{y:=y-2},\ (x+y=0))\\
\equiv\ &wp(\texttt{x:=x+2},\ wp(\texttt{y:=y-2},\ (x+y=0)))\\
\equiv\ &wp(\texttt{x:=x+2},\ (x+(y-2)=0))\\
\equiv\ &((x+2)+(y-2)=0)\\
\equiv\ &(x+y=0)
\end{aligned}$$

Weakest Preconditions for Conditionals (Rule 3a/4)

$$wp(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, Q) \equiv (b \Rightarrow wp(S_1, Q)) \wedge (\neg b \Rightarrow wp(S_2, Q))$$

Proof:

By cases on condition b,

- b is *true*: $RHS \equiv (True \Rightarrow wp(S_1, Q)) \wedge (False \Rightarrow wp(S_2, Q))$

  wp for the conditional is the weakest precondition for S_1 guaranteeing postcondition Q – that is, LHS is $wp(S_1, Q)$.

  The right hand side reduces to the same thing if we replace b with $True$.

- b is *false*:

  Similarly, both left hand and right hand sides reduce to $wp(S_2, Q)$

Conditional Example:

$$wp(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, Q) \;\equiv\; (b \Rightarrow wp(S_1, Q)) \;\wedge\; (\neg b \Rightarrow wp(S_2, Q))$$

$$wp(\texttt{if x>2 then y:=1 else y:=-1}, (y > 0))$$

$$\equiv ((x > 2) \Rightarrow wp(\texttt{y:=1}, (y > 0)) \;\wedge\; (\neg(x > 2) \Rightarrow wp(\texttt{y:=-1}, (y > 0))$$

$$\equiv ((x > 2) \Rightarrow (1 > 0)) \;\wedge\; (\neg(x > 2) \Rightarrow (-1 > 0))$$

$$\equiv ((x > 2) \Rightarrow True) \;\wedge\; ((x \le 2) \Rightarrow False)$$

$$\equiv x > 2$$

(If you are unhappy with the last step, draw a truth table.)

Alternative Rule for Conditionals (Rule 3b/4)

The conditional rule tends to produce complicated logical expressions which we then have to simplify.

It is often easier to deal with disjunctions and conjunctions than implications, so the following ***equivalent*** rule for conditionals is usually more convenient.

$$wp(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, Q) \equiv (b \wedge wp(S_1, Q)) \vee (\neg b \wedge wp(S_2, Q))$$

Conditional Example Again:

$$wp(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, Q) \equiv (b \wedge wp(S_1, Q)) \vee (\neg b \wedge wp(S_2, Q))$$

$$wp(\texttt{if x>2 then y:=1 else y:=-1}, (y > 0))$$

$$\equiv ((x > 2) \wedge wp(\texttt{y:=1}, (y > 0)) \vee (\neg(x > 2) \wedge wp(\texttt{y:=-1}, (y > 0))$$

$$\equiv ((x > 2) \wedge (1 > 0)) \vee (\neg(x > 2) \wedge (-1 > 0))$$

$$\equiv ((x > 2) \wedge True) \vee (\neg(x > 2) \wedge False)$$

$$\equiv (x > 2) \vee False$$

$$\equiv x > 2$$

(Again, any step you are unhappy with can be confirmed via truth table.)

Why The Rules are Equivalent

All that has changed is the form of the proposition. Rather than

$$(b \Rightarrow p) \wedge (\neg b \Rightarrow q)$$

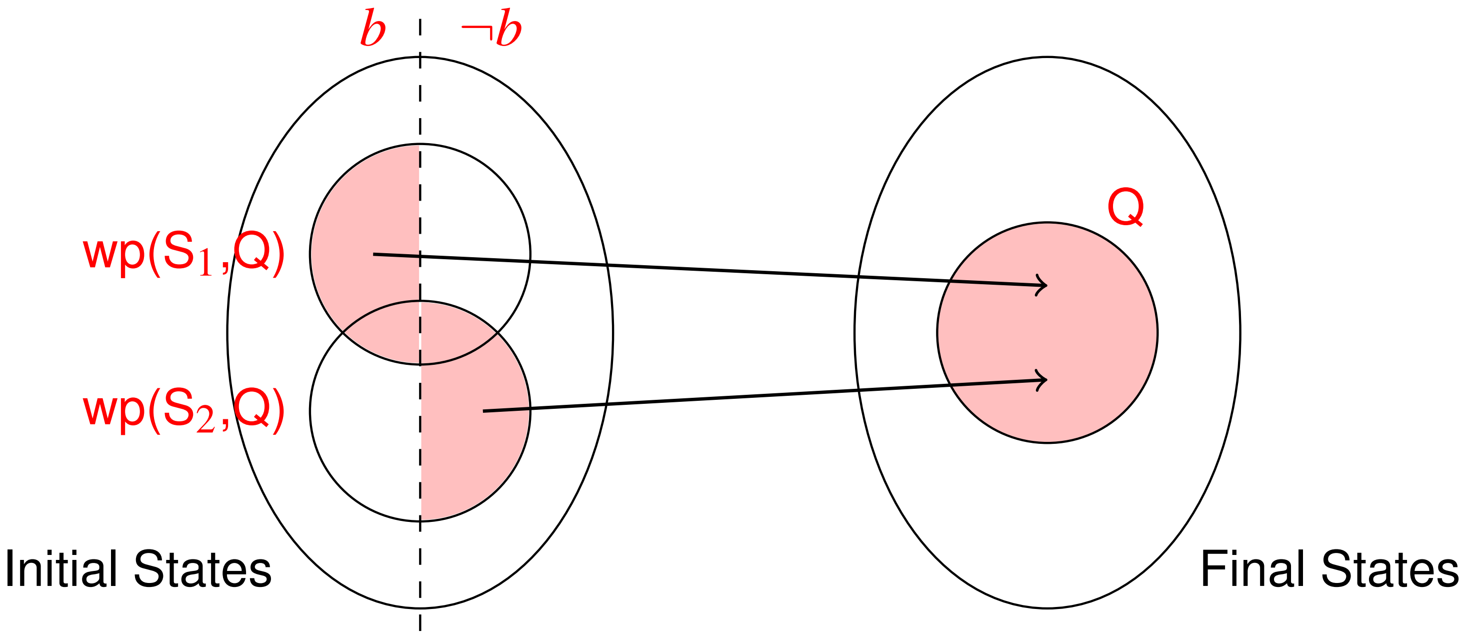
we have

$$(b \wedge p) \vee (\neg b \wedge q) \;\; :$$

b	p	q	$(b \Rightarrow p)$	$\wedge$	$(\neg b \Rightarrow q)$	$(b \wedge p)$	$\vee$	$(\neg b \wedge q)$
T	T	T	T	**T**	T	T	**T**	F
T	T	F	T	**T**	T	T	**T**	F
T	F	T	F	**F**	T	F	**F**	F
T	F	F	F	**F**	T	F	**F**	F
F	T	T	T	**T**	T	F	**T**	T
F	T	F	T	**F**	F	F	**F**	F
F	F	T	T	**T**	T	F	**T**	T
F	F	F	T	**F**	F	F	**F**	F

A Diagrammatic View of Conditionals

$$
\begin{aligned}
wp(\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2, Q) &\equiv (b \Rightarrow wp(S_1,Q)) \ \wedge \ (\neg b \Rightarrow wp(S_2,Q)) \\
&\equiv (b \wedge wp(S_1,Q)) \ \vee \ (\neg b \wedge wp(S_2,Q))
\end{aligned}
$$



Conditionals Without ‘Else’

It is sometimes convenient to have conditionals without `else`, i.e.

```
if b then S
```

recalling that this is just a compact way of writing

```
if b then S else x := x
```

We can derive wp rules for this case:

$$\begin{aligned} wp(\textbf{if } b \textbf{ then } S, Q) &\equiv (b \Rightarrow wp(S_1, Q)) \ \wedge \ (\neg b \Rightarrow Q) \\ &\equiv (b \wedge wp(S_1, Q)) \ \vee \ (\neg b \wedge Q) \end{aligned}$$

Loops

(The thing to do now is hang on tightly . . .)

Suppose we have a while loop and some postcondition Q.

The precondition P we seek is the weakest that:

- establishes Q
- *guarantees termination*

We can take hints for the first requirement from the corresponding rule for Hoare Logic. That is, think in terms of *loop invariants*.

But termination is a bigger problem...

Summary of Lecture I

Weak and Strong Conditions: P stronger than Q if $P \Rightarrow Q$ in first-order logic

Hoare Logic Rules: used this notion in rules precon equivalence, precond strengthening, postcond equivalence, postcond weakening

$wp(S,Q)$**:** Given S and Q, the assertion $wp(S,Q)$ is the ***weakest precondition*** that is true for *precisely* those initial states from which:

- S *must terminate*, and
- executing S must produce *a state satisfying Q*

$wp(S,Q)$: is a function ... so we can give the rules as equations