Derived rules for finding proofs

- **Suppose the goal is to prove** $\{Precondition\}\ Comman$
- **If there were a rule of the form**

$$\frac{\vdash\ H_1,\ \cdots,\vdash\ H_n}{\vdash\ \{P\}\ C\ \{Q\}}$$

  then we could instantiate

  $P \mapsto Precondition$, $C \mapsto Command$, $Q \mapsto Postcondition$

  to get instances of $H_1,\ \cdots, H_n$ **as subgoals**
- **Some of the rules are already in this form e.g. the s**
- **We will derive rules of this form for all commands**
- **Then we use these derived rules for mechanising Ho**

Derived Rules

- **We will establish derived rules for all commands**

$$\frac{\ldots}{\vdash \{P\}\ \texttt{SKIP}\ \{Q\}}$$

$$\frac{\ldots}{\vdash \{P\}\ V\texttt{:=}E\ \{Q\}}$$

$$\frac{\ldots}{\vdash \{P\}\ C_1\texttt{;}C_2\ \{Q\}}$$

$$\frac{\ldots}{\vdash \{P\}\ \texttt{BEGIN VAR}\ V_1\texttt{;}\ \ldots\texttt{;}\ \texttt{VAR}\ V_n\texttt{;}\ C\ \texttt{END}\ \{Q\}}$$

$$\frac{\ldots}{\vdash \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2\ \{Q\}}$$

$$\frac{\ldots}{\vdash \{P\}\ \texttt{WHILE}\ S\ \texttt{DO}\ C\ \{Q\}}$$

- **These support 'backwards proof' starting from a go**

The Derived Skip Rule

Derived Skip Rule

$$\frac{\vdash\ P \Rightarrow Q}{\vdash\ \{P\}\ \texttt{SKIP}\ \{Q\}}$$

- **Justifying proof schema:**

1. $\vdash\ P \Rightarrow Q$ — By assumption.
2. $\vdash\ \{Q\}\ \texttt{SKIP}\ \{Q\}$ — By the skip axiom.
3. $\vdash\ \{P\}\ \texttt{SKIP}\ \{Q\}$ — By precondition strengthening with 1 and 2.

- **This schematic proof shows that any inference with t Rule can be replaced by three inferences using prin**
- **Derived rules are 'proof subroutines'**
  - **the schema metavariables (P, Q above) are the parameter**
  - **the derivation schema is the body of the subroutine**

The Derived Assignment Rule

- **An example proof**

1. $\vdash$ $\{\texttt{R=X} \wedge \texttt{0=0}\}$ $\texttt{Q:=0}$ $\{\texttt{R=X} \wedge \texttt{Q=0}\}$ By the assignment axiom.
2. $\vdash$ $\texttt{R=X} \Rightarrow \texttt{R=X} \wedge \texttt{0=0}$ By pure logic.
3. $\vdash$ $\{\texttt{R=X}\}$ $\texttt{Q:=0}$ $\{\texttt{R=X} \wedge \texttt{Q=0}\}$ By precondition strengthening.

- **Can generalise this proof to a proof schema:**

1. $\vdash$ $\{Q[E/V]\}\ V := E\ \{Q\}$ By the assignment axiom.
2. $\vdash$ $P \Rightarrow Q[E/V]$ By assumption.
3. $\vdash$ $\{P\}\ C\ \{Q\}$ By precondition strengthening.

- **This proof schema justifies:**

Derived Assignment Rule

$$\frac{\vdash\ P \Rightarrow Q[E/V]}{\vdash\ \{P\}\ V := E\ \{Q\}}$$

- **Note: $Q[E/V]$ is the weakest liberal precondition**
- **Example proof above can now be done in one less s**

1. $\vdash$ $\texttt{R=X} \Rightarrow \texttt{R=X} \wedge \texttt{0=0}$ By pure logic.
2. $\vdash$ $\{\texttt{R=X}\}$ $\texttt{Q:=0}$ $\{\texttt{R=X} \wedge \texttt{Q=0}\}$ By derived assignment.

Derived Sequenced Assignment Rule

- The following rule will be useful later

Derived Sequenced Assignment Rule

$$\frac{\vdash \{P\}\ C\ \{Q[E/V]\}}{\vdash \{P\}\ C;V:=E\ \{Q\}}$$

- Exercise: give a proof schema to justify this
- Intuitively work backwards:
  - push Q 'through' $V:=E$, changing it to $Q[E/V]$
- Example: By the assignment axiom:

  $\vdash \{\texttt{X=x} \wedge \texttt{Y=y}\}\ \texttt{R:=X}\ \{\texttt{R=x} \wedge \texttt{Y=y}\}$

- Hence by the sequenced assignment rule

  $\vdash \{\texttt{X=x} \wedge \texttt{Y=y}\}\ \texttt{R:=X; X:=Y}\ \{\texttt{R=x} \wedge \texttt{X=y}\}$

Backward Hoare & forward Floyd assignme

- **Recall Hoare (backward) and Floyd (forward) assig**

  Hoare axiom: $\vdash \{P[E/V]\}\ V := E\ \{P\}$

  Floyd axiom: $\vdash \{P\}\ V := E\ \{\exists v.\ V = E[v/V] \ \wedge \ P[v/V]$

- **Exercise 1 (easy): derive forward axiom from Hoar**
  - **hint:** $P \ \Rightarrow \ \exists v.\ E = E[v/V] \ \wedge \ P[v/V]$
- **Exercise 2 (a bit harder): derive Hoare axiom from**
  - **hint: if** v **is a new variable then** $P[E/V][v/V] \ = \ P[E[v/$
- **Exercise 3: devise and justify a derived assignment the Floyd assignment axiom**

The Derived While Rule

Derived While Rule

$$\frac{\vdash P \Rightarrow R \quad \vdash \{R \wedge S\}\ \mathtt{C}\ \{R\} \quad \vdash R \wedge \neg S}{\vdash \{P\}\ \mathtt{WHILE\ S\ DO\ C}\ \{Q\}}$$

- **This follows from the While Rule and the rules of c**
- **Example: it is easy to show**

  $\vdash$ R=X $\wedge$ Q=0 $\Rightarrow$ X=R+(Y$\times$Q)

  $\vdash$ {X=R+(Y$\times$Q)$\wedge$Y$\leq$R} R:=R-Y; Q:=Q+1 {X=R+(Y$\times$Q)}

  $\vdash$ X=R+(Y$\times$Q)$\wedge\neg$(Y$\leq$R) $\Rightarrow$ X=R+(Y$\times$Q)$\wedge\neg$(Y$\leq$R)

- **Then, by the derived While rule**

  $\vdash$ {R=X $\wedge$ Q=0}
  WHILE Y$\leq$R DO
  R:=R-Y; Q:=Q+1
  {X=R+(Y$\times$Q) $\wedge$ $\neg$(Y$\leq$R)}

The Derived Sequencing Rule

- **The rule below follows from the sequencing and co**

The Derived Sequencing Rule

$$\frac{\begin{array}{ll} & \vdash\ P \Rightarrow P_1 \\ \vdash\ \{P_1\}\ C_1\ \{Q_1\} & \vdash\ Q_1 \Rightarrow P_2 \\ \vdash\ \{P_2\}\ C_2\ \{Q_2\} & \vdash\ Q_2 \Rightarrow P_3 \\ \vdots & \vdots \\ \vdash\ \{P_n\}\ C_n\ \{Q_n\} & \vdash\ Q_n \Rightarrow Q \end{array}}{\vdash\ \{P\}\ C_1;\ \ldots\ ;\ C_n\ \{Q\}}$$

- **Exercise: why no derived conditional rule?**

Example

- **By the assignment axiom**

  (i) $\vdash$ {X=x∧Y=y} R:=X {R=x∧Y=y}

  (ii) $\vdash$ {R=x∧Y=y} X:=Y {R=x∧X=y}

  (iii) $\vdash$ {R=x∧X=y} Y:=R {Y=x∧X=y}

- **Using the derived sequencing rule, it can be deduced (i), (ii), (iii) and the fact that for any P:** $\vdash P \Rightarrow$

  $\vdash$ {X=x ∧ Y=y} R:=X; X:=Y; Y:=R {Y=x ∧ X=y}

Derived Block Rule

- **From the derived sequencing rule and the block ru rule for blocks can be derived**

The Derived Block Rule

$$\frac{\begin{array}{ll} & \vdash P \Rightarrow P_1 \\ \vdash \{P_1\}\ C_1\ \{Q_1\} & \vdash Q_1 \Rightarrow P_2 \\ \vdash \{P_2\}\ C_2\ \{Q_2\} & \vdash Q_2 \Rightarrow P_3 \\ \vdots & \vdots \\ \vdash \{P_n\}\ C_n\ \{Q_n\} & \vdash Q_n \Rightarrow Q \end{array}}{\vdash \{P\}\ \texttt{BEGIN VAR}\ V_1\texttt{;}\ \ldots\ \texttt{VAR}\ V_m\texttt{;}C_1\texttt{;}\ \ldots\ \texttt{;}\ C_n\ \{Q}$$

where none of the variables $V_1, \ldots, V_m$ occur in P or Q.

Example

- **By the assignment axiom**

  (i) $\vdash$ {X=x∧Y=y} R:=X {R=x∧Y=y}

  (ii) $\vdash$ {R=x∧Y=y} X:=Y {R=x∧X=y}

  (iii) $\vdash$ {R=x∧X=y} Y:=R {Y=x∧X=y}

- **Using the derived block rule, it can be deduced** *in* **(ii) and (iii) that**

  $\vdash$ {X=x ∧ Y=y}
  BEGIN VAR R; R:=X; X:=Y; Y:=R END
  {Y=x ∧ X=y}

Deriving Rules for New Commands

- **Suppose we define a one-armed conditional by:**
  $\texttt{IF}\ S\ \texttt{THEN}\ C \quad \equiv \quad \texttt{IF}\ S\ \texttt{THEN}\ C\ \texttt{ELSE SKIP}$
- **We can derive the following rule:**

The One-Armed Conditional Rule

$$\frac{\vdash\ \{P \wedge S\}\ C\ \{Q\} \qquad \vdash\ P \wedge \neg S \Rightarrow Q}{\vdash\ \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C\ \{Q\}}$$

- **Derivation**

1.	$\vdash\ P \wedge \neg S\ \Rightarrow\ Q$	Hypothesis
2.	$\vdash\ \{P \wedge \neg S\}\ \texttt{SKIP}\ \{Q\}$	By derived `SKIP` rule
3.	$\vdash\ \{P \wedge S\}\ C\ \{Q\}$	Hypothesis
4.	$\vdash\ \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C\ \texttt{ELSE SKIP}\ \{Q\}$	By conditional rule
5.	$\vdash\ \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C\ \{Q\}$	By definition of one-a

Example

- **From**

  (i) $\vdash \{\mathtt{T} \wedge \mathtt{X} \geq \mathtt{Y}\}\ \mathtt{Y:=X}\ \{\mathtt{Y=max(X,Y)}\}$

  (ii) $\vdash \mathtt{T} \wedge \mathtt{Y} \geq \mathtt{X} \Rightarrow \mathtt{max(X,Y)=Y}$

- **Then by the derived one-armed conditional rule it**

  $\vdash \{\mathtt{T}\}\ \mathtt{IF\ X} \geq \mathtt{Y\ THEN\ Y:=X}\ \{\mathtt{Y=max(X,Y)}\}$

Aside: linear versus tree structured proofs

- **Tree structured proofs are fine (e.g. for exams)**
  - **advantages: familiar, maybe easier to construct**
  - **disadvantage: harder to present large proofs**
- **Examples (proof schemas, not concrete proofs)**

$$\frac{\vdash\ P \Rightarrow Q[E/V] \quad \dfrac{}{\vdash\ \{Q[E/V]\}\ V := E\ \{Q\}}\ ASS}{\vdash\ \{P\}\ V := E\ \{Q\}}\ PRE$$

$$\frac{\vdash\ P \Rightarrow Q \quad \dfrac{}{\vdash\ \{Q\}\ \texttt{SKIP}\ \{Q\}}\ SKP}{\vdash\ \{P\}\ \texttt{SKIP}\ \{Q\}}\ PRE$$

$$\frac{\vdash\ \{P\}C\{Q[E/V]\} \quad \dfrac{}{\vdash\ \{Q[E/V]\}\ V := E\ \{Q\}}\ AS}{\vdash\ \{P\}\ C;V := E\ \{Q\}}\ SE$$

$$\frac{\vdash\ \{P \wedge S\}\ C\ \{Q\} \quad \dfrac{\vdash\ P \wedge \neg S\ \Rightarrow\ Q}{\vdash\ \{P \wedge \neg S\}\ \texttt{SKIP}\ \{Q\}}\ DSKP}{\dfrac{\vdash\ \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C\ \texttt{ELSE SKIP}\ \{Q\}}{\vdash\ \{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C\ \{Q\}}\ DEF}\ IF$$

Forwards and backwards proof

- **Previously it was shown how to prove $\{P\}C\{Q\}$ by**
  - **proving properties of the components of C**
  - **and then putting these together, with the appropriate pr to get the desired property of C**
- **For example, to prove $\vdash \{P\}C_1;C_2\{Q\}$**
- **First prove $\vdash \{P\}C_1\{R\}$ and $\vdash \{R\}C_2\{Q\}$**
- **then deduce $\vdash \{P\}C_1;C_2\{Q\}$ by sequencing rule**
- **This method is called *forward proof***
  - **move forward from axioms via rules to conclusion**
- **The problem with forwards proof is that it is not al what you need to prove to get where you want to b**
- **It is more natural to work backwards**
  - **starting from the goal of showing $\{P\}C\{Q\}$**
  - **generate subgoals until problem solved**

Example

- **Suppose one wants to show**

  $\{X=x \wedge Y=y\}\ R:=X;\ X:=Y;\ Y:=R\ \{Y=x \wedge X=y\}$

- **By the assignment axiom and derived sequenced as is sufficient to show the subgoal**

  $\{X=x \wedge Y=y\}\ R:=X;\ X:=Y\ \{R=x \wedge X=y\}$

- **Similarly this subgoal can be reduced to**

  $\{X=x \wedge Y=y\}\ R:=X\ \{R=x \wedge Y=y\}$

- **This clearly follows from the assignment axiom**

Backwards versus Forwards Proof

- **Backwards proof just involves using the rules backw**
- **Given the rule**

$$\frac{\vdash S_1 \quad \ldots \quad \vdash S_n}{\vdash S}$$

- **Forwards proof says:**
  - **if we have proved** $\vdash S_1 \ \ldots \ \vdash S_n$ **we can deduce** $\vdash S$
- **Backwards proof says:**
  - **to prove** $\vdash S$ **it is sufficient to prove** $\vdash S_1 \ \ldots \ \vdash S_n$
- **Having proved a theorem by backwards proof, it is s a forwards proof**

Example Backwards Proof

- **To prove**

$\vdash$ {T}
R:=X;
Q:=0;
WHILE Y$\leq$R DO
BEGIN R:=R-Y; Q:=Q+1 END
{X=R+(Y$\times$Q) $\wedge$ R<Y}

- **By the sequencing rule, it is sufficient to prove**

(i) $\vdash$ {T} R:=X; Q:=0 {R=X $\wedge$ Q=0}

(ii) $\vdash$ {R=X $\wedge$ Q=0}
WHILE Y$\leq$R DO
BEGIN R:=R-Y; Q:=Q+1 END
{X=R+(Y$\times$Q) $\wedge$ R<Y}

- **Where does {R=X $\wedge$ Q=0} come from? (Answer later**

Example Continued (1)

- **From previous slide:**

  (i) $\vdash \{\mathtt{T}\}\ \mathtt{R:=X;\ Q:=0}\ \{\mathtt{R=X} \wedge \mathtt{Q=0}\}$

- **To prove (i), by the sequenced assignment axiom, w**

  (iii) $\vdash \{\mathtt{T}\}\ \mathtt{R:=X}\ \{\mathtt{R=X} \wedge \mathtt{0=0}\}$

- **To prove (iii), by the derived assignment rule, we n**

  $\vdash\ \mathtt{T} \Rightarrow \mathtt{X=X} \wedge \mathtt{0=0}$

- **This is true by pure logic**

Example continued (2)

- **From an earlier slide:**

(ii) $\vdash$ $\{R=X \wedge Q=0\}$
WHILE $Y \leq R$ DO
BEGIN R:=R-Y; Q:=Q+1 END
$\{X=R+(Y \times Q) \wedge R<Y\}$

- **To prove (ii), by the derived while rule, we must pr**

(iv) $R=X \wedge Q=0 \Rightarrow (X = R+(Y \times Q))$

(v) $X = R+Y \times Q \wedge \neg(Y \leq R) \Rightarrow (X = R+(Y \times Q) \wedge R<Y)$

and

$\{X = R+(Y \times Q) \wedge (Y \leq R)\}$
(vi) BEGIN R:=R-Y; Q:=Q+1 END
$\{X=R+(Y \times Q)\}$

- **(iv) and (v) are proved by pure arithmetic**

Example Continued (3)

- To prove (vi), by the block rule, we must prove

$$\begin{array}{ll} & \{\texttt{X = R+(Y}\times\texttt{Q)} \wedge \texttt{(Y}\leq\texttt{R)}\} \\ \textbf{(vii)} & \texttt{R:=R-Y; Q:=Q+1} \\ & \{\texttt{X=R+(Y}\times\texttt{Q)}\} \end{array}$$

- To prove (vii), by the sequenced assignment rule, w

$$\begin{array}{ll} & \{\texttt{X=R+(Y}\times\texttt{Q)} \wedge \texttt{(Y}\leq\texttt{R)}\} \\ \textbf{(viii)} & \texttt{R:=R-Y} \\ & \{\texttt{X=R+(Y}\times\texttt{(Q+1))}\} \end{array}$$

- To prove (viii), by the derived assignment rule, we

$$\textbf{(ix)}\ \texttt{X=R+(Y}\times\texttt{Q)} \wedge \texttt{Y}\leq\texttt{R} \Rightarrow \texttt{(X = (R-Y)+(Y}\times\texttt{(Q+1)))}$$

- This is true by arithmetic
- Exercise: Construct the forwards proof that corresp to this backwards proof

Annotations

- **The sequencing rule introduces a new statement R**

$$\frac{\vdash \{P\}\ C_1\ \{R\} \qquad \vdash \{R\}\ C_2\ \{Q\}}{\vdash \{P\}\ C_1;C_2\ \{Q\}}$$

- **To apply this backwards, one needs to find a suitab**
- **If C_2 is V :=E then sequenced assignment gives Q[E**
- **If C_2 isn't an assignment then need some other way**
- **Similarly, to use the derived While rule, must inver**

Annotate First

- It is helpful to think up these statements before you and then annotate the program with them
  - the information is then available when you need it in the
  - this can help avoid you being bogged down in details
  - the annotation should be true whenever control reaches t
- Example, the following program could be annotated P_1 and P_2 indicated by the arrows

```
{T}
 BEGIN
   R:=X;
   Q:=0; {R=X ∧ Q=0} ⟵P1
   WHILE Y≤R DO {X = R+Y×Q} ⟵P2
     BEGIN R:=R-Y; Q:=Q+1 END
 END
{X = R+Y×Q ∧ R<Y}
```

Summary

- We have looked at three ways of organizing proo
  easier for humans to apply them:
  - deriving "bigger step" rules
  - backwards proof
  - annotating programs
- Next we see how these techniques can be used to mec
  verification

NEW TOPIC: Mechanizing Program Verifi

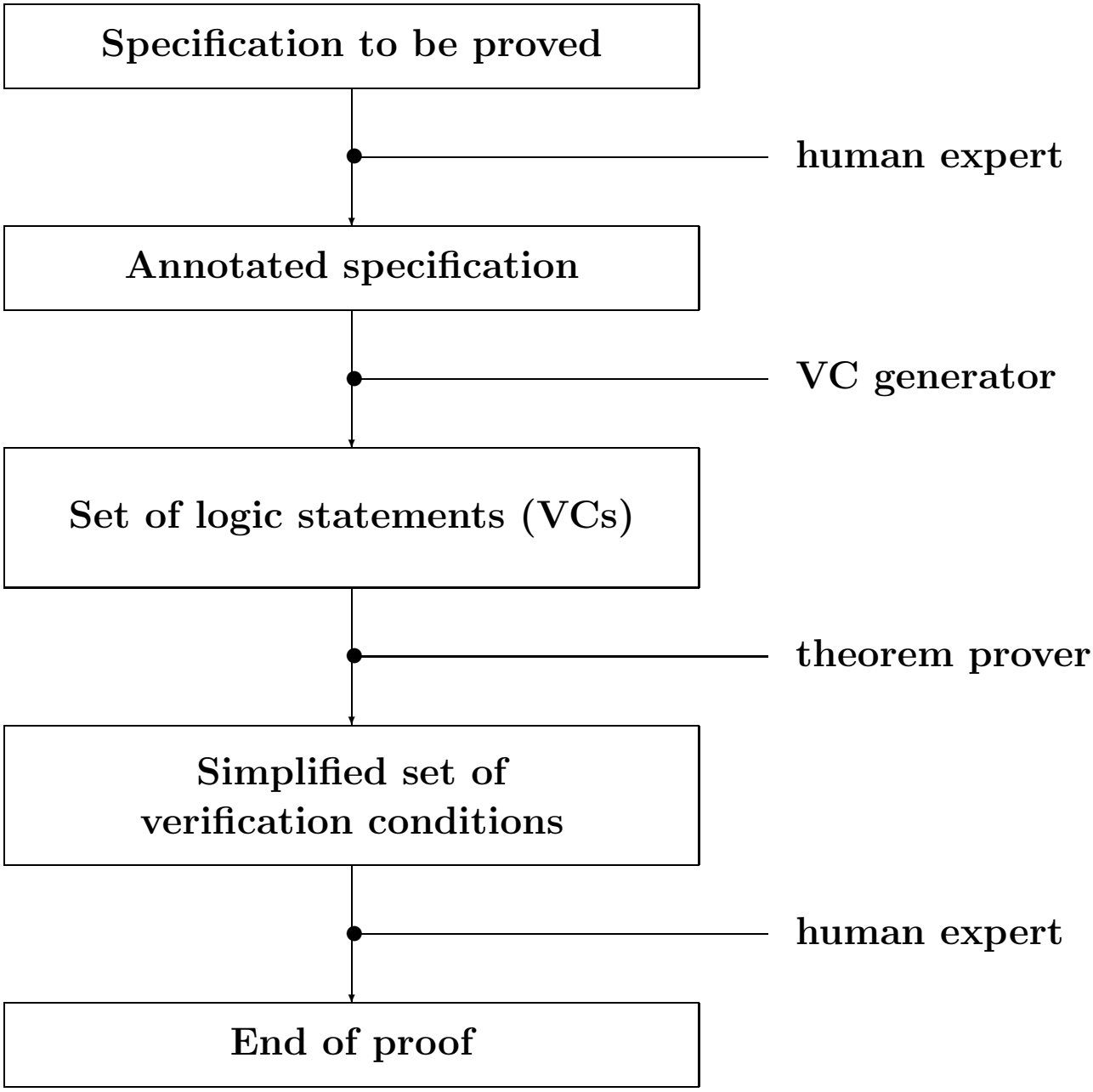
- The architecture of a simple program verifier will b
- Justified with respect to the rules of Floyd-Hoare l
- It is clear that
  - proofs are long and boring, even if the program being ver
  - lots of fiddly little details to get right, many of which are

$$\vdash \ (\mathtt{R}{=}\mathtt{X} \ \wedge \ \mathtt{Q}{=}0) \ \Rightarrow \ (\mathtt{X} = \mathtt{R} + \mathtt{Y}{\times}\mathtt{Q})$$

Mechanization

- **Goal:** automate the routine bits of proofs in Floyd-
- Unfortunately, logicians have shown that it is impos to design a *decision procedure* to decide automatica falsehood of an arbitrary mathematical statement
- This does not mean that one cannot have procedure many *useful* theorems
  - the non-existence of a general decision procedure merely s not hope to prove *everything* automatically
  - in practice, it is quite possible to build a system that boring and routine aspects of verification
- The standard approach to this will be described in
  - ideas very old (JC King's 1969 CMU PhD, Stanford veri
  - used by program verifiers (e.g. Gypsy and SPARK verifie
  - used to validate proof carrying code (PCC)
  - provides a verification front end to different provers (see

Architecture of a Verifier



Commentary

- **Input: a Hoare triple annotated with mathematical**
  - **these annotations describe relationships between variable**
- **The system generates a set of purely mathematical st** *verification conditions* **(or VCs)**
- **If the verification conditions are provable, then the cation can be deduced from the axioms and rules o**
- **The verification conditions are passed to a** *theorem* **which attempts to prove them automatically**
  - **if it fails, advice is sought from the user**

Verification conditions

- The three steps in proving $\{P\}C\{Q\}$ with a verifier
- 1 The program C is *annotated* by inserting statem expressing conditions that are meant to hold at inte
  - this step is tricky and needs intelligence and a good un the program works
  - automating it is an artificial intelligence problem
- 2 A set of logic statements called *verification conditio* generated from the annotated specification
  - this is purely mechanical and easily done by a program
- 3 The verification conditions are proved
  - needs automated theorem proving (i.e. more artificial int
- To improve automated verification one can try to
  - reduce the number and complexity of the annotations re
  - increase the power of the theorem prover
  - still a research area

Validity of Verification Conditions

- **It will be shown that**
  - **if one can prove all the verification conditions generated f**
  - **then** $\vdash \{P\}C\{Q\}$
- **Step** $\boxed{2}$ **converts a verification problem into a conv matical problem**
- **The process will be illustrated with:**

```
{T}
 BEGIN
   R:=X;
   Q:=0;
   WHILE Y≤R DO
     BEGIN R:=R-Y; Q:=Q+1 END
 END
{X = R+Y×Q ∧ R<Y}
```

Example

- **Step 1 is to insert annotations $\mathtt{P}_1$ and $\mathtt{P}_2$**

```
{T}
 BEGIN
   R:=X;
   Q:=0; {R=X ∧ Q=0} ⟵P1
   WHILE Y≤R DO {X = R+Y×Q} ⟵P2
     BEGIN R:=R-Y; Q:=Q+1 END
 END
{X = R+Y×Q ∧ R<Y}
```

- **The annotations $\mathtt{P}_1$ and $\mathtt{P}_2$ state conditions which hold *whenever* control reaches them**

Example Continued

```
{T}
 BEGIN
   R:=X;
   Q:=0; {R=X ∧ Q=0} ⟵P1
   WHILE Y≤R DO {X = R+Y×Q} ⟵P2
     BEGIN R:=R-Y; Q:=Q+1 END
 END
{X = R+Y×Q ∧ R<Y}
```

- **Control only reaches the point at which P_1 is placed**
- **It reaches P_2 each time the WHILE body is executed**
  - **whenever this happens X=R+Y×Q holds, even though the va**
  - **P_2 is an *invariant* of the WHILE-command**

Generating and Proving Verification Condit

- Step 2 will generate the following four verification

  (i) $\mathtt{T} \Rightarrow (\mathtt{X}=\mathtt{X} \wedge \mathtt{0}=\mathtt{0})$

  (ii) $(\mathtt{R}=\mathtt{X} \wedge \mathtt{Q}=\mathtt{0}) \Rightarrow (\mathtt{X} = \mathtt{R}+(\mathtt{Y}\times\mathtt{Q}))$

  (iii) $(\mathtt{X} = \mathtt{R}+(\mathtt{Y}\times\mathtt{Q})) \wedge \mathtt{Y}\leq\mathtt{R}) \Rightarrow (\mathtt{X} = (\mathtt{R}-\mathtt{Y})+(\mathtt{Y}\times(\mathtt{Q}+1)))$

  (iv) $(\mathtt{X} = \mathtt{R}+(\mathtt{Y}\times\mathtt{Q})) \wedge \neg(\mathtt{Y}\leq\mathtt{R}) \Rightarrow (\mathtt{X} = \mathtt{R}+(\mathtt{Y}\times\mathtt{Q}) \wedge \mathtt{R}<\mathtt{Y})$

- Notice that these are statements of arithmetic
  - the constructs of our programming language have been 'c
- Step 3 consists in proving the four verification con
  - easy with modern automatic theorem provers

Annotation of Commands

- **An annotated command is a command with statem embedded within it**
- **A command is *properly annotated* if statements have the following places**

  (i) before C_2 in $C_1;C_2$ if C_2 is *not* an assignment com

  (ii) after the word `DO` in `WHILE` (and `FOR`) commands

- **The inserted assertions should express the condition hold *whenever* control reaches the point at which the**
- **Can reduce number of annotations using weakest p (see later)**

Annotation of Specifications

- **A properly annotated specification is a specification C is a properly annotated command**
- **Example: To be properly annotated, assertions sho ① and ② of the specification below**

```
{X=n}
 BEGIN
   Y:=1; ⟵①
   WHILE X≠0 DO ⟵②
     BEGIN Y:=Y×X; X:=X-1 END
 END
{X=0 ∧ Y=n!}
```

- **Suitable statements would be**

at ①: {Y = 1 ∧ X = n}
at ②: {Y×X! = n!}

Verification Condition Generation

- **The VCs generated from an annotated specificati**
  obtained by considering the various possibilities for
- **We will describe it command by command using ru**
- **The VCs for** $C(C_1, C_2)$ **are**
  - vc_1, ... , vc_n
  - **together with the VCs for** $\mathtt{C}_1$ **and those for** $\mathtt{C}_2$
- **Each VC rule corresponds to either a primitive or**

A VC Generation Program

- The algorithm for generating verification condition
  the structure of commands
- The rule just given corresponds to the recursive pr

  $\mathbf{VC}\ (C(C_1, C_2)) = [vc_1, \ldots, vc_n]$@ $(\mathbf{VC}\ C_1)$ @ $(\mathbf{VC}\ C$
- The rules are chosen so that only one VC rule appl
  - applying them is then purely mechanical
  - the choice is based on the syntax
  - only one rule applies in each case so VC generation is det

Justification of VCs

- **This process will be justified by showing that $\vdash \{P$ verification conditions can be proved**
- **We will prove that for any `C`**
  - **assuming the VCs of $\{P\}C\{Q\}$ are provable**
  - **then $\vdash \{P\}C\{Q\}$ is a theorem of the logic**

Justification of Verification Conditions

- **The argument that the verification conditions are s by *induction* on the structure of C**
- **Such inductive arguments have two parts**
  - **show the result holds for atomic commands: skip and ass**
  - **show that when C is not an atomic command, then if t the constituent commands of C (this is called the *inductio* holds also for C**
- **The first of these parts is called the *basis* of the ind**
- **The second is called the *step***
- **The basis and step entail that the result holds for a**

VC for SKIP

The Skip command

The single verification condition generated by

$$\{P\}\ \texttt{SKIP}\ \{Q\}$$

is

$$P \Rightarrow Q$$

- **Example: The verification condition for**

  {X=0} SKIP {X=0}

  is

  X=0 $\Rightarrow$ X=0

  (which is clearly true)

Justification of SKIP VC

- **We must show that if the VCs of $\{P\}$ SKIP $\{Q\}$ are**
  then $\vdash$ $\{P\}$ SKIP $\{Q\}$

- **Proof:**
  - **Assume $P \Rightarrow Q$ is provable since it is the VC (i.e. assum**
  - **From derived Skip rule it follows that $\vdash$ $\{P\}$ SKIP $\{Q\}$ as**

VC for Assignments

Assignment commands

The single verification condition generated by

$$\{P\}\ V\texttt{:=}E\ \{Q\}$$

is

$$P\ \Rightarrow\ Q\texttt{[}E/V\texttt{]}$$

- **Example: The verification condition for**

  $\{\texttt{X=0}\}\ \texttt{X:=X+1}\ \{\texttt{X=1}\}$

  is

  $\texttt{X=0}\ \Rightarrow\ \texttt{(X+1)=1}$

  (which is clearly true)

- **Note:** $Q\texttt{[}E/V\texttt{]}\ =\ \texttt{wlp("}V\texttt{:=}E\texttt{"},Q)$

Justification of Assignment VC

- **We must show that if the VCs of $\{P\}\ V := E\ \{Q\}$ ar**
  then $\vdash\ \{P\}\ V := E\ \{Q\}$

- **Proof:**
  - **Assume $\vdash\ P\ \Rightarrow\ Q[E/V]$ as it is the VC**
  - **From derived assignment rule it follows that $\vdash\ \{P\}\ V :=$**

VCs for Two-Armed Conditional

Two-armed conditional

The verification conditions generated from

$$\{P\}\ \texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2\ \{Q\}$$

are

(i) the verification conditions generated by

$$\{P\ \wedge\ S\}\ C_1\ \{Q\}$$

(ii) the verifications generated by

$$\{P\ \wedge\ \neg S\}\ C_2\ \{Q\}$$

- **Example: The verification conditions for**

  {T} IF X≥Y THEN MAX:=X ELSE MAX:=Y {MAX=max(X,Y)}

  are

  (i) the VCs for {T ∧ X≥Y} MAX:=X {MAX=max(X,Y)}

  (ii) the VCs for {T ∧ ¬(X≥Y)} MAX:=Y {MAX=max(X,Y)}

Justification for the Conditional VCs (1)

- **Must show that if VCs of**
  $\{P\}$ `IF` S `THEN` C_1 `ELSE` C_2 $\{Q\}$
  are provable, then
  $\vdash$ $\{P\}$ `IF` S `THEN` C_1 `ELSE` C_2 $\{Q\}$

- **Proof:**
  - **Assume the VCs** $\{P \wedge S\}\ C_1\ \{Q\}$ **and** $\{P \wedge \neg S\}\ C_2\ \{Q\}$
  - **The inductive hypotheses tell us that if these VCs are corresponding Hoare Logic theorems are provable**
  - **i.e. by induction** $\vdash \{P \wedge S\}\ C_1\ \{Q\}$ **and** $\vdash \{P \wedge \neg S\}\ C_2$
  - **Hence by the conditional rule** $\vdash \{P\}$ `IF` S `THEN` C_1 `ELSE` C

Review of Annotated Sequences

- **If $C_1 ; C_2$ is properly annotated, then either**

  Case 1: **it is of the form $C_1 ; \{R\} C_2$ and C_2 is not an**

  Case 2: **it is of the form $C ; V := E$**

- **And C, C_1 and C_2 are properly annotated**

VCs for Sequences

Sequences

1. The verification conditions generated by

$$\{P\}\ C_1\ \{R\}\ C_2\ \{Q\}$$

(where C_2 is not an assignment) are the union of:

(a) the verification conditions generated by $\{P\}\ C$

(b) the verifications generated by $\{R\}\ C_2\ \{Q\}$

2. The verification conditions generated by

$$\{P\}\ C;V:=E\ \{Q\}$$

are the verification conditions generated by $\{P\}\ C$

Example

- **The verification conditions generated from**

  $\{X=x \wedge Y=y\}\ R:=X;\ X:=Y;\ Y:=R\ \{X=y \wedge Y=x\}$

- **Are those generated by**

  $\{X=x \wedge Y=y\}\ R:=X;\ X:=Y\ \{(X=y \wedge Y=x)[R/Y]\}$

- **This simplifies to**

  $\{X=x \wedge Y=y\}\ R:=X;\ X:=Y\ \{X=y \wedge R=x\}$

- **The verification conditions generated by this are tho**

  $\{X=x \wedge Y=y\}\ R:=X\ \{(X=y \wedge R=x)[Y/X]\}$

- **Which simplifies to**

  $\{X=x \wedge Y=y\}\ R:=X\ \{Y=y \wedge R=x\}$

Example Continued

- **The only verification condition generated by**

  $\{X{=}x \wedge Y{=}y\}\ R{:=}X\ \{Y{=}y \wedge R{=}x\}$

  is

  $X{=}x \wedge Y{=}y \Rightarrow (Y{=}y \wedge R{=}x)[X/R]$

- **Which simplifies to**

  $X{=}x \wedge Y{=}y \Rightarrow Y{=}y \wedge X{=}x$

- **Thus the single verification condition from**

  $\{X{=}x \wedge Y{=}y\}\ R{:=}X;\ X{:=}Y;\ Y{:=}R\ \{X{=}y \wedge Y{=}x\}$

  is

  $X{=}x \wedge Y{=}y \Rightarrow Y{=}y \wedge X{=}x$

Justification of VCs for Sequences (1)

- **Case 1: If the verification conditions for**

  $\{P\}\ C_1\ ;\ \{R\}\ C_2\ \{Q\}$

  are provable

- **Then the verification conditions for**

  $\{P\}\ C_1\ \{R\}$

  and

  $\{R\}\ C_2\ \{Q\}$

  must both be provable

- **Hence by induction**

  $\vdash\ \{P\}\ C_1\ \{R\}$ **and** $\vdash\ \{R\}\ C_2\ \{Q\}$

- **Hence by the sequencing rule**

  $\vdash\ \{P\}\ C_1;C_2\ \{Q\}$

Justification of VCs for Sequences (2)

- **Case 2: If the verification conditions for**

  $\{P\}\ C\,;V := E\ \{Q\}$

  are provable, then the verification conditions for

  $\{P\}\ C\ \{Q[E/V\}$

  are also provable

- **Hence by induction**

  $\vdash\ \{P\}\ C\ \{Q[E/V]\}$

- **Hence by the derived sequenced assignment rule**

  $\vdash\ \{P\}\ C\,;V := E\ \{Q\}$

VCs for Blocks

Blocks

The verification conditions generated by

$$\{P\}\ \texttt{BEGIN VAR}\ V_1\texttt{; ... ;VAR}\ V_n\texttt{;}C\ \texttt{END}\ \{Q\}$$

are

(i) the verification conditions generated by $\{P\}C\{Q$

(ii) none of V_1, ..., V_n occur in either P or Q.

Syntactic Conditions for Blocks

- Generating verification conditions from blocks invo
  syntactic condition
  - the local variables do not occur in the precondition or po
- The need for this is clear from the side condition in
- Example: the verification conditions for

  $\{$X=x $\wedge$ Y=y$\}$
  BEGIN VAR R; R:=X; X:=Y; Y:=R END
  $\{$X=y $\wedge$ Y=x$\}$

- Are the verification conditions generated by

  $\{$X=x $\wedge$ Y=y$\}$ R:=X; X:=Y; Y:=R $\{$X=y $\wedge$ Y=x$\}$

  and the syntactic check that R not in $\{$X=x $\wedge$ Y=y$\}$ or

- See previous example for verification conditions ger

Justification of VCs for Blocks

- **If the verification conditions for**

  $\{P\}$ `BEGIN VAR` V_1`;` $\ldots$ `;VAR` V_n`;`C `END` $\{Q\}$

  are provable

- **Then the verification conditions for**

  $\{P\}\ C\ \{Q\}$

  are provable and V_1, …, V_n do not occur in P or Q

- **By induction**

  $\vdash \{P\}\ C\ \{Q\}$

- **Hence by the block rule**

  $\vdash$ $\{P\}$ `BEGIN VAR` V_1`;` $\ldots$ `;VAR` V_n`;`C `END` $\{Q\}$

VCs for WHILE-Commands

- **A correctly annotated specification of a WHILE-comma**

  $\{P\}$ WHILE S DO $\{R\}$ C $\{Q\}$

- **The annotation R is called an invariant**

WHILE-commands

The verification conditions generated from

$\{P\}$ WHILE S DO $\{R\}$ C $\{Q\}$

are

(i) $P \Rightarrow R$

(ii) $R \wedge \neg S \Rightarrow Q$

(iii) the verification conditions generated by $\{R \wedge S\}\ C\{R\}$

Example

- **The verification conditions for**

  $\{R=X \wedge Q=0\}$
  WHILE $Y \leq R$ DO $\{X=R+Y \times Q\}$
  BEGIN $R:=R-Y$; $Q:=Q+1$ END
  $\{X = R+(Y \times Q) \wedge R<Y\}$

  are:

  (i) $R=X \wedge Q=0 \Rightarrow (X = R+(Y \times Q))$

  (ii) $X = R+Y \times Q \wedge \neg(Y \leq R) \Rightarrow (X = R+(Y \times Q) \wedge R<Y)$

  together with the verification condition for

  $\{X = R+(Y \times Q) \wedge (Y \leq R)\}$
  BEGIN $R:=R-Y$; $Q:=Q+1$ END
  $\{X=R+(Y \times Q)\}$

  which consists of the single condition

  (iii) $X = R+(Y \times Q) \wedge (Y \leq R) \Rightarrow X = (R-Y)+(Y \times (Q+1))$

Example Summarised

- **By previous transparency**

$$\vdash \{R{=}X \wedge Q{=}0\}$$
$$\texttt{WHILE } Y \leq R \texttt{ DO}$$
$$\texttt{BEGIN } R{:=}R{-}Y;\ Q{:=}Q{+}1 \texttt{ END}$$
$$\{X = R+(Y \times Q) \wedge R<Y\}$$

if

$$\vdash R{=}X \wedge Q{=}0 \Rightarrow (X = R+(Y \times Q))$$

and

$$\vdash X = R+(Y \times Q) \wedge \neg(Y \leq R) \Rightarrow (X = R+(Y \times Q) \wedge R<Y)$$

and

$$\vdash X = R+(Y \times Q) \wedge (Y \leq R) \Rightarrow X = (R-Y)+(Y \times (Q+1))$$

Justification of WHILE VCs

- **If the verification conditions for**

  $\{P\}$ WHILE S DO $\{R\}$ C $\{Q\}$

  are provable, then

  $\vdash \; P \Rightarrow R$

  $\vdash \; (R \; \wedge \; \neg S) \; \Rightarrow \; Q$

  and the verification conditions for

  $\{R \; \wedge \; S\}$ C $\{R\}$

  are provable

- **By induction**

  $\vdash \; \{R \; \wedge \; S\}$ C $\{R\}$

- **Hence by the derived WHILE-rule**

  $\vdash \; \{P\}$ WHILE S DO C $\{Q\}$

Summary

- Have outlined the design of an automated program
- Annotated specifications compiled to mathematical
  - if the statements (VCs) can be proved, the program is ve
- Human help is required to give the annotations and
- The algorithm was justified by an inductive proof
  - it appeals to the derived rules
- All the techniques introduced earlier are used
  - backwards proof
  - derived rules
  - annotation

Dijkstra's weakest preconditions

- **Weakest preconditions is a theory of refinement**
  - **idea is to calculate a program to achieve a postcondition**
  - **not a theory of post hoc verification**
- **Non-determinism a key idea in Dijksta's presentati**
  - **start with a non-deterministic high level pseudo-code**
  - **refine to deterministic and efficient code**
- **Weakest preconditions (wp) are for total correctnes**
- **Weakest *liberal* preconditions (wlp) for partial corre**
- **If C is a command and Q a predicate, then informa**
  - $\texttt{wlp}(C, Q)$ = '*The weakest predicate P such that $\{P\}$*
  - $\texttt{wp}(C, Q)$ = '*The weakest predicate P such that $[P]$ C*
- **If P and Q are predicates then $Q \Rightarrow P$ means P is**

Rules for weakest preconditions

- **Relation with Hoare specifications:**

$$\{P\}\ C\ \{Q\} \quad \Leftrightarrow \quad P \ \Rightarrow\ \texttt{wlp}(C, Q)$$
$$[P]\ C\ [Q] \quad \Leftrightarrow \quad P \ \Rightarrow\ \texttt{wp}(C, Q)$$

- **Dijkstra gives rules for computing weakest precond**

$$\begin{aligned}
&\texttt{wp}(\texttt{SKIP}, Q) &&= Q\\
&\texttt{wp}(V\,\texttt{:=}E, Q) &&= Q[E/V]\\
&\texttt{wp}(C_1\,\texttt{;}\,C_2,\ Q) &&= \texttt{wp}(C_1, \texttt{wp}(C_2,\ Q))\\
&\texttt{wp}(\texttt{IF}\ B\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2,\ Q) &&= (B\ \Rightarrow \texttt{wp}(C_1, Q))\ \wedge\ (
\end{aligned}$$

  for deterministic loop-free code the same equations

- **Rule for `WHILE`-commands doesn't give a first order**
- **Weakest preconditions closely related to verificatio**
- **VCs for $\{P\}\ C\ \{Q\}$ are related to $P\ \Rightarrow\ \texttt{wlp}(C, Q)$**
  - **VCs use annotations to ensure first order formulas can be**

Sequencing example

- **Swapping variables:**

$$\begin{aligned}
&\texttt{wlp}(\texttt{R:=X; X:=Y; Y:=R}, (\texttt{Y} = x \wedge \texttt{X} = y)) \\
&\quad = \texttt{wlp}(\texttt{R:=X}, \texttt{wlp}(\texttt{X:=Y}, \texttt{wlp}(\texttt{Y:=R}, (\texttt{Y} = x \wedge \texttt{X} = y)))) \\
&\quad = \texttt{wlp}(\texttt{R:=X}, \texttt{wlp}(\texttt{X:=Y}, (\texttt{Y} = x \wedge \texttt{X} = y)\texttt{[R/Y]})) \\
&\quad = \texttt{wlp}(\texttt{R:=X}, \texttt{wlp}(\texttt{X:=Y}, (\texttt{R} = x \wedge \texttt{X} = y))) \\
&\quad = \texttt{wlp}(\texttt{R:=X}, (\texttt{R} = x \wedge \texttt{Y} = y)) \\
&\quad = (\texttt{X} = x \wedge \texttt{Y} = y)
\end{aligned}$$

- **So since** $\{P\}\ C\ \{Q\} \Leftrightarrow P \Rightarrow \texttt{wlp}(C, Q)$

  to prove

  $\{\texttt{X} = x \wedge \texttt{Y} = y\}\ \texttt{R:=X; X:=Y; Y:=R}\ \{\texttt{Y} = x \wedge \texttt{X} = y\}$

  just need to prove:

  $(\texttt{X} = x \wedge \texttt{Y} = y) \Rightarrow (\texttt{X} = x \wedge \texttt{Y} = y)$

  which is clearly true (instance of $S \Rightarrow S$**)**

Conditional example

- **Compute `wlp` of the maximum program:**

  $\texttt{wlp}(\texttt{IF X<Y THEN MAX:=Y ELSE MAX:=X}, (\texttt{MAX} = max(x, y)$

  $= \ (\texttt{X<Y} \ \Rightarrow \ \texttt{wlp}(\texttt{MAX:=Y}, \ (\texttt{MAX} = max(x, y))))$
  $\wedge$
  $(\neg(\texttt{X<Y}) \ \Rightarrow \ \texttt{wlp}(\texttt{MAX:=X}, \ (\texttt{MAX} = max(x, y))))$

  $= \ (\texttt{X<Y} \ \Rightarrow \ \texttt{Y} = max(x, y)) \ \wedge \ (\neg(\texttt{X<Y}) \ \Rightarrow \ \texttt{X} = max($

  $= \ \textit{if} \ \texttt{X<Y} \ \textit{then} \ \texttt{Y} = max(x, y) \ \textit{else} \ \texttt{X} = max(x, y)$

- **So to prove**

  $\{\texttt{X} = x \wedge \texttt{Y} = y\} \ \texttt{IF X<Y THEN MAX:=X ELSE MAX:=Y} \ \{\texttt{MAX}$

  just prove:

  $(\texttt{X} = x \wedge \texttt{Y} = y) \Rightarrow (\texttt{X<Y} \ \Rightarrow \ \texttt{Y} = max(x, y)) \ \wedge \ (\neg(\texttt{X<Y}) \ \Rightarrow$

  which follows from the defining property of max

  $\vdash \quad \forall x \ y. \ (x \geq y \Rightarrow x = max(x, y)) \ \wedge \ (\neg(x \geq y) \Rightarrow y = m$

Using `wlp` to improve verification condition

- **If C is loop-free then VC for $\{P\}\ C\ \{Q\}$ is $P \Rightarrow$ `wlp`**
  - **no annotations needed in sequences!**
- **Cannot in general compute a finite formula for `wlp`**
- **The following holds**

  $\texttt{wlp}(\texttt{WHILE}\ S\ \texttt{DO}\ C,\ Q)\ =\ \mathit{if}\ S\ \mathit{then}\ \texttt{wlp}(C,\ \texttt{wlp}(\texttt{WHILE}\ S$
- **Above doesn't define $\texttt{wlp}(C, Q)$ as a finite statement**
- **We will describe a hybrid VC and `wlp` method**

wlp-based verification condition method

- **We define $\texttt{awp}(C, Q)$ and $\texttt{wvc}(C, Q)$**
  - $\texttt{awp}(C, Q)$ **is a statement sort of approximating** $\texttt{wlp}(C, Q)$
  - $\texttt{wvc}(C, Q)$ **is a set of verification conditions**
- **If C is loop-free then**
  - $\texttt{awp}(C, Q) = \texttt{wlp}(C, Q)$
  - $\texttt{wvc}(C, Q) = \{\}$
- **Denote by $\bigwedge\mathcal{S}$ the conjunction of all the statements**
  - $\bigwedge\{\} = \texttt{T}$
  - $\bigwedge(\mathcal{S}_1 \cup \mathcal{S}_2) = (\bigwedge\mathcal{S}_1) \wedge (\bigwedge\mathcal{S}_2)$
- **It will follow that** $\bigwedge\texttt{wvc}(C, Q) \Rightarrow \{\texttt{awp}(C, Q)\}\ C\ \{Q\}$
- **Hence to prove $\{P\}C\{Q\}$ it is sufficient to prove all the statements in** $\texttt{wvc}(C, Q)$ **and** $P \Rightarrow \texttt{awp}(C, Q)$

Definition of awp

- **Assume all WHILE-commands are annotated:** $\texttt{WHILE}\ S$
- **Define awp recursively by:**

$$
\begin{array}{ll}
\texttt{awp}(\texttt{SKIP},\ Q) & = Q \\
\texttt{awp}(V\ \texttt{:=}\ E,\ Q) & = Q\,[E/V] \\
\texttt{awp}(C_1\ \texttt{;}\ C_2,\ Q) & = \texttt{awp}(C_1,\ \texttt{awp}(C_2,\ Q)) \\
\texttt{awp}(\texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2,\ Q) & = (S\ \wedge \texttt{awp}(C_1,\ Q)) \vee (\neg \\
\texttt{awp}(\texttt{WHILE}\ S\ \texttt{DO}\ \{R\}\ C,\ Q) & = R
\end{array}
$$

- **Note:**
  $(S\ \wedge \texttt{awp}(C_1,\ Q)) \vee (\neg S \wedge \texttt{awp}(C_2,\ Q)\ =\ \textit{if}\ S\ \textit{then}\ \texttt{awp}(C_1,\ C$

Definition of wvc

- **Assume all WHILE-commands are annotated:** WHILE S
- **Define wvc recursively by:**

$$
\begin{array}{ll}
\mathtt{wvc}(\mathtt{SKIP},\ Q) & = \{\} \\
\mathtt{wvc}(V\ \mathtt{:=}\ E,\ Q) & = \{\} \\
\mathtt{wvc}(C_1\ \mathtt{;}\ C_2,\ Q) & = \mathtt{wvc}(C_1, \mathtt{awp}(C_2, Q)) \cup \mathtt{w} \\
\mathtt{wvc}(\mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2,\ Q) & = \mathtt{wvc}(C_1,\ Q) \cup \mathtt{wvc}(C_2,\ \\
\mathtt{wvc}(\mathtt{WHILE}\ S\ \mathtt{DO}\ \{R\}\ C,\ Q) & = \{R \wedge \neg S \Rightarrow Q,\ R \wedge S = \\
 & \quad \cup\, \mathtt{wvc}(C, R)
\end{array}
$$

Correctness of wlp-based verification conditi

- **Theorem:** $\bigwedge \mathtt{wvc}(C, Q) \Rightarrow \{\mathtt{awp}(C, Q)\}\ C\ \{Q\}$. **Proof by I**
  - $\bigwedge \mathtt{wvc}(\mathtt{SKIP}, Q) \Rightarrow \{\mathtt{awp}(C, Q)\}\ C\ \{Q\}$ **is** $\mathtt{T} \Rightarrow \{Q\}\ \mathtt{SKIP}\ \{Q\}$
  - $\bigwedge \mathtt{wvc}(V := E, Q) \Rightarrow \{\mathtt{awp}(C, Q)\}\ C\ \{Q\}$ **is** $\mathtt{T} \Rightarrow \{Q[E/V]\}\ V\ :=$
  - $\bigwedge \mathtt{wvc}(C_1; C_2, Q) \Rightarrow \{\mathtt{awp}(C_1; C_2, Q)\}\ C_1; C_2\ \{Q\}$ **is**
    $\bigwedge(\mathtt{wvc}(C_1, \mathtt{awp}(C_2, Q)) \cup \mathtt{wvc}(C_2, Q)) \Rightarrow \{\mathtt{awp}(C_1,\ \mathtt{awp}(C_2, Q))\}\ C_1$
    By induction $\bigwedge \mathtt{wvc}(C_2, Q) \Rightarrow \{\mathtt{awp}(C_2, Q)\}\ C_1\ \{Q\}$
    and $\bigwedge \mathtt{wvc}(C_1, \mathtt{awp}(C_2, Q)) \Rightarrow \{\mathtt{awp}(C_1, \mathtt{awp}(C_2, Q))\}\ C_2\ \{\mathtt{awp}(C_2, Q$
    hence result by the Sequencing Rule.
  - $\bigwedge \mathtt{wvc}(\mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2, Q)$
    $\Rightarrow \{\mathtt{awp}(\mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2, Q)\}\ \mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2\ \{Q\}$
    is $\bigwedge(\mathtt{wvc}(C_1,\ Q) \cup \mathtt{wvc}(C_2,\ Q))$
    $\Rightarrow \{(S\ \wedge \mathtt{awp}(C_1,\ Q)) \vee (\neg S \wedge \mathtt{awp}(C_2,\ Q)\}\ \mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}$
    By induction $\bigwedge \mathtt{wvc}(C_1, Q) \Rightarrow \{\mathtt{awp}(C_1, Q)\}\ C_1\ \{Q\}$
    and $\bigwedge \mathtt{wvc}(C_2, Q) \Rightarrow \{\mathtt{awp}(C_2, Q)\}\ C_2\ \{Q\}$**. Strengthening prec**
    gives $\bigwedge \mathtt{wvc}(C_1, Q) \Rightarrow \{\mathtt{awp}(C_1, Q) \wedge S\}\ C_1\ \{Q\}$
    and $\bigwedge \mathtt{wvc}(C_2, Q) \Rightarrow \{\mathtt{awp}(C_2, Q) \wedge \neg S\}\ C_2\ \{Q\}$**, hence**
    $\bigwedge \mathtt{wvc}(C_1, Q) \Rightarrow \{((S\ \wedge \mathtt{awp}(C_1,\ Q)) \vee (\neg S \wedge \mathtt{awp}(C_2,\ Q))) \wedge S\}\ C$
    and $\bigwedge \mathtt{wvc}(C_2, Q) \Rightarrow \{((S\ \wedge \mathtt{awp}(C_1,\ Q)) \vee (\neg S \wedge \mathtt{awp}(C_2,\ Q))) \wedge$
    hence result by the Conditional Rule.
  - $\bigwedge \mathtt{wvc}(\mathtt{WHILE}\ S\ \mathtt{DO}\ \{R\}\ C, Q) \Rightarrow \{\mathtt{awp}(\mathtt{WHILE}\ S\ \mathtt{DO}\ \{R\}\ C, Q)\}\ \mathtt{WH}$
    is $\bigwedge(\{R \wedge \neg S \Rightarrow Q,\ R \wedge S \Rightarrow \mathtt{awp}(C, R)\} \cup \mathtt{wvc}(C, R)) \Rightarrow \{R\}\ \mathtt{WH}$
    By induction $\bigwedge \mathtt{wvc}(C, R) \Rightarrow \{\mathtt{awp}(C, R)\}\ C\ \{R\}$**, hence result**

Example

$\texttt{wvc}(\texttt{WHILE}\ S\ \texttt{DO}\ \{R\}\ C,\ Q)\ =\ (R,\ \{R \wedge \neg S \Rightarrow Q,\ R \wedge S \Rightarrow \texttt{awp}(C, R)\} \cup$

$\texttt{wvc}(\texttt{R:=R-Y;Q:=Q+1},\ \texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q})$
$= (\texttt{wlp}(\texttt{R:=R-Y;Q:=Q+1},\ \texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q}), \{\})$
$= (\texttt{X} = \texttt{R-Y} + \texttt{Y} \times \texttt{Q+1},\ \{\})$

$\texttt{wvc}(\texttt{WHILE}\ \texttt{Y} \leq \texttt{R}\ \texttt{DO}\ \{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q}\}\ \texttt{R:=R-Y;Q:=Q+1},\ \texttt{X} = \texttt{R+Y}\times\texttt{Q} \wedge \texttt{R}<\texttt{Y}$
$= (\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q},$
$\{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \neg(\texttt{Y} \leq \texttt{R}) \Rightarrow \texttt{X} = \texttt{R+Y}\times\texttt{Q} \wedge \texttt{R}<\texttt{Y},$
$\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \texttt{Y} \leq \texttt{R} \Rightarrow \texttt{X} = \texttt{R-Y} + \texttt{Y} \times \texttt{Q+1}\} \cup \{\})$

$\texttt{wvc}(\texttt{Q:=0;WHILE}\ \texttt{Y} \leq \texttt{R}\ \texttt{DO}\ \{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q}\}\ \texttt{R:=R-Y;}\ \texttt{Q:=Q+1},\ \texttt{X} = \texttt{R+Y}\times\texttt{Q}$
$= (\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{0},$
$\{\} \cup \{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \neg(\texttt{Y} \leq \texttt{R}) \Rightarrow \texttt{X} = \texttt{R+Y}\times\texttt{Q} \wedge \texttt{R}<\texttt{Y},$
$\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \texttt{Y} \leq \texttt{R} \Rightarrow \texttt{X} = \texttt{R-Y} + \texttt{Y} \times \texttt{Q+1}\})$

$\texttt{wvc}(\texttt{R=X;Q:=0;WHILE}\ \texttt{Y} \leq \texttt{R}\ \texttt{DO}\ \{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q}\}\ \texttt{R:=R-Y;}\ \texttt{Q:=Q+1},\ \texttt{X} = \texttt{R+}$
$= (\texttt{X} = \texttt{X} + \texttt{Y} \times \texttt{0},$
$\{\} \cup \{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \neg(\texttt{Y} \leq \texttt{R}) \Rightarrow \texttt{X} = \texttt{R+Y}\times\texttt{Q} \wedge \texttt{R}<\texttt{Y},$
$\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \texttt{Y} \leq \texttt{R} \Rightarrow \texttt{X} = \texttt{R-Y} + \texttt{Y} \times \texttt{Q+1}\})$

As $\texttt{X} = \texttt{X} + \texttt{Y} \times \texttt{0}$ is $\texttt{T}$ the correctness theorem for $\texttt{wvc}$ gives

$\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \neg(\texttt{Y} \leq \texttt{R}) \Rightarrow \texttt{X} = \texttt{R+Y}\times\texttt{Q} \wedge \texttt{R}<\texttt{Y}\ \wedge\ \texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q} \wedge \texttt{Y} \leq \texttt{R}$
$\Rightarrow$
$\{\texttt{T}\}\ \texttt{R=X;Q:=0;WHILE}\ \texttt{Y} \leq \texttt{R}\ \texttt{DO}\ \{\texttt{X} = \texttt{R} + \texttt{Y} \times \texttt{Q}\}\ \{\texttt{X} = \texttt{R+Y}\times\texttt{Q} \wedge \texttt{R}<\texttt{Y}\}$

Strongest postconditions

- **Define $\texttt{sp}(C, P)$ to be 'strongest' Q such that $\{P\}\ C$**
  - **partial correctness:** $\{P\}\ C\ \{\texttt{sp}(C, P)\}$
  - **strongest means if** $\{P\}\ C\ \{Q\}$ **then** $\texttt{sp}(C, P) \Rightarrow Q$
- **Note that `wlp` goes 'backwards', but `sp` goes 'forwar**
  - **verification condition for** $\{P\}\ C\ \{Q\}$ **is:** $\texttt{sp}(C, P) \Rightarrow Q$
- **By 'strongest' and Hoare logic postcondition weake**
  - $\{P\}\ C\ \{Q\}$ **if and only if** $\texttt{sp}(C, P) \Rightarrow Q$

Strongest postconditions for loop-free code

- **Only consider loop-free code**
- $\texttt{sp}(\texttt{SKIP},\ P)\ =\ P$
- $\texttt{sp}(V\ \texttt{:=}\ E,\ P)\ =\ \exists v.\ V = E[v/V] \wedge P[v/V]$
- $\texttt{sp}(C_1\ \texttt{;}\ C_2,\ P)\ =\ \texttt{sp}(C_2,\ \texttt{sp}(C_1,\ P))$
- $\texttt{sp}(\texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2,\ P)\ =\ \texttt{sp}(C_1,\ P \wedge S)\ \vee\ \texttt{sp}(C$

- $\texttt{sp}(V\texttt{:=}E,\ P)$ **corresponds to Floyd assignment axion**
- **Can dynamically prune conditionals because** $\texttt{sp}(C,$

Sequencing example

- $\mathtt{sp}(\mathtt{R{:=}X};\ \mathtt{X{:=}Y};\ \mathtt{Y{:=}R},\ \mathtt{X} = x \wedge \mathtt{Y} = y)$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ \mathtt{sp}(\mathtt{X{:=}Y},\ \mathtt{sp}(\mathtt{R{:=}X},\ \mathtt{X} = x \wedge \mathtt{Y} = y)))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ \mathtt{sp}(\mathtt{X{:=}Y},\ (\exists v.\ \mathtt{R} = \mathtt{X}[v/\mathtt{R}] \wedge (\mathtt{X} = x \wedge \mathtt{Y} =$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ \mathtt{sp}(\mathtt{X{:=}Y},\ (\exists v.\ \mathtt{R} = \mathtt{X} \wedge (\mathtt{X} = x \wedge \mathtt{Y} = y))))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ \mathtt{sp}(\mathtt{X{:=}Y},\ (\mathtt{R} = \mathtt{X} \wedge \mathtt{X} = x \wedge \mathtt{Y} = y)))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ (\exists v.\ \mathtt{X} = \mathtt{Y}[v/\mathtt{X}] \wedge (\mathtt{R} = \mathtt{X} \wedge \mathtt{X} = x \wedge \mathtt{Y} = y)$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ (\exists v.\ \mathtt{X} = \mathtt{Y} \wedge (\mathtt{R} = v \wedge v = x \wedge \mathtt{Y} = y)))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ (\exists v.\ \mathtt{X} = \mathtt{Y} \wedge (\mathtt{R} = x \wedge v = x \wedge \mathtt{Y} = y)))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ (\mathtt{X} = \mathtt{Y} \wedge (\mathtt{R} = x \wedge (\exists v.\ v = x) \wedge \mathtt{Y} = y)))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ (\mathtt{X} = \mathtt{Y} \wedge (\mathtt{R} = x \wedge \mathtt{T} \wedge \mathtt{Y} = y)))$
  $= \mathtt{sp}(\mathtt{Y{:=}R},\ (\mathtt{X} = \mathtt{Y} \wedge \mathtt{R} = x \wedge \mathtt{Y} = y))$
  $= \exists v.\ \mathtt{Y} = \mathtt{R}[v/\mathtt{Y}] \wedge (\mathtt{X} = \mathtt{Y} \wedge \mathtt{R} = x \wedge \mathtt{Y} = y)[v/\mathtt{Y}]$
  $= \exists v.\ \mathtt{Y} = \mathtt{R} \wedge (\mathtt{X} = v \wedge \mathtt{R} = x \wedge v = y)$
  $= \exists v.\ \mathtt{Y} = \mathtt{R} \wedge (\mathtt{X} = y \wedge \mathtt{R} = x \wedge v = y)$
  $= \mathtt{Y} = \mathtt{R} \wedge (\mathtt{X} = y \wedge \mathtt{R} = x \wedge (\exists v.\ v = y))$
  $= \mathtt{Y} = \mathtt{R} \wedge (\mathtt{X} = y \wedge \mathtt{R} = x \wedge \mathtt{T})$
  $= \mathtt{Y} = \mathtt{R} \wedge \mathtt{X} = y \wedge \mathtt{R} = x$
  $= \mathtt{Y} = x \wedge \mathtt{X} = y \wedge \mathtt{R} = x$

- **So to prove** $\{\mathtt{X} = x \wedge \mathtt{Y} = y\}\ \mathtt{R{:=}X};\ \mathtt{X{:=}Y};\ \mathtt{Y{:=}R}\ \{\mathtt{Y} = x \wedge$
  just prove: $(\mathtt{Y} = x \wedge \mathtt{X} = y \wedge \mathtt{R} = x) \Rightarrow \mathtt{Y} = x \wedge \mathtt{X} = y$

Conditional example

- **Compute sp of the maximum program:**

$$
\begin{aligned}
&\mathtt{sp}(\mathtt{IF\ X<Y\ THEN\ MAX:=Y\ ELSE\ MAX:=X},\ (\mathtt{X} = x \wedge \mathtt{Y} = y)) \\
&= \mathtt{sp}(\mathtt{MAX:=Y},\ ((\mathtt{X} = x \wedge \mathtt{Y} = y) \wedge \mathtt{X} < \mathtt{Y})) \\
&\quad \vee \\
&\quad \mathtt{sp}(\mathtt{MAX:=X},\ ((\mathtt{X} = x \wedge \mathtt{Y} = y) \wedge \neg(\mathtt{X} < \mathtt{Y}))) \\
&= \exists v.\ \mathtt{MAX} = \mathtt{Y}[v/\mathtt{MAX}] \wedge ((\mathtt{X} = x \wedge \mathtt{Y} = y) \wedge \mathtt{X} < \mathtt{Y})[v/\mathtt{MA} \\
&\quad \vee \\
&\quad \exists v.\ \mathtt{MAX} = \mathtt{X}[v/\mathtt{MAX}] \wedge ((\mathtt{X} = x \wedge \mathtt{Y} = y) \wedge \neg(\mathtt{X} < \mathtt{Y}))[v/ \\
&= \exists v.\ \mathtt{MAX} = \mathtt{Y} \wedge ((\mathtt{X} = x \wedge \mathtt{Y} = y) \wedge \mathtt{X} < \mathtt{Y}) \\
&\quad \vee \\
&\quad \exists v.\ \mathtt{MAX} = \mathtt{X} \wedge \mathtt{X} = x \wedge \mathtt{Y} = y \wedge \neg(\mathtt{X} < \mathtt{Y})) \\
&= (\mathtt{MAX} = \mathtt{Y} \wedge \mathtt{X} = x \wedge \mathtt{Y} = y \wedge \mathtt{X} < \mathtt{Y}) \vee (\mathtt{MAX} = \mathtt{X} \wedge \mathtt{X} = x \\
&= (\mathtt{MAX} = y \wedge \mathtt{X} = x \wedge \mathtt{Y} = y \wedge x < y) \vee (\mathtt{MAX} = x \wedge \mathtt{X} = x \\
&= \mathit{if}\ x < y\ \mathit{then}\ (\mathtt{MAX} = y \wedge \mathtt{X} = x \wedge \mathtt{Y} = y)\ \mathit{else}\ (\mathtt{MAX} = \\
&= \mathtt{MAX} = (\mathit{if}\ x < y\ \mathit{then}\ y\ \mathit{else}\ x) \wedge \mathtt{X} = x \wedge \mathtt{Y} = y \\
&= \mathtt{MAX} = max(x, y) \wedge \mathtt{X} = x \wedge \mathtt{Y} = y
\end{aligned}
$$

Computing sp as symbolic execution: assign

- **Floyd assignment formula makes computing sp mes**
- **For a special case it becomes like symbolic executio**
- **Symbolic state:** $\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E_i \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge$
- **Suppose** $E_1, \ldots, E_n$ **or** R **doesn't contain** $\mathtt{X}_1, \ldots, \mathtt{X}_n$ **t**

$$
\begin{aligned}
&\mathtt{sp}(\mathtt{X}_i := E,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E_i \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)) \\
&= \exists v.\ \mathtt{X}_i = E[v/\mathtt{X}_i] \wedge (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E_i \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)[v \\
&= \exists v.\ \mathtt{X}_i = E[v/\mathtt{X}_i] \wedge (\mathtt{X}_1 = E_1 \wedge \cdots \wedge v = E_i \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R) \\
&= \mathtt{X}_i = E[E_i/\mathtt{X}_i] \wedge (\mathtt{X}_1 = E_1 \wedge \cdots \wedge (\exists v.\ v = E_i) \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R) \\
&= \mathtt{X}_i = E[E_i/\mathtt{X}_i] \wedge (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{T} \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R) \\
&= \mathtt{X}_i = E[E_i/\mathtt{X}_i] \wedge \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R \\
&= \mathtt{X}_i = E[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n] \wedge \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R \\
&= \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n] \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R
\end{aligned}
$$

- **Note** $E[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n]$ **doesn't contain** $\mathtt{X}_1, \ldots, \mathtt{X}_n$

Computing sp as symbolic execution: assign

- **Suppose $\mathtt{X} \neq \mathtt{X}_i$ and $\mathtt{X}$ doesn't occur in E_i or E for $1 \leq$**

$$\mathtt{sp}(\mathtt{X}:=E,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$$
$$= \exists v.\ \mathtt{X} = E[v/\mathtt{X}] \wedge (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)[v/\mathtt{X}]$$
$$= \exists v.\ \mathtt{X} = E \wedge \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R$$
$$= \mathtt{X} = E[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \wedge \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R$$

- **Summarising: we have two symbolic computation r**

  1. **if $E_1, \ldots, E_n$ or R doesn't contain $\mathtt{X}_1, \ldots, \mathtt{X}_n$ then:**

$$\mathtt{sp}(\mathtt{X}_i:=E,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E_i \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$$
$$= (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R$$

  2. **if $\mathtt{X} \neq \mathtt{X}_i$ and $\mathtt{X}$ doesn't occur in E_i or E for $1 \leq i \leq$**

$$\mathtt{sp}(\mathtt{X}:=E,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$$
$$= (\mathtt{X} = E[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \wedge \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)$$

Computing sp as symbolic execution: condi

- **Suppose if none of $E_1, \ldots, E_n$, R_1, R_2 contain $\mathtt{X}_1, \ldots,$**

$$\mathtt{sp}(C_1,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R_1)) \ = \ (\mathtt{X}_1 = E_{11} \wedge \cdots$$
$$\mathtt{sp}(C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R_2)) \ = \ (\mathtt{X}_1 = E_{21} \wedge \cdots$$

- **Conditional notation:** $(B \rightarrow E_1 \mid E_2) \ = \ \mathit{if}\ B\ \mathit{then}\ E_1$
- **Then for $E_1, \ldots, E_n$, R not containing $\mathtt{X}_1, \ldots, \mathtt{X}_n$**

$$\begin{aligned}
&\mathtt{sp}(\mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)) \\
&= \mathtt{sp}(C_1,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R) \wedge S) \\
&\quad \vee \\
&\quad \mathtt{sp}(C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R) \wedge \neg S) \\
&= \mathtt{sp}(C_1,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge (R \wedge S[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n]))) \\
&\quad \vee \\
&\quad \mathtt{sp}(C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge (R \wedge \neg S[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n]))) \\
&= (\mathtt{X}_1 = E_{11} \wedge \cdots \wedge \mathtt{X}_n = E_{1n} \wedge (R \wedge S[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n])) \qquad \textbf{(b} \\
&\quad \vee \\
&\quad (\mathtt{X}_1 = E_{21} \wedge \cdots \wedge \mathtt{X}_n = E_{2n} \wedge (R \wedge \neg S[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n])) \\
&= (\mathtt{X}_1 = (S[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \rightarrow E_{11} \mid E_{21})) \wedge \cdots \wedge (\mathtt{X}_n = (S[E_1/\mathtt{X}_1]\cdots[E_n
\end{aligned}$$

Computing sp as symbolic execution: condi

- From last slide if $E_1, \ldots, E_n$, R do not contain $\mathtt{X}_1, \ldots,$

  $\mathtt{sp}(\mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$

  $= (\mathtt{X}_1 = (S\,[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n] \rightarrow E_{11} | E_{21})) \wedge \cdots \wedge (\mathtt{X}_n = (S\,[E_1/\mathtt{X}_1] \cdots [E_n$

  where

  $\mathtt{sp}(C_1,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R_1)) = (\mathtt{X}_1 = E_{11} \wedge \cdots \wedge \mathtt{X}_n = E_{1n} \wedge$

  $\mathtt{sp}(C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R_2)) = (\mathtt{X}_1 = E_{21} \wedge \cdots \wedge \mathtt{X}_n = E_{2n} \wedge$

- If C_1 or C_2 don't assign to $\mathtt{X}_i$ then $E_i = E_{1i} = E_{2i}$ so
  $(S\,[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n] \rightarrow E_{1i} | E_{2i}) = E_i$
  so formula above can be further simplified

- If R determines the value of $S\,[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n]$ ther
  $(\mathtt{X}_i = (S\,[E_1/\mathtt{X}_1] \cdots [E_n/\mathtt{X}_n] \rightarrow E_{1i} \mid E_{2i}))$

Summary of sp loop-free code symbolic exec

- **Symbolic state:** $(\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)$
- $\mathtt{sp}(\mathtt{SKIP},\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)) = (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R$
- **If E_1, ..., E_n or R doesn't contain $\mathtt{X}_1$, ..., $\mathtt{X}_n$ then:**

  $\mathtt{sp}(\mathtt{X}_i\mathtt{:=}E,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E_i \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$

  $= (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_i = E[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R$
- **If $\mathtt{X} \neq \mathtt{X}_i$ and $\mathtt{X}$ doesn't occur in E_i or E for $1 \leq i \leq n$**

  $\mathtt{sp}(\mathtt{X}\mathtt{:=}E,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$

  $= (\mathtt{X} = E[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \wedge \mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R)$
- **If E_1, ...,E_n, R does not contain $\mathtt{X}_1$, ..., $\mathtt{X}_n$**

  $\mathtt{sp}(\mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R))$

  $= (\mathtt{X}_1{=}(S[E_1/\mathtt{X}_1]\cdots[E_n/\mathtt{X}_n] \rightarrow E_{11}|E_{21}))\wedge\cdots\wedge(\mathtt{X}_n{=}(S[E_1/\mathtt{X}_1]\cdots[E_n$

  where

  $\mathtt{sp}(C_1,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R_1)) = (\mathtt{X}_1 = E_{11} \wedge \cdots \wedge \mathtt{X}_n = E_{1n}$

  $\mathtt{sp}(C_2,\ (\mathtt{X}_1 = E_1 \wedge \cdots \wedge \mathtt{X}_n = E_n \wedge R_2)) = (\mathtt{X}_1 = E_{21} \wedge \cdots \wedge \mathtt{X}_n = E_{2n}$

Example symbolic computation using sp (1)

- $\mathtt{sp}(C_1 \ ; \ C_2, \ P) \ = \ \mathtt{sp}(C_2, \ \mathtt{sp}(C_1, \ P))$ **hence**

$\mathtt{sp}(\mathtt{R} := 0;$
$\quad \mathtt{K} := 0;$
$\quad \mathtt{IF}\ \mathtt{I} < \mathtt{J}\ \mathtt{THEN}\ \mathtt{K} := \mathtt{K} + 1\ \mathtt{ELSE}\ \mathtt{SKIP};$
$\quad \mathtt{IF}\ \mathtt{K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN}\ \mathtt{R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE}\ \mathtt{R} := \mathtt{I} - \mathtt{J},$
$\quad (\mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j)) \ =$

$\mathtt{sp}(\mathtt{K} := 0;$
$\quad \mathtt{IF}\ \mathtt{I} < \mathtt{J}\ \mathtt{THEN}\ \mathtt{K} := \mathtt{K} + 1\ \mathtt{ELSE}\ \mathtt{SKIP};$
$\quad \mathtt{IF}\ \mathtt{K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN}\ \mathtt{R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE}\ \mathtt{R} := \mathtt{I} - \mathtt{J},$
$\quad (\mathtt{R} = 0 \wedge \mathtt{I} = i \wedge \mathtt{J} = i \wedge i < j)) \ =$

$\mathtt{sp}(\mathtt{IF}\ \mathtt{I} < \mathtt{J}\ \mathtt{THEN}\ \mathtt{K} := \mathtt{K} + 1\ \mathtt{ELSE}\ \mathtt{SKIP};$
$\quad \mathtt{IF}\ \mathtt{K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN}\ \mathtt{R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE}\ \mathtt{R} := \mathtt{I} - \mathtt{J},$
$\quad (\mathtt{K} = 0 \wedge \mathtt{R} = 0 \wedge \mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j))$

Example symbolic computation using sp (2)

Considering each conditional branch and merging gives

$$\begin{array}{l}\texttt{sp}(\texttt{IF I} < \texttt{J THEN K} := \texttt{K} + 1 \texttt{ ELSE SKIP};\\ \quad \texttt{IF K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \texttt{ THEN R} := \texttt{J} - \texttt{I ELSE R} := \texttt{I} - \texttt{J},\\ \quad (\texttt{K} = 0 \wedge \texttt{R} = 0 \wedge \texttt{I} = i \wedge \texttt{J} = j \wedge i < j)) \;=\\ (\texttt{K}=(i<j \rightarrow E_{11} \mid E_{12}) \wedge \texttt{R}=(i<j \rightarrow E_{21} \mid E_{22}) \wedge \texttt{I}=(i<j \rightarrow E_{31} \mid E_{32}) \wedge \texttt{J}=(\end{array}$$

Noting '$\wedge\, i < j$' it follows that this simplifies to:

$$\begin{array}{l}\texttt{sp}(\texttt{IF I} < \texttt{J THEN K} := \texttt{K} + 1 \texttt{ ELSE SKIP};\\ \quad \texttt{IF K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \texttt{ THEN R} := \texttt{J} - \texttt{I ELSE R} := \texttt{I} - \texttt{J},\\ \quad (\texttt{K} = 0 \wedge \texttt{R} = 0 \wedge \texttt{I} = i \wedge \texttt{J} = j \wedge i < j)) \;=\\ (\texttt{K} = E_{11} \wedge \texttt{R} = E_{21} \wedge \texttt{I} = E_{31} \wedge \texttt{J} = E_{41} \wedge i < j)\end{array}$$

so need only consider the `I < J` branch

Example symbolic computation using sp (3)

$$\begin{array}{l}
\mathtt{sp}(\mathtt{R} := 0; \\
\quad \mathtt{K} := 0; \\
\quad \mathtt{IF\ I} < \mathtt{J\ THEN\ K} := \mathtt{K} + 1\ \mathtt{ELSE\ SKIP}; \\
\quad \mathtt{IF\ K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN\ R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE\ R} := \mathtt{I} - \mathtt{J}, \\
\quad (\mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j)) \ = \\
\\
\mathtt{sp}(\mathtt{IF\ K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN\ R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE\ R} := \mathtt{I} - \mathtt{J}, \\
\quad (\mathtt{K} = (\mathtt{K}+1)[0/\mathtt{K}] \wedge \mathtt{R} = 0 \wedge \mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j)) \ = \\
\\
\mathtt{sp}(\mathtt{IF\ K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN\ R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE\ R} := \mathtt{I} - \mathtt{J}, \\
\quad (\mathtt{K} = 1 \wedge \mathtt{R} = 0 \wedge \mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j)) \ = \quad \textbf{(by a similar argumen} \\
\\
(\mathtt{K} = 1 \wedge \mathtt{R} = j{-}i \wedge \mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j)
\end{array}$$

Hence by $\{P\}\ C\ \{Q\}\ =\ \mathrm{sp}(C, P) \Rightarrow Q$ it follows that

$$\begin{array}{l}
\{(\mathtt{I} = i \wedge \mathtt{J} = j \wedge i < j\} \\
\quad \mathtt{R} := 0; \\
\quad \mathtt{K} := 0; \\
\quad \mathtt{IF\ I} < \mathtt{J\ THEN\ K} := \mathtt{K} + 1\ \mathtt{ELSE\ SKIP}; \\
\quad \mathtt{IF\ K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN\ R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE\ R} := \mathtt{I} - \mathtt{J}) \\
\{R = j{-}i\}
\end{array}$$

Same example using wlp

$$
\begin{array}{l}
\texttt{wlp}(\texttt{R} := 0; \\
\quad \texttt{K} := 0; \\
\quad \texttt{IF I} < \texttt{J THEN K} := \texttt{K} + 1 \texttt{ ELSE SKIP}; \\
\quad \texttt{IF K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \texttt{ THEN R} := \texttt{J} - \texttt{I ELSE R} := \texttt{I} - \texttt{J}, \\
\quad (\texttt{R} = \texttt{J}{-}\texttt{I})) \; =
\end{array}
$$

$$
\begin{array}{l}
\texttt{wlp}(\texttt{R} := 0; \\
\quad \texttt{K} := 0; \\
\quad \texttt{IF I} < \texttt{J THEN K} := \texttt{K} + 1 \texttt{ ELSE SKIP}, \\
\quad (\texttt{K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \;\rightarrow\; \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J}))
\end{array}
$$

$$
\begin{array}{l}
\texttt{wlp}(\texttt{R} := 0; \\
\quad \texttt{K} := 0; \\
\quad (\texttt{I} < \texttt{J} \;\rightarrow\; (\texttt{K} + 1 = 1 \wedge \neg(\texttt{I} = \texttt{J}) \;\rightarrow\; \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J}) \\
\qquad\quad \mid \; (\texttt{K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \;\rightarrow\; \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J})) \; =
\end{array}
$$

$$
\begin{array}{l}
(\texttt{I} < \texttt{J} \;\rightarrow\; (0{+}1 = 1 \wedge \neg(\texttt{I} = \texttt{J}) \rightarrow \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J}) \\
\qquad\quad \mid \; (0 = 1 \wedge \neg(\texttt{I} = \texttt{J}) \rightarrow \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J})) \; =
\end{array}
$$

$$
(\texttt{I} < \texttt{J} \;\rightarrow\; (\neg(\texttt{I} = \texttt{J}) \rightarrow \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J}) \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J}) \; =
$$

$$
(\texttt{I} < \texttt{J} \;\rightarrow\; \texttt{J}{-}\texttt{I} = \texttt{J}{-}\texttt{I} \mid \texttt{J}{-}\texttt{I} = \texttt{I}{-}\texttt{J}) \; =
$$

$$
(\texttt{I} < \texttt{J})
$$

Example comparing sp with wlp

- **Going forward (simplifying on-the-fly)**

$$\begin{array}{l}\texttt{sp}(\texttt{R} := 0;\\ \quad \texttt{K} := 0;\\ \quad \texttt{IF } \texttt{I} < \texttt{J} \texttt{ THEN } \texttt{K} := \texttt{K} + 1 \texttt{ ELSE SKIP};\\ \quad \texttt{IF } \texttt{K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \texttt{ THEN } \texttt{R} := \texttt{J} - \texttt{I} \texttt{ ELSE } \texttt{R} := \texttt{I} - \texttt{J},\\ \quad (\texttt{I} = i \wedge \texttt{J} = j \wedge i < j)) \;=\\ (\texttt{K} = 1 \wedge \texttt{R} = j - i \wedge \texttt{I} = i \wedge \texttt{J} = j \wedge i < j)\end{array}$$

- **Going backwards**

$$\begin{array}{l}\texttt{wlp}(\texttt{R} := 0;\\ \quad \texttt{K} := 0;\\ \quad \texttt{IF } \texttt{I} < \texttt{J} \texttt{ THEN } \texttt{K} := \texttt{K} + 1 \texttt{ ELSE SKIP};\\ \quad \texttt{IF } \texttt{K} = 1 \wedge \neg(\texttt{I} = \texttt{J}) \texttt{ THEN } \texttt{R} := \texttt{J} - \texttt{I} \texttt{ ELSE } \texttt{R} := \texttt{I} - \texttt{J},\\ \quad (\texttt{R} = \texttt{J} - \texttt{I})) \;=\\ (\texttt{I} < \texttt{J} \;\rightarrow\; (0+1 = 1 \wedge \neg(\texttt{I} = \texttt{J}) \rightarrow \texttt{J} - \texttt{I} = \texttt{J} - \texttt{I} \mid \texttt{J} - \texttt{I} = \texttt{I} -\end{array}$$

Computing `sp` versus `wlp`

- Computing `sp` is like execution
  - can simplify as one goes along with the 'current state'
  - may be able to resolve branches, so can avoid executing t
  - Floyd assignment rule complicated in general
  - `sp` used for symbolically exploring 'reachable states'
    (see Specification and Verification II, especially model ch
- Computing `wlp` is like backwards proof
  - don't have 'current state', so can't simplify using it
  - can't determine conditional tests, so get big `if-then-else`
  - Hoare assignment rule simpler for arbitrary formulae
  - `wlp` used for improved verification conditions
    (see later slides)

Exercises

- **Compute**

  $\mathtt{sp}(\mathtt{R} := 0;$
  $\quad \mathtt{K} := 0;$
  $\quad \mathtt{IF\ I} < \mathtt{J\ THEN\ K} := \mathtt{K} + 1\ \mathtt{ELSE\ SKIP};$
  $\quad \mathtt{IF\ K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN\ R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE\ R} := \mathtt{I} - \mathtt{J},$
  $\quad (\mathtt{I} = i \wedge \mathtt{J} = j \wedge j \leq i))$

- **Hence show**

  $\{(\mathtt{I} = i \wedge \mathtt{J} = j \wedge j \leq i\}$
  $\quad \mathtt{R} := 0;$
  $\quad \mathtt{K} := 0;$
  $\quad \mathtt{IF\ I} < \mathtt{J\ THEN\ K} := \mathtt{K} + 1\ \mathtt{ELSE\ SKIP};$
  $\quad \mathtt{IF\ K} = 1 \wedge \neg(\mathtt{I} = \mathtt{J})\ \mathtt{THEN\ R} := \mathtt{J} - \mathtt{I}\ \mathtt{ELSE\ R} := \mathtt{I} - \mathtt{J})$
  $\{R = i - j\}$

- **Do same example use `wlp`**

Using sp to generate verification conditions

- **If C is loop-free then VC for $\{P\}\ C\ \{Q\}$ is $\mathtt{sp}(C,\ P)$**
- **Cannot in general compute a finite formula for $\mathtt{sp}($**
- **The following holds**

  $\mathtt{sp}(\mathtt{WHILE}\ S\ \mathtt{DO}\ C,\ P)\ = \mathtt{sp}(\mathtt{WHILE}\ S\ \mathtt{DO}\ C,\ \mathtt{sp}(C,\ (P \wedge S$
- **Above doesn't define $\mathtt{sp}(C, P)$ to be a finite stateme**
- **As with `wlp`, we will describe a hybrid VC and `sp` n**

sp-based verification conditions

- **Define $\mathtt{asp}(C, P)$ to be an approximate strongest pos**
- **Define $\mathtt{svc}(C, P)$ to be a set of verification condition**
- **Idea is that if $\wedge\mathtt{svc}(C, P) \Rightarrow \{P\}\ C\ \{\mathtt{asp}(C, P)\}$**
- **If C is loop-free then**
  - $\mathtt{asp}(C, P) = \mathtt{sp}(C, P)$
  - $\mathtt{svc}(C, P) = \{\}$

Definition of asp

- **Define asp recursively by:**

$$\begin{aligned}
&\texttt{asp}(P,\ \texttt{SKIP}) && = P \\
&\texttt{asp}(P,\ V\ \texttt{:=}\ E) && = \exists v.\ V = E[v/V] \wedge P[\\
&\texttt{asp}(P,\ C_1\ \texttt{;}\ C_2) && = \texttt{asp}(\texttt{asp}(P, C_1), C_2) \\
&\texttt{asp}(P,\ \texttt{IF}\ S\ \texttt{THEN}\ C_1\ \texttt{ELSE}\ C_2) && = \texttt{asp}(P \wedge S,\ C_1)\ \vee\ \texttt{asp} \\
&\texttt{asp}(P,\ \texttt{WHILE}\ S\ \texttt{DO}\ \{R\}\ C) && = R \wedge \neg S
\end{aligned}$$

Definition of svc

- **Define svc recursively by:**

$$
\begin{aligned}
&\mathtt{svc}(P,\ \mathtt{SKIP}) && = \{\} \\
&\mathtt{svc}(P,\ V\ \mathtt{:=}\ E) && = \{\} \\
&\mathtt{svc}(P,\ C_1\ \mathtt{;}\ C_2) && = \mathtt{svc}(P, C_1) \cup \mathtt{svc}(\mathtt{svc}_1(\\
&\mathtt{svc}(P,\ \mathtt{IF}\ S\ \mathtt{THEN}\ C_1\ \mathtt{ELSE}\ C_2) && = \mathtt{svc}(P \wedge S,\ C_1)\ \cup\ \mathtt{svc} \\
&\mathtt{svc}(P,\ \mathtt{WHILE}\ S\ \mathtt{DO}\ \{R\}\ C) && = \{P \Rightarrow R,\ \mathtt{asp}(R \wedge S,\ \\
& && \quad \cup\ \mathtt{svc}(R \wedge S,\ C)
\end{aligned}
$$

- **Theorem:** $\wedge\mathtt{svc}(P, C) \Rightarrow \{P\}\ C\ \{\mathtt{asp}(P, C)\}$
- **Proof by induction on C (exercise)**

Summary

- Annotate then generate VCs is the classical metho
  - practical tools: Gypsy (1970s), SPARK (current)
  - weakest preconditions are alternative explanation of VCs
  - `wlp` needs fewer annotations than VC method described e
  - `wlp` also used for refinement

- VCs and `wlp` go backwards, `sp` goes forward
  - `sp` provides verification method based on symbolic simula
  - widely used for loop-free code
  - current research potential for forwards full proof of corre
  - probably need mixture of forwards and backwards metho

Range of methods for proving $\{P\}C\{Q\}$

- Bounded model checking (BMC)
  - unwind loops a finite number of times
  - then symbolically execute
  - check states reached satisfy decidable properties
- Full proof of correctness
  - add invariants to loops
  - generate verification conditions
  - prove verification conditions with a theorem pro
- Research goal: unifying framework for a spectrum

decidable checking *proof*