FMSoft
Lecture 12 — expressivity, total correctness, decorated programs

(lecture version)

Tadeusz Litak

Jan 15, 2019

Informatik 8, FAU Erlangen-Nürnberg

expressivity

Define

$$wlp : \mathsf{Com} \to \mathsf{AssertHo} \to \mathsf{AssertHo} \qquad \text{by}$$

$$wlp(\mathtt{SKIP}, B) := B$$
$$wlp(\mathtt{X} \ \mathtt{:=} \ a, B) := B[a/\mathtt{X}]$$
$$wlp(c_1 \,\mathtt{;}\, c_2, B) := wlp(c_1, wlp(c_2, B))$$
$$wlp(\mathtt{IF}\ b\ \mathtt{THEN}\ c_1\ \mathtt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$
$$wlp(\mathtt{WHILE}\ b\ \mathtt{DO}\ c\ \mathtt{END}, B) := \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c}$$

where

$$A_0^{b,B,c} = \top$$
$$A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

Theorem
IMP *is expressive, with wlp being the witnessing function*

We have to show that for any c that

(*) $\vdash \{wlp(c, B)\}\, c\{B\}$

(!) ...and that for any I,
$[\![wlp(c, B)]\!]_I = \{\sigma \in \mathsf{States}_{\mathsf{IMP}} \mid [\![c]\!]\sigma \vDash^I B\}$

Note that one half of (!) we could get via (*) and the Soundness Theorem.

That is, if we show (*), then soundness yields for any I,

$[\![wlp(c, B)]\!]_I \subseteq \{\sigma \in \mathsf{States}_{\mathsf{IMP}} \mid [\![c]\!]\sigma \vDash^I B\}$

Proof.

The proof is by induction on c. We need to prove both (*) and (!) clauses together for each construct, as in some inductive steps we want to use both of them at the same time.

We do not present all steps in the logically right order. For convenience group them as follows:

1. present non-`WHILE` (*) clauses
2. present non-`WHILE` (!) clauses

   assuming previous inductive steps of (*) and (!) have been established
3. present (*) and (!) for `WHILE`

   assuming previous inductive steps of (*) and (!) have been established

The (*) claim is straightforward for almost every program construct apart from `WHILE` (apart from a trivial boolean transformation for `IF`).

The situation with non-WHILE clauses of the missing half of (!) is similar, but let us see it in more detail: for any σ,

- $[\![$ SKIP $]\!]\sigma$ $(= \sigma) \models^I B$ implies

  (in fact, is equivalent to)

  $\sigma \models^I B = wlp($ SKIP $, B)$
- $[\![$ X := $a]\!]\sigma$ $(= \sigma[[\![a]\!]\sigma/$ X $]) \models^I B$ is ...

  now we use an equivalence we proved before (when?) in the opposite direction

  ...equivalent to $\sigma \models^I B[a/\mathsf{X}] = wlp($ X := $a, B)$
- $[\![c_1$; $c_2]\!]\sigma$ $(= [\![c_2]\!]([\![c_1]\!]\sigma)) \models^I B$ by IH implies that $[\![c_1]\!]\sigma \models^I wlp(c_2, B)$ and using IH again gets us home
- $[\![$ IF b THEN c_1 ELSE $c_2]\!]\sigma \models^I B$ left as exercise (trivial splitting of cases)

- And thus, again, we're left with `WHILE`

- And thus, again, we're left with WHILE
- First, let us show the (*) claim, i.e., that

  $\vdash \{ \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c} \}$ WHILE b DO c END$\{B\}$

  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- And thus, again, we're left with WHILE
- First, let us show the (*) claim, i.e., that

$$\vdash \{ \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c} \} \texttt{WHILE } b \texttt{ DO } c \texttt{ END} \{ B \}$$

where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- Here's the derivation:

where

$wh = \texttt{WHILE } b \texttt{ DO } c \texttt{ END}$

$\bigwedge A = \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c}$

- And thus, again, we're left with WHILE
- First, let us show the (*) claim, i.e., that

$$\vdash \{\textstyle\bigwedge_{n\in\mathbb{N}} A_n^{b,B,c}\}\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\{B\}$$

where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- Here's the derivation:

$$\dfrac{\vdash \{wlp(c, \textstyle\bigwedge A)\}\ c\ \{\textstyle\bigwedge A\}}{\text{(IH)}}$$

where

$wh = \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}$

$\bigwedge A = \bigwedge_{n\in\mathbb{N}} A_n^{b,B,c}$

- And thus, again, we're left with WHILE
- First, let us show the (*) claim, i.e., that

$$\vdash \{ \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c} \} \texttt{WHILE } b \texttt{ DO } c \texttt{ END} \{ B \}$$

where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- Here's the derivation:

$$\frac{\vdash \{ wlp(c, \bigwedge A) \}\, c \,\{ \bigwedge A \} \qquad \models \bigwedge A \wedge b \rightarrow wlp(c, \bigwedge A)}{\vdash \{ \bigwedge A \wedge b \}\, c \,\{ \bigwedge A \}}$$

(IH) (?)

where

$wh =$ WHILE b DO c END

$\bigwedge A = \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c}$

- And thus, again, we're left with WHILE
- First, let us show the (*) claim, i.e., that

$$\vdash \{ \bigwedge_{n\in\mathbb{N}} A_n^{b,B,c} \}\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\{B\}$$

where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- Here's the derivation:

$$\dfrac{\dfrac{\dfrac{\vdash \{wlp(c, \bigwedge A)\}\, c\, \{\bigwedge A\} \quad \text{(IH)} \qquad \models \bigwedge A \wedge b \to wlp(c, \bigwedge A) \quad \text{(?)}}{\vdash \{\bigwedge A \wedge b\}\, c\, \{\bigwedge A\}}}{\vdash \{\bigwedge A\}\, wh\, \{\bigwedge A \wedge \neg b\}} \qquad \models \bigwedge A \wedge \neg b \to B \quad \text{(??)}}{\vdash \{\bigwedge A\}\, wh\, \{B\}}$$

where

$wh =$ WHILE b DO c END

$\bigwedge A = \bigwedge_{n\in\mathbb{N}} A_n^{b,B,c}$

(??) is immediate, distributing $\wedge$ first over $\bigwedge$ and then over $\vee$

- We now want to show (?), i.e., that
  $\vDash \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$

  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- We now want to show (?), i.e., that
  $\vDash \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$
- Using laws for finite/infinite conjunctions and disjunctions, this can be reduced to the question whether

$$\vDash \bigwedge wlp(c, A_n^{b,B,c}) \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$$

- We now want to show (?), i.e., that
  $\models \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$
- Using laws for finite/infinite conjunctions and disjunctions, this can be reduced to the question whether

$$\models \bigwedge wlp(c, A_n^{b,B,c}) \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$$

- In fact, the antecedent and the consequent are equivalent:

- We now want to show (?), i.e., that
  $\models \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$
- Using laws for finite/infinite conjunctions and disjunctions, this can be reduced to the question whether

$$\models \bigwedge wlp(c, A_n^{b,B,c}) \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$$

- In fact, the antecedent and the consequent are equivalent:

$$\sigma \models^I \bigwedge wlp(c, A_n^{b,B,c}) \Leftrightarrow \forall n, \sigma \models^I wlp(c, A_n^{b,B,c})$$

- We now want to show (?), i.e., that
  $\vDash \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$
- Using laws for finite/infinite conjunctions and disjunctions, this can be reduced to the question whether

$$\vDash \bigwedge wlp(c, A_n^{b,B,c}) \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$$

- In fact, the antecedent and the consequent are equivalent:

$$\begin{aligned} \sigma \vDash^I \bigwedge wlp(c, A_n^{b,B,c}) &\Leftrightarrow \forall n, \sigma \vDash^I wlp(c, A_n^{b,B,c}) \\ &\Leftrightarrow_{(IH)} \forall n, [\![c]\!]\sigma \vDash^I A_n^{b,B,c} \end{aligned}$$

- We now want to show (?), i.e., that
  $\vDash \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$
- Using laws for finite/infinite conjunctions and disjunctions, this can be reduced to the question whether

$$\vDash \bigwedge wlp(c, A_n^{b,B,c}) \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$$

- In fact, the antecedent and the consequent are equivalent:

$$\begin{aligned} \sigma \vDash^I \bigwedge wlp(c, A_n^{b,B,c}) &\Leftrightarrow \forall n, \sigma \vDash^I wlp(c, A_n^{b,B,c}) \\ &\Leftrightarrow_{(IH)} \forall n, [\![c]\!]\sigma \vDash^I A_n^{b,B,c} \\ &\Leftrightarrow [\![c]\!]\sigma \vDash^I \bigwedge A_n^{b,B,c} \end{aligned}$$

- We now want to show (?), i.e., that
  $\vDash \bigwedge A_n^{b,B,c} \wedge b \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$
- Using laws for finite/infinite conjunctions and disjunctions, this can be reduced to the question whether

$$\vDash \bigwedge wlp(c, A_n^{b,B,c}) \rightarrow wlp(c, \bigwedge A_n^{b,B,c})$$

- In fact, the antecedent and the consequent are equivalent:

$$\begin{aligned}
\sigma \vDash^I \bigwedge wlp(c, A_n^{b,B,c}) &\Leftrightarrow \forall n, \sigma \vDash^I wlp(c, A_n^{b,B,c}) \\
&\Leftrightarrow_{(IH)} \forall n, [\![c]\!]\sigma \vDash^I A_n^{b,B,c} \\
&\Leftrightarrow [\![c]\!]\sigma \vDash^I \bigwedge A_n^{b,B,c} \\
&\Leftrightarrow_{(IH)} \sigma \vDash^I wlp(c, \bigwedge A_n^{b,B,c})
\end{aligned}$$

- We're nearly home. All that's left is to show (!) for WHILE:

$$\text{for any } I, [\![\texttt{WHILE } b \texttt{ DO } c \texttt{ END }]\!]\sigma \models^I B \text{ implies}$$
$$\sigma \models^I \bigwedge_{n\in\mathbb{N}} A_n^{b,B,c}$$

where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- We're nearly home. All that's left is to show (!) for WHILE:

$$\text{for any } I, \llbracket \texttt{WHILE } b \texttt{ DO } c \texttt{ END} \rrbracket\sigma \models^I B \text{ implies}$$
$$\sigma \models^I \bigwedge_{n\in\mathbb{N}} A_n^{b,B,c}$$

where, recall, $A_0^{b,B,c} = \top$ and $A_{n+1}^{b,B,c} = (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$

- Recall also that $\llbracket \texttt{WHILE } b \texttt{ DO } c \texttt{ END} \rrbracket := \bigcup_{n\in\omega} \Phi_{b,c}^n(\varnothing)$

where $\Phi_{b,c}(f)(\sigma) := \begin{cases} \sigma & \text{if } \llbracket b \rrbracket\sigma = \bot \\ f(\llbracket c \rrbracket\sigma) & \text{if } \llbracket b \rrbracket\sigma = \top \end{cases}$

- Recall again

- Recall again

Lemma (WHILE helper)
for any n', σ, b, c *if* $\Phi_{b,c}^{n'+1}(\varnothing)(\sigma)$ *is defined, then there exists* $n \leq n'$ *s.t.*

- Recall again

Lemma (WHILE helper)
for any n', σ, b, c *if* $\Phi_{b,c}^{n'+1}(\varnothing)(\sigma)$ *is defined, then there exists* $n \leq n'$ *s.t.*

(a) $[\![c]\!]^n\sigma = [\![$ `WHILE` b `DO` c `END` $]\!]\sigma$ *and*

- Recall again

Lemma (WHILE helper)
for any n', σ, b, c *if* $\Phi_{b,c}^{n'+1}(\varnothing)(\sigma)$ *is defined, then there exists* $n \leq n'$ *s.t.*

(a) $[\![c]\!]^n\sigma = [\![$ `WHILE` b `DO` c `END` $]\!]\sigma$ *and*
(b) $[\![c]\!]^n\sigma \vDash^I \neg b$ *and*

- Recall again

Lemma (WHILE helper)
for any n', σ, b, c *if* $\Phi_{b,c}^{n'+1}(\varnothing)(\sigma)$ *is defined, then there exists* $n \leq n'$ *s.t.*

(a) $[\![c]\!]^n\sigma = [\![$ `WHILE` b `DO` c `END` $]\!]\sigma$ *and*
(b) $[\![c]\!]^n\sigma \vDash^I \neg b$ *and*
(c) *for all* $k < n$, $[\![c]\!]^k\sigma \vDash^I b$

- Recall again

Lemma (WHILE helper)
for any n', σ, b, c *if* $\Phi_{b,c}^{n'+1}(\varnothing)(\sigma)$ *is defined, then there exists* $n \leq n'$ *s.t.*

(a) $[\![c]\!]^n\sigma = [\![$ `WHILE` b `DO` c `END` $]\!]\sigma$ *and*
(b) $[\![c]\!]^n\sigma \vDash^I \neg b$ *and*
(c) *for all* $k < n$, $[\![c]\!]^k\sigma \vDash^I b$

- Using clauses (b) and (c), we can inductively prove that for every i,
  (!!) for every b, c, n, $\Phi_{b,c}^n(\varnothing)(\sigma) \vDash^I B$ implies $\sigma \vDash^I A_i^{b,B,c}$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{i+1}^{b,B,c} = (b \wedge wlp(c, A_i^{b,B,c})) \vee (\neg b \wedge B)$
  Btw, can you see what $wlp(c, \top)$ is?

- Recall again

Lemma (WHILE helper)
for any n', σ, b, c *if* $\Phi_{b,c}^{n'+1}(\varnothing)(\sigma)$ *is defined, then there exists* $n \leq n'$ *s.t.*

(a) $[\![c]\!]^n\sigma = [\![$ `WHILE` b `DO` c `END` $]\!]\sigma$ *and*
(b) $[\![c]\!]^n\sigma \vDash^I \neg b$ *and*
(c) *for all* $k < n$, $[\![c]\!]^k\sigma \vDash^I b$

- Using clauses (b) and (c), we can inductively prove that for every i,
  (!!) for every b, c, n, $\Phi_{b,c}^n(\varnothing)(\sigma) \vDash^I B$ implies $\sigma \vDash^I A_i^{b,B,c}$
  where, recall, $A_0^{b,B,c} = \top$ and $A_{i+1}^{b,B,c} = (b \wedge wlp(c, A_i^{b,B,c})) \vee (\neg b \wedge B)$
  Btw, can you see what $wlp(c, \top)$ is?
- Then using (a) and (!!), we get (!)

total correctness

- We did introduce the PCA $\perp\!\!\!\perp$-convention:

$$\text{for any } B \text{ and } I,\ \perp\!\!\!\perp \models^{I} B$$

- We did introduce the PCA $\perp\!\!\!\perp$-convention:

$$\text{for any } B \text{ and } I,\ \perp\!\!\!\perp \models^{I} B$$

- We noted also have the opposite total $\perp\!\!\!\perp$-convention:

$$\text{for any } B \text{ and } I,\ \perp\!\!\!\perp \not\models^{I}_{\mathsf{t}} B$$

- We did introduce the PCA $\lightning$-convention:

$$\text{for any } B \text{ and } I,\ \lightning \models^I B$$

- We noted also have the opposite total $\lightning$-convention:

$$\text{for any } B \text{ and } I,\ \lightning \not\models_{\mathsf{t}}^I B$$

- Let us now denote (PCA)-validity of Hoare triples under the total $\lightning$-convention:

$$\begin{array}{rl} \sigma \models^I \langle A \rangle c \langle B \rangle & \sigma \models^I A \text{ implies } [\![c]\!]\sigma \models_{\mathsf{t}}^I B \\ \sigma \models \langle A \rangle c \langle B \rangle & \text{for any } I,\ \sigma \models^I \langle A \rangle c \langle B \rangle \\ \models \langle A \rangle c \langle B \rangle & \text{for any } \sigma,\ \sigma \models \langle A \rangle c \langle B \rangle \end{array}$$

Another common notation: $[A]c[B]$

The one used here comes from *Effective axiomatizations of Hoare logic*, by Clarke, German, Halpern, *J. ACM* 1983

- We did introduce the PCA $\lightning$-convention:

$$\text{for any } B \text{ and } I,\ \lightning \models^I B$$

- We noted also have the opposite total $\lightning$-convention:

$$\text{for any } B \text{ and } I,\ \lightning \not\models_{\mathsf{t}}^I B$$

- Let us now denote (PCA)-validity of Hoare triples under the total $\lightning$-convention:

$$\begin{aligned} \sigma \models^I \langle\!| A |\!\rangle\, c\, \langle\!| B |\!\rangle &\quad \sigma \models^I A \text{ implies } [\![c]\!]\sigma \models_{\mathsf{t}}^I B \\ \sigma \models \langle\!| A |\!\rangle\, c\, \langle\!| B |\!\rangle &\quad \text{for any } I,\ \sigma \models^I \langle\!| A |\!\rangle\, c\, \langle\!| B |\!\rangle \\ \models \langle\!| A |\!\rangle\, c\, \langle\!| B |\!\rangle &\quad \text{for any } \sigma,\ \sigma \models \langle\!| A |\!\rangle\, c\, \langle\!| B |\!\rangle \end{aligned}$$

  Another common notation: $[\![A]\!]\, c\, [\![B]\!]$

  The one used here comes from *Effective axiomatizations of Hoare logic*, by Clarke, German, Halpern, *J. ACM* 1983

- Which rules in our proof system may require adjustment to ensure soundness?

...of course, the one for `WHILE` :

...of course, the one for WHILE :

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{==}i \rangle\ c\ \langle a_m{<}i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

...of course, the one for WHILE :

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{=}{=}i \rangle\ c\ \langle a_m{<}i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

a_m is called the termination metric or variant

...of course, the one for WHILE :

$$\frac{\vdash_{\mathsf{t}} \{\!|A \wedge b|\!\}\ c\ \{\!|A|\!\} \qquad \vdash_{\mathsf{t}} \{\!|A \wedge b \wedge a_m{=}{=}i|\!\}\ c\ \{\!|a_m{<}i|\!\} \qquad \vDash A \to a_m \geq 0}{\vdash_{\mathsf{t}} \{\!|A|\!\}\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \{\!|A \wedge \neg b|\!\}}$$

a_m is called the termination metric or variant

Soundness proof for our rule may require more fine-grained semantics

cf., e.g., Apt, Boer, Olderog, *Verification of sequential programs*

An example: Euclid's algorithm

First, recall /define ...

$$\texttt{j|k} := \exists i.\ \texttt{i} \cdot \texttt{j} == \texttt{k}$$

$$\texttt{k == gcd(i,j)} := \texttt{k|i} \wedge \texttt{k|j} \wedge \forall k'.\ (\texttt{k'|i} \wedge \texttt{k'|j} \to \texttt{k' <= k})$$

Recall ...

```
⟨ X == i ∧ Y == j ∧ i > 0 ∧ j > 0 ⟩
WHILE (not X == Y) DO
  IF (X <= Y) THEN Y := Y - X ELSE X := X - Y
     ENDIF
END
⟨ Y == gcd(i,j) ⟩
```

```
{ X == i ∧ Y == j ∧ i > 0 ∧ j > 0 }
WHILE (not X == Y) DO
  IF (X <= Y) THEN Y := Y - X ELSE X := X - Y
     ENDIF
END
{ Y == gcd(i,j) }
```

Recall ...

$$\frac{\vdash_t \{A \wedge b\}\, c\, \{A\} \qquad \vdash_t \{A \wedge b \wedge a_m{==}i\}\, c\, \{a_m{<}i\} \qquad \vDash A \to a_m \geq 0}{\vdash_t \{A\}\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \{A \wedge \neg b\}}$$

Let's fill in the blanks ...

```
⟨ X == i ∧ Y == j ∧ i > 0 ∧ j > 0 ⟩
WHILE (not X == Y) DO
  IF (X <= Y) THEN Y := Y - X ELSE X := X - Y
     ENDIF
END
⟨ Y == gcd(i,j) ⟩
```

Recall ...

$$\frac{\vdash_t \langle A \wedge b \rangle\, c\, \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{==}i \rangle\, c\, \langle a_m{<}i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

Let's fill in the blanks ...

$A :=$ `gcd(i,j) == gcd(X,Y) ∧ X >= 0 ∧ Y >= 0`

$a_m :=$ `X + Y`

Remaining details left as exercise

An aside

Note that one can easily write an unsound version of `WHILE` rule

An aside

Note that one can easily write an unsound version of WHILE rule

Compare

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{=}{=}i \rangle\ c\ \langle a_m{<}i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

with

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{=}{=}a_i \rangle\ c\ \langle a_m{<}a_i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

An aside

Note that one can easily write an unsound version of WHILE rule

Compare

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{==}i \rangle\ c\ \langle a_m{<}i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

with

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{==}a_i \rangle\ c\ \langle a_m{<}a_i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

Is the second one sound?

An aside

Note that one can easily write an unsound version of WHILE rule

Compare

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{==}i \rangle\ c\ \langle a_m{<}i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

with

$$\frac{\vdash_t \langle A \wedge b \rangle\ c\ \langle A \rangle \qquad \vdash_t \langle A \wedge b \wedge a_m{==}a_i \rangle\ c\ \langle a_m{<}a_i \rangle \qquad \vDash A \to a_m \geq 0}{\vdash_t \langle A \rangle\ \texttt{WHILE}\ b\ \texttt{DO}\ c\ \texttt{END}\ \langle A \wedge \neg b \rangle}$$

Is the second one sound?

Can we fix it using the *modifies* predicate from separation logic lectures?

Dijkstra's weakest preconditions

Clearly, weakest predictions (and predicate transformers) more suited to total correctness than Hoare rules

Clearly, weakest predictions (and predicate transformers) more suited to total correctness than Hoare rules

Recall weakest liberal preconditions:

$$wlp : \mathsf{Com} \to \mathsf{AssertHo} \to \mathsf{AssertHo} \qquad \text{by}$$

$$wlp(\mathtt{SKIP}, B) := B$$
$$wlp(\mathtt{X := a}, B) := B[\mathtt{X}/a]$$
$$wlp(c_1 \mathtt{;} c_2, B) := wlp(c_1, wlp(c_2, B))$$
$$wlp(\mathtt{IF}\ b\ \mathtt{THEN}\ c_1\ \mathtt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$
$$wlp(\mathtt{WHILE}\ b\ \mathtt{DO}\ c\ \mathtt{END}, B) := \bigwedge_{n \in \mathbb{N}} A_n^{b,B,c}$$

where

$$A_0^{b,B,c} := \top$$
$$A_{n+1}^{b,B,c} := (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

Now we can replace them with Dijkstra's original idea: his weakest preconditions were supposed to be total. Again, the only change is in the WHILE loops:

$$wp : \mathsf{Com} \to \mathsf{AssertHo} \to \mathsf{AssertHo}$$

$$wp(\mathtt{SKIP}, B) := B$$
$$wp(\mathtt{X} \ \mathtt{:=} \ a, B) := B[\mathtt{X}/a]$$
$$wp(c_1 \,\mathtt{;}\, c_2, B) := wp(c_1, wp(c_2, B))$$
$$wp(\mathtt{IF} \ b \ \mathtt{THEN} \ c_1 \ \mathtt{ELSE} \ c_2, B) :=$$
$$(b \wedge wp(c_1, B)) \vee (\neg b \wedge wp(c_2, B))$$
$$wp(\mathtt{WHILE} \ b \ \mathtt{DO} \ c \ \mathtt{END}, B) := \bigvee_{n \in \mathbb{N}} A'^{b,B,c}_n$$

where

$$A'^{b,B,c}_0 := \neg b \wedge B$$
$$A'^{b,B,c}_{n+1} := (b \wedge wp(c, A'^{b,B,c}_n))$$

Limits of relative completeness results

- We did not not prove here relative completeness for total correctness

  Doable, but might require different semantics

Limits of relative completeness results

- We did not not prove here relative completeness for total correctness

  Doable, but might require different semantics
- Furthermore, we have the often-mentioned fact that we're actually including oracle for first-order arithmetic

Limits of relative completeness results

- We did not not prove here relative completeness for total correctness

  Doable, but might require different semantics
- Furthermore, we have the often-mentioned fact that we're actually including oracle for first-order arithmetic
- Surprisingly, even with these restrictions (relative) completeness is not always available

  R. J. Lipton, *A necessary and sufficient condition for the existence of Hoare logics*, In Proc. 18th IEEE Symp. on Foundations of Comp. Sc. (1977)

  E. M. Clarke, S. M. German, J. Halpern *Effective axiomatizations of Hoare logics*, J. ACM (1983)

  E. M. Clarke, *The characterization problem for Hoare logics*, Phil. Trans. R. Sac. Land. A 312, 423-440 (1984)

decorated programs

Annotated (or decorated) programs DCom

No need to insist on annotating each command with both pre- and postcondition.

Annotated (or decorated) programs DCom

No need to insist on annotating each command with both pre- and postcondition. Instead, define:

$c_1, c_2 ::=$ SKIP $\{B\}$ | X := $a\{B\}$ | $c1$; $c2$ |
IF b THEN $\{A\}\, c_1$ ELSE $\{A'\}\, c_2$ ENDIF $\{B\}$ |
WHILE b DO $\{A\}\, c_1$ END $\{B\}$

An annotated/decorated PCA:

$d ::= \{A\}\, c\{B\}$, where $c\{B\} \in$ DCom (prove by induction: each element of DCom comes with a postcondition at the end!)

Convention: $\{A\}\{A'\}$ written as $\{A\} \to \{A'\}$

- We have the obvious function $erase : \mathsf{DCom} \to \mathsf{DCom}$

- We have the obvious function $erase : \mathsf{DCom} \to \mathsf{DCom}$
- For a decorated PCA $\{\!\{A\}\!\}\, c\{\!\{B\}\!\}$, we would like ensure that $c\{\!\{B\}\!\}$ is indeed correctly annotated

- We have the obvious function $erase : \mathsf{DCom} \to \mathsf{DCom}$
- For a decorated PCA $\{A\}\, c\{B\}$, we would like ensure that $c\{B\}$ is indeed correctly annotated
- In particular, we want to have that
  $\vdash \{A\}\, erase(c\{B\})\{B\}$

  (and that the same holds for each component of c, e.g., that candidates for loop invariants are indeed loop invariants …)

- We have the obvious function $erase : \mathsf{DCom} \to \mathsf{DCom}$
- For a decorated PCA $\{\!\{A\}\!\}\, c\, \{\!\{B\}\!\}$, we would like ensure that $c\{\!\{B\}\!\}$ is indeed correctly annotated
- In particular, we want to have that
  $\vdash \{\!\{A\}\!\}\, erase(c\{\!\{B\}\!\})\{\!\{B\}\!\}$

  (and that the same holds for each component of c, e.g., that candidates for loop invariants are indeed loop invariants …)
- We will do that by defining a function

$$vc : \mathsf{AssertHo} \times \mathsf{DCom} \to \mathsf{AssertHo}$$

s.t. $\vDash vc(A, c\{\!\{B\}\!\})$ implies $\vdash \{\!\{A\}\!\}\, erase(c\{\!\{B\}\!\})\{\!\{B\}\!\}$

- We have the obvious function $erase : \mathsf{DCom} \to \mathsf{DCom}$
- For a decorated PCA $\{A\}\, c\{B\}$, we would like ensure that $c\{B\}$ is indeed correctly annotated
- In particular, we want to have that
  $\vdash \{A\}\, erase(c\{B\})\{B\}$

  (and that the same holds for each component of c, e.g., that candidates for loop invariants are indeed loop invariants ...)
- We will do that by defining a function

$$vc : \mathsf{AssertHo} \times \mathsf{DCom} \to \mathsf{AssertHo}$$

  s.t. $\vDash vc(A, c\{B\})$ implies $\vdash \{A\}\, erase(c\{B\})\{B\}$
- Furthermore, the separation logic assertion obtained this way, while complex, can be broken down into pieces which tend to be manageable for automated tools ...

Verification conditions

$$vc(A,\ \texttt{SKIP}\ \{B\}) = A \to B$$

$$vc(A,\ \texttt{X}\ \texttt{:=}\ a\{B\}) = A \to B[a/\texttt{X}]$$

$$vc(A, c_1\{B\}\ \texttt{;}\ c_2) = vc(A, c_1\{B\}) \wedge vc(B, c_2)$$

$$\begin{aligned} vc(A,\ \texttt{IF}\ b\ \texttt{THEN}\ \{A_1\}\, c_1\{A_1'\} \\ \texttt{ELSE}\ \{A_2\}\, c_2\{A_2'\}\ \texttt{ENDIF}\ \{B\}) = \\ (A \wedge b \to A_1) \wedge (A \wedge \neg b \to A_2) \wedge \\ (A_1' \leftrightarrow B) \wedge (A_2' \leftrightarrow B) \wedge \\ vc(A_1, c_1\{A_1'\}) \wedge vc(A_2, c_2\{A_2'\}) \end{aligned}$$

$$\begin{aligned} vc(A,\ \texttt{WHILE}\ b\ \texttt{DO}\ \{A_1\}\, c_1\{A_1'\}\ \texttt{END}\ \{B\}) = \\ (A \to A_1') \wedge \\ (A_1 \leftrightarrow A_1' \wedge b) \wedge \\ (B \leftrightarrow A_1' \wedge \neg b) \wedge \\ vc(A_1, c_1\{A_1'\}) \end{aligned}$$

Some of these equivalences can be weakened, some not. Which ones?

- We can now prove the desired result:

- We can now prove the desired result:

Theorem
$\vDash vc(A, c\{B\})$ *implies* $\vdash \{A\}\, erase(c\{B\})\{B\}$.

- We can now prove the desired result:

Theorem
$\vDash vc(A, c\{B\})$ *implies* $\vdash \{A\}\, erase(c\{B\})\{B\}$.

Proof sketch.
Straightforward (if rather tedious) induction on c, using the definition of vc, (global backward) rules for program constructs and the consequence rule for preconditions. □

- We can now prove the desired result:

Theorem
$\models vc(A, c\{B\})$ *implies* $\vdash \{A\}\, erase(c\{B\})\{B\}$.

Proof sketch.
Straightforward (if rather tedious) induction on c, using the definition of vc, (global backward) rules for program constructs and the consequence rule for preconditions. □

- We just need to ensure that the tool has as many tactics/heuristics for verification conditions as possible ...

- We can now prove the desired result:

Theorem
$\vDash vc(A, c\{B\})$ *implies* $\vdash \{A\}\, erase(c\{B\})\{B\}$.

Proof sketch.
Straightforward (if rather tedious) induction on c, using the definition of vc, (global backward) rules for program constructs and the consequence rule for preconditions. □

- We just need to ensure that the tool has as many tactics/heuristics for verification conditions as possible ...
- ...and/or that our annotations are helpful enough

- We can now prove the desired result:

Theorem
$\vDash vc(A, c\{B\})$ *implies* $\vdash \{A\}\, erase(c\{B\})\{B\}$.

Proof sketch.
Straightforward (if rather tedious) induction on c, using the definition of vc, (global backward) rules for program constructs and the consequence rule for preconditions. □

- We just need to ensure that the tool has as many tactics/heuristics for verification conditions as possible ...
- ...and/or that our annotations are helpful enough
- Aside question: is the reverse implication true?

{X == i and i >= 0}

```
Y := 1;
WHILE not (X == 0) DO
     Y := Y * X;
     X := X - 1
END
```

{Y == i!}

Recall that (t_1!) == t_2 encoded in AssertHo as:

$\bigvee_{n \in \mathbb{N}}$ ((n == t_1) and (1 * 2 * ... * n == t_2))

$\{$X == i and i >= 0$\}$ $\rightarrow$
$\{$X == i and i >= 0 and 1 == 1$\}$

```
Y := 1;
```

$\{$X == i and i >= 0 and Y == 1$\}$

```
WHILE not (X == 0) DO
    Y := Y * X;
    X := X - 1
END
```

$\{$Y == $i!\}$

Recall that $(t_1!)$ == t_2 encoded in AssertHo as:

$\bigvee_{n \in \mathbb{N}}$((n == t_1) and (1 * 2 * ... * n == t_2))

$\{$X == i and i >= 0$\}\rightarrow$
$\{$X == i and i >= 0 and 1 == 1$\}$

```
Y := 1;
```
$\{$X == i and i >= 0 and Y == 1$\}\rightarrow$
$\{$Y * X! == i! and X >= 0 $\}$ $\Leftarrow$only here some creativity needed
```
WHILE not (X == 0) DO
    Y := Y * X;
    X := X - 1
END
```

$\{$Y == i!$\}$

Recall that (t_1!) == t_2 encoded in AssertHo as:

$\bigvee_{n\in\mathbb{N}}$((n == t_1) and (1 * 2 * ... * n == t_2))

```
{X == i and i >= 0} →
{X == i and i >= 0 and 1 == 1}

Y := 1;
{X == i and i >= 0 and Y == 1} →
{Y * X! == i! and X >= 0 }
WHILE not (X == 0) DO
      {Y * X! == i! and X >= 0 and not (X == 0) } →
      { Y * X * (X-1)! == i! and X > 0 }
          ⇑ derived from the encoding of (t1!) == t2 below
    Y := Y * X;
    X := X - 1
END
```

```
{Y == i!}
```

Recall that $(t_1!)$ == t_2 encoded in AssertHo as:

$$\bigvee_{n \in \mathbb{N}} (\texttt{(n == } t_1\texttt{) and (1 * 2 * ... * n == } t_2\texttt{)})$$

```
{X == i and i >= 0} →
{X == i and i >= 0 and 1 == 1}

Y := 1;
{X == i and i >= 0 and Y == 1} →
{Y * X! == i! and X >= 0 }
WHILE not (X == 0) DO
      {Y * X! == i! and X >= 0 and not (X == 0) } →
      {Y * X * (X-1)! == i! and X > 0 }
    Y := Y * X;
      {Y * (X-1)! == i! and X > 0 }
    X := X - 1
END
```

```
{Y == i!}
```

Recall that $(t_1!)$ == t_2 encoded in AssertHo as:

$\bigvee_{n \in \mathbb{N}}$ ((n == t_1) and (1 * 2 * ... * n == t_2))

```
{X == i and i >= 0} →
{X == i and i >= 0 and 1 == 1}

Y := 1;
{X == i and i >= 0 and Y == 1} →
{Y * X! == i! and X >= 0 }
WHILE not (X == 0) DO
      {Y * X! == i! and X >= 0 and not (X == 0) } →
      {Y * X * (X-1)! == i! and X > 0 }
    Y := Y * X;
      {Y * (X-1)! == i! and X > 0 } →
      {Y * (X − 1)! == i! and X − 1 >= 0 } →
    X := X - 1
      {Y * X! == i! and X >=0 }
END


{Y == i!}
```

Recall that $(t_1!)$ == t_2 encoded in AssertHo as:

$$\bigvee_{n \in \mathbb{N}} (\ (\texttt{n} == t_1)\ \texttt{and}\ (1 * 2 * \ldots * \texttt{n} == t_2))$$

```
{X == i and i >= 0} →
{X == i and i >= 0 and 1 == 1}

Y := 1;
{X == i and i >= 0 and Y == 1} →
{Y * X! == i! and X >= 0 }
WHILE not (X == 0) DO
      {Y * X! == i! and X >= 0 and not (X == 0) } →
      {Y * X * (X-1)! == i! and X > 0 }
    Y := Y * X;
      {Y * (X-1)! == i! and X > 0 } →
      {Y * (X-1)! == i! and X-1 >= 0 } →
    X := X - 1
      {Y * X! == i! and X >=0 }
END
{Y * X! == i! and X >=0 and X == 0 } →

{Y == i!}
```

Recall that $(t_1!)$ == t_2 encoded in AssertHo as:

$$\bigvee_{n \in \mathbb{N}} (\ (\texttt{n} == t_1)\ \texttt{and}\ (1 * 2 * \ldots * \texttt{n} == t_2))$$

More examples and further discussion when we come back to separation logic