Lecture 24 - Hoare logic, more logic

Vaishnavi Sundararajan

COL703 - Logic for Computer Science

Recap

- Wanted to verify that imperative programs operate as expected
- Programs as state transformers – function mapping inputs to outputs
- Try to obtain this function and check if it satisfies required guarantees
- Use Hoare logic for this
- Reason about assertions that hold before and after a program
- Hoare triples: $\{\alpha\}\, c\, \{\beta\}$
- c is the command, α is the precondition (should hold of the state before the command is run), β is the postcondition (should hold of the state after the command is run)

Recap: Big-step semantics for commands

$$s \text{—}[\mathit{skip}]\text{→} s$$

$$\frac{[\![e]\!]s = n}{s \text{—}[X = e]\text{→} s[X \mapsto n]}$$

$$\frac{s \text{—}[c_1]\text{→} s_1 \quad s_1 \text{—}[c_2]\text{→} s_2}{s \text{—}[c_1; c_2]\text{→} s_2}$$

$$\frac{s \vDash b \quad s \text{—}[c_1]\text{→} s'}{s \text{—}[\mathit{if}\ b\ \mathit{then\ do}\ c_1\ \mathit{else}\ c_2\ \mathit{end}]\text{→} s'}$$

$$\frac{s \nvDash b \quad s \text{—}[c_2]\text{→} s'}{s \text{—}[\mathit{if}\ b\ \mathit{then\ do}\ c_1\ \mathit{else}\ c_2\ \mathit{end}]\text{→} s'}$$

$$\frac{s \nvDash b}{s \text{—}[\mathit{while}\ b\ \mathit{do}\ c\ \mathit{end}]\text{→} s}$$

$$\frac{s \vDash b \quad s \text{—}[c]\text{→} s_1 \quad s_1 \text{—}[\mathit{while}\ b\ \mathit{do}\ c\ \mathit{end}]\text{→} s_2}{s \text{—}[\mathit{while}\ b\ \mathit{do}\ c\ \mathit{end}]\text{→} s_2}$$

where

$$(s[X \mapsto n])(Y) = \begin{cases} n & \text{if } Y = X \\ s(Y) & \text{otherwise} \end{cases}$$

Recap: Hoare logic rules

$$\frac{}{\{\alpha\}\ \mathit{skip}\ \{\alpha\}}\ \textbf{Skip} \qquad \frac{}{\{\alpha(e)\}\ X = e\ \{\alpha(X)\}}\ \textbf{Assign}$$

$$\frac{\{\alpha\}\ c\ \{\beta\} \quad \{\beta\}\ c'\ \{\varphi\}}{\{\alpha\}\ c; c'\ \{\varphi\}}\ \textbf{Seq} \qquad \frac{\vDash \alpha' \supset \alpha \quad \{\alpha\}\ c\ \{\beta\} \quad \vDash \beta \supset \beta'}{\{\alpha'\}\ c\ \{\beta'\}}\ \textbf{Con}$$

$$\frac{\{\alpha \wedge b\}\ c\ \{\beta\} \quad \{\alpha \wedge \neg b\}\ c'\ \{\beta\}}{\{\alpha\}\ \mathit{if}\ b\ \mathit{then}\ \mathit{do}\ c\ \mathit{else}\ c'\ \mathit{end}\ \{\beta\}}\ \textbf{If} \qquad \frac{\{b \wedge \iota\}\ c\ \{\iota\}}{\{\iota\}\ \mathit{while}\ b\ \mathit{do}\ c\ \mathit{end}\ \{\iota \wedge \neg b\}}\ \textbf{While}$$

We say that $\vdash \{\alpha\}\ c\ \{\beta\}$ if there is a proof of $\{\alpha\}\ c\ \{\beta\}$ using these rules.

Showed that this system was sound. Also showed it was complete assuming the theorem on the next slide.

WLP theorem

Theorem (Weakest liberal precondition): For every assertion ψ and command c, there is an assertion $wlp(c, \psi)$ such that:

1. for all states s, we have that $s \models wlp(c, \psi)$ iff for all states s', if $s \xrightarrow{[c]} s'$, then $s' \models \psi$, and
2. $\vdash \{wlp(c, \psi)\}\, c\, \{\psi\}$.

- $wlp(c, \psi)$ is essentially the least restrictive α such that running c in **any** state that satisfies α leads the system to a state that satisfies ψ.
- Need to inductively construct a $wlp(c, \psi)$ for every ψ
- $wlp(skip, \psi) \coloneqq \psi$ and $wlp(X = e, \psi(X)) \coloneqq \psi(e)$
- **Exercise**: Prove (1) and (2) for the *skip* and $X = e$ cases.
- Rest of the proof by induction on the structure of commands.

WLP theorem: $c = c_1; c_2$ case

- $wlp(c_1; c_2, \psi) :=$

WLP theorem: $c = c_1; c_2$ case

- $wlp(c_1; c_2, \psi) \coloneqq wlp(c_1, wlp(c_2, \psi))$
- (1) Have to show that for all states s, we have that $s \vDash wlp(c, \psi)$ iff for all states s', if $s \text{—}[c]\text{→} s'$, then $s' \vDash \psi$.
- When does $s \text{—}[c_1; c_2]\text{→} s'$ hold?

WLP theorem: $c = c_1; c_2$ case

- $wlp(c_1; c_2, \psi) \coloneqq wlp(c_1, wlp(c_2, \psi))$
- (1) Have to show that for all states s, we have that $s \models wlp(c, \psi)$ iff for all states s', if $s \xrightarrow{[c]} s'$, then $s' \models \psi$.
- When does $s \xrightarrow{[c_1; c_2]} s'$ hold? When there is an s'' such that $s \xrightarrow{[c_1]} s''$ and $s'' \xrightarrow{[c_2]} s'$.
- Two applications of IH yield $s'' \models wlp(c_2, \psi)$ and $s \models wlp(c_1, wlp(c_2, \psi))$.
- (2) Have to show that $\vdash \{wlp(c_1, wlp(c_2, \psi))\}\ c_1; c_2\ \{\psi\}$.
- Subproofs: $\{wlp(c_1, wlp(c_2, \psi))\}\ c_1\ \{\beta\}$ and $\{\beta\}\ c_2\ \{\psi\}$
- What β do we choose?

WLP theorem: $c = c_1; c_2$ case

- $wlp(c_1; c_2, \psi) \coloneqq wlp(c_1, wlp(c_2, \psi))$
- (1) Have to show that for all states s, we have that $s \models wlp(c, \psi)$ iff for all states s', if $s \text{—}[c]\text{→} s'$, then $s' \models \psi$.
- When does $s \text{—}[c_1; c_2]\text{→} s'$ hold? When there is an s'' such that $s \text{—}[c_1]\text{→} s''$ and $s'' \text{—}[c_2]\text{→} s'$.
- Two applications of IH yield $s'' \models wlp(c_2, \psi)$ and $s \models wlp(c_1, wlp(c_2, \psi))$.
- (2) Have to show that $\vdash \{wlp(c_1, wlp(c_2, \psi))\}\ c_1; c_2\ \{\psi\}$.
- Subproofs: $\{wlp(c_1, wlp(c_2, \psi))\}\ c_1\ \{\beta\}$ and $\{\beta\}\ c_2\ \{\psi\}$
- What β do we choose? What do we get from IH?
- **Exercise**: Fill in the details to complete this case

WLP theorem: *if* b *then do* c_1 *else* c_2 *end* case

- $wlp(\textit{if } b \textit{ then do } c_1 \textit{ else } c_2 \textit{ end}, \psi)$

WLP theorem: *if b then do* c_1 *else* c_2 *end* case

- $wlp(\textit{if } b \textit{ then do } c_1 \textit{ else } c_2 \textit{ end}, \psi) \coloneqq (b \land wlp(c_1, \psi)) \lor (\neg b \land wlp(c_2, \psi))$
- Consider s such that $s \vDash (b \land wlp(c_1, \psi)) \lor (\neg b \land wlp(c_2, \psi))$
- Then, s satisfies at least one of the two disjuncts
- Suppose $s \vDash (b \land wlp(c_1, \psi))$
- Then, $s \vDash b$ and $s \vDash wlp(c_1, \psi)$
- Consider any s' such that $s \text{---}[\textit{if } b \textit{ then do } c_1 \textit{ else } c_2 \textit{ end}]\!\!\rightarrow s'$
- When is this true? If $s \vDash b$ and $s \text{---}[c_1]\!\!\rightarrow s'$.
- By IH, for all states s', if $s \text{---}[c_1]\!\!\rightarrow s'$, then $s' \vDash \psi$. So done!
- Similarly for the case when $s \vDash (\neg b \land wlp(c_2, \psi))$

WLP theorem: *if b then do c_1 else c_2 end* case

- $wlp(\textit{if } b \textit{ then do } c_1 \textit{ else } c_2 \textit{ end}, \psi) \coloneqq (b \wedge wlp(c_1, \psi)) \vee (\neg b \wedge wlp(c_2, \psi))$
- We denote by IF the command *if b then do c_1 else c_2 end*
- Let s be a state. Suppose for every s' s.t. $s \xrightarrow{[\text{IF}]} s'$, $s' \models \psi$
- Suppose $s \models b$. Then, since $s \xrightarrow{[\text{IF}]} s'$, it must be that $s \xrightarrow{[c_1]} s'$.
- By IH $s \models wlp(c_1, \psi)$. So $s \models (b \wedge wlp(c_1, \psi))$
- Similarly, in the other case, $s \models (\neg b \wedge wlp(c_2, \psi))$
- So $s \models (b \wedge wlp(c_1, \psi)) \vee (\neg b \wedge wlp(c_2, \psi))$, i.e. $s \models wlp(\text{IF}, \psi)$
- Now we have to show that $\vdash \{wlp(\text{IF}, \psi)\}\ \text{IF}\ \{\psi\}$
- By IH, $\vdash \{wlp(c_i, \psi)\}\ c_i\ \{\psi\}$ for $i \in \{1, 2\}$
- Get $\vdash \{wlp(\text{IF}, \psi)\}\ \text{IF}\ \{\psi\}$ using these proofs and **Con** and **If**

WLP theorem: *if b then do c_1 else c_2 end* case

$$
\dfrac{\dfrac{\vDash \psi_b \supset wlp(c_1,\psi) \quad \dfrac{\text{IH}}{\{wlp(c_1,\psi)\}\, c_1\, \{\psi\}}}{\{\psi_b\}\, c_1\, \{\psi\}}\,\textbf{Con} \qquad \dfrac{\vDash \psi_{\neg b} \supset wlp(c_2,\psi) \quad \dfrac{\text{IH}}{\{wlp(c_2,\psi)\}\, c_2\, \{\psi\}}}{\{\psi_{\neg b}\}\, c_2\, \{\psi\}}\,\textbf{Con}}{\{wlp(\text{IF},\psi)\}\, \text{IF}\, \{\psi\}}\,\textbf{If}
$$

where $\psi_b = b \wedge wlp(\text{IF},\psi)$ and $\psi_{\neg b} = \neg b \wedge wlp(\text{IF},\psi)$

WLP theorem: *while b do c end* case

- Suppose c is such that $wlp(c, \theta)$ is defined for all assertions θ
- We denote by WHILE the command *while b do c end*
- We look at WHILE with postcondition ψ
- Suppose X and Y are the only program variables appearing in b, c, and ψ
- We want a $wlp(\text{WHILE}, \psi)$ which satisfies $s \models wlp(\text{WHILE}, \psi)$ iff $s' \models \psi$ for all s' s.t. $s \text{—[WHILE]→} s'$.
- That is, for every sequence of states $s_0, s_1, \dots, s_k$ such that
  - $s = s_0$,
  - c transforms s_i to s_{i+1} for all $0 \leqslant i < k$,
  - $s_i \models b$ for all $0 \leqslant i < k$, and
  - $s_k \models \neg b$,

  $s_k \models \psi$.

WLP theorem: *while b do c end* case

- What (potentially) changes from s_i to s_{i+1}?

WLP theorem: *while b do c end* case

- What (potentially) changes from s_i to s_{i+1}? Values of X and Y
- What determines whether b is true or not?

WLP theorem: *while b do c end* case

- What (potentially) changes from s_i to s_{i+1}? Values of X and Y
- What determines whether b is true or not? Again, the values of X and Y
- Denote s_i by $s(m_i, n_i)$, where $s_i(X) = m_i$ and $s_i(Y) = n_i$ for each i
- Then, $s \vDash wlp(\text{WHILE}, \psi)$ iff $\mathbb{N} \vDash \psi(m_k, n_k)$ for all sequences $(m_0, n_0), (m_1, n_1), \ldots, (m_k, n_k)$ s.t. the following hold:
- for all $i < k$, $s(m_i, n_i) \text{—}[c]\!\rightarrow s(m_{i+1}, n_{i+1})$, and
- for all $i < k$, $\mathbb{N} \vDash b(m_i, n_i)$, and
- $\mathbb{N} \vDash \neg b(m_k, n_k)$

WLP theorem: *while b do c end* case

- For every i, $s_{i+1} \vDash (X = m_{i+1}) \land (Y = n_{i+1})$ (and for each i, there is a **unique** s_{i+1} obtained by running c at s_i – Why?)
- So use IH and $s_i \text{—}[c]\text{→} s_{i+1}$ to get $s_i \vDash wlp(c, (X = m_{i+1}) \land (Y = n_{i+1}))$
- Since executing c at s_i DOES yield a next state, $s_i \vDash \neg wlp(c, 0 = 1)$
- So $s_i \vDash wlp(c, (X = m_{i+1}) \land (Y = n_{i+1})) \land \neg wlp(c, 0 = 1)$
- What if $wlp(c, (X = m_{i+1}) \land (Y = n_{i+1}))$ contains X and/or Y?
- In state s_i, X and Y should get meaning m_i and n_i respectively
- But I get s_i from s by modifying only X and Y (to make them m_i and n_i)
- Can therefore evaluate the wlp formulas at s itself, with this substitution applied!
- Substitution lemma again: Apply the substitution to the formula whose satisfaction we check, not to the interpretation

WLP theorem: *while b do c end* case

- $s_i \text{—}[c]\rightarrow s_{i+1}$ iff $s_i \models \left[wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1})) \wedge \neg wlp(c, 0 = 1)\right]$
  iff $s(m_i, n_i) \models \left[wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1})) \wedge \neg wlp(c, 0 = 1)\right]$
  iff $s \models \left[wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1})) \wedge \neg wlp(c, 0 = 1)\right](m_i, n_i)$
- So $s \models wlp(\textsc{while}, \psi)$ iff

$$s \models \forall\, k, m_0, n_0, m_1, n_1, \ldots, m_k, n_k : X = m_0 \wedge Y = n_0$$
$$\wedge \Big\{\forall i < k : [b \wedge wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1}))$$
$$\wedge \neg wlp(c, 0 = 1)](m_i, n_i)$$
$$\wedge \neg b(m_k, n_k) \wedge \Big\} \supset \psi(m_k, n_k)$$

- But we cannot quantify over sequences of natural numbers like this
- Ring any bells?

WLP theorem: *while b do c end* case

- $s_i \text{—}[c]\text{→} s_{i+1}$ iff $s_i \vDash \left[wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1})) \wedge \neg wlp(c, 0 = 1)\right]$
  iff $s(m_i, n_i) \vDash \left[wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1})) \wedge \neg wlp(c, 0 = 1)\right]$
  iff $s \vDash \left[wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1})) \wedge \neg wlp(c, 0 = 1)\right](m_i, n_i)$
- So $s \vDash wlp(\text{WHILE}, \psi)$ iff

$$s \vDash \forall k, m_0, n_0, m_1, n_1, \ldots, m_k, n_k : X = m_0 \wedge Y = n_0$$
$$\wedge \left\{\forall i < k : [b \wedge wlp(c, (X = m_{i+1}) \wedge (Y = n_{i+1}))\right.$$
$$\wedge \neg wlp(c, 0 = 1)](m_i, n_i)$$
$$\left.\wedge \neg b(m_k, n_k) \wedge\right\} \supset \psi(m_k, n_k)$$

- But we cannot quantify over sequences of natural numbers like this
- Ring any bells? Use Gödel's β-function lemma to get around it.

WLP theorem: *while b do c end* case

- So this proved (1) for the while case
- To prove (2), we need some extra work.
- One major ingredient is to show that
  $\models (b \wedge wlp(\textsc{while}, \psi)) \supset wlp(c, wlp(\textsc{while}, \psi))$.
- One can use this, IH, and the **Con** and **While** rules to get the proof.

FOL?

- Most applications we saw so far used first-order logic
- But we also saw that certain things are not expressible in FOL
- The FO theory of the natural numbers is incomplete
- Does it help to go one level up?
- To go from propositional ("zeroth-order") to first-order, we added quantification on variables
- How does one go from first-order to **second-order logic**?

FOL?

- Most applications we saw so far used first-order logic
- But we also saw that certain things are not expressible in FOL
- The FO theory of the natural numbers is incomplete
- Does it help to go one level up?
- To go from propositional ("zeroth-order") to first-order, we added quantification on variables
- How does one go from first-order to **second-order logic**?
- Quantify on **sets of** variables; Can quantify over predicates now!
- **Exercise**: Think of how to express paths in a graph using second-order logic

Second order logic: Naturals

- What about the second-order theory of the naturals?
- Recall $(A7_\varphi)$: $\varphi(0) \supset \forall x. \left[\varphi(x) \supset \varphi(s(x))\right] \supset \forall x. \left[\varphi(x)\right]$
- Does this give you the full power of induction?

Second order logic: Naturals

- What about the second-order theory of the naturals?
- Recall $(A7_{\varphi})$: $\varphi(0) \supset \forall x.\ [\varphi(x) \supset \varphi(s(x))] \supset \forall x.\ [\varphi(x)]$
- Does this give you the full power of induction? **No!**
- Only applies to predicates definable in the language (as $\varphi(x)$)!
- Second-order logic lets you say this for any set P
- You can express $+$ and $\times$ in the language, so no need for $A3, A4, A5, A6$!
- $A7 : \forall P.\ [P(0) \supset \forall x.\ [P(x) \supset P(s(x))] \supset \forall x.\ [P(x)]]$
- **Theorem (Dedekind)**: A mathematical structure satisfies $A1, A2, A7$ iff it is isomorphic to $(\mathbb{N}, 0, s)$
- So why not move to second-order logic?

Second order logic: Naturals

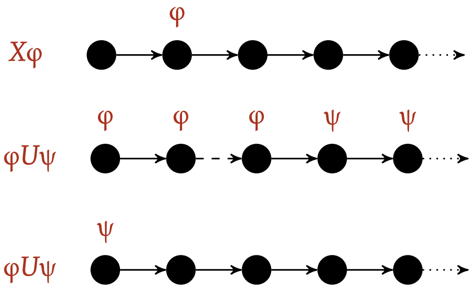
- What about the second-order theory of the naturals?
- Recall $(A7_\varphi)$: $\varphi(0) \supset \forall x.\ \left[\varphi(x) \supset \varphi(s(x))\right] \supset \forall x.\ \left[\varphi(x)\right]$
- Does this give you the full power of induction? **No!**
- Only applies to predicates definable in the language (as $\varphi(x)$)!
- Second-order logic lets you say this for any set P
- You can express $+$ and $\times$ in the language, so no need for $A3, A4, A5, A6$!
- $A7 : \forall P.\ [P(0) \supset \forall x.\ [P(x) \supset P(s(x))] \supset \forall x.\ [P(x)]]$
- **Theorem (Dedekind)**: A mathematical structure satisfies $A1, A2, A7$ iff it is isomorphic to $(\mathbb{N}, 0, s)$
- So why not move to second-order logic? It has no "nice" proof system!

Other logics

- Recall our model for Tic-Tac-Toe
- Could not easily express that $\bigcirc$ and $\times$ always alternate
- Could not say that eventually either one wins or draw
- Need logic for expressing properties that hold always or sometimes
- Enter **temporal logic**
- $\varphi, \psi \coloneqq p \mid \neg\varphi \mid \varphi \vee \psi \mid X\varphi \mid \varphi U \psi$, where $p \in AP$
- X: "In the next state (φ holds)", U: "(φ holds) until (ψ)"
- System moves from state to state at each (global) clock tick
- Crucial to system verification for dynamic systems!
- Equivalent to the first-order logic of $<$ with only unary predicates

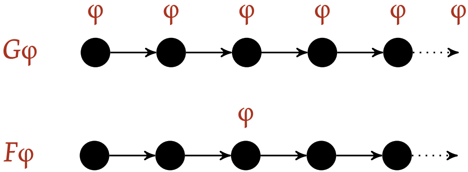
(Linear) Temporal logic

- States form a directed path (the model)
- An edge in this path is one clock tick
- Can talk about formulas being true at a particular node in this path
- What semantics do these formulas get now?



Linear temporal logic

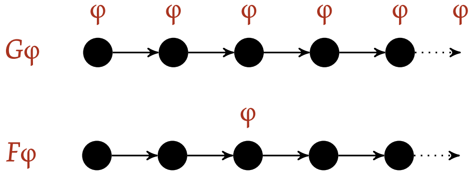
- Can define new unary operators
- $G\varphi$: (**Globally**) φ holds on the entire subsequent path
- $F\varphi$: (**In Future**) φ holds at some state on the subsequent path



- **Exercise**: Express G and F using X and U
- ○ and × always alternate:

Linear temporal logic

- Can define new unary operators
- $G\varphi$: (**Globally**) φ holds on the entire subsequent path
- $F\varphi$: (**In Future**) φ holds at some state on the subsequent path



- **Exercise**: Express G and F using X and U
- ○ and × always alternate: $G((\bigcirc \wedge X\times) \vee (\times \wedge X\bigcirc))$
- **Exercise**: Formalize "eventually either one wins or draw" using our earlier formulas for win and empty cells

Other temporal logics

- LTL looks at individual system executions as **paths**
- **Computation tree logic (CTL)** talks about the entire transition system
- CTL can talk about "along all paths" (A) and "along some path" (E)
- Can talk about $AX\varphi$, for example (but $X\varphi$ is not allowed in the syntax)
- Useful for reasoning about multiple executions of the system simultaneously

What more can I do in this area?

- Logics are inherently interesting of course
- Various logics; choose the one that is “most useful”
- Many mathematical questions to be posed/answered
- About expressive power, about structural restrictions...
- Model theoretic investigations into truth and satisfiability
- Many connections to computer science as well!
- Verification/modelling applications
- Proof theoretic investigations into provability and feasibility
- Questions of algorithms/complexity wrt satisfiability/provability also