Scribe: Alan Shieh, Gun Srijuntongsiri Lecturer: Dexter Kozen

1 Review: Relative Completeness of Hoare Logic

Last time, we proved the soundness and relative completeness of Hoare logic. Recall that Hoare logic was defined in terms of partial correctness assertions (PCA) of the form

$\{Precondition\}c\{Postcondition\}$

where *Precondition* and *Postcondition* are both sentences in the language of first-order number theory $Th(\mathbb{N})$. A PCA states that if the precondition holds, and c halts, then the postcondition holds. The **weakest precondition** $wp(c, B)$ is defined as the weakest assertion A such that $\{A\}c\{B\}$ holds, that is, if $\{A\}c\{B\}$, then $A \to wp(c, B)$.

Cook's relative completeness theorem is composed of several steps.

- Get the weakest precondition in *some* proof system. PCAs are broken down inductively in terms of $wp(c, A)$. For instance,

$$\begin{array}{rcl} wp(x := e, B) & = & B[e/x] \\ wp(c_1; c_2, B) & = & wp(c_1, wp(c_2, B)) \end{array}$$

- Show that weakest preconditions are expressible in $Th(\mathbb{N})$. This is fairly complicated, since the construction of the weakest precondition for **while**-loops results in infinite expressions. The Gödel β-function is used to show how $Th(\mathbb{N})$ can encode sequences in a finite set of numbers.

While all steps in the relative completeness theorem are constructive, and hence, in principle, implementable, it is still a complicated endeavor. We would like to use the programmer's knowledge of the program to make life easier for the verifier.

2 Verification conditions

We wish to augment the grammar of the language with annotations to allow the programmer to specify **verification conditions** to help the verifier. A challenging (and undecidable) problem for a verifier is extracting adequate loop-invariants. If we allow the programmer to break loops using annotations inserted at every loop head, then program verification will become much easier.

Grammar for annotated commands

$$c ::= x := e \;\mid\; c; x := e \;\mid\; c_1; \{D\}c_2 \;\mid\; \textbf{skip} \;\mid\; \textbf{if}\, b\, \textbf{then}\, c_1\, \textbf{else}\, c_2 \;\mid\; \textbf{while}\, b\, \textbf{do}\, \{D\}c \qquad (1)$$

Following the annotated command, annotated PCA is written as $\{A\}c\{B\}$, where c is an annotated command.

Verification conditions: Associate a conjunction of first-order sentences of number theory inductively with each annotated PCA. Denoted with $VC(\{\mathrm{A}\}\mathrm{c}\{\mathrm{B}\})$

[Parentheses are used in each PCA to disambiguate the precondition and postcondition of a PCA from the embedded annoations in the commands]

$$\begin{array}{rcl} VC(\{A\}\textbf{skip}\{B\}) & \equiv & A \Rightarrow B \\ VC(\{A\}x := e\{B\}) & \equiv & A \Rightarrow B[e/x] \\ VC(\{A\}c; x := e\{B\}) & \equiv & VC(\{A\}c\{B[e/x]\}) \end{array}$$

$$
\begin{aligned}
VC(\{A\}(c_1; \{D\}c_2)\{B\}) &\equiv VC(\{A\}c_1\{D\}) \wedge VC(\{D\}c_2\{B\}) \\
VC(\{A\}\ \textbf{if}\, b\, \textbf{then}\, c_1\ \textbf{else}\, c_2\{B\}) &\equiv VC\{A \wedge b\}c_1\{B\}) \wedge VC(\{A \wedge \neg b\}c_2\{B\}) \\
VC(\{A\}\ (\textbf{while}\, b\, \textbf{do}\ \{D\}c)\{B\}) &\equiv VC(\{D \wedge b\}c\{D\}) \wedge (A \Rightarrow D) \wedge ((D \wedge \neg b) \Rightarrow B)
\end{aligned}
$$

Note that in the while rule, the precondition A is sufficient to satisfy the loop invariant, and the loop invariant D is sufficient to satisfy the postcondition B. Also, in contrast with the weakest precondition constructions, there are no infinite expressions.

Theorem If $VC(\{A\}c\{B\})$ is true, then $\{A\}c'\{B\}$ is true, where c' is c with all annotations deleted.
Proof Induction on structure of c.

We will show only part of the induction.

Base case of assignment: $VC(\{A\}\ x := e\ \{B\}) \equiv A \Rightarrow B[x/e]$

Proved by using the weakening rule of Hoare's logic as follow:

$$
\frac{A \Rightarrow B[x/e] \quad \{B[x/e]\}\ x := e\ \{B\} \quad B \Rightarrow B}{\{A\}\ x := e\ \{B\}}
$$

Induction step of while:

$$
VC(\{A\}\ (\textbf{while}\, b\, \textbf{do}\ \{D\}c)\ \{B\}) = VC(\{D \wedge b\}c\{D\}) \wedge (A \Rightarrow D) \wedge (D \wedge \neg b \Rightarrow B)
$$

We would like to show that if the verification condition is true, then the corresponding PCA on the unannotated program is also true. For the conjunction to be true, each term must be true.
Since $VC(\{D \wedge b\}c\{D\})$ is true, the inductive hypothesis shows that $\{D \wedge b\}c\{D\}$ has a proof in Hoare logic.

We have $A \Rightarrow D$ and $D \wedge \neg b \Rightarrow B$ by the premise that the verification condition is true. Note that these statements about the precondition A and postcondition B of the **while** PCA are compatible with the Hoare weakening rule. By the Hoare **while** rule, the strengthened version of the **while**-loop has a premise $\{D \wedge b\}c\{D\}$. As stated above, this statement has a proof in Hoare logic. Therefore, we can construct the following proof tree:

$$
\dfrac{A \Rightarrow D \quad \dfrac{\dfrac{\vdots}{\{D \wedge b\}c\{D\}}}{\{D\}\ \textbf{while}\, b\, \textbf{do}\ c\ \{D \wedge \neg b\}} \quad D \wedge \neg b \Rightarrow B}{\{A\}\ \textbf{while}\, b\, \textbf{do}\ c\ \{B\}}
$$

3 Dynamic Logic (Pratt 1976)

Idea: treat a program as a binary input/output relation on *states*. For instance, a program is a relation mapping between register and memory states, or between variable states.

Weakest preconditions behave like necessity operators in modal logic.

Modal Logic: Modal logic is the logic of possibility and necessity. We start with a logic system, and add the $\Box$ (necessity) and $\Diamond$ (possibility) operators.

$$
\begin{aligned}
\Box\varphi &= \text{“It is necessary that } \varphi \text{ is true.”} \\
\Diamond\varphi &= \text{“It is possible that } \varphi \text{ is true.”} \\
&= \neg\Box\neg\varphi \text{ (Duality)}
\end{aligned}
$$

and the weakening rule

$$
\frac{\Box(\varphi \rightarrow \psi)}{\Box\varphi \rightarrow \Box\psi}
$$

[In the following section, the lecture assumes that an evaluation function is present for defining $s \models \varphi$, where s is a state and φ. In the case of our program semantics, it is simply the evaluation of an assertion (precondition/postcondition) in a particular state.]

Kripke Model: $M = (S, R)$, where S = states, and R is a binary relation on states. This allows us to define semantics for a state-based logic system. In the Kripke model, R defines the possible transitions between states. Then, we can define $\Box\varphi$ as "a statement that evaluates to true for a state s only if it is necessary that φ holds in any successor states of s". Formally,

$s \models \Box\varphi \equiv \forall t.(s,t) \quad \in \quad R \rightarrow t \models \varphi$

The Kripke model allows us to model commands as input/output relations on program states. We define $[c]\varphi$ to mean "a statement is true only in states for which φ must be true after executing c" (first equation below). This expression can express the weakest preconditions of PCAs (second equation below); given this, we can readily rewrite our inductive rules for weakest preconditions in dynamic logic.

We should show $[c]\varphi = wp(c, \varphi)$ so that this definition makes sense. Suppose $[c]\varphi$ holds for some state s. Then, c sends s to some states for which φ holds. We claim that s must satisfy $wp(c, \varphi)$. Otherwise, construct a sentence C that is true only for s. Since all successors of s satisfy φ, $C \Rightarrow \mathcal{C}[\![c]\!](s) \models^I \varphi$. In state s, C is true, but $wp(c, \varphi)$ is false, so $C \not\Rightarrow wp(c, \varphi)$. This contradicts the definition of $wp(c, \varphi)$.

Suppose $wp(c, \varphi)$ is true for s. Then, after executing c, φ must hold. This means that $\forall t.(s,t) \in c^M \Rightarrow t \models \varphi$, and so $[c]\varphi$ must be true in s. Hence, the two statements are equivalent. $\Box$

$$
\begin{aligned}
s \models [c]\varphi &\equiv \forall t.(s,t) \in c^M \Rightarrow t \models \varphi \\
[c]\varphi &= wp(c, \varphi) \\
\{A\}c\{B\} &= A \Rightarrow wp(c, B) = A \Rightarrow [c]B
\end{aligned}
$$

$$
\begin{aligned}
[c_1; c_2]B &\Leftrightarrow [c_1]([c_2]B) \\
wp(c_1; c_2, B) &= wp(c_1, wp(c_2, B))
\end{aligned}
$$

Several rules from modal logic carry over to dynamic logic, including the weakening rule

$$\frac{[c](A \Rightarrow B)}{[c]A \Rightarrow [c]B}$$