Problem Sheet 5: Program Proofs

Chris Poskitt
ETH Zürich

1 Axiomatic Semantics Recap

This section provides some additional questions on Hoare logic. The proof rules are given again in Figure 3.

i. Devise an axiom for the command $\texttt{havoc}(\texttt{x}_0, \ldots, \texttt{x}_n)$, which assigns arbitrary values to the variables $\texttt{x}_0, \ldots, \texttt{x}_n$.

ii. Write a program that computes the factorial of a natural number stored in variable `x` and assigns the result to variable `y`. Prove that the program is correct using our Hoare logic.

iii. Sarah Proofgood has successfully shown that given an arbitrary program P and postcondition *post*, the triple:

$$\{\text{WP}[P, post]\}\ P\ \{post\}$$

can be proven in our Hoare logic, i.e. $\vdash \{\text{WP}[P, post]\}\ P\ \{post\}$. Here, $\text{WP}[P, post]$ is an assertion expressing the *weakest (liberal) precondition* relative to P and *post*; that is, the weakest condition that must be satisfied for P to establish *post* (without guaranteeing termination).

Using Sarah's result, show that *any* valid triple $\models \{p\}\ P\ \{q\}$ is provable in our Hoare logic, i.e. $\vdash \{p\}\ P\ \{q\}$.

2 Separation Logic Recap

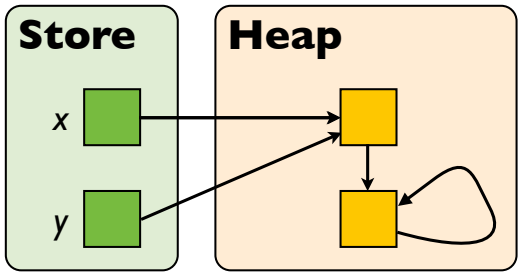
This section provides some additional practice on using separation logic. The small axioms and frame rule of separation logic are given in Figure 4.

i. Are the following assertions satisfiable? Justify your answers.

$$p * \neg p$$

$$x = y * \neg(x = y)$$

ii. Consider the following program state:



Which of the following assertions does this state satisfy? For the assertions it does not satisfy: why not?

(a) $\exists v.\ x \mapsto v * v \mapsto v$

(b) $\exists v.\ x \mapsto v * v \mapsto v * y \mapsto v$

(c) $y \mapsto \_$

(d) $(x = y) \wedge (y \mapsto \_ * \text{true})$

(e) $(x = y) * \text{true}$

iii. Starting from precondition {emp}, apply the axioms and inference rules of separation logic to derive a postcondition expressing exactly the contents of the store and heap at termination (assume that `x` and `y` are the only variables). Then, depict this state using the store and heap diagrams presented in the lectures.

```
x := cons(5,9);
y := cons(6,7);
x := [x];
[y+1] := 9;
dispose(y);
```

3 Graph-Based Reasoning and Verification

The questions in this section are about modelling programs using graph transformation, asserting properties of their states, and Hoare-style proof rules facilitating reasoning. Introductory details are given in the lecture notes:

http://se.inf.ethz.ch/courses/2013b_fall/sv/slides/07-GraphBasedReasoning.pdf

We provide proof rules of a Hoare logic in Figure 5.

i. Define, in English, the meaning of $\models \{c\}\ P\ \{d\}$, where P is a graph program, and c, d are E-conditions. How does the definition handle nondeterminism and program failure?

ii. Consider the graph program in Figure 1. What does it compute? Trace an execution on an input graph that results in the no-branch being executed, and another that results in the yes-branch being executed.

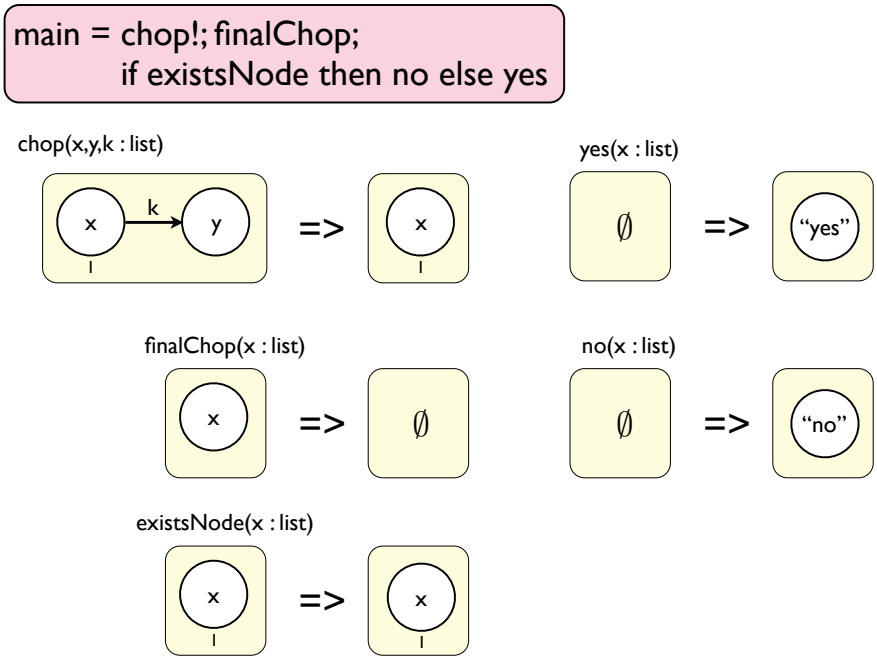


Figure 1: Graph program for exercise 3-ii

iii. Write a graph program that iteratively adds loops to nodes such that the partial correctness specification in Figure 2 is fulfilled (you do not need to prove this). Informally justify that your program terminates.

iv. The assertion language of E-conditions we proposed is not yet powerful enough to specify framing properties. Demonstrate this by writing a program that always returns the empty graph $\emptyset$ (the graph containing no nodes and no edges), yet also fulfills the partial correctness specification of Figure 2. Then, strengthen the specification for your program in part (iii), by informally (i.e. in English) adding frame axioms to the pre- and postcondition.

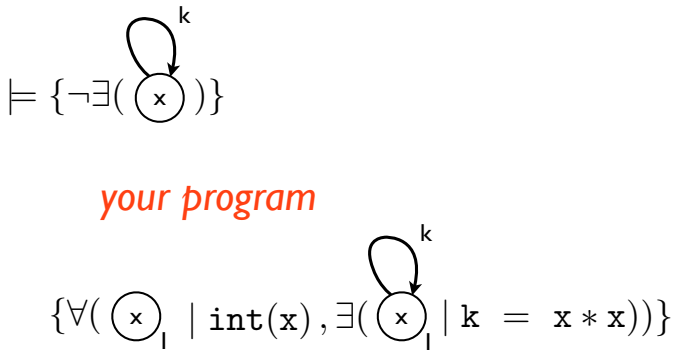


Figure 2: Partial correctness specification for exercises 3-iii and 3-iv

v. Propose a partial correctness proof rule for the new control construct:

$$P \texttt{ or } Q$$

which nondeterministically executes one of the programs P or Q.

vi. Propose a partial correctness proof rule for the derived control construct:

$$\texttt{if } \mathcal{R} \texttt{ then } P$$

which is equivalent to $\texttt{if } \mathcal{R} \texttt{ then } P \texttt{ else } \emptyset \Rightarrow \emptyset$.

vii. Write an E-condition expressing that: "there exists *exactly* one node incident to a loop". **Hint:** you will need nesting.

Appendix: Proof Rules

$$[\text{ass}] \quad \vdash \{p[e/x]\}\ x := e\ \{p\}$$

$$[\text{skip}] \quad \vdash \{p\}\ \texttt{skip}\ \{p\}$$

$$[\text{comp}]\ \frac{\vdash \{p\}\ P\ \{r\} \quad \vdash \{r\}\ Q\ \{q\}}{\vdash \{p\}\ P; Q\ \{q\}}$$

$$[\text{if}]\ \frac{\vdash \{b \wedge p\}\ P\ \{q\} \quad \vdash \{\neg b \wedge p\}\ Q\ \{q\}}{\vdash \{p\}\ \texttt{if}\ b\ \texttt{then}\ P\ \texttt{else}\ Q\ \{q\}}$$

$$[\text{while}]\ \frac{\vdash \{b \wedge p\}\ P\ \{p\}}{\vdash \{p\}\ \texttt{while}\ b\ \texttt{do}\ P\ \{\neg b \wedge p\}}$$

$$[\text{cons}]\ \frac{p \Rightarrow p' \quad \vdash \{p'\}\ P\ \{q'\} \quad q' \Rightarrow q}{\vdash \{p\}\ P\ \{q\}}$$

Figure 3: A Hoare logic for partial correctness

$$\vdash \{e \mapsto \_\}\ [e] := f\ \{e \mapsto f\}$$

$$\vdash \{e \mapsto \_\}\ \texttt{dispose}(e)\ \{\text{emp}\}$$

$$\vdash \{X = x \wedge e \mapsto Y\}\ x := [e]\ \{e[X/x] \mapsto Y \wedge Y = x\}$$

$$\vdash \{\text{emp}\}\ x := \texttt{cons}(e_0, \ldots, e_n)\ \{x \mapsto e_0, \ldots, e_n\}$$

$$\frac{\vdash \{p\}\ P\ \{q\}}{\vdash \{p * r\}\ P\ \{q * r\}}$$

side condition: no variable modified by P appears free in r

Figure 4: The small axioms and frame rule of separation logic

$$[\text{ruleapp}]\ \frac{}{\{\text{Pre}(r,c)\}\ r\ \{c\}}$$

$$[\text{nonapp}]\ \frac{}{\{\neg\text{App}(\{r\})\}\ r\ \{\texttt{false}\}}$$

$$[\text{ruleset}]\ \frac{\{c\}\ r\ \{d\} \text{ for each } r \in \mathcal{R}}{\{c\}\ \mathcal{R}\ \{d\}}$$

$$[\text{comp}]\ \frac{\{c\}\ P\ \{e\} \quad \{e\}\ Q\ \{d\}}{\{c\}\ P;\ Q\ \{d\}}$$

$$[\text{if}]\ \frac{\{c \wedge \text{App}(\mathcal{R})\}\ P\ \{d\} \quad \{c \wedge \neg\text{App}(\mathcal{R})\}\ Q\ \{d\}}{\{c\}\ \texttt{if}\ \mathcal{R}\ \texttt{then}\ P\ \texttt{else}\ Q\ \{d\}}$$

$$[\text{try}]\ \frac{\{c \wedge \text{App}(\mathcal{R})\}\ \mathcal{R};\ P\ \{d\} \quad \{c \wedge \neg\text{App}(\mathcal{R})\}\ Q\ \{d\}}{\{c\}\ \texttt{try}\ \mathcal{R}\ \texttt{then}\ P\ \texttt{else}\ Q\ \{d\}}$$

$$[!]\ \frac{\{inv\}\ \mathcal{R}\ \{inv\}}{\{inv\}\ \mathcal{R}!\ \{inv \wedge \neg\text{App}(\mathcal{R})\}}$$

$$[\text{cons}]\ \frac{c \Rightarrow c' \quad \{c'\}\ P\ \{d'\} \quad d' \Rightarrow d}{\{c\}\ P\ \{d\}}$$

where $\text{App}(\mathcal{R})$ is an E-condition satisfied exactly by those graphs that a rule in $\mathcal{R}$ can be applied to; and

$\text{Pre}(r, c)$ is an E-condition expressing the *weakest* property that must hold for r to establish c (without guaranteeing termination)

Figure 5: A Hoare logic for partial correctness of graph programs