Assignment 2
Dynamic Logic
Sample Solution

15-316: Software Foundations of Security & Privacy
Frank Pfenning

Due Tuesday, September 17, 2024
80 points

Note that this is a *sample solution*! There are often multiple correct ways to solve a problem and we do not try to comprehensive in any way.

1 Skip (15 points)

In this problem we consider adding a new program **skip** to the language already containing **assert** P (that is, unsafe behavior). **skip** does nothing, like a "nop" instruction in assembly language. For example, we can write a trivial nonterminating loop as **while** $\top$ **skip**.

Task 1 (2 pts) Give a semantic definition of $\omega[\![\mathbf{skip}]\!]\nu$.

$$\omega[\![\mathbf{skip}]\!]\nu \quad \text{iff} \quad \omega = \nu$$

Task 2 (2 pts) Give a semantic definition of $\omega[\![\mathbf{skip}]\!]\lightning$.

$$\omega[\![\mathbf{skip}]\!]\lightning \quad \text{iff} \quad \text{never}$$

Task 3 (2 pts) Give a *valid axiom* characterizing **skip** in the form $[\mathbf{skip}]Q \leftrightarrow ??$. Your task is to fill in "??". You do not need to prove the validity of your axiom.

$$[\mathbf{skip}]Q \leftrightarrow Q$$

Task 4 (2 pts) Assuming the validity of your axiom, write out right ($[\mathbf{skip}]R$) and left ($[\mathbf{skip}]L$) rules for **skip** in the sequent calculus.

$$\frac{\Gamma \vdash Q, \Delta}{\Gamma \vdash [\mathbf{skip}]Q, \Delta}\ [\mathbf{skip}]R \qquad \frac{\Gamma, Q \vdash \Delta}{\Gamma, [\mathbf{skip}]Q \vdash \Delta}\ [\mathbf{skip}]L$$

Task 5 (5 pts) Using the right and left rules for sequential composition ($[;]R$ and $[;]L$) and your own rules from the previous task, prove that

$$\cdot \vdash [\mathbf{skip}\ ;\ \alpha]Q \leftrightarrow [\alpha]Q$$

(using the general definition of $P_1 \leftrightarrow P_2 \triangleq (P_1 \rightarrow P_2) \wedge (P_2 \rightarrow P_1)$). Do not use a semantic argument.

$$\dfrac{\dfrac{\dfrac{}{[\alpha]Q \vdash [\alpha]Q}\ \mathsf{id}}{[\mathbf{skip}]([\alpha]Q) \vdash [\alpha]Q}\ [\mathbf{skip}]L}{[\mathbf{skip}\ ;\ \alpha]Q \vdash [\alpha]Q}\ [;]L}{\cdot \vdash [\mathbf{skip}\ ;\ \alpha]Q \rightarrow [\alpha]Q}\ {\rightarrow}R \qquad \dfrac{\dfrac{\dfrac{}{[\alpha]Q \vdash [\alpha]Q}\ \mathsf{id}}{[\alpha]Q \vdash [\mathbf{skip}]([\alpha]Q)}\ [\mathbf{skip}R]}{[\alpha]Q \vdash [\mathbf{skip}\ ;\ \alpha]Q}\ [;]R}{\cdot \vdash [\alpha]Q \rightarrow [\mathbf{skip}\ ;\ \alpha]Q}\ {\rightarrow}R$$

We define that two programs α and β are *equivalent* if they have the same meaning, that is,

$$(\omega[\![\alpha]\!]\nu \text{ iff } \omega[\![\beta]\!]\nu) \quad \text{and} \quad (\omega[\![\alpha]\!]\lightning \text{ iff } \omega[\![\beta]\!]\lightning)$$

Task 6 (2 pts) Give two different programs (using disjoint constructs) that are equivalent to **skip**. You do not need to prove the equivalences.

$$\begin{array}{lcl} \mathbf{skip} & \triangleq & \mathbf{assert}\ \top \\ \mathbf{skip} & \triangleq & x := x \end{array}$$

2 For Loops (65 points)

The general form of **while** loops and the absence of explicitly given loop invariants can make it difficult to prove safety properties. In this problem you will consider **for** loops that have a more restricted pattern of iteration, possibly making it easier to prove safety.

We give an informal description of our kind of **for** loops and your task will be to formalize and prove some properties of it. We use the syntax

$$\mathbf{for}\ 0 \leq i < n\ \mathbf{do}\ \alpha$$

The loop body α may depend on the variables i and n (which must be different variables), but α may not assign to i or n. You should assume these properties are checked by the parser and your answers below can depend on them.

The **for** loop above executes as follows:

1. If $n < 0$, the construct is considered *unsafe*.
2. Execute α for $i = 0, 1, \ldots, n-1$ in this order. If $n = 0$ then α is not executed at all.
3. After the loop exits, i should be equal to n.

Task 7 (5 pts) Using **for** (and not **while**), write a program to compute the sum $1 + 3 + 5 + \cdots + (2k + 1)$ under the precondition $k \geq 0$.

$$a := 0 \ ; \ \textbf{for } 0 \leq i < k + 1 \textbf{ do } \{ \ a := a + 2 * i + 1 \ \}$$

Task 8 (5 pts) Define $\omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]\nu$ inductively, analogously to the way we defined the meaning of $\omega[\![\textbf{while } P \ \alpha]\!]\nu$.

$$\begin{array}{lll} \omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]\nu & \text{iff} & \omega(n) \geq 0 \text{ and } \omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]^0\nu \\ \omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]^k\nu & \text{iff} & k < \omega(n) \text{ and } \omega'[\![\alpha]\!]\mu \text{ and } \mu[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]^{k+1}\nu \\ & & \quad \text{for some } \mu \text{ where } \omega' = \omega[i \mapsto k] \\ & & \text{or } k = \omega(n) \text{ and } \nu = \omega[i \mapsto n] \end{array}$$

Task 9 (5 pts) Define $[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]\lightning$ inductively, analogously to the way we defined the meaning of $[\![\textbf{while } P \ \alpha]\!]\lightning$.

$$\begin{array}{lll} \omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]\lightning & \text{iff} & \omega(n) < 0 \text{ or } \omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]^0\lightning \\ \omega[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]^k\lightning & \text{iff} & k < \omega(n) \text{ and } (\omega'[\![\alpha]\!]\lightning \text{ or } (\omega'[\![\alpha]\!]\mu \text{ and } \mu[\![\textbf{for } 0 \leq i < n \textbf{ do } \alpha]\!]^{k+1}\lightning \\ & & \quad \text{for some } \mu \text{ where } \omega' = \omega[i \mapsto k])) \end{array}$$

Task 10 (20 pts) Give a right rule $[\textbf{for}]R$ for $[\textbf{for } 0 \leq i < n \textbf{ do } \alpha]Q(i)$ in analogy to our proof rule $[\textbf{while}]R$.

You should allow for an arbitrary loop invariant $J(i)$ in the premises, analogously to $[\textbf{while}]R$. Your rule should incorporate assumptions about i that hold for all safe **for** loops so they don't need to be expressed explicitly in J every time the proof rule is used.

Furthermore, the only explicit *program properties* in your premises should be for α, although formulas do not need to get smaller.

$$\frac{\Gamma, i' = 0 \vdash n \geq 0 \land J(i'), \Delta \quad J(i), 0 \leq i < n, i' = i + 1 \vdash [\alpha]J(i') \quad J(i), i = n \vdash Q(i)}{\Gamma \vdash [\textbf{for}\ 0 \leq i < n\ \textbf{do}\ \alpha]Q(i), \Delta}\ [\textbf{for}]R^{i'}$$

where i' is chosen fresh (not in Γ, $J(i)$, Δ, α or $Q(i)$).

Task 11 (15 pts) Prove the correctness of the following **for** loop using your rule from the previous task. Explicitly state the loop invariant J you used.

$$n \geq 0, a = 0, b = 0 \vdash [\text{square}]\, a = n * n$$

where "square" is the program

$$\textbf{for}\ 0 \leq i < n\ \textbf{do}\ \{\ a := a + b + 1\ ;\ b := b + 2\ \}$$

For space reasons, state the proof of each premise of your $[\textbf{for}]R$ rule separately, but make it clear which proof is of which premise. You do not need to justify any sequent of pure arithmetic (that is, not containing any programs), but of course such sequents must be valid and you should check that to your own satisfaction.

We use $J(i) \triangleq a = i * i \land b = 2 * i$
First premise:

(by arithmetic)

$$n \geq 0, a = 0, b = 0, i' = 0 \vdash n \geq 0 \land a = i' * i' \land b = 2 * i'$$

Second premise:

(by arithmetic)

$$\dfrac{\dfrac{\dfrac{a = i * i \land b = 2 * i, 0 \leq i < n, i' = i + 1, a' = a + b + 1, b' = b + 2 \vdash a' = i' * i' \land b' = 2 * i'}{a = i * i \land b = 2 * i, 0 \leq i < n, i' = i + 1, a' = a + b + 1 \vdash [b := b + 2]a' = i' * i' \land b = 2 * i'}\ [:=]R}{a = i * i \land b = 2 * i, 0 \leq i < n, i' = i + 1 \vdash [a := a + b + 1]([b := b + 2]a = i' * i' \land b = 2 * i')}\ [:=]R}{a = i * i \land b = 2 * i, 0 \leq i < n, i' = i + 1 \vdash [a := a + b + 1\ ;\ b := b + 2]a = i' * i' \land b = 2 * i'}\ [;]R$$

Third premise:

(by arithmetic)

$$a = i * i \land b = 2 * i, i = n \vdash a = n * n$$

Task 12 (10 pts) If possible, provide a translation of **for** $0 \leq i \leq n$ **do** α into our language without **for** (but with **while**) with the same meaning (which you do not need to prove). If you believe it is not possible, explain briefly why you think so.

$$i := 0 \,;\, \textbf{while}\ 0 \leq i \wedge i \leq n\ \textbf{do}\ (\alpha \,;\, i := i + 1)$$

Task 13 (5 pts) If possible, provide a translation of **while** $P\ \alpha$ into our language without **while** (but with **for**) with the same meaning (which you do not need to prove). If you believe it is not possible, explain briefly why you think so.

This is not possible, because safe **for** loops (that is, those where $n \geq 0$) are terminating.