Assignment 2
Dynamic Logic

15-316: Software Foundations of Security & Privacy

Due Tue, Feb 3, 2026
75 points

Your solution should be handed in as file `hw2.pdf` to Gradescope. If at all possible, write your solution in LaTeX. The handout `hw2-dl.zip` includes the LaTeX sources for Lectures 3 and 4 and the necessary style files which provide some examples for rules, derivations, and proofs.

1 Abort (20 points)

In this problem we consider adding a new program **abort** to the language already containing **assert** P (that is, unsafe behavior). The command **abort** represents an immediate runtime error or crash: it is always *unsafe* and it has no poststate.

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

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

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

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

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

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

Do not use a semantic argument. You should provide two sequent derivations, one for $[\mathbf{abort}\ ;\ \alpha]Q \rightarrow [\mathbf{abort}]Q$ and another for $[\mathbf{abort}]Q \rightarrow [\mathbf{abort}\ ;\ \alpha]Q$.

Task 6 (5 pts) State whether $[\alpha\,;\,\mathbf{abort}]Q \leftrightarrow [\mathbf{abort}]Q$. If it is, then provide a sequent derivation proving it like in Task 5. If it isn't, then briefly explain why, and provide a concrete program α that serves as a counterexample, i.e. and α such that $\not\models [\alpha\,;\,\mathbf{abort}]Q \leftrightarrow [\mathbf{abort}]Q$.

2 For Loops (55 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

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

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 $k \geq 0$.

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 $\omega[\![\textbf{while } P\ \alpha]\!]\nu$.

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

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.

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.

$$n \geq 0, a = 0, b = 1 \vdash [\text{oddsum}]\,(b = 2 * n + 1)$$

where "oddsum" is the program (which computes the sum of the first n odd numbers)

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

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.

Task 12 (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.