Assignment 2: Dynamic Logic
15-316 Software Foundations of Security and Privacy

1. **Arbitrary conditions (25 points).** Sometimes when modeling a computation, we need to avoid making assumptions about what exactly might transpire at runtime. For example, suppose that we wish to write a program that accepts user input, and branches on the value that they provide. Because there is no way of knowing what the user will do advance, a safety analysis needs to cover all possible cases. Nondeterminism is the appropriate way to handle cases like this.

   Part 1 (5 points). Extend the language discussed in lecture by defining the semantics of a nondeterministic branching command, $\texttt{if}(*)\,\alpha\,\texttt{else}\,\beta$. Informally, this command arbitrarily selects either α or β to run; the choice does not depend on the state in which the command is executed.

   Solution.

   Part 2 (10 points). Design an axiom that allows you to reason about box modalities around nondeterministic branches:

$$[\texttt{if}(*)\,\alpha\,\texttt{else}\,\beta]p(x) \leftrightarrow \ldots$$

   The right side of this equivalence should not contain a box or diamond modality, but only first-order formulas. Prove that your axiom is valid using your semantics from Part 1.

   Solution.

Part 3 (10 points). Suppose that two programs α and β are identical in every way, except that all of the branches in β are nondeterministic, and all of those in α are deterministic. For example, the following programs would match this description:

$$\begin{aligned} \alpha \equiv &\quad x := 1; \mathtt{if}(y < 0)\, z := y \,\mathtt{else}\, z := -y \\ \beta \equiv &\quad x := 1; \mathtt{if}(*)\, z := y \,\mathtt{else}\, z := -y \end{aligned}$$

If β satisfies a given safety property Φ, then will α necessarily satisfy it as well? Likewise, if α satisfies Φ, then must β? For both questions, if you believe that both will satisfy Φ, then use your semantics from Part 1 and the definition of safety properties to justify your example. Otherwise, provide a counterexample set of α, β, and Φ where only one of α, β satisfy Φ.

Solution.

2. **Verified safety (15 points).** In the previous homework, you looked for ways to exploit a flawed runtime memory safety monitor. Recall that the safety policy aimed to ensure that a given program cannot write outside the range `0x800300`–`0x8003FF` (inclusive). Another way to enforce this policy is to verify that the program will not write outside the bounds, before executing it; this removes the need for any runtime monitors.

   However, non-determinism arising from inputs that are unknown before execution can pose a challenge. Consider the program α below, which uses both non-deterministic branching, as well as non-deterministic assignment.

$$\alpha \equiv s := *; \texttt{if}(*)\,\{p := p + s\}\,\texttt{else}\,\{\texttt{assert}(\texttt{false})\}$$

   The `assert` command serves to signal an exception when the policy is violated. You should understand the non-deterministic assignment as updating the state by mapping the target variable to an arbitrary integer. The following axiom characterizes its behavior:

$$[x := *]p(x) \leftrightarrow \forall x.p(x)$$

   Part 1 (5 points). Explain why the following formula is not valid by giving a trace of α that violates the safety policy.

$$\texttt{0x8000300} \leqslant p \leqslant \texttt{0x8000400} \rightarrow [\alpha]\texttt{0x8000300} \leqslant p \leqslant \texttt{0x8000400}$$

   Solution.

Part 2 (10 points). Identify an expression e to place in the conditional (i.e., a branch condition that removes the non-determinism from the `if`) that will make the program satisfy the safety policy. Is the formula from Part 1 now valid with your fix? If so, provide a sequent deduction using the axioms of dynamic logic. If not, then identify the premise in an attempted sequent deduction that is not valid.

Solution.