15-812 Programming Language Semantics Spring 2018

Homework 3: Methods and fun with games April 5

Name: _____

Andrew ID: _____

1 Methods in Programs (35 points)

This question will give you the opportunity of extending the basic (deterministic) while programming language with methods/procedures.

Grading will follow a mix of generality and correctness. Precedence will, however, be given to a correct development, instead of the most general possible approach that, unfortunately, is incorrect. Assume that you have a single procedure $m()$ without arguments but the ability to change state.

(Syntax)

0 **Task 1** Change the syntax of the usual dynamic logic with while programs where all variables have type $\mathbb{Z}$ to incorporate calls to the procedure $m()$.

(Semantics)

10 **Task 2** Assume that you are given a fixed program γ in the above syntax that is to be used as the function body for procedure $m()$. Define the denotational semantics $[\![\alpha]\!] \subseteq \mathcal{S} \times \mathcal{S}$ of programs in the presence of procedure calls.

(Axioms)

5 **Task 3** One way of reasoning about procedures is by macro expansion to replace their call by their definition. Express this as an axiom in dynamic logic and indicate limitations or side conditions if there are any.

(Proof Rules)

10 **Task 4** Another way of reasoning about procedures is by proving and then using a pre/postcondition contract. Express this as a proof rule in dynamic logic and indicate limitations or side conditions if there are any.

(Soundness)

10 **Task 5** Prove that the axioms or proof rules you gave in the previous tasks are sound in the semantics you defined.

2 Security Games Against Evil Hackers™ (15 points)

This question considers a system program that is under security attack and studies it as a game. Angel plays the role of the Evil Genius Attacker. Demon plays the role of the Defender. The system has push and pop operations on a stack that is unguarded and just implemented naively in C by pointer operations. The system also has a guarded peek operation that returns the top of the stack into z without changing the stack but requires the stack to be nonempty. The entire operating system can also be force-resetted in which case the stack will restart with an initial value that we will simply also call z. The exact C implementations is amusing, diverting, and entertaining, but the question can already be understood without looking into the exact data representation and code implementation. That is why it will be enough for our purposes to simply represent the stack sizes as integers in the game without the need to explore more data types.

This leads to the following formula in Game Logic:

$$\Big[\big(\; (x := x+1 \cup x := x-1)^*; \quad \text{push or pop} \;\; (?x>0; z := x; \cup x := z)^d \quad \text{guarded peek or reset} \;\big)^*\Big] x \geq 0$$

10 **Task 1** Identify the weakest possible condition on the initial state under which the above formula is true and give a sequent calculus proof in Game Logic.

5 **Task 2** What the above game rendition elides is that resets take significantly more time than push, pop, and guarded peeks. Both a runtime semantics that you have already explored in another assignment as well as additional cost variables could model this in the game. Assuming this is taken care of, describe and explain the winning strategy you follow that wins while exhibiting minimal cost.