15-812 Programming Language Semantics Spring 2018

Homework 1: Imperative program reasoning and undefinedness Feb 13

Name: _____

Andrew ID: _____

1 Imperative Programs with Undefinedness (40 points)

While the Programming Language Semantics lectures explain the most important foundations of giving and using semantics to programs, your future career will often explore extensions of the concepts from class. This question gives you the opportunity to develop one such extension. You will extend dynamic logic beyond what we have seen in lecture to also cover the case of undefinedness. This will give you an opportunity to reflect back on what we have seen in class while simultaneously scrutinizing it critically to see what changes in the presence of undefinedness and what can stay as is without compromising the approach. 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.

Even if imperative programs can have any number of different sources of undefinedness in C, we will isolate the matter by focusing exclusively on the undefinedness coming from the fact that divisions by zero are undefined.

(Syntax)

0 **Task 1** Change the syntax of dynamic logic to incorporate division, preparing for the challenge of handling the undefinedness resulting from divisions by zero. If you choose a development that limits the occurrences of divisions, then also describe in what way your syntactic choice covers all relevant cases.

(Semantics) Define the semantics of dynamic logic in the presence of undefinedness.

5 **Task 2** Define the semantics of terms as a function $Z(\cdot)$ that defines the meaning $Z(\theta)$ for each term θ. Begin by specifying the semantic domain by giving the type of $Z(\cdot)$.

5 **Task 3** Define the semantics of formulas as a function $Z(\cdot)$ that defines the meaning $Z(\phi)$ for each formula ϕ. Begin by specifying the semantic domain by giving the type of $Z(\cdot)$.

5 **Task 4** Define the semantics of programs as a function $Z(\cdot)$ that defines the meaning $Z(\alpha)$ for each program α. Begin by specifying the semantic domain by giving the type of $Z(\cdot)$.

(Axioms) For each top-level operator in the box modality of the dynamic logic axioms discussed in class was there one (or occasionally two) axioms.

10 **Task 5** Give corresponding axioms or proof rules for dynamic logic in the presence of undefinedness. Recall that soundness is more important than generality.

15 **Task 6** Prove that the axioms or proof rules you gave in the previous task are sound in the semantics you defined.

2 Proving Practice (10 points)

Let's go back to ordinary dynamic logic from class.

Consider a fixed interpretation in which gcd is interpreted to be the standard greatest common divisor function. Give a sequent calculus proof, using the axioms of dynamic logic, that the following formula is valid.

$$0 < a \land 0 < b \rightarrow [x := a; y := b; \mathsf{while}(x \neq y)\, \{\mathsf{if}(x > y)\, x := x - y\, \mathsf{else}\, y := y - x](x = gcd(a, b))$$

Hint: Feel free to use the usual identities of gcd in the proof such as

$$\forall a, b\ gcd(a, b) = gcd(a, b - a)$$