CS 599: Assignment 1

Due Wednesday, February 7, 2024

Total: 50 pts

Ankush Das

- This assignment is due on the above date and it must be submitted electronically on Gradescope. Please create an account on Gradescope, if you haven't already done so.
- Please use the template provided on the course webpage to typeset your assignment and please include your name and BU ID in the Author section (above).
- Although it is not recommended, you can submit handwritten answers that are scanned as a PDF and clearly legible.
- You should hand in one file, named as ⟨first-name⟩_⟨last-name⟩_⟨BU-ID⟩_asgn1.pdf containing the solutions to the problems below.
- You will be provided a tex file, named asgn1.tex. It contains environments called solution. Please enter your solutions inside these environments.

Problem 1 (5 pts) *In class, we looked at a simple arithmetic expression language. Similarly, for this assignment, we will study a simple boolean expression language. This language contains 3 expressions written using the following syntax:*

$$\textit{Expressions} \quad e ::= \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e\ \mathsf{else}\ e$$

The first two expressions are just the values true and false respectively. The expression if b then e_1 else e_2 *describes the standard* if *behavior, i.e., if b evaluates to true, then the expression reduces to e_1, else if b evaluates to false, then the expression reduces to e_2.*

(3 pts) Write and explain each rule of semantics for this language. Note that the semantics will be written using the two standard judgments: e value *and $e \mapsto e'$.*

(2 pts) Consider the following expression:

if (if true then false else true) then (if false then false else true) else (if true then true else false)

Evaluate this expression using the rules of the semantics you just defined above.

Problem 2 (16 pts) *Now, we will try to express booleans using λ-calculus. The general technique is to represent the two values, i.e., true and false using normal forms (or values) in λ-calculus. Furthermore, they should be closed, that is, not contain any free variables. Since we need to distinguish between the two values, we need to introduce two variables. We then define rather arbitrarily one to be true and the other to be false*

$$\mathsf{true} = \lambda x.\lambda y.x \qquad \mathsf{false} = \lambda x.\lambda y.y$$

(8 pts) Our first task is to express the if *expression defined in the previous problem in λ-calculus. To do this, let's consider two constants, T and E. T represents the expression for the 'then' branch, while E represents the expression for the 'else' branch. In other words, construct the expression* if b then T else E *in λ-calculus, i.e., using λ expressions, function applications, and variables.*

(2 pts) *Verify that the expression you constructed is indeed equivalent to* if b then T else E. *To do this, suppose the expression you constructed above is* I. *Then,* $(\lambda b.I)$ true *should evaluate to* T. *And similarly,* $(\lambda b.I)$ false *should evaluate to* E, *where* true *and* false *are the expressions above. Evaluate these two expressions using the semantics rules of* λ*-calculus to verify that they indeed evaluate to* T *and* F *respectively.*

(6 pts) *Use the expression above to construct the* and, or, *and* not *functions.*

- and *should have the form* $\lambda b_1.\lambda b_2.\ldots$. *It takes two booleans as parameters and returns a boolean.*
- *Similarly,* or *should have the same form.*
- not *should have the form* $\lambda b.\ldots$, *i.e., takes a boolean as a parameter and returns a boolean.*

Problem 3 (4 pts) *Consider the following expression, popularly known as the Y-combinator, defined as* $(\lambda x.x\ x)\ (\lambda x.x\ x)$. *What does this expression evaluate to? Show each step of the evaluation. Is this expression a value or not? Does this violate the progress theorem we discussed in the lecture?*

Problem 4 (25 pts) *Let's return to the LL1 language discussed in the lecture. We now add an expression for multiplication in the language, so the formal syntax is*

$$\textit{Expressions} \quad e ::= \overline{n} \mid e + e \mid e \times e$$

(5 pts) *First, define the rules of (small-step) semantics for this language.*

(5 pts) *You will notice there is some non-determinism in the semantics rules, i.e., for a given expression* e, *it will evaluate to two different values based on the order in which the rules are applied. Give an example of such an expression* e *and show that it evaluates to two different values in two different derivations. Again, show each step of the derivation.*

(10 pts) *Our goal is to eliminate this non-determinism. For this, we have to come up with evaluation precedence, i.e., the order in which operators are applied. We arbitrarily decide that multiplication has higher precedence than addition, i.e.,* ***all multiplication must occur before any addition****. Define the semantics rules for such an evaluation.* ***Hint:*** *It might help to define an auxiliary judgment:* e mult−value, *which holds iff expression* e *contains no addition operators.*

(5 pts) *Take the same expression* e *defined in sub-task 2 and evaluate it using the semantics rules defined in sub-task 3. Is there still any non-determinism in these rules? Can* e *still evaluate to different values?*