HW2 – Syntax and Type Checking

CS 476, Fall 2023

1 Instructions

Begin by downloading the file `hw2-base.ml` from the course website, and renaming it to `hw2.ml`. Then fill in your answers to the problems, adding or modifying definitions as you see fit. Make sure to answer the problems in both Part 2 (Grammars and Abstract Syntax) and Part 3 (Type Checking). Submit your completed `hw2.ml` via Gradescope. As always, please don't hesitate to ask for help on Piazza (`https://piazza.com/class/l6qm516ecm36kb`).

2 Grammars and Abstract Syntax

The type `term` defined in `hw2.ml` corresponds to this grammar:

$$T ::= T \texttt{ until (} T \texttt{)} \ \mid\ \langle\text{string}\rangle \texttt{ = } \langle\text{int}\rangle \ \mid\ \langle\text{string}\rangle \texttt{ \^{} } \langle\text{int}\rangle$$

Every OCaml value of type `term` represents a term that can be built by applying the cases of this grammar. For instance, the term `x ^ 5` is represented as `Up("x", 5)` in OCaml.

Recall that this grammar is "just syntax"; you don't need to know what any of these operations mean in order to solve the problems.

1. (3 points) What term is represented by the following OCaml code?

   ```
   Until (Up ("x", 2), Until (Up ("y", 4), Eq ("y", 9)))
   ```

   Write your answer in the **comments** in the space provided; it will not be valid OCaml code, so it will break your file if you write it outside of comments.

2. (3 points) Suppose we wanted to extend the grammar with a case

   $$\texttt{for } \langle\text{string}\rangle \texttt{ from } \langle\text{int}\rangle \texttt{ do } T$$

   Extend the OCaml definition of `term` with a case that represents this syntax.

3. (4 points) Define an OCaml value `my_term : term` representing the following term:

   ```
   for i from 2 do i ^ 1 until i = 10
   ```

   You should represent the term directly in OCaml, **not** compute the result of running any operations.

3 Type Checking

The function `typecheck` contains the incomplete typechecker for expressions that we wrote in class.

4. (5 points) Add a case for `If`, according to the following typing rule:

   $$\frac{e : \text{bool} \qquad e_1 : \tau \qquad e_2 : \tau}{\text{if } e \text{ then } e_1 \text{ else } e_2 : \tau}$$

   Note that the types of e_1 and e_2 must match the type of the overall if expression. Once you have added this case, the examples in the code should return the indicated values. You may want to add some test cases of your own as well.

5. (for graduate students) Add a case for `Eq`, according to the following typing rule:

   $$\frac{e_1 : \tau \qquad e_2 : \tau}{e_1 = e_2 : \text{bool}}$$

   Then write at least one test case involving `Eq` that should typecheck, and at least one test case that should not typecheck.

   Hint: The type τ is either int or bool.