COMP 321 Fall 2021 Homework 1

For this homework, please hand in `hw01.sml` and `hw01-written.pdf` to your Google Drive handin folder.

1 Dynamic Typing

In Lecture 3, we discussed three approaches to handling situations where an operation is applied to the wrong kind of data, like $\mathsf{xor}(17, \mathsf{true})$. In this problem, you will finish the "dynamic type error" approach to these situations.

For this problem, we work with the following syntax:

$$e ::= \mathsf{num}(k) \mid e_1 + e_2 \mid \mathsf{true} \mid \mathsf{false} \mid e_1 \mathsf{\ xor\ } e_2 \mid \mathsf{iszero}(e) \mid \mathsf{error}$$

Here, $+$ is supposed to add numbers, xor is boolean exclusive or (true if one input is true and the other false, and false if both are true or both are false), and iszero is supposed to check if a number is 0 (and return true), or a non-zero number (and return false). The error term represents a dynamic type error.

In class, we discussed the following operational semantics rules

$$\frac{}{\mathsf{num}(k)\ \mathsf{done}} \qquad \frac{}{\mathsf{num}(k_1) + \mathsf{num}(k_2) \mapsto \mathsf{num}(k_1 + k_2)} \qquad \frac{e_1 \mapsto e_1'}{e_1 + e_2 \mapsto e_1' + e_2} \qquad \frac{e_2 \mapsto e_2'}{e_1 + e_2 \mapsto e_1 + e_2'}$$

$$\frac{}{\mathsf{true}\ \mathsf{done}} \qquad \frac{}{\mathsf{false}\ \mathsf{done}}$$

$$\frac{}{\mathsf{true\ xor\ true} \mapsto \mathsf{false}} \qquad \frac{}{\mathsf{false\ xor\ true} \mapsto \mathsf{true}} \qquad \frac{}{\mathsf{true\ xor\ false} \mapsto \mathsf{true}} \qquad \frac{}{\mathsf{false\ xor\ false} \mapsto \mathsf{false}}$$

$$\frac{e_1 \mapsto e_1'}{e_1 \mathsf{\ xor\ } e_2 \mapsto e_1' \mathsf{\ xor\ } e_2} \qquad \frac{e_2 \mapsto e_2'}{e_1 \mathsf{\ xor\ } e_2' \mapsto e_1 \mathsf{\ xor\ } e_2'}$$

Task 1 (10 points). Give operational semantics rules for iszero(e) (you can ignore the possibility of errors until the next task).

Task 2 (10 points). In class, we discussed the following (incomplete) operational semantics rules for dynamic type errors:

$$\frac{}{\mathsf{error}\ \mathsf{done}} \qquad \frac{}{\mathsf{num}(k) \mathsf{\ xor\ } e_2 \mapsto \mathsf{error}} \qquad \frac{}{\mathsf{error\ xor\ } e_2 \mapsto \mathsf{error}}$$

Give sufficient additional rules for errors: xor should error when given a non-boolean, + should error when given a non-number, and iszero should error when given a non-number, and there should be enough error propogation rules that progress is true.

Task 3 (10 points). Prove progress for all of the above rules (extending the proof discussed in class):

For all expressions e, either e done or there exists an e' such that $e \mapsto e'$.

Task 4 (10 points). Implement your progress proof as a step function `progress` in `hw01.sml`, inside the `Dynamic` module (extending the implementation given in Lecture 3). To test, you can run it interactively in SMLNJ, e.g.

```
use "hw01.sml"; open Dynamic;

- progress (Xor(IsZero(Plus(Num 0, Num 1)),False));
val it = Stepped (Xor (IsZero (Num 1),False)) : result

- progress (Xor (IsZero (Num 1),False));
val it = Stepped (Xor (False,False)) : result

- progress (Xor (IsZero (Num 1),False));
val it = Stepped (Xor(False,False)) : result

- progress (Xor (Num 17, True));
val it = Stepped Error : resul
```

For convenience, we have also provided a many-step function:

```
- stepUntilDone (Xor(IsZero(Plus(Num 0, Num 1)),False));
val it = False : exp
```

Note that `open` "opens" a module, bringing all of the variables defined in it into scope. This way, you can refer to `progress` rather than `Dynamic.progress`, etc.

2 Static Typing

In this section, we will instead address situations where an operation is applied to the wrong kind of input with a static (compile-time) type system. In class, we discussed the following typing rules:

$$\tau ::= \mathsf{int} \mid \mathsf{bool}$$

$$\frac{}{\mathsf{num}(k) : \mathsf{int}} \qquad \frac{e_1 : \mathsf{int} \quad e_2 : \mathsf{int}}{e_1 + e_2 : \mathsf{int}} \qquad \frac{}{\mathsf{true} : \mathsf{bool}} \qquad \frac{e_1 : \mathsf{bool} \quad e_2 : \mathsf{bool}}{\mathsf{false} : \mathsf{bool} \quad e_1 \ \mathsf{xor}\ e_2 : \mathsf{bool}} \qquad \frac{e : \mathsf{int}}{\mathsf{iszero}(e) : \mathsf{bool}}$$

Task 1 (15 points). Implement these rules as a function `typecheck` inside the `Static` module. The output of this function is the following:

```
datatype typOrError =
    WellTyped of typ
  | IllTyped of string
```

where `typecheck`(e) should return `WellTyped`(τ) iff $e : \tau$, and should return `IllTyped`(s) with some informative error message s otherwise.

Here are some example tests:

```
- use "hw01.sml"; open Static;

- typecheck (Xor(IsZero(Plus(Num 0, Num 1)),False));
val it = WellTyped Bool : typOrError

- typecheck (Xor(IsZero(Plus(Num 0, False)),False));
val it = IllTyped "second argument of Plus must be an int" : typOrError
```

Task 2 (5 points). Since the static type system will rule out errors before the program is run, the operational semantics is simpler: none of the error rules are necessary. Implement the operational semantics in a function `progress` inside the `Static` module; this version of progress can *assume* that its input is well-typed. **(You should not run the type checker from your `progress` function; you should assume that someone else has run it, and is giving your progress function, as input, an expression that has already passed the type checker.)** Copy your solution from Task 1.4 and delete the unnecessary parts.

Task 3 (10 points). Show the cases of the *type preservation theorem* pertaining to your operational semantics rules from Task 1.1 for iszero(e). Recall from Lecture 4 that type preservation says

> For all expressions e and types τ, if $e \mapsto e'$ and $e : \tau$ then $e' : \tau$.

and that the proof is by induction on the derivation of $e \mapsto e'$.