Exercise 2F-2

The issue is with the sentence Pick any arbitrary $x \in Y \cap Y'$. Suppose that $n = 1$, so X has size $|X| = n+1 = 2$ with elements $X = \{f, f'\}$. Then $Y = \{f'\}$ and $Y' = \{f\}$ and thus $Y \cap Y' = \emptyset$. So there is no way to pick an $x \in Y \cap Y'$.

Meta note: I feel betrayed by the hint. The problematic part didn't say "obviously" :(

Exercise 2F-3

We induct on the derivation sequence D.

Case 1: The last rule used in D was `while false`:

Then we have

$$D :: \frac{\langle b, \sigma \rangle \Downarrow \texttt{false}}{\langle \texttt{while } b \texttt{ do } x := x + 2, \sigma \rangle \Downarrow \sigma}$$

Note this is the base case. Essentially, we have that $\sigma' = \sigma$, and so $\sigma'(x) = \sigma(x)$. Thus, when $\sigma(x)$ is even, of course $\sigma'(x)$ will also be even.

Case 2: The last rule used in D was `while true`:

Then we have

$$D :: \frac{\langle b, \sigma \rangle \Downarrow \texttt{true} \quad \langle x := x + 2, \sigma \rangle \Downarrow \sigma' \quad D_1 :: \langle \texttt{while } b \texttt{ do } x := x + 2, \sigma' \rangle \Downarrow \sigma''}{\langle \texttt{while } b \texttt{ do } x := x + 2, \sigma \rangle \Downarrow \sigma''}$$

Note that $\sigma'(x) = \sigma(x) + 2$, so when $\sigma(x)$ is even, then $\sigma'(x)$ is even. By the inductive hypothesis on D_1, if $\sigma'(x)$ is even, then $\sigma''(x)$ is even. And so, we have that if $\sigma(x)$ is even, then $\sigma''(x)$ is even, as wanted. $\square$

Question assigned to the following page: 4

Peer Review ID: 305807804 — enter this when you fill out your peer evaluation via gradescope

Exercise 2F-4

For throw, we need only one rule:

$$\frac{\langle e, \sigma \rangle \Downarrow n}{\langle \texttt{throw}\ e, \sigma \rangle \Downarrow \sigma\ \texttt{exc}\ n}$$

We use the following rule for try catch when c_1 executes without an exception:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma'}{\langle \texttt{try}\ c_1\ \texttt{catch}\ x\ c_2, \sigma \rangle \Downarrow \sigma'}$$

We use the following rule for try catch when c_1 throws an exception:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma'\ \texttt{exc}\ n \quad \langle x := n; c_2, \sigma' \rangle \Downarrow t}{\langle \texttt{try}\ c_1\ \texttt{catch}\ x\ c_2, \sigma \rangle \Downarrow t}$$

We use the following rule for after finally when c_1 executes without an exception:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma' \quad \langle c_2, \sigma' \rangle \Downarrow t}{\langle \texttt{after}\ c_1\ \texttt{finally}\ c_2, \sigma \rangle \Downarrow t}$$

We use the following rule for after finally when c_1 throws an exception but c_2 executes without an exception:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma'\ \texttt{exc}\ n \quad \langle c_2, \sigma' \rangle \Downarrow \sigma''}{\langle \texttt{after}\ c_1\ \texttt{finally}\ c_2, \sigma \rangle \Downarrow \sigma''\ \texttt{exc}\ n}$$

We use the following rule for after finally when both c_1 and c_2 throw an exception:

$$\frac{\langle c_1, \sigma \rangle \Downarrow \sigma'\ \texttt{exc}\ n_1 \quad \langle c_2, \sigma' \rangle \Downarrow \sigma''\ \texttt{exc}\ n_2}{\langle \texttt{after}\ c_1\ \texttt{finally}\ c_2, \sigma \rangle \Downarrow \sigma''\ \texttt{exc}\ n_2}$$

Exercise 2F-5

I believe it is more natural to describe "IMP with exceptions" using contextual semantics. Exceptions are fundamentally a *local* phenomena - a single call to `throw` creates an exception and derails the remainder of the program, modulo any `try` or `finally` commands. Intuitively, contextual semantics proceeds through and evaluates the program "line by line"; when it hits an exception, it can halt immediately without considering the rest of the program. This is in contrast to operational semantics, which has more of a big-picture focus and does not capture the temporal nature of exceptions as naturally. Thus, I believe contextual semantics can more naturally capture the local nature of exceptions than operational semantics can.