2 Exercise 2F-2. Mathematical Induction

We are given the following faulty proof:

Proof: Let F be the set of all flowers and let $\mathsf{smells}(f)$ be the smell of the flower $f \in F$. (The range of smells is not so important, but we'll assume that it admits equality.) We'll also assume that F is countable. Let the property $P(n)$ mean that all subsets of F of size at most n contain flowers that smell the same.

$$P(n) \stackrel{\text{def}}{=} \forall X \in \mathcal{P}(F).\ |X| \leq n \implies (\forall f, f' \in X.\ \mathsf{smells}(f) = \mathsf{smells}(f'))$$

(the notation $|X|$ denotes the number of elements of X)

One way to formulate the statement to prove is $\forall n \geq 1.P(n)$. We'll prove this by induction on n, as follows:

Base Case: $n = 1$. Obviously all singleton sets of flowers contain flowers that smell the same (by the definition of $P(n)$).

Induction Step: Let n be arbitrary and assume that all subsets of F of size at most n contain flowers that smell the same. We will prove that the same thing holds for all subsets of size at most $n+1$. Pick an arbitrary set X such that $|X| = n+1$. Pick two distinct flowers $f, f' \in X$ and let's show that $\mathsf{smells}(f) = \mathsf{smells}(f')$. Let $Y = X - \{f\}$ and $Y' = X - \{f'\}$. Obviously Y and Y' are sets of size at most n so the induction hypothesis holds for both of them. Pick any arbitrary $x \in Y \cap Y'$. Obviously, $x \neq f$ and $x \neq f'$. We have that $\mathsf{smells}(f') = \mathsf{smells}(x)$ (from the induction hypothesis on Y) and $\mathsf{smells}(f) = \mathsf{smells}(x)$ (from the induction hypothesis on Y'). Hence $\mathsf{smells}(f) = \mathsf{smells}(f')$, which proves the inductive step, and the theorem.

Explanation of Error: In order for this inductive step to hold, it must work with any n $\geq$ 1(and the corresponding n+1 $\geq$ 2). In the case of $n = 1$, $|X| = 2$. If we pick a distinct $f, f' \in X$, then $Y = X - \{f\} = \{f'\}$ and $Y' = X - \{f'\} = \{f\}$, so $Y \cap Y'$ is empty, and thus there is no $x \in Y \cap Y'$ and the proof cannot continue. Because the proof fails at $n = 1$, it cannot be used to prove the inductive step down to the base case, and the proof is invalid.

3 Exercise 2F-3. While Induction

Base Cases: When b is false, the do statement is not executed, then $\sigma(x)$ remains unchanged, and is thus even.

$$\dfrac{\dfrac{\dfrac{\langle b, \sigma\rangle \Downarrow \mathit{false} \quad \langle \text{while } b \text{ do } x := x + 2, \sigma\rangle \Downarrow \sigma'}{\langle \text{skip}, \sigma\rangle \Downarrow \sigma'} \quad \dfrac{}{\langle \text{skip}, \sigma\rangle \Downarrow \sigma}}{\sigma == \sigma'} \quad \sigma(x) \text{ is even}}{\sigma'(x) \text{ is even}}$$

Inductive Case: When b is true, then x is incremented by two. Even numbers are closed under addition, so $x + 2$ is also even.

Questions assigned to the following page: 3 and 4

Peer Review ID: 305608827 — enter this when you fill out your peer evaluation via gradescope

Lemma 1:

$$\frac{\dfrac{\langle x := x+2, \sigma\rangle \Downarrow \sigma_1}{\sigma_1(x) = \sigma(x) + 2} \quad \sigma(x) \text{ is even}}{\sigma_1(x) \text{ is even}}$$

Final Proof:

$$\frac{\dfrac{\dfrac{\langle b, \sigma\rangle \Downarrow \mathit{true} \quad \langle \text{while } b \text{ do } x := x+2, \sigma\rangle \Downarrow \sigma'}{\langle x := x+2; \text{while } b \text{ do } x := x+2, \sigma\rangle \Downarrow \sigma'} \quad \langle x := x+2, \sigma\rangle \Downarrow \sigma_1}{\langle \text{while } b \text{ do } x := x+2, \sigma_1\rangle \Downarrow \sigma'} \text{ Lemma 1}}{\sigma_1(x) \text{ is even}}$$

After each iteration, b will either be true or false, thus by induction upon iterations, the final state of $\sigma'(x)$ will be even.

4 Exercise 2F-4. Language Features, Large-Step

Throw Command: The throw command adds an exception to the current state, with the edge case that if there is already an exception, it yields to the existing one. This design is implemented so the user knows the first exception derived among multiple, allowing them to pinpoint the source of an issue.

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

Try Command: There are two cases, if c_1 terminates without exception, the catch statement is skipped. If c_1 ends in exception, then x is assigned the exception value and c_2 is executed immediately after.

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

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma' \texttt{ exc } n \quad \langle x := n; c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \text{try } c_1 \text{ catch } x \ c_2, \sigma\rangle \Downarrow \sigma''}$$

Finally Command: There are three different cases. If c_1 terminates without exception, c_2 is ran and output is returned as usual. If c_1 has an exception, c_2 executes and the exception is carried, unless c_2 throws its own exception which will override it.

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

Questions assigned to the following page: 4 and 5

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

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

5 Exercise 2F-4. Language Features, Analysis

Small step semantics are more natural than large step semantics to describe exceptions, because they describe a sequence of execution one step at a time. Exceptions can be naturally thought of in terms of a sequence of execution, where the exception is carried in subsequent steps. This also holds for the try-catch and the after-finally clauses, where the continuation and catching/rethrowing behavior is more reasonable when considered in small steps instead of all at once.