1 2F-1 Bookkeeping

- **0 pts** Correct

Exercise 2F-2. Mathematical Induction [5 points]. Find the flaw in the following inductive proof that "All flowers smell the same". Please indicate exactly which sentences are wrong in the proof via highlighting or underlining.

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.

(One indication that the proof might be wrong is the large number of occurrences of the word "obviously" `:-)`)

The highlighed section is incorrect since the value x does not exist in the $n + 1 = 2$ case, so the inductive step does not apply for all values greater than 1. In the $n = 2$ case, $X = \{f, f'\}$, so $Y = X - \{f\} = \{f'\}$ and $Y' = X - \{f'\} = \{f\}$. $Y \cap Y' = \{f'\} \cap \{f\} = \{\}$. Since $Y = \{\}$, no x exists such that $x \in Y$. Thus, the rest of the highlighted section does not apply for the case of $n = 2$, since this x value does not exist. This proves this inductive step is incorrect, since it does not hold for all $n + 1 = 2$ case.

2 2F-2 Mathematical Induction

- **0 pts** Correct

Exercise 2F-3. While Induction [10 points]. Prove by induction the following statement about the operational semantics:

For any BExp b and any initial state σ such that $\sigma(x)$ is even, if

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

then $\sigma'(x)$ is even. Make sure you state what you induct on, what the base case is and what the inductive cases are. Show representative cases among the latter. Do not do a proof by mathematical induction!

To Prove: $\forall b \in \texttt{Bexp}.\langle \texttt{while}\ b\ \texttt{do}\ x := x+2, \sigma\rangle \Downarrow \sigma', \sigma'(x)\%2 = 0$ given that $\sigma(x)\%2 = 0$.

Pick an arbitrary σ such that $\sigma(x)\%2 = 0$ and $D :: \langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle$

Prove by structure of the derivation D

Case 1: Pick arbitrary b such that $D1 :: \langle b, \sigma\rangle \Downarrow \texttt{false}$

$$\text{D::}\ \frac{D1 :: \langle b, \sigma\rangle \Downarrow \texttt{false}}{\langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma}$$

It is given that $\sigma(x)$ is even, so because $\langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma$, then $\sigma'(x) = \sigma$ and the assertion holds that $\sigma'(x)$ is even.

This is a base case.

Case 2: Pick arbitrary b such that $D2 :: \langle b, \sigma\rangle \Downarrow \texttt{true}$

$$\text{D::}\ \frac{\begin{array}{c} D2 :: \langle b, \sigma\rangle \Downarrow \texttt{true} \quad \dfrac{\overline{\langle x, \sigma\rangle \Downarrow \sigma(x)}\ \overline{\langle 2, \sigma\rangle \Downarrow 2}}{\langle x := x + 2, \sigma \Downarrow \sigma[x := \sigma(x) + 2]} \\ D3 :: \langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma[x := \sigma(x) + 2]\rangle \downarrow \sigma' \end{array}}{\langle \texttt{while}\ b\ \texttt{do}\ x := x + 2, \sigma\rangle \Downarrow \sigma'}$$

By mathematical rules, $(\sigma(x) + 2)\%2 = 0$, meaning $\sigma[x := \sigma(x) + 2]$ fufills the requirement of the value of x being even. For brevity, we will refer to $\sigma[x := \sigma(x) + 2]$ as σ''

By inversion, $D3$ will follow the same rules as D.

If $\langle b, \sigma''\rangle \Downarrow \texttt{false}$, then we have reached Case 1. Because Case 1 has already been proven true, Case 2 is also true in this case. This is a basic inductive case.

If $\langle b, \sigma'' \rangle \Downarrow \texttt{true}$, then we have reached Case 2 again. Since a single step has proven to hold that $\sigma''(x)$ is even, by the inductive hypothesis, $\sigma'(x)$ is even. This is a base case.

Since both possible evaluated values for b in $D3$ have been considered and proven that $\sigma'(x)$ remains even in both cases, the assertion that $\sigma'(x)$ is even has been proven in Case 2.

Since the assertion has been proven in Case 1 and Case 2, the assertion has been proven for all possible evaluated values for $b \in BExp$ and the assertion has been proven true on the whole.

3 2F-3 While Induction

- **0 pts** Correct

Exercise 2F-4. Language Features, Large-Step [12 points]. We extend IMP with a notion of integer-valued *exceptions* (or *run-time errors*), as in Java, ML or C#. We introduce a new type T to represent command terminations, which can either be normal or exceptional (with an exception value $n \in \mathbb{Z}$).

$$\begin{array}{l}\mathsf{throw}\ e \\ \mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2 \\ \mathsf{after}\ c_1\ \mathsf{finally}\ c_2\end{array}$$

Give the large-step operational semantics inference rules (using our new judgment) for the three new commands presented here. You should present six (6) new rules total.

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

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

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exe}\ n \quad \langle c_2, \sigma'[x := n]\rangle \Downarrow t}{\langle \mathsf{try}\ c_1\ \mathsf{catch}\ x\ c_2, \sigma\rangle \Downarrow t}$$

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

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exe}\ e_1 \quad \langle c_2, \sigma'\rangle \Downarrow \sigma''}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ x\ c_2, \sigma\rangle \Downarrow \sigma''\ \mathsf{exe}\ e_1}$$

$$\frac{\langle c_1, \sigma\rangle \Downarrow \sigma'\ \mathsf{exe}\ e_1 \quad \langle c_2, \sigma'\rangle \Downarrow \sigma''\ \mathsf{exe}\ e_2}{\langle \mathsf{after}\ c_1\ \mathsf{finally}\ x\ c_2, \sigma\rangle \Downarrow \sigma''\ \mathsf{exe}\ e_2}$$

4 2F-4 Language Features, Large Step

- **0 pts** Correct

Exercise 2F-5. Language Features, Analysis [6 points]. Argue for or against the claim that it would be more natural to describe "IMP with exceptions" using small-step contextual semantics. You may use "simpler" or "more elegant" instead of "more natural" if you prefer. Do not exceed two paragraphs (one should be sufficient). Both your ideas and also the clarity with which they are expressed (i.e., your English prose) matter.

I agree with the assertion that it would be more natural to describe "IMP with exceptions" using small-step contextual semantics. This is due to the use of contexts in small-step contextual semantics. Much of the behavior of the execption handling changes based on in what context the current command is executing from. For example, in after finally , there is prescidence of an exception thrown in the finally block over the exception thrown in the after block. However, in large step semantics it may be less elegant to keep track of the separate execption statements as there is no specific ordering in large step, ordering is only differentiated by having separate variables. Having the concept of contexts can better model the different meanings for exceptions depending on what statement they appear in. Also, small step semantics has ordering of execution as part of the structure, so this can make the expression for order of exceptions easier to follow and more natural. This makes small-step contextual semantics more natural than large-step, and better conveys the different meanings of exceptions in different statements.

5 2F-5 Language Features, Analysis

- **0 pts** Correct