Problem 1 Operational Semantics: Postfix + \{sdup\}

Alyssa P. Hacker extended the PostFix language with a new command called sdup: smart dup. This allows us to compute $\text{square}(x) = x^2$ without hurting the termination property of PostFix programs. The informal semantics for sdup is as follows: duplicate the top of the stack if it is a number or a command sequence that doesn’t contain sdup; otherwise, report an error.

Formally, the operational semantics has been extended with the following two transition rules:

\[
\langle \text{sdup} . Q_{\text{rest}}, N . S \rangle \Rightarrow \langle Q_{\text{rest}}, N . N . S \rangle \quad [\text{sdup-numeral}]
\]

\[
\langle \text{sdup} . Q_{\text{rest}}, Q . S \rangle \Rightarrow \langle Q_{\text{rest}}, Q . Q . S \rangle \quad [\text{sdup-sequence}]
\]

where \( \text{contains}_{\text{sdup}} Q \)

\(\text{contains}_{\text{sdup}} : \text{Command}^* \rightarrow \text{Bool} \) is a helper function that takes a sequence of commands and checks whether it contains sdup or not (yes, \(\text{contains}_{\text{sdup}}\) handles even nested sequences of commands).

As a new graduate student in Alyssa’s AHRG (Advanced Hacking Research Group), you were assigned to give a proof that all PostFix + \{sdup\} programs terminate. However, you are not alone! Alyssa already took care of most of the mathematical weaponry:

Consider the product domain $P = \text{Nat} \times \text{Nat}$ (as usual, \(\text{Nat}\) is the set of natural numbers, starting with 0). On this domain, we define the relation $<_P$ as follows:

**Definition 1 (lexicographic order)** \(\langle a_1, b_1 \rangle <_P \langle a_2, b_2 \rangle \) if:

\[ a. \quad a_1 < a_2 \text{ or} \]
\[ b. \quad a_1 = a_2 \text{ and } b_1 < b_2. \]

E.g. \(\langle 3, 10000 \rangle <_P \langle 4, 0 \rangle, \langle 5, 2 \rangle <_P \langle 5, 3 \rangle\).

**Definition 2** A strictly decreasing chain in $P$ is a finite or infinite sequence of elements $p_1, p_2, \ldots$ such that $p_i \in P, \forall i$ and $p_{i+1} <_P p_i, \forall i$.

After a long struggle, Alyssa proved the following lemma for you:

**Lemma 1** There is no infinite strictly decreasing chain in $P$. 

Give a rigorous proof that each PostFix + \{sdup\} program terminates by using a cleverly defined energy function \( E_{\text{config}} \). Hint: Each transition of Postfix reduces the energy function \( E_{\text{config}} \) you saw in class. Try to see what is reduced by the two new rules, and how you can combine these two things into a single energy function.

Note: If you need to use some helper functions that are intuitively easy to describe but tedious to define (e.g. \text{contains}_{sdup}), just give an informal description of them.

**Problem 2 Operational and Denotational Semantics**

In this problem we ask you to develop both an operational and denotational semantics for the language EL. The grammar for EL programs is given on page 25 of the text.

a. Give a small-step operational semantics for EL. Remember, there are five parts to an operational semantics (see section 3.2 of the text).

Your Program domain should be the set \( \text{NumExp} \times \text{IntegerLiteral} \), that is, an EL expression paired with a value for the input variable. The Answer domain should be \( \text{IntegerLiteral} + \text{Error} \) (division by zero should cause an error in EL). Technically, you will have to define two rewriting relations for your semantics, one for NumExps and one for BoolExps.

You may take as given the functions \( \text{calculate} \), \( \text{logical} \), and \( \text{relational} \) which evaluate arithmetic, logical, and relational operators in the usual way, e.g.,

\[
(\text{calculate} \ + \ 3.2 \ 18.4) = 21.6,
(\text{logical} \ \text{or} \ \text{true} \ \text{false}) = \text{true},
(\text{relational} \ < \ 26 \ 4.1) = \text{false},
\]
e tc.

b. Define a denotational semantics for EL. You should give three meaning functions, \( \mathcal{P} \) for Programs, \( \mathcal{N} \) for NumExps, and \( \mathcal{B} \) for BoolExps. The meaning of a NumExp, \( \mathcal{N}[NE] \), should be a function from the value of the input variable to the final answer, and similarly, the meaning of a BoolExp, \( \mathcal{B}[BE] \), should be a function from the value of the input variable to a boolean answer.

You may take as given the meaning functions \( \mathcal{M} \), \( \mathcal{A} \), \( \mathcal{R} \), and \( \mathcal{L} \), for IntegerLiterals, ArithmeticOperators, RelationalOperators, and LogicalOperators respectively, except that you must clearly specify any error-related behavior. (\( \mathcal{A} \), \( \mathcal{R} \), and \( \mathcal{L} \) correspond to the functions \text{calculate}, \text{relational}, and \text{logical} from the operational semantics.)

You must provide the definition of all semantic domains, and the signatures of the meaning functions \( \mathcal{P} \), \( \mathcal{N} \), \( \mathcal{B} \), \( \mathcal{M} \), \( \mathcal{A} \), \( \mathcal{R} \), and \( \mathcal{L} \).