Consider the following program:

\[
P ≡ \begin{align*}
y &:= 1; \\
\text{while } 0 < x \text{ do} \\
(y &:= 2 \times y; \\
x &:= x - 1)
\end{align*}
\]

Program \(P\) can be written as \(P ≡ y := 1; w\) where

\[
w ≡ \text{while } 0 < x \text{ do} \ (y := 2 \times y; x := x - 1)
\]

**Exercise 1**

Use the ML implementation of the operational semantics to determine the values of \(x\) and \(y\) in the resulting state \(\sigma'\) after executing \(P\) in a state \(\sigma\) with \(\sigma(x) = 3\)? Include a printout of your ML run to show how you obtained the values.

Next show that the result of the ML run is correct by drawing by hand the complete derivation of \((P, \sigma) \rightarrow \sigma'\).

**Exercise 2**

Prove the following:

(a) If \(\sigma(x) = n \geq 0, \sigma(y) = m\) and \(\langle w, \sigma \rangle \rightarrow \sigma'\) then \(\sigma'(y) = m \times 2^n\).

(b) If \(\sigma(x) = n \geq 0\) and \(\langle P, \sigma \rangle \rightarrow \sigma'\) then \(\sigma'(y) = 2^n\).

**Exercise 3**

Prove that if a Boolean expression \(b \in \text{Bexp}\) contains no variables then for any states \(\sigma\) and \(\sigma'\) if \(\langle b, \sigma \rangle \rightarrow t\) and \(\langle b, \sigma' \rangle \rightarrow t'\) then \(t = t'\). (Hint: to save some work use the theorem proven in Handout 1)

Imagine introducing a catch construct to the Imp language. Command \(c\) catch \(h\) where \(c, h \in \text{Com}\) behaves as follows: command \(c\) is executed as usual. If the execution of \(c\) is interrupted by an error then the error handler \(h\) is invoked. Otherwise (no error has happened) the handler is ignored.

The only source of errors is a special error command which immediately breaks any execution reporting an error. If this happens inside a catch construct then an error handler will be called as described above. Otherwise no more actions are taken.

The extended syntax of commands is presented below

\[
c ::= \text{skip} | \text{error} | x := a | c_0 ; c_1 | \text{if } b \text{ then } c_0 \text{ else } c_1 \\
| c_0 \text{ catch } c_1 | \text{while } b \text{ do } c.
\]

**Exercise 4**

Give a complete operational semantic definition for executing commands of the Imp language extended with catch and error. You don’t have to repeat the rules for arithmetic and Boolean expressions.

Hint: you will need to modify the transition relation so that it does not simply return a new state but also information about whether an error occurred or not.
Exercise 5

Modify the ML implementation of the operational semantics to include the language extension above. Submit a printout of the ML file with your modifications marked using a pen. Include a printout of an ML run where you demonstrate the changes by executing some simple example programs of your own invention.