1 Semantic Domain - *Python*

Expressions' transitions rules:

$$s, b \vdash exp \Downarrow s', b', v$$

Commands' transition rule:

$$s, b \vdash C \Downarrow s', b'$$

Bindings:

$$(identifier \mapsto cell) \cup (identifier \mapsto closure).$$

Store:

$$cell \mapsto value$$

$$value = string\_literal \cup num\_literal \cup bool\_literal \cup list$$

2 Big Step Semantics for Python Programming Language

2.1 Expressions:

$$\textit{(access-list)}\ \frac{s, b \vdash exp1 \Downarrow s', b', vL \qquad s', b' \vdash exp2 \Downarrow s'', b'', vI}{s, b \vdash exp1[exp2] \Downarrow s''', b''', v} vL \in list$$

$$\textit{(string-add)}\ \frac{s, b \vdash exp1 \Downarrow s', b', v' \qquad s', b' \vdash exp2 \Downarrow s'', b'', v''}{s'', b'' \vdash exp1 + exp2 \Downarrow s''', b''', v'''} v''' = v'.v''$$

... Add all kinds of expressions for math domain: (add, subtract, division, exponentiation)
... Add all kinds of expressions for logic domain: (and, or, not, <, >, <=, >=, ==, !=, ternaryoperator)

2.2 Commands:

Assignments: Assignment of non-list element that is not present on the bindings:

$$\textit{(list-attrib-1)}\frac{s, b \vdash exp \Downarrow s', b', v}{s, b \vdash identifier = exp \Downarrow s', b'[identifier \mapsto v]} identifier \notin dom(b), v \notin list$$

Assignment of non-list element that is present on the bindings:

$$\textit{(list-attrib-2)}\frac{s, b \vdash exp \Downarrow s', b', v}{s, b \vdash identifier = exp \Downarrow s', b'[b(identifier) \mapsto v]} identifier \in dom(b), v \notin list$$

Assignment of list element that is not present on the bindings:

$$\textit{(list-attrib-3)}\frac{s, b \vdash exp \Downarrow s', b', v \qquad k := alloc(s', v)}{s, b \vdash identifier = exp \Downarrow s'[k \mapsto v], b'[identifier \mapsto k]} identifier \notin dom(b), v \in list$$

Assignment of list element that is present on the bindings:

$$\textit{(list-attrib-4)}\frac{s, b \vdash exp \Downarrow s', b', v}{s, b \vdash identifier = exp \Downarrow s'[b'(identifier) \mapsto v], b'} identifier \in dom(b), v \in list$$

List assignments:

$$\textit{(list-attrib)}\frac{\begin{array}{c} s, b \vdash exp1 \Downarrow s', b', vK \qquad s', b' \vdash exp2 \Downarrow s'', b'', vI \\ s'', b'' \vdash exp3 \Downarrow s''', b''', vV \qquad update(s'''(vK), vI, vV) = l \end{array}}{s, b \vdash exp1\,[exp2] = exp3 \Downarrow s'''[k \mapsto l], b'''}$$

Conditional Clauses:

$$\textit{(if-true)}\frac{s, b \vdash exp \Downarrow s', b', v' \qquad s', b' \vdash C1 \Downarrow s'', b''}{s, b \vdash if\, exp : C1\, else : C2 \Downarrow s'', b''} \qquad v' = True$$

$$\textit{(if-false)}\frac{s, b \vdash exp \Downarrow s', b', v' \qquad s', b' \vdash C2 \Downarrow s'', b''}{s, b \vdash if\, exp : C1\, else : C2 \Downarrow s'', b''} \qquad v' = False$$

For Loop

$$\textit{(for-base-case)}\frac{}{s, b, [\ ], i \vdash command \Downarrow s, b}$$

$$\textit{(for-induction)}\frac{s, b[i \mapsto head] \vdash command \Downarrow s', b' \qquad s', b', tail, i \vdash command \Downarrow s'', b''}{s, b, [head, tail], i \vdash command \Downarrow s'', b''}$$

$$\textit{(for-execution)}\frac{s, b \vdash exp \Downarrow s', b', [head, tail] \qquad \textit{(for-induction)}}{s, b \vdash for\ i\ in\ exp :\ command \Downarrow s'', b''}$$

Function Definition

$$\textit{(func-def)}\frac{}{s, b \vdash def\ identifier(listParamFormais)\ command \Downarrow s, b'[identifier \mapsto (listParamFormais, command)]}$$

New Rule for commands (update previous definitions):

$$s, b \vdash command \Downarrow s', b', t, v, z$$

Where

$$t = Returnparam$$

$$v = ReturnedValue$$

$$z = ExitLoopParam$$

Loop exitwhen(exp)

$$\textit{(loop-1)}\frac{s, b \vdash command \Downarrow s', b', t, v, true}{s, b \vdash loop : command \Downarrow s', b', t, v, true}$$

$$\textit{(loop-2)}\frac{s, b \vdash command \Downarrow s', b', false, v, false \qquad s', b' \vdash loop : command \Downarrow s', b', false, v, true}{s, b \vdash loop : command \Downarrow s', b', false, v, true}$$

$$\textit{(exitwhen-false)}\frac{s, b \vdash exp \Downarrow s', b', false}{s, b \vdash exitwhen(exp) \Downarrow s', b', false, None, false}$$

$$\textit{(exitwhen-true)}\frac{s, b \vdash exp \Downarrow s', b', true}{s, b \vdash exitwhen(exp) \Downarrow s', b', false, None, true}$$

Function call + Return

$$\textit{(return)}\frac{s, b \vdash exp \Downarrow s', b', v}{s, b \vdash return(exp) \Downarrow s', b', true, v, false}$$

$$\textit{(functioncall)}\frac{\begin{array}{c} s, b[P \mapsto (\overline{x}, command)] \vdash \overline{e} \Downarrow s', b', \overline{v} \\ s', b'[P \mapsto (\overline{x}, command), Xi \mapsto Vi] \vdash command \Downarrow s'', b'', t, v', z \end{array} \qquad b''' = b'' - \{Xi \mapsto Vi\}}{s, b[P \mapsto (\overline{x}, command)] \vdash P(\overline{e}) \Downarrow s'', b''', false, v', false}$$

Where

$$\overline{x} = FormalParametersTuple$$

$$\overline{e} = RealParameterstuple$$

$$\overline{v} = Tupleofevaluatedvalues$$

$$P = FunctionIdentifier$$

New Rule for commands (update previous definitions):

$$s, b \vdash command \Downarrow s', b', t, v, z, w, q$$

Where

$$t = Returnparam$$
$$v = ReturnedValue$$
$$z = ExitLoopParam$$
$$w = ExceptionParam$$
$$q = ExceptionType$$

Try - Except: No exception

$$\textit{(noException)} \quad \frac{\begin{array}{c} s, b \vdash addBind(idCommandList) \Downarrow s, b'[ExcepI \mapsto Ci], False, None, False, False, None \\ s, b'[ExcepI \mapsto Ci] \vdash command \Downarrow s', b''[ExcepI \mapsto Ci], t, v, z, false, None \\ s', b''[ExcepI \mapsto Ci \vdash removeBind(idCommandList) \Downarrow s', b''', False, None, False, False, None \end{array}}{s, b \vdash try : command - except : idCommandList \Downarrow s', b''', t, v, z, False, None}$$

Try - Except

$$\textit{(tryExcept)} \quad \frac{\begin{array}{c} s, b \vdash addBind(idCommandList) \Downarrow s, b'[ExcepI \mapsto Ci], False, None, False, False, None \\ s, b'[ExcepI \mapsto Ci] \vdash command \Downarrow s', b''[ExcepI \mapsto Ci], t, v, z, True, ExceptionA \\ s', b''[ExcepI \mapsto Ci] \vdash handleException(ExceptionA) \Downarrow s', b''[ExcepI \mapsto Ci], t', v', z', w', q' \\ s', b''[ExcepI \mapsto Ci] \vdash removeBind(idCommandList) \Downarrow s', b''', False, None, False, False, None \end{array}}{s, b \vdash try : command - except : idCommandList \Downarrow s', b''', t', v', z', w', q'}$$

$$\textit{(Recover)} \frac{s, b[ExcepI \mapsto Ci] \vdash if(ExceptionA \in \{ExceptionStack\})Ca \Downarrow s', b', t, v, z, w, q}{s, b[ExcepI \mapsto Ci] \vdash handleException(ExceptionA) \Downarrow s', b', t, v, z, w, q}$$

$$\textit{(Kill)} \frac{s, b[ExcepI \mapsto Ci] \vdash if(ExceptionA \notin \{ExceptionStack\})kill \Downarrow s, b, False, None, False, True, SIGKILL}{s, b[ExcepI \mapsto Ci] \vdash handleException(ExceptionA) \Downarrow s, b, False, True, SIGKILL}$$