TinyImp

A program that's well behaved but not ok?

```
var x ·
if 1 then
   x := 3161
else
   skip ⇝ ∅
fi;
x := x + 1
```

We can see that the else branch never runs, but static semantics can't!

Due to the skip, this definitely writes to nothing. Thus, we lose knowledge of x being initialised!

A useful interpretation of this is that no variables are used out of scope.

Suppose we know that $\emptyset; \emptyset \vdash s \text{ ok} \rightsquigarrow \emptyset$. What does this mean within a

- Crash-and-burn semantics? Is neve going to crash due to a variable access.

  Never accesses uninit. variables

- Default value semantics? (Hint: deterministic?) The execution of any program entails deteministic evaluation.

  (every identical execution evaluates to the same state)

- Junk data semantics? Same as above.

  Programs that are ok don't use uninit. variables

How could we formalise these properties (i.e. in logic, with judgments)?

"No variable in s is used out of scope" $\equiv$ $FV(s) =$ ___

Alternatively we could say that "the result of evaluation doesn't depend on any of the state's free variables"

$\equiv \forall \sigma, \sigma', x.\ (\sigma, s) \Downarrow \sigma' \Rightarrow$ ___

"Deterministic evaluation"

$\equiv \forall \sigma_1, \sigma_2, \sigma_3.$ ___ $\Rightarrow$ ___

"Never crashes due to an uninit. variable" $\stackrel{?}{\equiv} \forall \sigma.\ \exists \sigma'.\ (\sigma, s) \Downarrow \sigma'$

Reasonable? Any problems with this? (Hint: what about non-termination?)

Does it matter if sequential composition is left or right associative?

CLAIM $(\sigma_1, (s_1; s_2); s_3) \Downarrow \sigma_2$ if and only if $(\sigma_1, s_1; (s_2; s_3)) \Downarrow \sigma_2$

PROOF (of the if direction. Opposite direction: exercise!)

Suppose that $(\sigma_1, (s_1; s_2); s_3) \Downarrow \sigma_2$. Then

$$\dfrac{\dfrac{\dfrac{\vdots}{(\sigma_1, s_1) \Downarrow \sigma''} \quad \dfrac{\vdots}{(\sigma'', s_2) \Downarrow \sigma'}}{(\sigma_1, s_1; s_2) \Downarrow \sigma'} \qquad \dfrac{\vdots}{(\sigma', s_3) \Downarrow \sigma_2}}{(\sigma_1, (s_1; s_2); s_3) \Downarrow \sigma_2}$$

Now, we can derive the evaluation of the right-associative variant:

$$\dfrac{\dfrac{\checkmark}{(\sigma_1, s_1) \Downarrow \sigma''} \qquad \dfrac{\dfrac{\checkmark}{(\sigma'', s_2) \Downarrow \sigma'} \quad \dfrac{\checkmark}{(\sigma', s_3) \Downarrow \sigma_2}}{(\sigma'', s_2; s_3) \Downarrow \sigma_2}}{(\sigma_1, s_1; (s_2, s_3)) \Downarrow \sigma_2}$$

do s until e

Big-step semantics for this loop:

$$\frac{}{(\sigma_1, \text{do } s \text{ until } e) \Downarrow \sigma_2} \text{ Iteration}$$

$$\frac{}{(\sigma_1, \text{do } s \text{ until } e) \Downarrow \sigma_2} \text{ Termination}$$

do s until e $\equiv$

Derive

We will assume this

$$\frac{\{\varphi\}\ s\ \{\varphi\}}{\{\varphi\}\ \text{do } s \text{ until } e\ \{\varphi \wedge e\}}\ \text{DoUntil}$$

★ Need to use the rule of consequence.

$$\cfrac{\{\varphi\}\ s\ \{\varphi\} \qquad \cfrac{\cfrac{\overline{}\quad\overline{}\quad\overline{}}{}\,\star \qquad \overline{}}{\{\varphi\}\ \text{while } \neg e \text{ do } s \text{ od } \{\varphi \wedge e\}}\,\star}{\{\varphi\}\ s;\ \text{while } \neg e \text{ do } s \text{ od } \{\varphi \wedge e\}}$$

Loop

$$\frac{\{\varphi \wedge e\}\ s\ \{\varphi\}}{\{\varphi\}\ \texttt{while}\ e\ \texttt{do}\ s\ \texttt{od}\ \{\varphi \wedge \neg e\}}$$

Sequential composition

$$\frac{\{\varphi\}\ s_1\ \{\alpha\} \qquad \{\alpha\}\ s_2\ \{\psi\}}{\{\varphi\}\ s_1; s_2\ \{\psi\}}$$

$$\frac{\varphi \Rightarrow \alpha \qquad \{\alpha\}\ s\ \{\beta\} \qquad \beta \Rightarrow \psi}{\{\varphi\}\ s\ \{\psi\}}$$

Rule of consequence! Useful, but tricky to use.