1 Structural Control Flow

We add a few structural control flow constructs to the language:

$$\begin{aligned} \mathcal{S} \quad += \quad & \text{if } \mathcal{E} \text{ then } \mathcal{S} \text{ else } \mathcal{S} \\ & \text{while } \mathcal{E} \text{ do } \mathcal{S} \end{aligned}$$

The big-step operational semantics is straightforward and is shown on Fig. 1.

In the concrete syntax for the constructs we add the closing keywords " fi " and "od" as follows:

$$\begin{aligned} & \text{if } e \text{ then } s_1 \text{ else } s_2 \text{ fi} \\ & \text{while } e \text{ do } s \text{ od} \end{aligned}$$

$$\frac{\sigma \underset{\mathcal{E}}{\Longrightarrow} n \neq 0 \qquad \langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{S_1}{\Longrightarrow}} c'}{\langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{\text{if } e \text{ then } S_1 \text{ else } S_2}{\Longrightarrow}} c'} \quad [\text{If-True}]$$

$$\frac{\sigma \underset{\mathcal{E}}{\overset{e}{\Longrightarrow}} 0 \qquad \langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{S_2}{\Longrightarrow}} c'}{\langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{\text{if } e \text{ then } S_1 \text{ else } S_2}{\Longrightarrow}} c'} \quad [\text{If-False}]$$

$$\frac{\sigma \underset{\mathcal{E}}{\overset{e}{\Longrightarrow}} n \neq 0 \qquad \langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{S}{\Longrightarrow}} c' \qquad c' \underset{\mathcal{S}}{\overset{\text{while } e \text{ do } S}{\Longrightarrow}} c''}{\langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{\text{while } e \text{ do } S}{\Longrightarrow}} c''} \quad [\text{While-True}]$$

$$\frac{\sigma \underset{\mathcal{E}}{\overset{e}{\Longrightarrow}} 0}{\langle \sigma, w \rangle \underset{\mathcal{S}}{\overset{\text{while } e \text{ do } S}{\Longrightarrow}} \langle \sigma, w \rangle} \quad [\text{While-False}]$$

Figure 1: Big-step operational semantics for control flow statements

2 Syntax Extensions

With the structural control flow constructs already implemented, it is rather simple to "saturate" the language with more elaborated control constructs, using the method of syntactic extension. Namely, we may introduce the following constructs

```
  if e1 then s1
elif e2 then s2
...
elif ek then sk
[ else sk+1 ]
  fi
```

and

```
for s1, e, s2 do s3 od
```

only at the syntactic level, directly parsing these constructs into the original abstract syntax tree, using the following conversions:

```
  if e1 then s1                    if e1 then s1
elif e2 then s2                  else if e2 then s2
...                                ...
elif ek then sk        ⇝         else if ek then sk
else sk+1                        else sk+1
fi                                    fi
                                      ...
                                      fi
```

```
  if e1 then s1                    if e1 then s1
elif e2 then s2                  else if e2 then s2
...                                ...
elif ek then sk        ⇝         else if ek then sk
fi                               else skip
                                      fi
                                      ...
                                      fi
```

```
for s1, e, s2 do s3 od     ⇝     s1;
                                 while e do
                                   s3;
                                   s2
                                 od
```

The advantage of syntax extension method is that it makes it possible to add certain constructs with almost zero cost — indeed, no steps have to be made in order to implement the extended constructs (besides parsing). Note, the semantics of extended constructs is provided for free as well (which is not always desirable). Another potential problem with syntax extensions is that

they can easily provide unreasonable results. For example, one may be tempted to implement a post-condition loop using syntax extension:

```
repeat s until e   ⇝   s;
                       while e == 0 do
                         s
                       od
```

However, for nested repeat constructs the size of extended program is exponential w.r.t. the nesting depth, which makes the whole idea unreasonable.