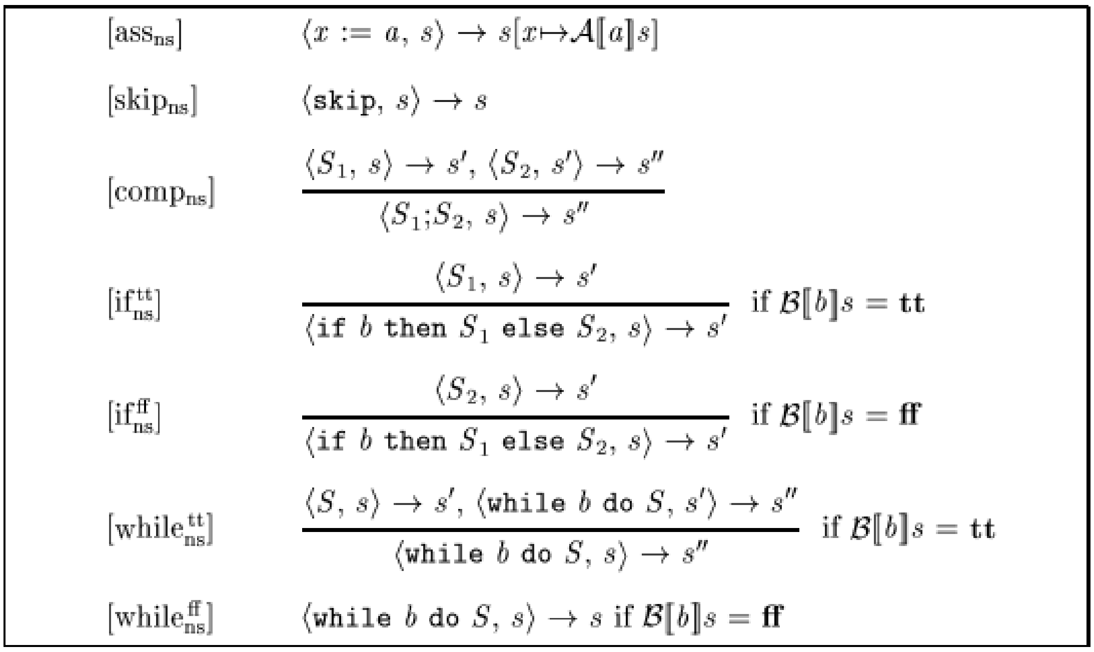
CSC 2125
Homework Operational Semantics



$[\text{ass}_{\text{ns}}]$	$\langle x := a,\ s\rangle \to s[x \mapsto \mathcal{A}[\![a]\!]s]$
$[\text{skip}_{\text{ns}}]$	$\langle \texttt{skip},\ s\rangle \to s$
$[\text{comp}_{\text{ns}}]$	$\dfrac{\langle S_1,\ s\rangle \to s',\ \langle S_2,\ s'\rangle \to s''}{\langle S_1;S_2,\ s\rangle \to s''}$
$[\text{if}^{\text{tt}}_{\text{ns}}]$	$\dfrac{\langle S_1,\ s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \to s'}$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{if}^{\text{ff}}_{\text{ns}}]$	$\dfrac{\langle S_2,\ s\rangle \to s'}{\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \to s'}$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$
$[\text{while}^{\text{tt}}_{\text{ns}}]$	$\dfrac{\langle S,\ s\rangle \to s',\ \langle \texttt{while}\ b\ \texttt{do}\ S,\ s'\rangle \to s''}{\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s''}$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{while}^{\text{ff}}_{\text{ns}}]$	$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \to s$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$

Table 2.1: Natural semantics for **While**

1. Consider following statement
   repeat S until b

   a. Extend the natural operational ("big-step") semantics of the WHILE language (Table 2.1 from [1]) by a rule for relation → for the **`repeat`**-construct. (The semantics for the repeat-construct should not rely on the existence of a while-construct)

The definition is analogous to the definition of the while construct. We need two separate rules for the cases b is true and b is false.

$$\left[repeat^{tt}_{ns}\right] \quad \langle S,s\rangle \to s' \quad \text{iff}\,\mathcal{B}[\![b]\!] = tt$$

$$\left[repeat^{ff}_{ns}\right] \quad \frac{\langle S,s\rangle \to s' \quad \langle repeat\ S\ until\ b,\ s'\rangle \to s''}{\langle repeat\ S\ until\ b,\ s\rangle \to s''} \quad \text{iff}\,\mathcal{B}[\![b]\!] = ff$$

b. Two statements in a natural semantic are considered equivalent if for all states s and s':

   $\langle S_1, s\rangle \to s' \ iff \ \langle S_2, s\rangle \to s'$

   How can you show that the repeat construct is semantically equivalent to **`S; while ¬b do S`**.

   Why does this lead to the conclusion that the extended semantics is deterministic?

We can give a proof by induction over the number of applied rules for the derivation tree (called induction on the shape of the derivation tree in [1]).
For all constructs in the WHILE language, you find the complete solution in [1], pp.26f.
Here we consider only the extension, namely **`repeat S until b`**.

Two statements S$_1$ and S$_2$ are considered semantically equivalent iff:
$\langle S_1, s\rangle \to s' \ iff \ \langle S_2, s\rangle \to s'$

$\langle repeat\ S\ until\ b,\ s\rangle \to s'$ (1)
iff
$\langle S; while\ \neg b\ do\ \ S,\ s\rangle \to s'$ (2)

Here we show only that each step preserves the property:
Base case: For one application of the repeat-rules for the derivation tree, we can either apply $\left[repeat_{ns}^{tt}\right]$ or $\left[repeat_{ns}^{ff}\right]$ depending on $\mathcal{B}[\![b]\!]$.

We get

$$\langle S, s\rangle \to s' \quad \text{iff} \mathcal{B}[\![b]\!] = tt$$

$$\frac{\langle S, s\rangle \to s' \quad \langle repeat\ S\ until\ b,\ s'\rangle \to s''}{\langle repeat\ S\ until\ b,\ s\rangle \to s''} \quad \text{iff} \mathcal{B}[\![b]\!] = ff$$

For the program **`S; while ¬b do S`**, we first apply the composition rule $\left[comp_{ns}\right]$, and afterwards we can apply either $\left[while_{ns}^{ff}\right]$ or $\left[while_{ns}^{tt}\right]$ to get a derivation tree with depth one bigger than the corresponding repeat-derivation.

$$\frac{\langle S, s\rangle \to s' \quad \langle while\ \neg b\ do\ S,\ s'\rangle \to s'}{\langle S; while\ \neg b\ do\ S,\ s\rangle \to s'} \quad \text{iff} \mathcal{B}[\![\neg b]\!] = ff$$

$$\frac{\langle S, s\rangle \to s' \quad \dfrac{\langle S, s'\rangle \to s'' \quad \langle while\ \neg b\ do\ S,\ s''\rangle \to s''}{\langle while\ \neg b\ do\ S,\ s'\rangle \to s''}}{\langle S; while\ \neg b\ do\ S,\ s\rangle \to s''} \quad \text{iff} \mathcal{B}[\![\neg b]\!] = tt$$

For $\mathcal{B}[\![b]\!] = tt$ respectively $\mathcal{B}[\![\neg b]\!] = ff$ we get

$$\langle repeat\ S\ until\ b,\ s\rangle \rightarrow s' \ \Leftrightarrow\ \frac{\langle S,s\rangle \rightarrow s' \quad \langle while\ \neg b\ do\ S,\ s'\rangle \rightarrow s'}{\langle S; while\ \neg b\ do\ S,\ s\rangle \rightarrow s'}$$

And for $\mathcal{B}[\![b]\!] = ff$ respectively $\mathcal{B}[\![\neg b]\!] = tt$ we get

$$\frac{\langle S,s\rangle \rightarrow s' \quad \langle repeat\ S\ until\ b,\ s'\rangle \rightarrow s''}{\langle repeat\ S\ until\ b,\ s\rangle \rightarrow s''}$$

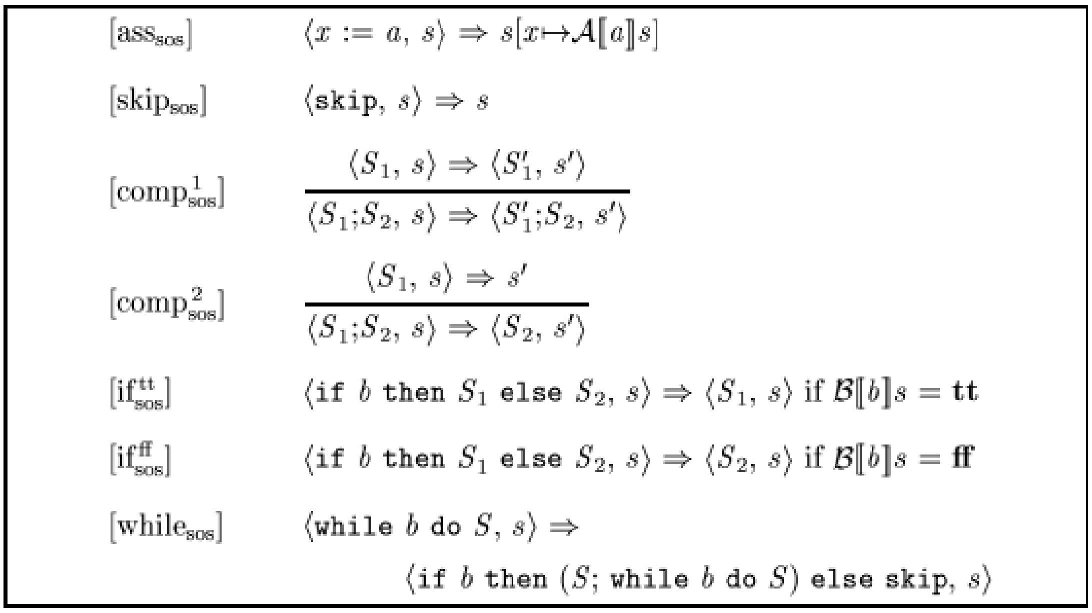
$\Leftrightarrow$

$$\frac{\langle S,s\rangle \rightarrow s' \quad \dfrac{\langle S,s'\rangle \rightarrow s'' \quad \langle while\ \neg b\ do\ S,\ s''\rangle \rightarrow s''}{\langle while\ \neg b\ do\ S,\ s'\rangle \rightarrow s''}}{\langle S; while\ \neg b\ do\ S,\ s\rangle \rightarrow s''}$$

So in both cases
$\langle repeat\ S\ until\ b,\ s\rangle \rightarrow s' \ \Leftrightarrow \langle S; while\ \neg b\ do\ \ S,\ s\rangle \rightarrow s'$

The induction step for compositional trees works is now trivial, we assume that for n applications of the repeat-rules, we can use n-applications of the while-rule plus the additional rule for the composition to gain the same result.

[1] shows that the natural semantics of the WHILE language as given in Table 2.1. is deterministic. Since we just proved that we can express the repeat-construct in the WHILE language by `S; while ¬b do S`, the extended version of the natural semantics preserves its deterministic nature.



$[\text{ass}_{\text{sos}}]$	$\langle x := a,\ s\rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$
$[\text{skip}_{\text{sos}}]$	$\langle \texttt{skip},\ s\rangle \Rightarrow s$
$[\text{comp}^1_{\text{sos}}]$	$\dfrac{\langle S_1,\ s\rangle \Rightarrow \langle S_1',\ s'\rangle}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_1';S_2,\ s'\rangle}$
$[\text{comp}^2_{\text{sos}}]$	$\dfrac{\langle S_1,\ s\rangle \Rightarrow s'}{\langle S_1;S_2,\ s\rangle \Rightarrow \langle S_2,\ s'\rangle}$
$[\text{if}^{\text{tt}}_{\text{sos}}]$	$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \Rightarrow \langle S_1,\ s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{tt}$
$[\text{if}^{\text{ff}}_{\text{sos}}]$	$\langle \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2,\ s\rangle \Rightarrow \langle S_2,\ s\rangle$ if $\mathcal{B}[\![b]\!]s = \mathbf{ff}$
$[\text{while}_{\text{sos}}]$	$\langle \texttt{while}\ b\ \texttt{do}\ S,\ s\rangle \Rightarrow$ $\langle \texttt{if}\ b\ \texttt{then}\ (S;\ \texttt{while}\ b\ \texttt{do}\ S)\ \texttt{else skip},\ s\rangle$

Table 2.2: Structural operational semantics for **While**

2. Consider following statement
   `repeat S until b`

   a. Define the structural operational ("small-step") semantics as in Table 2.2 from [1] for the `repeat`-construct. (The semantics for the repeat-construct should not rely on the existence of a while-construct)

Analogous to the while-construct we define:

$$[\textit{repeat}_{sos}] \quad \langle \textit{repeat } S \textit{ until } b,\ s\rangle \Rightarrow \langle S;\ \text{if } b \text{ then } \textit{skip} \text{ else } (\textit{repeat } S \textit{ until } b),\ s\rangle$$

Notice that we do not need two rules here as in the natural semantics. We define the loop recursively with a semantic construct of "if then else".

   b. How must the notion of semantic equivalence be defined for structural operational semantics?

Two statements S_1 and S_2 are equivalent if starting from a Statement S_1 in a state s there is a derivation sequence to configuration γ iff starting from S_2 in the same state s there is some derivation sequence (not necessary the same!) leading to configuration γ.

$$\langle S_1,\ s\rangle \Rightarrow^* \gamma \text{ iff } \langle S_2,\ s\rangle \Rightarrow^* \gamma$$

$\Rightarrow^*$ denotes a derivation sequence ("path"), that can be possible infinite.
γ is a resulting configuration, which is either terminal or "stuck".

3. What distinguishes the two notions of semantic equivalence in 1) and 2)?

The small-step semantics captures the intermediate steps of a loop. So, we would obtain an infinite derivation sequence for e.g.

```
x:=0; repeat (x:=x+1) until (false)
```

where always the else branch is followed.

$$\langle \mathit{repeat}\,(x := x+1)\ \mathit{until}\ (\mathit{false}),\ s\rangle \Rightarrow$$

$$\langle \mathit{repeat}\,(x := x+1)\ \mathit{until}\ (\mathit{false});\ \text{if}\ b\ \text{then}\ \mathit{skip}\ \mathit{else}\ (\mathit{repeat}\,(x := x+1)\ \mathit{until}\ (\mathit{false})),\ s\rangle$$

However, we can write down this derivation sequence up to n iterations. At each iteration we have an intermediate state that executes `(x:=x+1)` and results in a new state. That way, we could compare even non-terminating programs in respect to semantic equivalence. We just need to show that from some point on, the endless loop is equivalent. Since in the example non-termination can only occur in conditionals and these are all represented by “if then else”, the notion of semantic equivalence reduced to the equivalent conditionals.

For the big-step semantics, it is not possible to describe infinite looping. The derivation tree is not complete without termination. The proof of the root of the tree relies on termination. Furthermore, we can not discuss semantic equivalence for non-terminating programs, since they might use completely different derivation trees (see exercise 1b).

[1] Nielson, H., Nielson, F.: “Semantics with Applications: A Formal Introduction”, Wiley Professional Computing, 1992.