Syntax and Semantics
Exercise Session 12

Exercise 1.
Given S_1 and S_2 defined as follows

$$\texttt{x} := 0;\ \texttt{while}\ (\texttt{x} \geq 0)\ \texttt{do}\ \texttt{x} := \texttt{x} + 1 \tag{S_1}$$

$$\texttt{x} := 3;\ \texttt{y} := 4;\ \texttt{while}\ (\texttt{x} \leq \texttt{y})\ \texttt{do}\ (\texttt{x} := 2 * \texttt{x};\ \texttt{y} := 3 * \texttt{y}) \tag{S_2}$$

check whether S_1 and S_2 are semantically equivalent in the big-step semantics, that is $S_1 \sim_{bs} S_2$. Motivate your answer.

Solution 1.
By definition, $S_1 \sim_{bs} S_2$ if for all $s, s' \in \mathbf{States}$ we have

$$\langle S_1, s\rangle \to s' \text{ if and only if } \langle S_2, s\rangle \to s'\,.$$

We will prove that the above holds by showing that, for any $s, s' \in \mathbf{States}$, $\langle S_1, s\rangle \not\to s'$ and $\langle S_2, s\rangle \not\to s'$.

To show that $\langle S_2, s\rangle \not\to s'$ it suffices to show that there is no finite proof tree for $\langle S_2, s\rangle \to s'$, for any $s, s' \in \mathbf{States}$. To this end, notice that for $s'' \in \mathbf{States}$ such that $0 \leq s''(x) \leq s''(y)$ we have that, for any $s' \in \mathbf{States}$,

$$\langle \texttt{while}\ (\texttt{x} \leq \texttt{y})\ \texttt{do}\ (\texttt{x} := 2 * \texttt{x};\ \texttt{y} := 3 * \texttt{y}), s''\rangle \not\to s'\,,$$

since by $s''(x) \leq s''(y)$ only rule [while-$\top$] can be applied, and after the execution of the body of the while-loop we obtain the state $s''' = s''[x \mapsto 2s''(x), y \mapsto 3s''(y)]$ for which it holds again $0 \leq s'''(x) \leq s'''(y)$. Moreover, note that $\langle S_2, s\rangle \to s'$ requires us to show that

$$\frac{\langle \texttt{x} := 3;\ \texttt{y} := 4, s\rangle \to s'' \quad \langle \texttt{while}\ (\texttt{x} \leq \texttt{y})\ \texttt{do}\ (\texttt{x} := 2 * \texttt{x};\ \texttt{y} := 3 * \texttt{y}), s''\rangle \to s'}{\langle \texttt{x} := 3;\ \texttt{y} := 4;\ \texttt{while}\ (\texttt{x} \leq \texttt{y})\ \texttt{do}\ (\texttt{x} := 2 * \texttt{x};\ \texttt{y} := 3 * \texttt{y}), s\rangle \to s'}$$

where $s'' = s[x \mapsto 3, y \mapsto 4]$. By what we have said before, however, we have that the second premise of the rule cannot be proven, therefore

$$\langle \mathtt{x} := \mathtt{3};\ \mathtt{y} := \mathtt{4};\ \mathtt{while}\ (\mathtt{x} \leq \mathtt{y})\ \mathtt{do}\ (\mathtt{x} := \mathtt{2} * \mathtt{x};\ \mathtt{y} := \mathtt{3} * \mathtt{y}), s\rangle \not\to s'$$

Analogously, one can prove that $\langle S_1, s\rangle \not\to s'$, for all $s, s' \in$ **States**. $\square$

Exercise 2.
Given S_3 and S_4 defined as follows

$$\mathtt{y} := \mathtt{n_1};\ \mathtt{for}\ \mathtt{x} := \mathtt{1}\ \mathtt{to}\ \mathtt{n_2}\ \mathtt{do}\ \mathtt{y} := \mathtt{y} + \mathtt{1} \qquad (S_3)$$

$$\mathtt{y} := \mathtt{n_1} + \mathtt{n_2} * (\mathtt{n_2} + \mathtt{1})/\mathtt{2};\ \mathtt{x} := \mathtt{n_2} + \mathtt{1} \qquad (S_4)$$

check whether S_3 and S_4 are semantically equivalent in the big-step semantics, that is $S_3 \sim_{bs} S_4$, for all $\mathtt{n_1}, \mathtt{n_2} \in \mathbb{N}$. Motivate your answer.

Solution 2.
S_3 and S_4 are not semantically equivalent. Take for instance $n_1 = \mathtt{50}$ and $n_2 = 2$. One can show that $\langle S_3, s\rangle \to s[x \mapsto 3,\ y \mapsto 52]$ whereas $\langle S_4, s\rangle \to s[x \mapsto 3,\ y \mapsto 53]$ (the proof trees for these derivations are left as exercise). This counterexample solves the exercise. (For those who are curious, S_3 would be equivalent to S_4 if $y := y + 1$ would be replaced by $y := y + x$.)

Exercise 3.
Consider the following extension of **Bims** which adds the following formation rule to those of **Stm**, for $m > 0$,

$$S ::= \cdots \mid \mathtt{foreach}\ x\ \mathtt{in}\ [n_1, \ldots, n_m]\ \mathtt{do}\ S\ .$$

Intuitively, the above construct executes the body S m-times, and at each execution of the body S, the value of the variable x is set to v_i, the value of the numeral n_i, for $i = 1 \ldots m$. At the end of the execution of the `foreach` construct, x assumes the value v_m of the numeral n_m. Give both the big-step and the small-step semantics that formalize the above description.

Solution 3.
The big-step semantics is given by the following two rules

$$[\text{FOREACH-1}_{\text{BS}}]\ \frac{\langle \texttt{S}, s[x \mapsto v]\rangle \to s'' \quad \langle \texttt{foreach x in } [\texttt{n}_2, \ldots, \texttt{n}_\texttt{m}] \texttt{ do S}, s''\rangle \to s'}{\langle \texttt{foreach x in } [\texttt{n}_1, \ldots, \texttt{n}_\texttt{m}] \texttt{ do S}, s\rangle \to s'} \quad \text{if } v = \mathcal{N}[\![\texttt{n}_1]\!]$$

$$[\text{FOREACH-2}_{\text{BS}}]\ \frac{\langle \texttt{S}, s[x \mapsto v]\rangle \to s'}{\langle \texttt{foreach x in } [\texttt{n}] \texttt{ do S}, s\rangle \to s'} \quad \text{if } v = \mathcal{N}[\![n]\!]$$

The small-step semantics is given by the following two rules

$$[\text{FOREACH-1}_{\text{SS}}]\ \langle \texttt{foreach x in } [\texttt{n}_1, \ldots, \texttt{n}_\texttt{m}] \texttt{ do S}, s\rangle \Rightarrow \langle \texttt{x} := \texttt{n}_1;\ \texttt{S};\ \texttt{foreach x in } [\texttt{n}_2, \ldots, \texttt{n}_\texttt{m}] \texttt{ do S}, s\rangle$$

$$[\text{FOREACH-2}_{\text{SS}}]\ \langle \texttt{foreach x in } [\texttt{n}] \texttt{ do S}, s\rangle \Rightarrow \langle \texttt{x} := \texttt{n};\ \texttt{S}, s\rangle$$

□

Exercise 4.
Consider the following statements in **Bims**

$$\texttt{y} := \texttt{x} + 4;\ (\texttt{for x} := 1 \texttt{ to } 3 \texttt{ do y} := \texttt{y} * \texttt{x});\ \texttt{y} := \texttt{y} + \texttt{x} \qquad (S_5)$$

$$(\texttt{if x} < 0 \texttt{ then x} := 2 * \texttt{x} \texttt{ else x} := 2 + \texttt{x});\ \texttt{x} := \texttt{x} * (-1) \qquad (S_6)$$

$$\texttt{repeat } S_6 \texttt{ until } (\texttt{x} \geq 200) \qquad (S_7)$$

$$\texttt{while } (\texttt{x} < 200) \texttt{ do } S_6 \qquad (S_8)$$

Find all the transitions (if there are any) in the SS-semantics, for each of the following cases:

(i) $\langle S_5, [x \mapsto -2]\rangle \Rightarrow^* ?$

(ii) $\langle S_7, [x \mapsto 100]\rangle \Rightarrow^* ?$

(iii) $\langle S_8, [x \mapsto 100]\rangle \Rightarrow^* ?$

Solution 4.

First we will define a SS-semantics for `repeat` as we will need it in (ii):

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

(i) $\langle S_5, [x \mapsto -2]\rangle$

$\Rightarrow \langle(\texttt{for x} := 1 \texttt{ to } 3 \texttt{ do y} := \texttt{y} * \texttt{x}); \texttt{y} := \texttt{y} + \texttt{x}, [x \mapsto -2, y \mapsto 2]\rangle$

$\Rightarrow \langle(\texttt{for x} := 2 \texttt{ to } 3 \texttt{ do y} := \texttt{y} * \texttt{x}); \texttt{y} := \texttt{y} + \texttt{x}, [x \mapsto 1, y \mapsto 2]\rangle$

$\Rightarrow \langle(\texttt{for x} := 3 \texttt{ to } 3 \texttt{ do y} := \texttt{y} * \texttt{x}); \texttt{y} := \texttt{y} + \texttt{x}, [x \mapsto 2, y \mapsto 4]\rangle$

$\Rightarrow \langle(\texttt{for x} := 4 \texttt{ to } 3 \texttt{ do y} := \texttt{y} * \texttt{x}); \texttt{y} := \texttt{y} + \texttt{x}, [x \mapsto 3, y \mapsto 12]\rangle$

$\Rightarrow \langle \texttt{y} := \texttt{y} + \texttt{x}, [x \mapsto 4, y \mapsto 12]\rangle$

$\Rightarrow [x \mapsto 4, y \mapsto 16]$

(ii) $\langle S_7, [x \mapsto 100]\rangle$

$\Rightarrow \langle S_6;\ \texttt{if x} \geq 200 \texttt{ then skip else (repeat } S_6 \texttt{ until x} \geq 200), [x \mapsto 100]\rangle$

$\Rightarrow \langle \texttt{x} := \texttt{x} + 2; \texttt{x} := \texttt{x} * (-1);$
$\quad \texttt{if x} \geq 200 \texttt{ then skip else (repeat } S_6 \texttt{ until x} \geq 200), [x \mapsto 100]\rangle$

$\Rightarrow \langle \texttt{x} := \texttt{x} * (-1);\ \texttt{if x} \geq 200 \texttt{ then skip else (repeat } S_6 \texttt{ until x} \geq 200), [x \mapsto 102]\rangle$

$\Rightarrow \langle \texttt{if } x \geq 200 \texttt{ then skip else (repeat } S_6 \texttt{ until } x \geq 200), [x \mapsto -102]\rangle$

$\Rightarrow \langle \texttt{repeat } S_6 \texttt{ until } x \geq 200, [x \mapsto -102]\rangle$

$\Rightarrow \langle S_6;\ \texttt{if x} \geq 200 \texttt{ then skip else (repeat } S_6 \texttt{ until x} \geq 200), [x \mapsto -102]\rangle$

$\Rightarrow \langle \texttt{x} := \texttt{x} * 2; \texttt{x} := \texttt{x} * (-1);\ \texttt{if x} \geq 200 \texttt{ then skip else}$
$\quad \texttt{(repeat } S_6 \texttt{ until x} \geq 200), [x \mapsto -102]\rangle$

$\Rightarrow \langle \texttt{x} := \texttt{x} * (-1);\ \texttt{if x} \geq 200 \texttt{ then skip else}$
$\quad \texttt{(repeat } S_6 \texttt{ until x} \geq 200), [x \mapsto -204]\rangle$

$\Rightarrow \langle \texttt{if } x \geq 200 \texttt{ then skip else (repeat } S_6 \texttt{ until } x \geq 200), [x \mapsto 204]\rangle$

$\Rightarrow \langle \texttt{skip}, [x \mapsto 204]\rangle$

$\Rightarrow [x \mapsto 204]$

(iii) $\langle S_8, [x \mapsto 100]\rangle$

$\Rightarrow \langle \texttt{if x} < 200 \texttt{ then } (S_6;\ \texttt{while (x} < 200\texttt{) do } S_6) \texttt{ else skip}, [x \mapsto 100]\rangle$

$\Rightarrow \langle S_6;\ \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto 100]\rangle$

$\Rightarrow \langle \texttt{x} := 2 + \texttt{x}; \texttt{x} := \texttt{x} * (-1);\ \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto 100]\rangle$

$\Rightarrow \langle \texttt{x} := \texttt{x} * (-1);\ \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto 102]\rangle$

$\Rightarrow \langle \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto -102]\rangle$

$\Rightarrow \langle \texttt{if x} < 200 \texttt{ then } (S_6;\ \texttt{while (x} < 200\texttt{) do } S_6) \texttt{ else skip}, [x \mapsto -102]\rangle$

$\Rightarrow \langle S_6;\ \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto -102]\rangle$

$\Rightarrow \langle \texttt{x} := 2 * \texttt{x}; \texttt{x} := \texttt{x} * (-1);\ \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto -102]\rangle$

$\Rightarrow \langle \texttt{x} := \texttt{x} * (-1);\ \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto -204]\rangle$

$\Rightarrow \langle \texttt{while (x} < 200\texttt{) do } S_6, [x \mapsto 204]\rangle$

$\Rightarrow \langle \texttt{if x} < 200 \texttt{ then } (S_6;\ \texttt{while (x} < 200\texttt{) do } S_6) \texttt{ else skip}, [x \mapsto 204]\rangle$

$\Rightarrow \langle \texttt{skip}, [x \mapsto 204]\rangle$

$\Rightarrow [x \mapsto 204]$

Exercise 5.

Prove or disprove that the following languages are regular or context-free.

$$L_1 = \{ww \mid w \in \{a,b\}^*\}$$
$$L_2 = \{w_1 w_2 \mid w_1, w_2 \in \{a,b\}^*, |w_1| = |w_2|\}$$
$$L_3 = \{a^n w b^n \mid w \in \{a,b\}^*,\ |w| = n\}$$

Solution 5.

L_1) Assume L_1 to be context-free and let $p \geq 0$ be its pumping length. Let $s = a^p b^p a^p b^p$. Clearly $|s| \geq p$ and $s \in L_1$. Consider an arbitrary split of s of the form $uvxyz$ such that $|vy| > 0$ and $|vxy| \leq p$. By $|vy| > 0$, v and y cannot be both empty. Let σ and σ' be respectively the first and last symbol of vxy. We consider the following cases:

$(\sigma = \sigma')$ By $|vxy| \leq p$ we have that $vxy = \sigma^k$ for some $k \leq p$, meaning that vxy is entirely contained in one of the two substrings of the form a^p or one of the form b^p. Consider now the string $s' = uv^2xy^2z$ and assume (without loss of generality) that vxy was within the first sequence of a's. Then s' is of the form $a^k b^p a^p b^p$ for some $k > p$. The string $s' \notin L_1$, indeed if we assume that there exists $w \in \{a, b\}^*$ such that $s' = ww$, then w has to start with a and end in b because s' does so. The only way to do so is to cut s' exactly when the first sequence of b's ends. But this implies that $a^k b^p = a^p b^p$ which is impossible since $k > p$.

($\sigma \neq \sigma'$) Let $s' = uv^0xy^0z$. Consider the following 3 sub-cases according to where vxy is positioned in s.

- If vxy is entirely contained in the first half of the string s. Then the string s' is of the form $a^h b^k a^p b^p$ for some $h, k > 0$ such that $h + k < 2p$. If we assume that $s' \in L_1$ then there exists $w \in \{a, b\}^*$ such that $s' = ww$. Hence w has to start with a and end in b because s' does so. The only way to do so is to cut s' exactly when the first sequence of b's ends, but this implies that $a^h b^k = a^p b^p$ which is impossible since $h + k < 2p$.

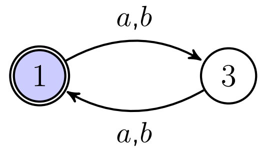
- If vxy is entirely contained in the second half of s. Analogously, to the previous case one can show that $s' = a^p b^p a^h b^k$ for some $h, k > 0$ such that $h + k < 2p$, proving that $s' \notin L_1$.

- If vxy is in between the first and the second half of s. Analogously to the previous cases, one can show that $s' = a^p b^h a^k b^p$ for some $h, k > 0$ such that $h + k < 2p$, proving that $s' \notin L_1$.

This is in contradiction with the Pumping Lemma for context free languages. Thus L_1 is not a context-free language. This also proves that L_1 is not regular either.

Since L_1 is not context-free, it is also not regular. However, if we were only asked to show that L_1 is not regular, we could have used directly the Pumping Lemma for regular languages as follows. Assume that L_1 is regular and let p be its pumping length. Then, $w = a^p b^p a^p b^p \in L_1$ and $|w| \geq p$. Take now a split for $w = xyz$ with $|y| > 0$ (i.e., y non empty) and $|xy| \leq p$. By construction xy has only $a's$. Thus we have that $xy^0z = xz \notin L_1$ since it is of the form $a^q b^p a^p b^p$ for some $q < p$. This contradicts the Pumping Lemma (for regular languages), thus L_1 is not regular.

L_2) The following NFA N generates the language $L_2 = \mathcal{L}(N)$.



This proves that L_2 is a regular language and, therefore, is also context-free. As a sanity check, we also provide a context-free grammar that generates L_2.

$$S \to \varepsilon \mid T\,S\,T$$
$$T \to a \mid b$$

L_3) Assume L_3 to be context-free and let p be its pumping length. Let $s = a^{2p}b^pa^pb^{2p}$. Then $s \in L_3$ and $|s| \geq p$. Consider a split of s of the form $uvxyx$ such that $|vy| > 0$ and $|vxy| \leq p$. By $|vy| > 0$, v and y cannot be both empty. Let σ and σ' be respectively the first and the last symbol of vxy. We consider the following cases:

($\sigma = \sigma'$). By $|vxy| \leq p$ we have that $vxy = \sigma^k$ for some $k \leq p$. Then the string $uv^0xy^0z = uxz \notin L_3$. Indeed if $\sigma = a$ then in $uxz = a^qb^pa^rb^{2p}$ where either $q < 2p$ or $r < p$ (that implies $|b^pa^r| < 2p$).

Analogous arguments apply for $\sigma = b$.

($\sigma \neq \sigma'$). By $|vxy| \leq p$ we have that vxy we have three possible sub-cases

- vxy is entirely contained in the first half of s. Then $uv^0xy^0z \notin L_3$, since it is of the form $a^qb^ra^pb^{2p}$ for some $q \leq 2p$ and $r \leq p$ such that $q + r < 3p$. Therefore, $|a^q| < |b^{2p}| = 2p$ or $|b^ra^p| < 2p$.

- vxy is entirely contained in the second half of s. Then $uv^0xy^0z \notin L_3$ by arguments similar to the previous case.

- vxy is entirely contained in the substring a^pb^p in the middle of s. Then, $uv^0xy^0z \notin L_3$ since it is of the form $a^{2p}b^qa^rb^{2p}$ with $|b^qa^r| < 2p$.

This contradicts the Pumping Lemma for context-free languages. Thus L_3 is not context-free. This also proves that L_3 is not regular.