Exercise session 5
– Type systems –

Exercise 20. Proving Lemma 2.4.3

Prove the following lemma (see slide p.182) we used for the type safety proof of the noninterference type system:

$$\Gamma, \texttt{high} \vdash s \wedge \langle s, \sigma \rangle \rightarrow \sigma' \Rightarrow \sigma \equiv_L \sigma'$$

Exercise 21. Secure or not?

Determine whether the following statements are secure or not. First try to use the type system to show security of a statement. If that attempt fails then try to use operational semantics. If that also fails then give an example showing that the statement is insecure.
Assume that $\Gamma = \{\, \texttt{h} \mapsto \texttt{high},\ \texttt{l} \mapsto \texttt{low} \}$.

1. `if` $\texttt{l} \leq 0$ `then` $\texttt{h} := -1; \texttt{l} := -1$
   `else while` $\texttt{h} \geq \texttt{l}$ `do` $\texttt{h} := \texttt{h} - \texttt{l}$ `end`
   `end`

2. `if` $\texttt{l} \leq 0$ `or` $\texttt{h} < 0$ `then` $\texttt{h} := -1; \texttt{l} := -1$
   `else while` $\texttt{h} \geq \texttt{l}$ `do` $\texttt{h} := \texttt{h} - \texttt{l}$ `end`; $\texttt{l} := -1$
   `end`

3. `if` $\texttt{l} \leq 0$ `or` $\texttt{h} < 0$ `then` $\texttt{h} := -1; \texttt{l} := -1$
   `else while` $\texttt{h} \geq \texttt{l}$ `do` $\texttt{h} := \texttt{h} - \texttt{l}$ `end`;
   `end`

Solutions

Exercise 20. Proving Lemma 2.4.3

The proof goes by induction on the shape of the derivation tree for $\langle s, \sigma\rangle \to \sigma'$.

Base cases.

Case `skip`: transition $\langle \texttt{skip}, \sigma\rangle \to \sigma'$ yields that $\sigma = \sigma'$. Thus, we get $\sigma \equiv_L \sigma'$.

Case $\texttt{x} := e$: according to our assumption the assignment is typed `high`, thus `x` must be a `high` variable. We have to show that $\sigma \equiv_L \sigma[\texttt{x} \mapsto \mathcal{A}[\![e]\!]\sigma]$ holds when `x` is a `high` variable. This means (using the definition of $\equiv_L$) we have to show that $\forall\, y : \Gamma(y) = \texttt{low} \Rightarrow \sigma(y) = \sigma[\texttt{x} \mapsto \mathcal{A}[\![e]\!]\sigma](y)$ holds. Since `x` is a `high` variable, the left-hand side of the implication is false and thus the property holds for variable `x`. Since other variables have not been modified the right-hand side of the implication yields true and thus the property holds for all other variables.
(Intuitively: assigning any value to a `high` variable cannot have any effect on the `low` variables.)

Step cases.

Case $\texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2\ \texttt{end}$: as the statement is typed `high` we know that b, s_1 and s_2 are also typed `high`. Depending on $\mathcal{B}[\![b]\!]\sigma$, either s_1 or s_2 will be executed and by the induction hypothesis we know that

$$\langle s_1, \sigma\rangle \to \sigma' \Rightarrow \sigma \equiv_L \sigma' \quad \text{and} \quad \langle s_2, \sigma\rangle \to \sigma' \Rightarrow \sigma \equiv_L \sigma'$$

holds. Thus, independently of the evaluation of $\mathcal{B}[\![b]\!]\sigma$ we get $\sigma \equiv_L \sigma'$.

Case $\texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}$: as the statement is typed `high` we know that b and s are also typed `high`.
If $\mathcal{B}[\![b]\!]\sigma$ yields false then the `while`-axiom gives $\sigma = \sigma'$, thus $\sigma \equiv_L \sigma'$.
If $\mathcal{B}[\![b]\!]\sigma$ yields true then the `while`-rule gives transitions $\langle s, \sigma\rangle \to \sigma''$ and $\langle \texttt{while}\ b\ \texttt{do}\ s\ \texttt{end}, \sigma''\rangle \to \sigma'$. By the induction hypothesis we get $\sigma \equiv_L \sigma''$ and $\sigma'' \equiv_L \sigma'$. By transitivity of $\equiv_L$ we get $\sigma \equiv_L \sigma'$.

Case s_1; s_2: as the statement is typed `high` we know that s_1 and s_2 are also typed `high`. The sequential composition rule gives transitions $\langle s_1, \sigma\rangle \to \sigma''$ and $\langle s_2, \sigma''\rangle \to \sigma'$. By the induction hypothesis we get $\sigma \equiv_L \sigma''$ and $\sigma'' \equiv_L \sigma'$. By transitivity of $\equiv_L$ we get $\sigma \equiv_L \sigma'$.

Exercise 21. Secure or not?

1.
The first statement can be typed `low` as follows:

$$\dfrac{\dfrac{\Gamma(\texttt{l}) \neq \texttt{high}}{\Gamma \vdash \texttt{l} \leq 0 :: \texttt{low}}, \quad \dfrac{\dfrac{\Gamma(\texttt{h}) = \texttt{high}}{\Gamma, \texttt{low} \vdash \texttt{h:=} -1}, \quad \dfrac{\Gamma \vdash -1 :: \texttt{low},\ \Gamma(\texttt{l}) = \texttt{low}}{\Gamma, \texttt{low} \vdash \texttt{l:=} -1}}{\Gamma, \texttt{low} \vdash \texttt{h:=} -1\texttt{;l:=} -1}, \quad \dfrac{\Gamma \vdash \texttt{h} \geq \texttt{l} :: \texttt{high}, \quad \dfrac{\Gamma(\texttt{h}) = \texttt{high}}{\Gamma, \texttt{high} \vdash \texttt{h:=h} - \texttt{l}}}{\Gamma, \texttt{high} \vdash \texttt{while h} \geq \texttt{l do h:=h} - \texttt{l end}}}{\Gamma, \texttt{low} \vdash \texttt{while h} \geq \texttt{l do h:=h} - \texttt{l end}}}{\Gamma, \texttt{low} \vdash \texttt{if l} \leq 0\ \texttt{then h:=} -1\texttt{;l:=} -1\ \texttt{else while h} \geq \texttt{l do h:=h} - \texttt{l end end}}$$

Note that if one builds the derivation tree from left to right then the first statement that "forces" Δ to be `low` is assignment `l := -1` in the "then" branch (until then Δ could be both `high` and `low`). Furthermore, the loop can be typed `low` only by using the subsumption rule.

2.
The second statement cannot be typed: the condition can only be typed `high`, thus both branches should be typed `high` too. This is not possible, because for the "then" branch we would get the following **illegal** derivation:

$$\frac{\Gamma(\mathtt{h}) = \mathtt{high}}{\Gamma, \mathtt{high} \vdash \mathtt{h} := -1}, \quad \Gamma, \mathtt{high} \vdash \mathtt{l} := -1$$
$$\Gamma, \mathtt{high} \vdash \mathtt{h} := -1; \mathtt{l} := -1$$

The right-hand side premise cannot be verified by the type system. We get the same situation for the other branch.

Let's try to show that the statement is secure by using Natural Semantics. That is, we have to show that the noninterference property holds:

$$\forall \sigma_1, \sigma_2 : \sigma_1 \equiv_L \sigma_2 \Rightarrow \mathcal{S}_{NS}[\![s]\!]\sigma_1 \approx_L \mathcal{S}_{NS}[\![s]\!]\sigma_2$$

Let's use notation σ_{ab} for state $\sigma[\mathtt{l} \mapsto a][\mathtt{h} \mapsto b]$. We set σ_1 to $\{\mathtt{l} \mapsto l, \mathtt{h} \mapsto x\}$ and σ_2 to $\{\mathtt{l} \mapsto l, \mathtt{h} \mapsto y\}$ where we do not assume anything about x, y and l.

Let's first see what $\mathcal{S}_{NS}[\![s]\!]\sigma_1$ gives. As the first step we evaluate the condition and choose one of the two branches.
Case A: the "then" branch is selected. We get the derivation tree:

$$\frac{\dfrac{\langle \mathtt{h} := -1, \sigma_1\rangle \to \sigma_{l-1}, \quad \langle \mathtt{l} := -1, \sigma_{l-1}\rangle \to \sigma_{-1-1}}{\langle \mathtt{h} := -1; \mathtt{l} := -1, \sigma_1\rangle \to \sigma_{-1-1}}}{\langle \mathtt{if\ l} \leq 0 \ \mathtt{or\ h} < 0 \ \mathtt{then\ h} := -1; \mathtt{l} := -1 \ \mathtt{else\ while\ h} \geq \mathtt{l\ do\ h} := \mathtt{h} - \mathtt{l\ end}; \mathtt{l} := -1 \ \mathtt{end}, \sigma_1\rangle \to \sigma_{-1-1}}$$

Thus, the final state is σ_{-1-1}.

Case B: the "else" branch is selected. Using the lemma proved in Exercise 20, we know that the `while`-loop (that can be typed `high`) does not change the values of `low` variables. Thus, after the execution of the loop we get a state σ_{lv}, where v is some value. After executing the assignment "$\mathtt{l} := -1$" we finally get state σ_{-1v}.

We can now observe that both branches yield a final state in which `l` equals -1 independently of the starting value of `l` and `h`. Thus, executing the statement in state σ_2 would result in the same final value for the `low` variable. This means that $\mathcal{S}_{NS}[\![s]\!]\sigma_1 \approx_L \mathcal{S}_{NS}[\![s]\!]\sigma_2$ holds, the statement is secure.

3.
The third statement cannot be typed as the "then" branch is not typeable in `high` (just as in 2.).
We will use the same method of applying Natural Semantics to the statement as in 2.
When starting in state σ_1 we again branch depending on the evaluation of the conditional.

Case A: the "then" branch is selected. Just as in 2. the final state will be σ_{-1-1}.

Case B: the "else" branch is selected. Just as in 2. we apply the lemma and get that the final state is σ_{lv}, where v is some value.

We can observe that the final `low` value might differ (-1 or the initial value, l) depending on which branch was selected. Thus, if we can find two pairs of initial values for `h` and `l` so that in one case the "then" branch, in the other case the "else" branch is selected, then we have showed that $\mathcal{S}_{NS}[\![s]\!]\sigma_1 \approx_L \mathcal{S}_{NS}[\![s]\!]\sigma_2$ does not hold and the statement is not secure.
An example is initial states $\{\mathtt{l} \mapsto 2, \mathtt{h} \mapsto 0\}$ and $\{\mathtt{l} \mapsto 2, \mathtt{h} \mapsto -1\}$.