Exercise 29 (Most General Unifier)

Let $S$ and $T$ be unification problems. Moreover, let $\sigma$ be a most general unifier for $S$ and $\theta$ be a most general unifier for $\sigma(T)$. Show that $\theta\sigma$ is a most general unifier for $S \cup T$.

Solution

- $\theta\sigma$ is a unifier of $S \cup T$:
  
  \[
  \text{Given: } \sigma \in \mathcal{U}(S), \ \theta \in \mathcal{U}(\sigma(T)) \\
  \text{Claim: } \theta\sigma \in \mathcal{U}(S \cup T)
  \]
  
  \[
  \begin{align*}
  \sigma(s) = \sigma(t) & \implies \theta\sigma(s) = \theta\sigma(t) \\
  \sigma(s) \neq \sigma(t) & \implies s \not\sim t \in T \\
  & \implies \sigma(s) \not\sim \sigma(t) \in \sigma(T) \\
  & \implies \theta\sigma(s) = \theta\sigma(t) \quad \text{(as } \theta \in \mathcal{U}(\sigma(T)))
  \end{align*}
  \]

- $\theta\sigma$ is most general unifier of $S \cup T$:
  
  \[
  \text{Proof. Let } s \not\sim t \in S \cup T. \ \text{Case distinction over } \sigma(s) = \sigma(t).
  \]
  
  \[
  \begin{align*}
  \sigma(s) = \sigma(t) & \implies \theta\sigma(s) = \theta\sigma(t) \\
  \sigma(s) \neq \sigma(t) & \implies s \not\sim t \in T \\
  & \implies \sigma(s) \not\sim \sigma(t) \in \sigma(T) \\
  & \implies \theta\sigma(s) = \theta\sigma(t) \quad \text{(as } \theta \in \mathcal{U}(\sigma(T)))
  \end{align*}
  \]

Exercise 30 (Equivalence Relations)

Prove that the notions of a equivalence relation and a partition coincide.

Reminder: A set of sets $P$ is called a partition of a set $M$ iff all elements of $P$ are pairwise disjoint, $P$ does not contain the empty set, and $\bigcup P = M$. 
Solution

Every equivalence relation gives rise to a partition, and vice versa. We give a construction for each direction.

- Let \( \simeq \subseteq M \times M \) be an equivalence relation over the set \( M \). Thus, reflexivity, symmetry and transitivity holds for \( \simeq \). We can define a partition \( P \) as follows:

\[
P = \{ \{ y \mid y \in M \wedge x \simeq y \} \mid x \in M \}
\]

To show: \( P \) is a partition of \( M \).

Proof.
- Let \( P' \in P \) which is induced by some \( x \in M \). Since \( \simeq \) is reflexive, at least \( x \) itself is in \( P' \). Hence, \( P' \neq \emptyset \).
- Let \( P_1, P_2 \in P \) two different elements of \( P \). Assume that there is an \( x \) such that \( x \in P_1 \) and \( x \in P_2 \). By symmetry and transitivity, that means that all elements of \( P_1 \) and \( P_2 \) are in the equivalence relation, which means that \( P_1 = P_2 \). Hence, \( P_1 \) and \( P_2 \) must be disjoint.
- \( \bigcup P \subseteq M \) is trivial.
- \( M \subseteq \bigcup P \) follows from reflexivity.

- Let \( P \) be a partition of \( M \). We construct \( \simeq \) as follows:

\[
x \simeq y \iff \exists P' \in P, x \in P' \wedge y \in P'
\]

To show: \( \simeq \) is an equivalence relation.

Proof.
- Let \( x \in M \). Pick the \( P' \in P \) such that \( x \in P' \). By definition of \( \simeq \), \( x \simeq x \).
- Let \( x, y \in M \). Assume \( x \simeq y \). Obtain \( P' \in P \) such that \( x, y \in P' \). By definition of \( \simeq \), \( y \simeq x \).
- Let \( x, y, z \in M \). Assume \( x \simeq y \) and \( y \simeq z \). For both, obtain \( P_1, P_2 \in P \) such that \( x, y \in P_1 \) and \( y, z \in P_2 \). Since \( y \in P_1 \cap P_2 \) and because \( P \) is a partition, \( P_1 \) must be the same as \( P_2 \). Finally, \( x, z \in P_1 = P_2 \), hence \( x \simeq z \).

Both constructions are inverses of each other. That is, starting with an equivalence relation, constructing a partition, then constructing an equivalence relation again, produces the original relation, and vice versa. (The proof is left as an exercise to the reader.)
Exercise 31 (Termination)

A term rewriting system $R$ is called right reduced, if for all $(l \rightarrow r) \in R$, the right hand side $r$ is irreducible. Show that every right reduced and right ground term rewriting system terminates.

*Hint:* Consider the positions in the term at which rules from $R$ may be applied, and specify a suitable order on terms. Is there a simpler way to get this lemma as a corollary from a lemma that was presented in the lecture?

**Solution**

The argument that the number of positions where a reduction may be applied decreases in each step does not work. *Counterexample:* TRS $f(c) \rightarrow d$ and $b \rightarrow c$

Consider the starting term $f(b)$. It has exactly one redex, namely $b$. We get: $f(b) \rightarrow f(c)$. But now we again have one redex: $f(c) \rightarrow d$

However, the above example demonstrates that the new redex lays over the old one. This leads to the following proof:

*Proof.* For every term $t$, all reduction sequences starting in $t$ are finite. Informally, this holds as every term $t$ has only finitely many redexes. If a rule is applied at position $p$ in $t$, then below $p$ no further rules can be applied (as all right hand sides are irreducible, and as they contain no variables, they stay irreducible). Above position $p$, further rules may be applied, but this is only possible finitely often, until the root of the term is reached.

A more formal proof uses the multiset order to prove termination:

*Proof.* On the set of positions in $t$, we have the following terminating order:

$$p_1 < p_2 \iff p_1 \text{ properly above } p_2 \quad (\text{i.e. } \exists x \in \mathbb{N}^+. \ p_1 x = p_2)$$

Let $P(t)$ be the multiset of positions of $t$, at which a rule of $R$ can be applied. A position $p$ occurs multiple times in $P(t)$, if more than one rule can be applied at $p$.

For each application of a rule $l \rightarrow r$ at position $p$ in $t_1$ that yields $t_1 \rightarrow t_2$, we have:

$$P(t_1) \succ_{\text{mul}} P(t_2), \text{ as } P(t_2) = (P(t_1) - X) \cup Y \land \forall y \in Y. \exists x \in X. \ y < x$$

where $X$ is the set of positions at which rules can be applied in $t_1$ but not in $t_2$. $X$ contains at least the position $p$.

$Y$ is the set of positions, at which rules can be applied in $t_2$ but not in $t_1$.

If $Y$ is not empty, we have for all $p' \in Y$: $p' < p$.

Thus, for every reduction sequence, there is a sequence of multisets, that decreases due to the multiset ordering. As this multiset ordering terminates, all reduction sequences terminate. \(\square\)
There is also a simpler way, to get this lemma as a corollary from Lemma 5.1.8:

Let $R$ be a finite right-ground term rewriting system. Then, the following statements are equivalent:

a) $R$ does not terminate
b) There exists a rule $l \rightarrow r \in R$ and a term $t$ such that $r \xrightarrow{+} R t$ and $t$ contains $r$ as a subterm.

In our case, a reduction sequence starting with $r$ does not exists, as $r$ is irreducible, by assumption. Thus, $R$ terminates.

**Homework 32 (Compactness)**

Prove that every satisfiable set of equations over a finite set of variables contains a finite subset that has the same solutions.

Note that equations are interpreted in the term algebra.

*Hint:* Select a countable subset of the set of equations.

**Homework 33 (Deciding Termination for Right-Ground TRSs)**

In the lecture, we discussed a procedure to decide termination of right-ground term rewriting systems. It is important that we use a breadth-first search strategy, as you shall demonstrate in this exercise.

a) Given the following procedure that uses a depth-first approach:

   **Input** A right ground term rewriting system $R = \{l_1 \rightarrow r_1, \ldots, l_n \rightarrow r_n\}$.

   **Procedure** Enumerate all reduction sequences that start with $r_1$, in depth-first order.
   If one of those sequences contains $r_1$ as a subterm, output *non-terminating*.
   Otherwise continue with the sequences starting at $r_2$, and so on. If all right hand sides have been processed, output *terminating*.

   Find a right-ground term rewriting system such that the above procedure does not terminate.

b) Determine whether the following rewriting systems terminate using the breadth-first approach:

   \[
   \begin{align*}
   R_1 = \{ f(x, x) \rightarrow f(a, b), b \rightarrow c \} \\
   R_2 = \{ f(x, x) \rightarrow f(a, b), b \rightarrow a, b \rightarrow c \}
   \end{align*}
   \]

   c) Implement the correct algorithm in Haskell. More instructions can be found on the lecture website.