Exercise 1 (Examples for Well-foundedness and Confluence): (1 + 1 + 2 = 4 points)

a) Let $\sim \subseteq \mathbb{N}^2$ be the relation such that $x \sim y$ iff $\exists z \in \mathbb{N} \setminus \{1\} : x = y \cdot z$.

Prove or disprove *well-foundedness* and *confluence* of $\sim$.

b) Consider the relation $\supsetneq_A \subseteq \mathcal{P}(A)^2$ for an arbitrary finite set A. Here $\mathcal{P}(A)$ denotes the *power set* of A, i.e., the set of all sets with elements from A. Then $M \supsetneq_A N$ holds iff $M \neq N$ and for all $a \in N$ also $a \in M$ holds. Prove or disprove *well-foundedness* and *confluence* of $\supsetneq_A$.

c) A term s is called *linear* iff for each $x \in \mathcal{V}(s)$ there is one and only one position π such that $s|_\pi = x$. Let $\mathcal{T}_\ell(\Sigma, \mathcal{V})$ be the set of all linear terms over Σ and $\mathcal{V}$ and let $\sqsupseteq \subseteq \mathcal{T}_\ell(\Sigma, \mathcal{V})^2$ be the inverse matching relation, i.e., we have $s \sqsupseteq t$ iff there is a substitution σ such that $s = t\sigma$. We define $s \sqsupset t$ iff $s \sqsupseteq t \wedge t \not\sqsupseteq s$. Prove or disprove *well-foundedness* and *confluence* of $\sqsupset$.

Solution:

a)
- The relation $\sim$ is *not well founded*.
  We have $0 \sim 0 \sim 0 \sim \ldots$ since $0 = 0 \cdot 0$.
- The relation $\sim$ is *confluent*.
  We have $x = 1 \cdot x$ for all $x \neq 1$.

b)
- The relation $\supsetneq_A$ is *well founded*.
  Since A is finite, we have $|M| \in \mathbb{N}$ for each $M \in \mathcal{P}(A)$. By definition, $M \supsetneq_A N$ implies $a \in N \implies a \in M$. Hence, we have $|M| \geq |N|$. By definition, we also have $M \neq N$, such that we get $|M| > |N|$. Hence, well-foundedness of $\supsetneq_A$ follows from well foundedness of $>$.
- The relation $\supsetneq_A$ is *confluent*.
  Let $P, S, T \in \mathcal{P}(A)$ where $P \supsetneq_A^* S$ and $P \supsetneq_A^* T$. Then there is also a value $Q \in \mathcal{P}(A)$ such that $S \supsetneq_A^* Q$ and $T \supsetneq_A^* Q$ — choose $Q = \emptyset$ (or $Q = S \cap T$).

c)
- The relation $\sqsupset$ is *well founded*.
  Let $>_\Sigma \subseteq \mathcal{T}_\ell(\Sigma, \mathcal{V})^2$ be the relation such that $s >_\Sigma t$ iff $|s|_\Sigma > |t|_\Sigma$ (i.e., s contains more function symbols than t). Obviously, $>_\Sigma$ is well founded, since $>$ is well founded. Assume there are terms s, t such that $s \sqsupset t$, but $s \not>_\Sigma t$. Then, by definition of $\sqsupset$, there is a substitution σ such that $s = t\sigma$. Since $s \not>_\Sigma t$ (and applying σ cannot "remove" function symbols), we have $|s|_\Sigma = |t|_\Sigma$. Since s and t are linear, this means that there is a variable renaming μ (i.e., an injective substitution $\mu : \mathcal{V} \to \mathcal{V}$) such that $s = t\mu$. But then, there is also a variable renaming ν such that $s\nu = t$, which contradicts $s \sqsupset t$. Hence, well-foundedness of $>_\Sigma$ implies well-foundedness of $\sqsupset$.
- The relation $\sqsupset$ is *not confluent*.
  We have, e.g., $\mathsf{f}(x) \sqsupset x$ and $\mathsf{f}(x) \sqsupset y$, but x and y are normal forms w.r.t. $\sqsupset$.

Exercise 2 (Well-foundedness and Confluence): (3 + 2 = 5 points)

Consider an arbitrary binary relation $\to \subseteq A \times A$. Now consider the following three relations:

1. $\to$

2. $\rightarrow^+$
3. $\rightarrow^{*,\neq}$, where $s \rightarrow^{*,\neq} t$ iff $s \rightarrow^* t$ and $s \neq t$.

a) Investigate for each pair of these relations if well-foundedness of the first relation implies well-foundedness of the second relation (6 cases). For this, either give a proof or a counterexample.

b) Investigate for each pair of these relations if confluence of the first relation implies confluence of the second relation (6 cases). For this, either give a proof or a counterexample.

Solution:

a) Note: If we have $\rightarrow_1 \supseteq \rightarrow_2$, then obviously well-foundedness of $\rightarrow_1$ also implies well-foundedness of $\rightarrow_2$.

- $\rightarrow^+$ well founded $\Rightarrow$ $\rightarrow$ well founded (superset)
- $\rightarrow^+$ well founded $\Rightarrow$ $\rightarrow^{*,\neq}$ well founded (superset)
- $\rightarrow^{*,\neq}$ well founded $\not\Rightarrow$ $\rightarrow$ well founded ($\rightarrow = \{(a,a)\}, \rightarrow^{*,\neq} = \emptyset$)
- $\rightarrow^{*,\neq}$ well founded $\not\Rightarrow$ $\rightarrow^+$ well founded ($\rightarrow = \{(a,a)\}, \rightarrow^{*,\neq} = \emptyset, \rightarrow^+ = \{(a,a)\}$)
- $\rightarrow$ well founded $\Rightarrow$ $\rightarrow^+$ well founded
  Assume $a_1 \rightarrow^+ a_2 \rightarrow^+ a_3 \rightarrow \ldots$. Then we have $a_1 \rightarrow b_{1,1} \rightarrow \cdots \rightarrow b_{1,n_1} \rightarrow a_2 \rightarrow b_{2,1} \rightarrow \cdots \rightarrow b_{2,n_2} \rightarrow a_3 \rightarrow \ldots$, contradicting well-foundedness of $\rightarrow$.
- $\rightarrow$ well founded $\Rightarrow$ $\rightarrow^{*,\neq}$ well founded ($\rightarrow$ well founded $\Rightarrow$ $\rightarrow^+$ well founded $\Rightarrow$ $\rightarrow^{*,\neq}$ well founded)

b) Recall: $\rightarrow$ is confluent iff for all p, s, t with $p \rightarrow^* s, p \rightarrow^* t$ there exists some q such that $s \rightarrow^* q$ and $t \rightarrow^* q$ hold. In set notation: $\leftarrow^* \circ \rightarrow^* \subseteq \rightarrow^* \circ \leftarrow^*$.
Obviously we have $\rightarrow^* = (\rightarrow^+)^* = (\rightarrow^{*,\neq})^*$, which implies equivalence of confluence for the three relations $\rightarrow, \rightarrow^+, \rightarrow^{*,\neq}$.

Exercise 3 (Programming in Term Rewriting): (2 + 3 = 5 points)

a) Define a term rewrite system $\mathcal{R}$ for reverse, which can be used to reverse a list $[x_1, x_2, \ldots, x_n]$ resulting in the list $[x_n, x_{n-1}, \ldots, x_1]$.

To represent lists, use the two symbols $\mathsf{Nil} \in \Sigma_0$ and $\mathsf{Cons} \in \Sigma_2$. Nil represents the empty list and $\mathsf{Cons}(x, xs)$ represents a list where x is the first list element and xs is the representation of the remaining list elements. Thus, the term $\mathsf{Cons}(x, \mathsf{Cons}(y, \mathsf{Nil}))$ corresponds to the list $[x, y]$. For example, your TRS should allow the reduction $\mathsf{reverse}(\mathsf{Cons}(\mathcal{O}, \mathsf{Cons}(\mathsf{s}(\mathcal{O}), \mathsf{Nil}))) \to_{\mathcal{R}}^* \mathsf{Cons}(\mathsf{s}(\mathcal{O}), \mathsf{Cons}(\mathcal{O}, \mathsf{Nil}))$.

b) Integers can be represented by terms using the function symbols $\mathcal{O} \in \Sigma_0$ and $\mathsf{s}, \mathsf{neg} \in \Sigma_1$, where neg denotes negative numbers. Hence, integers are mapped to terms by the following function $term : \mathbb{Z} \to \mathcal{T}(\{\mathcal{O}, \mathsf{s}, \mathsf{neg}\}, \emptyset)$:

$$term(x) = \begin{cases} \mathsf{s}^{|x|}(\mathcal{O}) & \text{If } x \geq 0 \\ \mathsf{neg}(\mathsf{s}^{|x|}(\mathcal{O})) & \text{Otherwise} \end{cases}$$

Define a term rewrite system $\mathcal{R}$ for plus, which can be used to add integers. For example, your TRS should allow the reduction $\mathsf{plus}(\mathsf{neg}(\mathsf{s}(\mathcal{O})), \mathsf{s}(\mathsf{s}(\mathcal{O}))) \to_{\mathcal{R}}^* \mathsf{s}(\mathcal{O})$. You can assume that the arguments of plus are of the form $\mathsf{s}^x(\mathcal{O})$ or $\mathsf{neg}(\mathsf{s}^x(\mathcal{O}))$, $x \in \mathbb{N}$.

Solution:

a)

$$\begin{aligned} \mathsf{append}(\mathsf{Nil}, ys) &\to ys \\ \mathsf{append}(\mathsf{Cons}(x, xs), ys) &\to \mathsf{Cons}(x, \mathsf{append}(xs, ys)) \\ \mathsf{reverse}(\mathsf{Nil}) &\to \mathsf{Nil} \\ \mathsf{reverse}(\mathsf{Cons}(x, xs)) &\to \mathsf{append}(\mathsf{reverse}(xs), \mathsf{Cons}(x, \mathsf{Nil})) \end{aligned}$$

b)

$$\mathcal{R}_{\mathsf{plus}} = \left\{ \begin{array}{rcl} \mathsf{plus}(x, \mathcal{O}) & \to & x \\ \mathsf{plus}(\mathcal{O}, x) & \to & x \\ \mathsf{neg}(\mathcal{O}) & \to & \mathcal{O} \\ \mathsf{plus}(\mathsf{s}(x), \mathsf{s}(y)) & \to & \mathsf{s}(\mathsf{plus}(\mathsf{s}(x), y)) \\ \mathsf{plus}(\mathsf{neg}(x), \mathsf{neg}(y)) & \to & \mathsf{neg}(\mathsf{plus}(x, y)) \\ \mathsf{plus}(\mathsf{neg}(x), \mathsf{s}(y)) & \to & \mathsf{plus}(\mathsf{s}(y), \mathsf{neg}(x)) \\ \mathsf{plus}(\mathsf{s}(x), \mathsf{neg}(\mathsf{s}(y))) & \to & \mathsf{plus}(x, \mathsf{neg}(y)) \end{array} \right\}$$

Exercise 4 (Term Rewriting and WHILE Programs): (5 points)

WHILE programs are known to be Turing complete. The goal of this exercise is to prove Turing completeness of term rewriting by simulating WHILE programs with TRSs.

Given a set of variables $\mathcal{V} = \{x_1, \ldots, x_n\}$, the set $\mathcal{W}$ of all WHILE programs operating on $\mathcal{V}$ is the smallest set such that:

- "$x := y + c$"$\in \mathcal{W}$ for all $x, y \in \mathcal{V}, c \in \mathbb{Z}$

- if $w, u \in \mathcal{W}$, then "$w; u$"$\in \mathcal{W}$
- if $w \in \mathcal{W}$, then "WHILE $x \neq 0$ DO w END" $\in \mathcal{W}$ for all $x \in \mathcal{V}$

So we have, e.g., add ="WHILE $y \neq 0$ DO $x = x + 1; y = y + (-1)$ END" $\in \mathcal{W}$.

Given an initial valuation $v : \mathcal{V} \to \mathbb{Z}$, a WHILE program w computes the result $eval(v, w)$ where

$$eval : (\mathcal{V} \to \mathbb{Z}) \times \mathcal{W} \to (\mathcal{V} \to \mathbb{Z})$$

is a partial function which is defined as follows:

$$\begin{aligned}
eval(v, x := y + c) &= v' \text{ where } v'(z) = \begin{cases} v(y) + c & \text{if } z = x \\ v(z) & \text{otherwise} \end{cases} \\
eval(v, w; u) &= eval(eval(v, w), u) \\
eval(v, \text{WHILE } x \neq 0 \text{ DO } w \text{ END}) &= \begin{cases} v & \text{if } v(x) = 0 \\ eval(eval(v, w), \text{WHILE } x \neq 0 \text{ DO } w \text{ END}) & \text{otherwise} \end{cases}
\end{aligned}$$

So for our example program add, we have $eval(v, add) = v(x) + v(y)$ for all valuations v such that $v(y) \geq 0$.

The program w terminates on the valuation v iff $eval(v, w)$ is defined. So our example program add terminates on the valuation v iff $v(y) \geq 0$.

With $term^{-1}$, we refer to the partial function such that $term^{-1}(term(x)) = x$ (where $term$ is defined as in the previous exercise, i.e., $term^{-1}$ can be used to map terms of the form $\mathsf{s}^x(\mathcal{O})$ or $\mathsf{neg}(\mathsf{s}^x(\mathcal{O}))$ to integers in the expected way). For a given WHILE program w, construct a corresponding term rewrite system $\mathcal{R}$, such that the following holds for all valuations v:

- If w terminates on v (i.e., $eval(v, w)$ is defined), then $\mathsf{start}(term(v(x_1)), \ldots, term(v(x_n)))$ has one and only one $\to_{\mathcal{R}}$-normal form $\mathsf{end}(t_1, \ldots, t_n)$ where $term^{-1}(t_i) = (eval(v, w))(x_i)$ for all $i \in \{1, \ldots, n\}$.
- Otherwise, $\mathsf{start}(term(v(x_1)), \ldots, term(v(x_n)))$ does not have a $\to_{\mathcal{R}}$-normal form.

You do not have to provide a formal proof for your construction, but you should explain it.

Solution:

First note that, by construction, every WHILE program has a unique entry- and exit-point. For each WHILE program w, we define a TRS $\mathcal{R}_w$ as follows:

- If $w = x_i := x_j + c$, then $\mathcal{R}_w = \{\mathsf{start}(x_1, \ldots, x_n) \to \mathsf{end}(x_1, \ldots, x_{i-1}, \mathsf{plus}(x_j, term(c)), x_{i+1}, \ldots, x_n)\}$.
- If $w = u; u'$, then $\mathcal{R}_w = \mathcal{R}'_u \cup \mathcal{R}'_{u'}$ where $\mathcal{R}'_u$ results from $\mathcal{R}_u$ by replacing the function symbol end by f and $\mathcal{R}'_{u'}$ results from $\mathcal{R}_{u'}$ by replacing the function symbol start by f where f is fresh (i.e., f does neither occur in $\mathcal{R}_u$, nor in $\mathcal{R}_{u'}$).
- If $w = \text{WHILE } x_i \neq 0 \text{ DO } u \text{ END}$, then $\mathcal{R}_w$ contains the following rules

$$\begin{aligned}
\mathsf{start}(x_1, \ldots, x_{i-1}, \mathcal{O}, x_{i+1}, \ldots, x_n) &\to \mathsf{end}(x_1, \ldots, x_{i-1}, \mathcal{O}, x_{i+1}, \ldots, x_n), \\
\mathsf{start}(x_1, \ldots, x_{i-1}, \mathsf{s}(x), x_{i+1}, \ldots, x_n) &\to \mathsf{f}(x_1, \ldots, x_{i-1}, \mathsf{s}(x), x_{i+1}, \ldots, x_n), \\
\mathsf{start}(x_1, \ldots, x_{i-1}, \mathsf{neg}(x), x_{i+1}, \ldots, x_n) &\to \mathsf{f}(x_1, \ldots, x_{i-1}, \mathsf{neg}(x), x_{i+1}, \ldots, x_n), \\
\mathsf{g}(x_1, \ldots, x_n) &\to \mathsf{start}(x_1, \ldots, x_n)
\end{aligned}$$

  and, in addition, all rules from $\mathcal{R}'_u$, which results from $\mathcal{R}_u$ by replacing the function symbol start with f and the function symbol end with g where f and g are fresh.

Then, a given WHILE program w is translated to $\mathcal{R}_w \cup \mathcal{R}_{\mathsf{plus}}$.