Prof. Dr. Jürgen Giesl

Marc Brockschmidt, Carsten Fuhs, Thomas Ströder

Exercise 1 (Examples for Well-foundedness and Confluence): (2 + 2 + 1 = 5 points)

a) Let $\delta \in \mathbb{Q}^+ = \{x \in \mathbb{Q} \mid x > 0\}$ and $c \in \mathbb{Q}$. Consider the relation $>_{\delta,c} \subseteq \mathbb{Q} \times \mathbb{Q}$, where $x >_{\delta,c} y$ holds iff both $x \geq y + \delta$ and $y > c$ hold. Prove or disprove *well-foundedness* and *confluence* of $>_{\delta,c}$.

b) Consider the relation $\supsetneq_A \subseteq \mathcal{P}(A)^2$ for an arbitrary 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) From the lecture we know that the strict *subterm relation* $\rhd \subseteq \mathcal{T}(\Sigma, \mathcal{V})^2$ is well founded. Now prove or disprove *confluence* of $\rhd$.

Solution:

a)
- The relation $>_{\delta,c}$ is *well founded*.
  From a value x at most $\frac{x-c}{\delta}$ repeated steps in the relation $>_{\delta,c}$ are possible, so there cannot exist any infinite sequence in the relation $>_{\delta,c}$.
- The relation $>_{\delta,c}$ is *not confluent*.
  Consider $p = c + 2 \cdot \delta$, $s = c + \frac{1}{2} \cdot \delta$, and $t = c + \frac{1}{3} \cdot \delta$. We have $p >^*_{\delta,c} s$ and $p >^*_{\delta,c} t$. However, both s and t are normal forms w.r.t. $>_{\delta,c}$, and $s \neq t$. Thus, there does not exist any q with $s >^*_{\delta,c} q$ and $t >^*_{\delta,c} q$, so $>^*_{\delta,c}$ cannot be confluent.

b)
- The relation $\supsetneq_A$ in general is *not well founded*.
  Consider e.g. $A = \mathbb{N}$. We have $\mathbb{N} \supsetneq_A \mathbb{N} \setminus \{0\} \supsetneq_A \mathbb{N} \setminus \{0, 1\} \supsetneq_A \ldots$.
- 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 $\rhd$ is *not confluent*.
As a counterexample, consider the terms $p = \mathsf{f}(\mathsf{a}, \mathsf{b})$, $s = \mathsf{a}$, and $t = \mathsf{b}$. We have $p \rhd^* s$ and $p \rhd^* t$. Since s and t are normal forms for the relation $\rhd$ with $s \neq t$, we can conclude that there is no term q with $s \rhd^* q$ and $t \rhd^* q$.

Exercise 2 (Well-foundedness and Confluence): (3 + 3 = 6 points)

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

1. $\to$
2. $\to^+$
3. $\to^{*,\neq}$, where $s \to^{*,\neq} t$ iff $s \to^* t$ 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 $\to_1 \supseteq \to_2$, then obviously well-foundedness of $\to_1$ also implies well-foundedness of $\to_2$.

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

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

Exercise 3 (Programming in Term Rewriting): (1 + (1 + 1 + 1) + 1 = 5 points)

a) Define a term rewrite system for append, which can be used to append two lists $[x_1, x_2, \ldots, x_n]$ and $[y_1, y_2, \ldots, y_m]$ to obtain the list $[x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m]$.

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 with x as first value and xs is the representation of the remaining the list elements. Thus, the term $\mathsf{Cons}(x, \mathsf{Cons}(y, \mathsf{Nil}))$ corresponds to the list $[x, y]$. For example, your system should describe the reduction $\mathsf{append}(\mathsf{Cons}(\mathcal{O}, \mathsf{Nil}), \mathsf{Cons}(\mathsf{s}(\mathcal{O}), \mathsf{Nil})) \to^* \mathsf{Cons}(\mathcal{O}, \mathsf{Cons}(\mathsf{s}(\mathcal{O}), \mathsf{Nil}))$.

b) i) Translate the following functional Haskell program into a term rewrite system:

```
take n          Nil = Nil
take 0           xs = Nil
take (n+1) (Cons x xs) = (Cons x (take n xs))

from n = (Cons n (from (n+1)))
```

Each of the equations from the functional program should be translated to one term rewrite rule. Use $\mathcal{O}, \mathsf{s}$ to represent natural numbers and Cons, Nil to represent lists.

ii) How many normal forms does the term $t_1 = \mathsf{from}(\mathcal{O})$ have? Is there an infinite reduction that starts in t_1?

iii) How many normal forms does the term $t_2 = \mathsf{take}(\mathsf{s}(\mathcal{O}), \mathsf{from}(\mathcal{O}))$ have? Is there an infinite reduction that starts in t_2?

c) Consider the following term rewrite system:

$$\begin{aligned}
\mathsf{p}(\mathcal{O}) &\to \mathcal{O} \\
\mathsf{p}(\mathsf{s}(n)) &\to n \\
\mathsf{copy}(\mathcal{O}, xs) &\to \mathsf{Nil} \\
\mathsf{copy}(n, \mathsf{Nil}) &\to \mathsf{Nil} \\
\mathsf{copy}(n, \mathsf{Cons}(x, xs)) &\to \mathsf{Cons}(x, \mathsf{copy}(\mathsf{p}(n), xs))
\end{aligned}$$

Here, p is the predecessor function. The system is not confluent. Give an example for a term which has at least two normal forms.

Solution:

a)

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

b) i)

$$\begin{aligned}
\mathsf{take}(\mathcal{O}, xs) &\to \mathsf{Nil} \\
\mathsf{take}(n, \mathsf{Nil}) &\to \mathsf{Nil} \\
\mathsf{take}(\mathsf{s}(n), \mathsf{Cons}(x, xs)) &\to \mathsf{Cons}(x, \mathsf{take}(n, xs)) \\
\mathsf{from}(n) &\to \mathsf{Cons}(n, \mathsf{from}(\mathsf{s}(n)))
\end{aligned}$$

ii) t_1 has no normal form. There is an infinite reduction:

$$\begin{aligned}\mathsf{from}(\mathcal{O}) &\to \mathsf{Cons}(\mathcal{O}, \mathsf{from}(\mathsf{s}(\mathcal{O})))\\ &\to \mathsf{Cons}(\mathcal{O}, \mathsf{Cons}(\mathsf{s}(\mathcal{O}), \mathsf{from}(\mathsf{s}(\mathsf{s}(\mathcal{O})))))\\ &\dots\end{aligned}$$

iii) t_2 has one normal form:

$$\begin{aligned}\mathsf{take}(\mathsf{s}(\mathcal{O}), \mathsf{from}(\mathcal{O})) &\to \mathsf{take}(\mathsf{s}(\mathcal{O}), \mathsf{Cons}(\mathcal{O}, \mathsf{from}(\mathsf{s}(\mathcal{O}))))\\ &\to \mathsf{Cons}(\mathcal{O}, \mathsf{take}(\mathcal{O}, \mathsf{from}(\mathsf{s}(\mathcal{O}))))\\ &\to \mathsf{Cons}(\mathcal{O}, \mathsf{Nil})\end{aligned}$$

However, there is also an infinite reduction starting in t_2:

$$\begin{aligned}\mathsf{take}(\mathsf{s}(\mathcal{O}), \mathsf{from}(\mathcal{O})) &\to \mathsf{take}(\mathsf{s}(\mathcal{O}), \mathsf{Cons}(\mathcal{O}, \mathsf{from}(\mathsf{s}(\mathcal{O}))))\\ &\to \mathsf{take}(\mathsf{s}(\mathcal{O}), \mathsf{Cons}(\mathcal{O}, \mathsf{Cons}(\mathsf{s}(\mathcal{O}), \mathsf{from}(\mathsf{s}(\mathsf{s}(\mathcal{O}))))))\\ &\dots\end{aligned}$$

c) $t = \mathsf{copy}(\mathcal{O}, \mathsf{Cons}(\mathcal{O}, \mathsf{Nil}))$ is such a term. Consider the following two reductions:

$$\begin{aligned}\mathsf{copy}(\mathcal{O}, \mathsf{Cons}(\mathcal{O}, \mathsf{Nil})) &\to \mathsf{Nil}\\ \mathsf{copy}(\mathcal{O}, \mathsf{Cons}(\mathcal{O}, \mathsf{Nil})) &\to \mathsf{Cons}(\mathcal{O}, \mathsf{copy}(\mathsf{p}(\mathcal{O}), \mathsf{Nil}))\\ &\to \mathsf{Cons}(\mathcal{O}, \mathsf{copy}(\mathcal{O}, \mathsf{Nil}))\\ &\to \mathsf{Cons}(\mathcal{O}, \mathsf{Nil})\end{aligned}$$

Exercise 4 (Term Rewriting and Turing Machines): (4 + 4* points)

A deterministic Turing machine is a 6-tuple $(\mathcal{Q}, \Gamma, \varepsilon, q_s, q_e, \delta)$ with a finite set of states $\mathcal{Q}$, a finite alphabet Γ, the blank symbol $\varepsilon \in \Gamma$, the initial state $q_s \in \mathcal{Q}$, the final state $q_e \in \mathcal{Q}$, and the step function $\delta : \mathcal{Q} \setminus \{q_e\} \times \Gamma \to \mathcal{Q} \times \{\mathsf{left}, \mathsf{right}\} \times \Gamma$. A configuration $\mathcal{K}$ of a Turing machine is a triple (q, p, b), where q is a state from $\mathcal{Q}$ and $p \in \mathbb{Z}$ is a position on the tape b. The tape b is a memory which is modeled as a function from $\mathbb{Z}$ to Γ, where only finitely many memory cells are used, i.e., $b(p) \neq \varepsilon$ only holds for finitely many positions p.
The initial configuration $\mathcal{K}_s$ is $(q_s, 0, b_{initial})$, where $b_{initial}(x) = \varepsilon$ for all $x \in \mathbb{Z}$. The computation relation $\vdash$ is defined as follows:

$$\begin{aligned}(q, p, b) \vdash (q', p', b') \quad \textit{iff} \quad & q \neq q_e,\\ & \delta(q, b(p)) = (q', \textit{direction}, a),\\ & p' = \left\{ \begin{array}{ll} p+1, & \text{if } \textit{direction} = \mathsf{right} \\ p-1, & \text{otherwise} \end{array} \right\} \text{ and}\\ & b' \text{ is the tape with } b'(x) = \left\{ \begin{array}{ll} a, & \text{if } x = p \\ b(x), & \text{otherwise} \end{array} \right\}\end{aligned}$$

A Turing machine is *terminating* iff there is no infinite sequence $\mathcal{K}_s \vdash \mathcal{K}_1 \vdash \mathcal{K}_2 \vdash \dots$. The question whether a Turing machine terminates - the so-called halting problem - is generally known to be undecidable.

a) For a given arbitrary Turing machine $(\mathcal{Q}, \Gamma, \varepsilon, q_s, q_e, \delta)$, construct a corresponding term rewrite system $\mathcal{R}$, such that every computation step of the Turing machine can be simulated by one or more rewrite steps of the term rewrite system, i.e., for each two configurations $\mathcal{K}_1$ and $\mathcal{K}_2$ w.r.t. the Turing machine with

$\mathcal{K}_1 \vdash \mathcal{K}_2$, there are two terms t_1 and t_2 such that t_1 is uniquely representing $\mathcal{K}_1$, t_2 is uniquely representing $\mathcal{K}_2$, and $t_1 \to_{\mathcal{R}}^* t_2$. You do not have to provide a formal proof for your construction, but you should explain it.

Hints:

- In order to represent a configuration (q, p, b) as a term, you can for example use a binary function symbol f_q for each state $q \in \mathcal{Q}$. The two arguments ℓ and r of a term $f_q(\ell, r)$ then represent the symbols before (including) and after (excluding) the position p on the tape. In the following example, the position p points to the symbol a_3. Then ℓ represents the sequence of symbols $a_3 a_2 a_1$ and r represents the sequence of symbols $a_4 a_5 a_6$.

Tape b	$\cdots$	ε	ε	a_1	a_2	a_3	a_4	a_5	a_6	ε	ε	$\cdots$

b)* The construction from the previous exercise part is probably not useful to show undecidability of the question whether an arbitrary given term rewrite system terminates. The reason is that a solution for the first exercise part may start with terms representing unreachable system configurations w.r.t. the Turing machine which cause non-termination of the corresponding term rewrite system. As an example, consider the Turing machine $(\{q_{es}, q_1\}, \{\varepsilon\}, \varepsilon, q_{es}, q_{es}, \delta)$ with $\delta(q_1, \varepsilon) = (q_1, \mathsf{right}, \varepsilon)$. Since initial and final state are the same, the Turing machine obviously terminates. However, starting from the (unreachable) configuration $\mathcal{K} = (q_1, 0, b)$, we obtain the infinite sequence $\mathcal{K} \vdash (q_1, 1, b) \vdash (q_1, 2, b) \vdash \ldots$. Therefore, a term representing the configuration $\mathcal{K}$ might not terminate w.r.t. the constructed term rewrite system.

Adapt your construction from the previous exercise part in such a way that the Turing machine is terminating iff the corresponding term rewrite system is terminating. Again, you do not have to provide a formal proof for your construction, but you should explain it.

Solution:

a) We first consider the representation of the tape. We use the binary concatenation operator : (like Cons in Exercise 3) and interpret $\mathsf{a} : \mathsf{b} : \mathsf{nil}$ as the sequence of symbols ab. Now we insert rules for the state transitions. For this we need the binary function symbols f_q for each state $q \in \mathcal{Q}$. The first argument represents the tape contents left from and at the current position while the second argument represents the tape contents right from the current position. Although the tape is infinite, it suffices to use finite terms, since only finitely many positions may contain symbols different from the blank symbol: we do not represent the infinite sequences of blank symbols left and right from the other tape contents explicitly. The finite alphabet Γ is represented by a finite set of constants, so we use $\Gamma \subseteq \Sigma_0$.

Now we insert two rules to add blank symbols to our representation of the tape:

$$\mathsf{f}_q(xs, \mathsf{nil}) \to \mathsf{f}_q(xs, \varepsilon : \mathsf{nil}) \qquad \mathsf{f}_q(\mathsf{nil}, ys) \to \mathsf{f}_q(\varepsilon : \mathsf{nil}, ys)$$

Depending on the movement of the Turing machine, we insert corresponding rules for the step function.

left movement: Let $\delta(q, \mathsf{a}) = (q', \mathsf{left}, \mathsf{b})$. Then we insert the following rule into the TRS:

$$\mathsf{f}_q(\mathsf{a} : l, x : r) \to \mathsf{f}_{q'}(l, \mathsf{b} : x : r)$$

right movement: Let $\delta(q, \mathsf{a}) = (q', \mathsf{right}, \mathsf{b})$. Then we insert the following rule into the TRS:

$$\mathsf{f}_q(\mathsf{a} : l, x : r) \to \mathsf{f}_{q'}(x : \mathsf{b} : l, r)$$

To easily detect termination of the Turing machine, we additionally insert the rule $\mathsf{f}_{q_e}(x, y) \to \mathsf{term}$. Moreover, we add the "initial" rule $\mathsf{init} \to \mathsf{f}_{q_s}(\mathsf{nil}, \mathsf{nil})$.

b)* First we notice that half of the work is already done: *If* the constructed TRS from the previous exercise part terminates, then every computation starting from the initial state also terminates. Thus, the Turing machine terminates.

However, the other direction does not hold. One possible solution is to simulate the Turing machine in an incremental way. We simulate the computation of the Turing machine only for at most i steps. If the Turing machine terminates within i steps, we are done. Otherwise we re-start the simulation from the initial configuration and try to perform the simulation with $i+1$ steps. This way, we ensure that we only simulate a single step from unreachable system configurations before we re-start the simulation from the initial state. Hence, we avoid the problem of unreachable system configurations and we have that if the Turing machine is terminating, then the constructed TRS is also terminating.

Formally, we now construct the corresponding TRS as follows: For each rule $l \to r$ in the TRS from the previous exercise part, we have the rule

$$\mathsf{trans}(l, y, \mathsf{s}(x)) \to \mathsf{trans}(r, y, x)$$

in our new TRS. Thus, x is our counter. We need y to store the current maximal number of simulation steps. Furthermore, we need the rule

$$\mathsf{trans}(x, y, 0) \to \mathsf{trans}(\mathsf{f}_{q_s}(\mathsf{nil}, \mathsf{nil}), \mathsf{s}(y), \mathsf{s}(y))$$

for the re-start process.
