assignment07

June 23, 2019

1 Functional Programming SS19

2 Assignment 07 Solutions

2.0.1 Exercise 1 (Transformation to Simple Haskell)

The data structure `List a` *is defined by*

```
data List a = Nil | Cons a (List a)
```

Transform the following Haskell-expression to an equivalent simple Haskell-expression using the transformation rules (1)-(9) from the lecture. Please give all intermediate expressions and indicate in each step which transformation rule was used. You may apply rules to several identical subterms simultaneously.

```
let length Nil = 0
    length (Cons x xs) = 1 + length xs
in length xs
```

Please simplify the intermediate expressions according to the following rules:

(i) `if (isa`$_{ntuple}$ `exp) then exp1 else exp2` should be replaced by `exp1` for any $n \leq 0$

(ii) `(\var -> exp1) exp` should be replaced by `exp1'`, where `exp1'` results from `exp1` by replacing all (free) occurrences of var by `exp`.

```
let length Nil = 0
    length (Cons x xs) = 1 + length xs
in length xs
```

(1) $\Longrightarrow$

```
let length = \x1 -> case x1 of Nil -> 0
                               (Cons x xs) -> 1 + length xs
in length xs
```

(4) $\Longrightarrow$

```
let length = \x1 -> match Nil x1 0
                    (match (Cons x xs) (1 + length xs) bot)
in length xs
```

(7) $\Longrightarrow$

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then (match () (argof_{Nil} x1) 0
                           (if (isa_{Cons} x1)
                            then match (x, xs)
                                       (argof_{Cons} x1)
                                       (1 + length xs)
                                       bot)
                            else bot))
               else (if (isa_{Cons} x1)
                     then (match (x, xs)
                                 (argof_{Cons} x1)
                                 (1 + length xs)
                                 bot)
                     else bot)
in length xs
```

(8) $\Longrightarrow$

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then (if (isa_{0-tuple} (argof_{Nil} x1))
                     then 0
                     else (if (isa_{Cons} x1)
                            then match (x, xs)
                                       (argof_{Cons} x1)
                                       (1 + length xs)
                                       bot)
                            else bot))
               else (if (isa_{Cons} x1)
                     then (match (x, xs)
                                 (argof_{Cons} x1)
                                 (1 + length xs)
                                 bot)
                     else bot)
in length xs
```

(9) $\Longrightarrow$

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then (if (isa_{0-tuple} (argof_{Nil} x1))
                     then 0
                     else (if (isa_{Cons} x1)
                            then (if (isa_{2-tuple} (argof_{Cons} x1))
                                  then match x (sel_{2,1} (argof_{Cons} x1))
                                       (match xs (sel_{2,2} (argof_{Cons} x1))
```

```
                                        (1 + length xs)
                                        bot))
                             else bot))
               else (if (isa_{Cons} x1)
                     then (if (isa_{2-tuple} (argof_{Cons} x1))
                           then match x (sel_{2,1} (argof_{Cons} x1))
                                (match xs (sel_{2,2} (argof_{Cons} x1))
                                     (1 + length xs)
                                     bot))
                     else bot)
in length xs
```

Since we know that `isa_{0-tuple} (argof_{Nil} x1)` is indeed the case based on the given definition of `List`, we can replace this expression by `0`. Similarly, we know that `isa_{2-tuple} (argof_{Cons} x1)` is also the case. We thus have

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then 0
               else (if (isa_{Cons} x1)
                     then match x (sel_{2,1} (argof_{Cons} x1))
                          (match xs (sel_{2,2} (argof_{Cons} x1))
                               (1 + length xs)
                               bot)
                     else bot)
in length xs
```

(7) $\Longrightarrow$

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then 0
               else (if (isa_{Cons} x1)
                     then (\x -> (\xs -> (1 + length xs))
                                    (sel_{2,2} (argof_{Cons} x1)))
                              (sel_{2,1} (argof_{Cons} x1)))
in length xs
```

We can now apply the second hint two successive times. After the first application to the innermost lambda, we have:

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then 0
               else (if (isa_{Cons} x1)
                     then (\x -> (1 + length (sel_{2,2} (argof_{Cons} x1))))
                              (sel_{2,1} (argof_{Cons} x1)))
in length xs
```

After the second application, we obtain:

```
let length = \x1 ->
               if (isa_{Nil} x1)
               then 0
               else (if (isa_{Cons} x1)
                     then (1 + length (sel_{2,2} (argof_{Cons} x1))))
in length xs
```

None of the reduction rules are applicable anymore, so this is our resulting simple Haskell expression.

2.0.2 Exercise 2 (Substitutions)

Identify the free variables of the following lambda terms t_1, t_2, and t_3, and also apply the three substitutions $\sigma_1 = [x/\lambda x.x\ y]$, $\sigma_2 = [y/v\ y]$, and $\sigma_3 = [z/\lambda z.x\ z]$ respectively, to each of these terms (so that you obtain in total 9 possibly different lambda terms $t_i\sigma_j$ with $i, j \in \{1, 2, 3\}$):

a) $t_1 = \lambda y.x\ (\lambda x.y\ x)$

b) $t_2 = \lambda x.(\lambda y.x\ y)\ x\ y$

c) $t_3 = \lambda y.(\lambda z.z\ x\ y)\ z\ y$

Before we start with the identification of the free variables of t_1, t_2, and t_3 and the substitutions, let us identify the free variables of the terms that will be substituted.

$$\begin{aligned}
\text{free}(\lambda x.x\ y) &= \text{free}(x\ y) \setminus \{x\} \\
&= (\text{free}(x) \cup \text{free}(y)) \setminus \{x\} \\
&= (\{x\} \cup \{y\}) \setminus \{x\} \\
&= \{x, y\} \setminus \{x\} \\
&= \{y\}
\end{aligned}$$

$$\begin{aligned}
\text{free}(v\ y) &= \text{free}(v) \cup \text{free}(y) \\
&= \{v\} \cup \{y\} \\
&= \{v, y\}
\end{aligned}$$

$$\begin{aligned}
\text{free}(\lambda z.x\ z) &= \text{free}(x\ z) \setminus \{z\} \\
&= (\text{free}(x) \cup \text{free}(z)) \setminus \{z\} \\
&= (\{x\} \cup \{z\}) \setminus \{z\} \\
&= \{x, z\} \setminus \{z\} \\
&= \{x\}
\end{aligned}$$

a) $t_1 = \lambda y.x\ (\lambda x.y\ x)$

$$
\begin{aligned}
\text{free}(\lambda y.x\ (\lambda x.y\ x)) &= \text{free}(x\ (\lambda x.y\ x)) \setminus \{y\} \\
&= (\text{free}(x) \cup \text{free}(\lambda x.y\ x)) \setminus \{y\} \\
&= (\{x\} \cup \text{free}(\lambda x.(y\ x))) \setminus \{y\} \\
&= (\{x\} \cup \text{free}(y\ x) \setminus \{x\}) \setminus \{y\} \\
&= (\{x\} \cup (\text{free}(y) \cup \text{free}(x)) \setminus \{x\}) \setminus \{y\} \\
&= (\{x\} \cup (\{y\} \cup \{x\}) \setminus \{x\}) \setminus \{y\} \\
&= (\{x\} \cup \{x, y\} \setminus \{x\}) \setminus \{y\} \\
&= (\{x, y\} \setminus \{x\}) \setminus \{y\} \\
&= \{y\} \setminus \{y\} \\
&= \varnothing
\end{aligned}
$$

Applying the substitution rules in definition 3.1.3, we have:

$$
\begin{aligned}
t_1\sigma_1 &= (\lambda y.x\ (\lambda x.y\ x))[x/\lambda x.x\ y] \\
&= \lambda y'.((x\ (\lambda x.y\ x))[y/y'][x/\lambda x.x\ y]) \\
&= \lambda y'.((x)[y/y'][x/\lambda x.x\ y]\ (\lambda x.y\ x)[y/y'][x/\lambda x.x\ y]) \\
&= \lambda y'.(\lambda x.x\ y)\ (\lambda x.y'\ x)
\end{aligned}
$$

$$
\begin{aligned}
t_1\sigma_2 &= (\lambda y.x\ (\lambda x.y\ x))[y/v\ y] \\
&= \lambda y.x\ (\lambda x.y\ x)
\end{aligned}
$$

$$
\begin{aligned}
t_1\sigma_3 &= (\lambda y.x\ (\lambda x.y\ x))[z/\lambda z.x\ z] \\
&= \lambda y.((x\ (\lambda x.y\ x))[z/\lambda z.x\ z]) \\
&= \lambda y.((x)[z/\lambda z.x\ z]\ (\lambda x.y\ x)[z/\lambda z.x\ z]) \\
&= \lambda y.x\ (\lambda x.y\ x)
\end{aligned}
$$

b) $t_2 = \lambda x.(\lambda y.x\ y)\ x\ y$

$$
\begin{aligned}
\mathrm{free}(\lambda x.(\lambda y.x\ y)\ x\ y) &= \mathrm{free}(\lambda x.((\lambda y.x\ y)\ x)\ y) \\
&= \mathrm{free}(((\lambda y.x\ y)\ x)\ y) \setminus \{x\} \\
&= (\mathrm{free}((\lambda y.x\ y)\ x) \cup \mathrm{free}(y)) \setminus \{x\} \\
&= (\mathrm{free}((\lambda y.x\ y)\ x) \cup \{y\}) \setminus \{x\} \\
&= ((\mathrm{free}(\lambda y.x\ y) \cup \mathrm{free}(x)) \cup \{y\}) \setminus \{x\} \\
&= (((\mathrm{free}(x\ y) \setminus \{y\}) \cup \{x\}) \cup \{y\}) \setminus \{x\} \\
&= ((((\mathrm{free}(x) \cup \mathrm{free}(y)) \setminus \{y\}) \cup \{x\}) \cup \{y\}) \setminus \{x\} \\
&= ((((\{x\} \cup \{y\}) \setminus \{y\}) \cup \{x\}) \cup \{y\}) \setminus \{x\} \\
&= (((\{x,y\} \setminus \{y\}) \cup \{x\}) \cup \{y\}) \setminus \{x\} \\
&= ((\{x\} \cup \{x\}) \cup \{y\}) \setminus \{x\} \\
&= (\{x\} \cup \{y\}) \setminus \{x\} \\
&= \{x,y\} \setminus \{x\} \\
&= \{y\}
\end{aligned}
$$

For the substitutions, we have:

$$
\begin{aligned}
t_2\sigma_1 &= (\lambda x.(\lambda y.x\ y)\ x\ y)[x/\lambda x.x\ y] \\
&= \lambda x.(\lambda y.x\ y)\ x\ y
\end{aligned}
$$

$$
\begin{aligned}
t_2\sigma_2 &= (\lambda x.(\lambda y.x\ y)\ x\ y)[y/v\ y] \\
&= \lambda x.(((\lambda y.x\ y)\ x\ y)[y/v\ y]) \\
&= \lambda x.((((\lambda y.x\ y)\ x)\ y)[y/v\ y]) \\
&= \lambda x.(((\lambda y.x\ y)\ x)[y/v\ y]\ (y)[y/v\ y]) \\
&= \lambda x.(((\lambda y.x\ y)[y/v\ y]\ (x)[y/v\ y])\ v\ y) \\
&= \lambda x.((\lambda y.x\ y)\ x)\ v\ y \\
&= \lambda x.(\lambda y.x\ y)\ x\ v\ y
\end{aligned}
$$

$$
\begin{aligned}
t_2\sigma_3 &= (\lambda x.(\lambda y.x\ y)\ x\ y)[z/\lambda z.x\ z] \\
&= \lambda x.(((\lambda y.x\ y)\ x\ y)[z/\lambda z.x\ z]) \\
&= \lambda x.((((\lambda y.x\ y)\ x)\ y)[z/\lambda z.x\ z]) \\
&= \lambda x.(((\lambda y.x\ y)\ x)[z/\lambda z.x\ z]\ (y)[z/\lambda z.x\ z]) \\
&= \lambda x.(((\lambda y.x\ y)[z/\lambda z.x\ z]\ (x)[z/\lambda z.x\ z])\ y) \\
&= \lambda x.(((\lambda y.((x\ y)[z/\lambda z.x\ z]))\ x)\ y) \\
&= \lambda x.((\lambda y.((x)[z/\lambda z.x\ z]\ (y)[z/\lambda z.x\ z])\ x)\ y) \\
&= \lambda x.(((\lambda y.(x\ y))\ x)\ y) \\
&= \lambda x.((\lambda y.(x\ y))\ x)\ y \\
&= \lambda x.(\lambda y.(x\ y))\ x\ y \\
&= \lambda x.(\lambda y.x\ y)\ x\ y
\end{aligned}
$$

c) $t_3 = \lambda y.(\lambda z.z\ x\ y)\ z\ y$

$$
\begin{aligned}
\text{free}(\lambda y.(\lambda z.z\ x\ y)\ z\ y) &= \text{free}(\lambda y.((\lambda z.z\ x\ y)\ z)\ y) \\
&= \text{free}((\lambda z.(z\ x\ y)\ z)\ y) \setminus \{y\} \\
&= \text{free}(((\lambda z.(z\ x)\ y)\ z)\ y) \setminus \{y\} \\
&= (\text{free}((\lambda z.(z\ x)\ y)\ z) \cup \text{free}(y)) \setminus \{y\} \\
&= ((\text{free}(\lambda z.(z\ x)\ y) \cup \text{free}(z)) \cup \{y\}) \setminus \{y\} \\
&= (((\text{free}((z\ x)\ y) \setminus \{z\}) \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= ((((\text{free}(z\ x) \cup \text{free}(y)) \setminus \{z\}) \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= (((((\text{free}(z) \cup \text{free}(x)) \cup \{y\}) \setminus \{z\}) \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= (((((\{z\} \cup \{x\}) \cup \{y\}) \setminus \{z\}) \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= ((((\{x, z\} \cup \{y\}) \setminus \{z\}) \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= (((\{x, y, z\} \setminus \{z\}) \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= ((\{x, y\} \cup \{z\}) \cup \{y\}) \setminus \{y\} \\
&= (\{x, y, z\} \cup \{y\}) \setminus \{y\} \\
&= \{x, y, z\} \setminus \{y\} \\
&= \{x, z\}
\end{aligned}
$$

The substitutions are given as follows:

$$
\begin{aligned}
t_3\sigma_1 &= (\lambda y.(\lambda z.z\ x\ y)\ z\ y)[x/\lambda x.x\ y] \\
&= \lambda y'.(((\lambda z.z\ x\ y')\ z\ y')[x/\lambda x.x\ y]) \\
&= \lambda y'.((((\lambda z.z\ x\ y')\ z)\ y')[x/\lambda x.x\ y]) \\
&= \lambda y'.(((\lambda z.z\ x\ y')\ z)[x/\lambda x.x\ y]\ (y')[x/\lambda x.x\ y]) \\
&= \lambda y'.(((\lambda z.z\ x\ y')[x/\lambda x.x\ y]\ (z)[x/\lambda x.x\ y])\ y') \\
&= \lambda y'.((\lambda z.(z\ x\ y')[x/\lambda x.x\ y]\ z)\ y') \\
&= \lambda y'.((\lambda z.((z\ x)\ y')[x/\lambda x.x\ y]\ z)\ y') \\
&= \lambda y'.((\lambda z.((z\ x)[x/\lambda x.x\ y]\ (y')[x/\lambda x.x\ y])\ z)\ y') \\
&= \lambda y'.((\lambda z.(((z)[x/\lambda x.x\ y]\ (x)[x/\lambda x.x\ y])\ y')\ z)\ y') \\
&= \lambda y'.((\lambda z.((z\ (\lambda x.x\ y))\ y')\ z)\ y') \\
&= \lambda y'.((\lambda z.(z\ (\lambda x.x\ y)\ y')\ z)\ y') \\
&= \lambda y'.(\lambda z.(z\ (\lambda x.x\ y)\ y')\ z)\ y' \\
&= (\lambda y'.(\lambda z.z\ (\lambda x.x\ y)\ y')\ z)\ y' \\
&= \lambda y'.(\lambda z.z\ (\lambda x.x\ y)\ y')\ z\ y'
\end{aligned}
$$

$$
\begin{aligned}
t_3\sigma_2 &= (\lambda y.(\lambda z.z\ x\ y)\ z\ y)[y/v\ y] \\
&= \lambda y.(\lambda z.z\ x\ y)\ z\ y
\end{aligned}
$$

$$
\begin{aligned}
t_3\sigma_3 &= (\lambda y.(\lambda z.z\ x\ y)\ z\ y)[z/\lambda z.x\ z] \\
&= \lambda y.(((\lambda z.z\ x\ y)\ z\ y)[z/\lambda z.x\ z]) \\
&= \lambda y.((((\lambda z.z\ x\ y)\ z)\ y)[z/\lambda z.x\ z]) \\
&= \lambda y.(((\lambda z.z\ x\ y)\ z)[z/\lambda z.x\ z]\ (y)[z/\lambda z.x\ z]) \\
&= \lambda y.(((\lambda z.z\ x\ y)[z/\lambda z.x\ z]\ (z)[z/\lambda z.x\ z])\ y) \\
&= \lambda y.(((\lambda z.z\ x\ y)\ (\lambda z.x\ z))\ y) \\
&= \lambda y.((\lambda z.z\ x\ y)\ (\lambda z.x\ z)\ y) \\
&= \lambda y.(\lambda z.z\ x\ y)\ (\lambda z.x\ z)\ y
\end{aligned}
$$

2.0.3 Exercise 3 (β-reduction)

a) *Give all reduction sequences with $\rightarrow_\beta$ starting with the following term:*

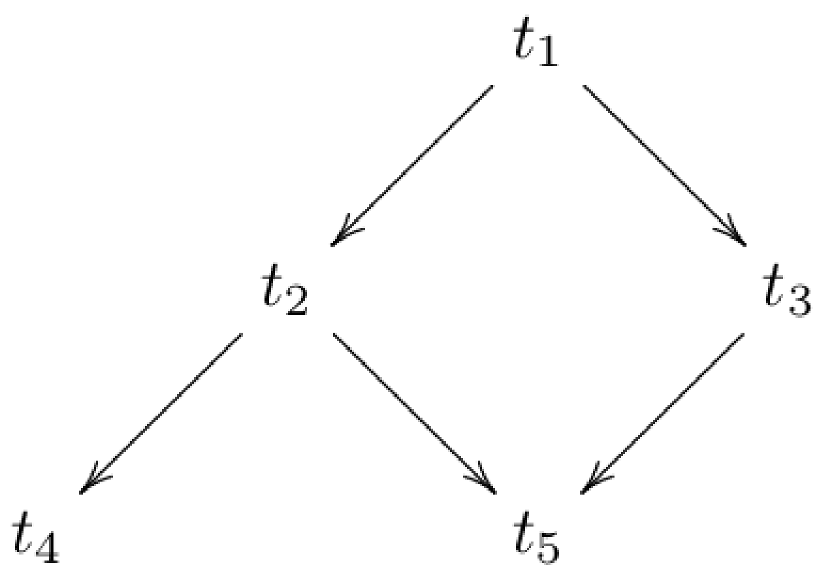
$$(\lambda x\ y.x\ y)\ \text{exp}\ ((\lambda x\ z.\text{mult}\ x\ (\text{mult}\ z\ z))\ 3\ 2)$$

b) *Give all reduction sequences $\rightarrow_\beta$ with at most 3 steps starting with the following term:*

$$(\lambda f.(\lambda x.\ f\ (x\ x))\ (\lambda x.\ f\ (x\ x)))\ (\lambda y.y)$$

Hints:

- *You can save space by representing the reduction sequences as directed graphs. For example, $t_1 \rightarrow_\beta t_2 \rightarrow_\beta t_4$, $t_1 \rightarrow_\beta t_2 \rightarrow_\beta t_5$, and $t_1 \rightarrow_\beta t_3 \rightarrow_\beta t_5$ can be represented as:*



reduction_sequence_graph

- *In a), you may abbreviate* `exp` *to* `e` *and* `mult` *to* `m` *to save space.*
- *In b), you may write A_f instead of $\lambda x.f\ \ (x\ \ x)$, B instead of $\lambda y.y$ and C instead of $\lambda x.x\ \ x$ to save space. The start term could then be represented as $(\lambda f.A_f\ \ A_f)\ B$. Furthermore, the term $\lambda x.(\lambda y.y)\ (x\ x)$ can be abbreviated to A_B.*

a) The graph showing the $\rightarrow_\beta$ reduction sequences for $(\lambda x\ y.x\ y)$ exp $((\lambda x\ z.\text{mult}\ x\ (\text{mult}\ z\ z))\ 3\ 2)$ is given below.

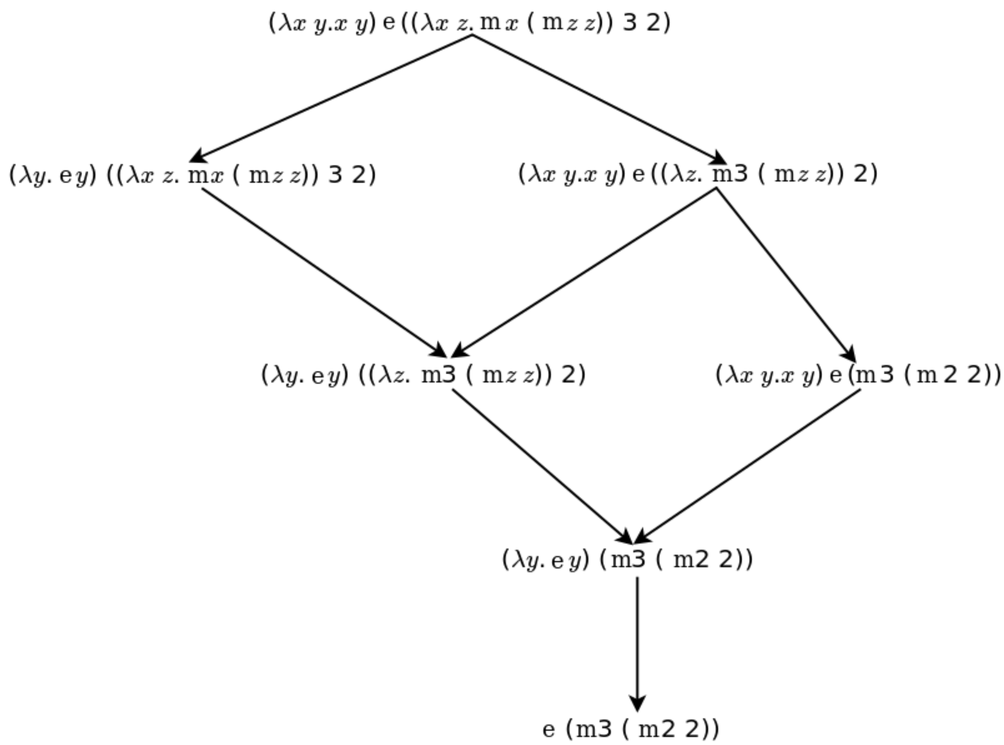
2.0.4 Exercise 4 (Confluence)

a) *Let $\mathbb{N} = \{0, 1, 2, ...\}$ be the natural numbers with the standard relation $>$. Indicate whether the following properties hold. Explain your solution!*

- *For any $n \in \mathbb{N}$ there is an $m \in \mathbb{N}$ such that m is a normal form of n w.r.t. $>$.*
- *$>$ is confluent.*

Based on definition 3.2.3, the transitive-reflexive closure of $>$ is the relation $\geq$ since for any $t_1, t_2, t_3 \in \mathbb{N}$, it holds that * $t_1 > t_2 \implies t_1 \geq t_2$ * $t_1 > t_2 \geq t_3 \implies t_1 \geq t_3$ * $t_1 \geq t_1$

According to the same definition, a number $m \in \mathbb{N}$ is a normal form of $n \in \mathbb{N}$ w.r.t $>$ iff $n \geq m$ and m is a normal form, which is the case iff there is no $q \in \mathbb{N}$ for which $m > q$. The normal form of $\mathbb{N}$ is $m = 0$ since there is no $q \in \mathbb{N}$ such that $m > q$. Since it also holds that $n \geq 0$ for any $n \in \mathbb{N}$, we see that the first property holds. □



3a_beta_reduction_sequences

$>$ is confluent if for every $t, q_1, q_2 \in \mathbb{N}$ we have that, whenever $t \geq q_1$ and $t \geq q_2$, there is a $q \in \mathbb{N}$ such that $g_1 \geq q$ and $g_2 \geq q$. Let us investigate four exhaustive cases to show that the confluence property holds:

1. If $t = q_1 = q_2$, the property clearly holds since we can then take $q = t$ to satisfy it.
2. Without loss of generality, if we assume $t = q_1 \neq q_2$, we can take $q = q_2$ to satisfy it since $q_1 = t \geq q$ and $q_2 = q \geq q$.
3. If $q_1 = q_2 \neq t$, we can, without loss of generality, take $q = q_1$ to satisfy the property since $q_1 = q \geq q$ and $q_2 = q \geq q$.
4. In the most general case, $t \neq q_1 \neq q_2$. Since we know that $>$ has a normal form for any $n \in \mathbb{N}$, we can satisfy the property by taking $q = 0$ since $q_1 \geq 0$ and $q_2 \geq 0$.

This proves that $>$ is confluent. $\square$

b) *Let $\mathbb{N} = \{0, 1, 2, ...\}$ be the natural numbers with the following relation $\succ$: $m \succ n$ if there is some $0 < r \in \mathbb{N}$ with $m = r \cdot n$, i.e. n divides m. Indicate whether the following properties hold. Explain your solution!*

- *For any $n \in \mathbb{N}$ there is an $m \in \mathbb{N}$ such that m is a normal form of n w.r.t. $\succ$.*
- *$\succ$ is confluent.*

The relation $\succ$ is its own transitive-reflexive closure since for any $t_1, t_2, t_3 \in \mathbb{N}$, we have:

- $t_1 \succ t_2 \implies t_2 \succ t_2$

- $t_1 \succ t_2 \succ t_3 \implies t_1 \implies t_3$ (proof: if we write $t_1 = mt_2$ and $t_2 = nt_3$ for some $0 < m, n \in \mathbb{N}$, then $t_1 = mt_2 = m(nt_3) = kt_3$ where $k = mn$).
- $t_1 \succ t_1$ (this clearly holds for any $m > 0$ since any number is divisible by itself; for the given definition of $\succ$, this also holds for $m = 0$ since we can then take any $0 < r \in \mathbb{N}$ to satisfy the definition)

$m \in \mathbb{N}$ is a normal form of $n \in \mathbb{N}$ w.r.t. $>$ iff $n \succ m$ and m is a normal form. m will be a normal form iff a $q \in \mathbb{N}$ for which $m \succ q$ does not exist; however, as we know from before, $\succ$ is reflexive and so for any m, we can take $q = m$ to satisfy the relation. Thus, m is not a normal form and can therefore not be a normal form of n. $\square$

The relation $\succ$ will be confluent if for any $t, q_1, q_2 \in \mathbb{N}$ such that $t \succ q_1$ and $t \succ q_2$, we can find a $q \in \mathbb{N}$ for which $q_1 \succ q$ and $q_2 \succ q$. This clearly holds since we can take $q = 1$ to satisfy the relation for any arbitrary t, q_1, q_2; thus, $\succ$ is confluent. $\square$

c) *Let $\mathbb{Z}$ be the integers with the standard relation $>$. Indicate whether the following properties hold. Explain your solution!*

- *For any $z \in \mathbb{Z}$ there is a $q \in \mathbb{Z}$ such that q is a normal form of z w.r.t. $>$.*
- *$>$ is confluent.*

As in part a), the transitive-reflexive closure of $>$ will be the relation $\geq$.

A number $q \in \mathbb{Z}$ is a normal form of $z \in \mathbb{Z}$ w.r.t. $>$ iff $z \geq q$ and q is a normal form. Now, q is a normal form if there is no $q' \in \mathbb{Z}$ such that $q > q'$; however, for any q, we can always take $q' = q - 1 \in \mathbb{Z}$ so that $q > q'$. This means that q cannot be a normal form and is thus not a normal form of z w.r.t. $>$.

Let's proceed to show that q is confluent as in part a), namely by considering $t, q_1, q_2 \in \mathbb{Z}$.

1. The first three cases, namely $t = q_1 = q_2$, $t = q_1 \neq q_2$, and $q1 = q_2 \neq t$, can be treated the same as in a)
2. For the most general case of $t \neq q_1 \neq q_2$, let us, without loss of generality, assume that $q_1 \geq q_2$. Then, we can take $q = q_2 - 1$, in which case we have $q_1 \geq q$ and $q_2 \geq q$.

Thus, $>$ is confluent. $\square$

d) *Let $\mathrm{Pot}_{\neq\varnothing}(\mathbb{Z})$ be the set of nonempty subsets of the integers with the relation $\subsetneq$: $A \subsetneq B$ whenever $A \neq B$ and A is a subset of B. Indicate whether the following properties hold. Explain your solution!*

- *For any $\varnothing \neq A \subseteq \mathbb{Z}$ there is a $B \subseteq \mathbb{Z}$ such that B is a normal form of A w.r.t. $\subsetneq$.*
- *$\subsetneq$ is confluent.*

The transitive-reflexive closure of $\subsetneq$ will be the relation $\subseteq$ since for any $A, B, C \subseteq \mathbb{Z}$, we have:

- $A \subsetneq B \implies A \subseteq B$
- $A \subsetneq B \subseteq C \implies A \subseteq C$
- $A \subseteq A$

A set $\varnothing \neq B \subseteq Z$ will be a normal form of $\varnothing \neq A \subseteq Z$ iff $A \subseteq B$ and B is a normal form, which is the case if there is no $\varnothing \neq C \subseteq Z$ such that $B \subsetneq C$. The normal form of $\mathbb{Z}$ is $B = \mathbb{Z}$ since there is then no $C \subseteq \mathbb{Z}$ with $B \subsetneq C$. Since every $A \subseteq \mathbb{Z} \subseteq B = \mathbb{Z}$, the first property holds. $\square$

The relation $\subsetneq$ is confluent if for any $A, B, C \subseteq \mathbb{Z}$, whenever $A \subseteq B$ and $A \subseteq C$, then there is a $Q \subseteq \mathbb{Z}$ such that $B \subseteq Q$ and $C \subseteq Q$. This clearly holds since we can take $Q = \mathbb{Z}$, in which case $B \subseteq Q$ and $C \subseteq Q$. Thus, $\subsetneq$ is confluent. $\square$