11. Term rewriting programming

11.1. Computing with TRSs and rewrite strategies

TRSs can be used as computing frameworks. Indeed, if $\mathcal{T} := (\mathcal{S}, \mathsf{V}, \to)$ is a TRS,

- rewrite relations $\Rightarrow$ describe how to make a **computation progress**: if $t \Rightarrow t'$, where t and t' are $\mathcal{S}, \mathsf{V}$-terms, t' is the *next step of computation* from t;
- normal forms of $\mathcal{T}$ are the **values** of the computation: if $t' \in t^{\Rightarrow}$ where t is an $\mathcal{S}, \mathsf{V}$-term, then t' is a *computation result* of t.

Thus, TRSs form a **programming paradigm** where $\mathcal{S}$ encodes what are the **syntactically valid expressions** and V encodes what are the **allowed variables**.

Example

Let $\mathcal{T} := (\mathcal{S}, \mathbb{N}, \to)$ be the TRS such that $\mathcal{S}$ is the signature containing three nullary constants **true**, **false**, and **zero**, two unary constants **succ** and even, and a ternary constant if.
Let $\to$ be the elementary rewrite relation defined by $\mathsf{if}\,\mathbf{true}\,1\,2 \to 1$, $\mathsf{if}\,\mathbf{false}\,1\,2 \to 2$, $\mathsf{even}\,\mathbf{zero} \to \mathbf{true}$, $\mathsf{even}\,(\mathbf{succ}\,\mathbf{zero}) \to \mathbf{false}$, and $\mathsf{even}\,(\mathbf{succ}\,(\mathbf{succ}\,1)) \to \mathsf{even}\,1$.
Then, the $\mathcal{S}, \mathbb{N}$-term $t := \mathsf{if}\,(\mathsf{even}\,(\mathbf{succ}\,(\mathbf{succ}\,(\mathbf{succ}\,\mathbf{zero}))))\,t_1\,t_2$ encodes the expression

```
if even 3 then t1 else t2.
```

Its computation in $\mathcal{T}$ is

$$t \;\Rightarrow\; \mathsf{if}\,(\mathsf{even}\,(\mathbf{succ}\,\mathbf{zero}))\,t_1\,t_2 \;\Rightarrow\; \mathsf{if}\,\mathbf{false}\,t_1\,t_2 \;\Rightarrow\; t_2.$$

Hence, a normal form of t_2 is a computation result of t.

There are two main approaches:

1. a **program** is encoded by a TRS $\mathcal{T} := (\mathcal{S}, \mathsf{V}, \to)$ where the elementary rewrite relation $\to$ represents a **set of instructions**, and **input data** is an $\mathcal{S}, \mathsf{V}$-term t;
2. a TRS $\mathcal{T} := (\mathcal{S}, \mathsf{V}, \to)$ is fixed once and for all, and an $\mathcal{S}, \mathsf{V}$-term t encodes **both a program and input data**.

Here are some remarks about these two approaches when working with a TRS $\mathcal{T} := (\mathcal{S}, \mathsf{V}, \to)$:

- when $\mathcal{T}$ is **not terminating**, for some input, the computation can **diverge**;
- when $\mathcal{T}$ is **not confluent**, for some input, the computation can be **non-deterministic**;
- when t is a terminating $\mathcal{S}, \mathsf{V}$-term, the computation of a result from t in $\mathcal{T}$ may require **more or fewer computation steps**.

Definition

A *rewrite strategy* on an ARS $\mathcal{A} := (X, \Rightarrow)$ is a partial function s from X to X such that

1. for any $x \in X$, $s{\cdot}x$ is defined iff x is not a normal form of $\mathcal{A}$;
2. if $s{\cdot}x$ is defined, then $x \Rightarrow s{\cdot}x$.

Let $\mathcal{A} := (X, \Rightarrow)$ be an ARS.

For any strategy s on $\mathcal{A}$ and any $x, x' \in X$, we write $x \Rightarrow_s x'$ for the fact that $x' = s{\cdot}x$. In particular, $\Rightarrow_s$ is a rewrite relation on X. Moreover, for any $x \in X$, let $s^*{\cdot}x := x^{\Rightarrow_s{}^*}$.

A rewrite strategy s on $\mathcal{A}$ is *normalizing* for $x \in X$ if the set $s^*{\cdot}x$ contains a normal form of $\mathcal{A}$. When s is normalizing for all $x \in X$, s is *normalizing*.

Given two rewrite strategies s and s' on $\mathcal{A}$ and $x \in X$, s *dominates* s' for x if

- either s is normalizing for x and s' is not normalizing for x;
- or both s and s' are normalizing for x and $\#_{\underline{\lfloor s^*{\cdot}x \rfloor}} \leqslant \#_{\underline{\lfloor s'^*{\cdot}x \rfloor}}$.

When s dominates s' for all $x \in X$, s *dominates* s'.

Example

Let the ARS $\mathcal{A} := (\mathbb{N} \setminus \{0\}, \Rightarrow)$ where $\Rightarrow$ is defined by $n \Rightarrow k$ for any $n, k \in \mathbb{N}$ such that k is a proper divisor of n. The only normal form of $\mathcal{A}$ is 1.
Let s be the strategy on $\mathcal{A}$ such that for any $n \in \mathbb{N}$, $s{\cdot}n$ is the greatest proper divisor of n. For instance, $s{\cdot}123 = 41$ since $123 = 3 \times 41$. Since $\mathcal{A}$ is normalizing, s is normalizing.

Example

Let the ARS $\mathcal{A} := (\mathbb{N}, \Rightarrow)$ where $\Rightarrow$ is defined by $n \Rightarrow k$ for any $n, k \in \mathbb{N}$ such that n is even and $k \geqslant n$. The normal forms of $\mathcal{A}$ are the odd natural numbers. Moreover, since for any $n \in \mathbb{N}$ such that n is even, we have that $n \Rightarrow n+1$ and that $n+1$ is odd, so a normal form, $\mathcal{A}$ is normalizing.

- Let s_1 be the strategy on $\mathcal{A}$ such that for any $n \in \mathbb{N}$ such that n is even, $s_1{\cdot}n := 2n$. This strategy is not normalizing.
- Let s_2 be the strategy on $\mathcal{A}$ such that for any $n \in \mathbb{N}$ such that n is even, $s_2{\cdot}n := n+1$. This strategy is normalizing.
- Let s_3 be the strategy on $\mathcal{A}$ such that for any $n \in \mathbb{N}$, if n is divisible by 4, then $s_3{\cdot}n := n+1$, and if n is divisible by 2 but not by 4, $s_3{\cdot}n := 2n$. This strategy is normalizing.

The strategy s_1 is dominated by s_2, and s_3 is dominated by s_2.

Let $\mathcal{T} := (\mathcal{S}, \mathcal{V}, \to)$ be a TRS and $\leqslant$ be a total order relation on the rewrite rules of $\mathcal{T}$.

- The *left* (resp. *right*) *$\leqslant$-innermost rewrite strategy* is the rewrite strategy $s_{\mathrm{i}}^{\leqslant}$ such that for any $t \in \mathfrak{T}\cdot\mathcal{S}\cdot\mathcal{V}$, if t is not a normal form of $\mathcal{T}$, then $s_{\mathrm{i}}^{\leqslant}\cdot t := t'$ where t' is obtained from t by modifying the leftmost (resp. rightmost) factor such that its root has no proper descendent on which a one-step rewrite can be made.

  This rewrite strategy is analogous to the call-by-value evaluation strategy of programming languages.

- The *left* (resp. *right*) *$\leqslant$-outermost rewrite strategy* is the rewrite strategy $s_{\mathrm{o}}^{\leqslant}$ such that for any $t \in \mathfrak{T}\cdot\mathcal{S}\cdot\mathcal{V}$, if t is not a normal form of $\mathcal{T}$, then $s_{\mathrm{o}}^{\leqslant}\cdot t := t'$ where t' is obtained from t by modifying the leftmost (resp. rightmost) factor such that its root has no proper ancestor on which a one-step rewrite can be made.

  This rewrite strategy is analogous to the call-by-name evaluation strategy of programming languages.

If several rewrite rules of $\mathcal{T}$ can be considered at a given position, the smallest w.r.t. $\leqslant$ is considered.

Exercise ●●●○○

Provide a description of the rewrite strategies $s_{\mathrm{i}}^{\leqslant}$ and $s_{\mathrm{o}}^{\leqslant}$ by using rewrite positions and properties on these (like lexicographic order).

Example

Let the TRS $\mathcal{T} := (\mathcal{S}_{\mathbb{N}^2}, \mathbb{N}, \to)$ such that $r_1 := \mathsf{c}_2 1 \mathsf{c}_1 \to 1$, $r_2 := \mathsf{c}_2 12 \to \mathsf{c}_0$, and $r_3 := \mathsf{c}_2 12 \to \mathsf{c}_2 1 \mathsf{c}_0$. Let the total order relation $\leqslant$ on the rewrite rules of $\mathcal{T}$ such that $r_1 \leqslant r_2 \leqslant r_3$.
Let

$$\mathrm{t} := \mathsf{c}_3 \lfloor \mathsf{c}_2 \lfloor \mathsf{c}_2 23 \rfloor \lfloor \mathsf{c}_1 \mathsf{c}_0 \rfloor \rfloor 2 \lfloor \mathsf{c}_2 2 \lfloor \mathsf{c}_1 1 \rfloor \rfloor.$$

- By the left $\leqslant$-innermost rewrite strategy, t is rewritten at position 11 by using r_2.
- By the right $\leqslant$-innermost rewrite strategy, t is rewritten at position 3 by using r_1.
- By the left $\leqslant$-outermost rewrite strategy, t is rewritten at position 1 by using r_2.
- By the right $\leqslant$-outermost rewrite strategy, t is rewritten at position 3 by using r_1.

In general, neither the innermost strategy nor the outermost strategy dominates the other.

For the next examples, let $\mathcal{S}$ be the signature containing four nullary constants a, b, c, and d, two unary constants f and g, and one binary constant h.

We shall also consider only the left variations of these strategies and we need not specify any total order $\leqslant$ on the considered rewrite rules.

Example [s_{i} normalizing but s_{o} not]

Let the TRS $\mathcal{T} := (\mathcal{S}, \mathbb{N}, \to)$ such that $f\lfloor g\lfloor 1\rfloor\rfloor \to f\lfloor g\lfloor g\lfloor 1\rfloor\rfloor\rfloor$ and $ga \to a$.
From the $\mathcal{S}, \mathbb{N}$-term $\mathrm{t} := f\lfloor ga\rfloor$, we have

$$\mathrm{t} \Rightarrow_{s_{\mathrm{i}}} fa$$

and

$$\mathrm{t} \Rightarrow_{s_{\mathrm{o}}} f\lfloor g\lfloor ga\rfloor\rfloor \Rightarrow_{s_{\mathrm{o}}} f\lfloor g\lfloor g\lfloor ga\rfloor\rfloor\rfloor \Rightarrow_{s_{\mathrm{o}}} f\lfloor g\lfloor g\lfloor g\lfloor ga\rfloor\rfloor\rfloor\rfloor \Rightarrow_{s_{\mathrm{o}}} \cdots.$$

Hence, for a same $\mathcal{S}, \mathbb{N}$-term, the innermost rewrite strategy on $\mathcal{T}$ leads to a normal form while the outermost rewrite strategy leads to an infinite rewrite sequence.

Example [s_{o} normalizing but s_{i} not]

Let the TRS $\mathcal{T} := (\mathcal{S}, \mathbb{N}, \to)$ such that $f1 \to a$ and $g1 \to g\lfloor g1\rfloor$.
From the $\mathcal{S}, \mathbb{N}$-term $\mathrm{t} := f\lfloor ga\rfloor$, we have

$$\mathrm{t} \Rightarrow_{s_{\mathrm{i}}} f\lfloor g\lfloor ga\rfloor\rfloor \Rightarrow_{s_{\mathrm{i}}} f\lfloor g\lfloor g\lfloor ga\rfloor\rfloor\rfloor \Rightarrow_{s_{\mathrm{i}}} f\lfloor g\lfloor g\lfloor g\lfloor ga\rfloor\rfloor\rfloor\rfloor \Rightarrow_{s_{\mathrm{i}}} \cdots$$

and

$$\mathrm{t} \Rightarrow_{s_{\mathrm{o}}} a.$$

Hence, for a same $\mathcal{S}, \mathbb{N}$-term, the outermost rewrite strategy on $\mathcal{T}$ leads to a normal form while the innermost rewrite strategy leads to an infinite rewrite sequence.

Example [s_i dominates s_o]

Let the TRS $\mathcal{T} := (\mathcal{S}, \mathbb{N}, \to)$ such that $f1 \to h11$, $haa \to b$, and $c \to a$.
From the $\mathcal{S}, \mathbb{N}$-term $\mathrm{t} := fc$, we have

$$\mathrm{t} \Rightarrow_{s_\mathrm{i}} fa \Rightarrow_{s_\mathrm{i}} haa \Rightarrow_{s_\mathrm{i}} b$$

and

$$\mathrm{t} \Rightarrow_{s_\mathrm{o}} hcc \Rightarrow_{s_\mathrm{o}} hac \Rightarrow_{s_\mathrm{o}} haa \Rightarrow_{s_\mathrm{o}} b.$$

Hence s_i dominates s_o for t in $\mathcal{T}$.

Example [s_o dominates s_i]

Let the TRS $\mathcal{T} := (\mathcal{S}, \mathbb{N}, \to)$ such that $f1 \to a$, $ga \to b$, $gb \to c$, $gc \to d$.
From the $\mathcal{S}, \mathbb{N}$-term $\mathrm{t} := f\underline{\underline{g\underline{g\underline{ga}}}}$, we have

$$\mathrm{t} \Rightarrow_{s_\mathrm{i}} f\underline{\underline{g\underline{gb}}} \Rightarrow_{s_\mathrm{i}} f\underline{gc} \Rightarrow_{s_\mathrm{i}} fd \Rightarrow_{s_\mathrm{i}} a$$

and

$$\mathrm{t} \Rightarrow_{s_\mathrm{o}} a.$$

Hence s_o dominates s_i for t in $\mathcal{T}$.

11.2. Constructor term rewrite systems

Definition

A *constructor TRS* (*CTRS*) is a quadruple $(\mathbf{C}, \mathsf{F}, \mathsf{V}, \to)$ where $\mathbf{C}$ is a signature, called the *underlying signature of constructors*, F is a signature, called the *underlying signature of functions*, and such that

1. the underlying sets of $\mathbf{C}$ and F are disjoint;
2. the triple $\mathcal{T} := (\mathbf{C} \sqcup \mathsf{F}, \mathsf{V}, \to)$ is a TRS;
3. for any rewrite rule $(\mathrm{t}, \mathrm{t}')$ of $\mathcal{T}$, t is a $\mathbf{C} \sqcup \mathsf{F}, \mathsf{V}$-term of the form $\mathrm{t} = f\,\mathrm{t}_1 \dots \mathrm{t}_n$ where $f \in \mathsf{F}{\cdot}n$, $n \in \mathbb{N}$, and $\mathrm{t}_1, \dots, \mathrm{t}_n \in \mathfrak{T}{\cdot}\mathbf{C}{\cdot}\mathsf{V}$.

A CTRS $\mathcal{C} := (\mathbf{C}, \mathsf{F}, \mathsf{V}, \to)$ defines the TRS $(\mathbf{C} \sqcup \mathsf{F}, \mathsf{V}, \to)$, called the *TRS of* $\mathcal{C}$. We use the previous notations and notions in the context of TRSs, here on the TRS of $\mathcal{C}$.

Example

Let $\mathcal{C} := (\mathbf{C}, \mathsf{F}, \mathbb{N}, \to)$ be the CTRS defined as follows. Let the signature $\mathbf{C}$ containing the three nullary constants $\mathbf{true}$, $\mathbf{false}$, and $\mathbf{zero}$, and the unary constant $\mathbf{succ}$. Let the signature F containing the three unary constants not, even, and odd. Let $\to$ be the elementary rewrite relation defined by $\mathsf{not}\,\mathbf{true} \to \mathbf{false}$, $\mathsf{not}\,\mathbf{false} \to \mathbf{true}$, $\mathsf{even}\,\mathbf{zero} \to \mathbf{true}$, $\mathsf{even}\,\lfloor\mathbf{succ}\,\mathbf{zero}\rfloor \to \mathbf{false}$, $\mathsf{even}\,\lfloor\mathbf{succ}\,\lfloor\mathbf{succ}\,1\rfloor\rfloor \to \mathsf{even}\,1$, and $\mathsf{odd}\,1 \to \mathsf{not}\,\lfloor\mathsf{even}\,1\rfloor$.
The quadruple $\mathcal{C}$ is a CTRS.

CTRSs can be used as a programming framework:

- constructors make the data manipulated by the program;
- functions are the computing units of the program;
- elementary rewrite relations describe how to compute the result of applying a function to a sequence of values.

They form a simplified version of pattern matching of functional programming languages.

There are some interesting points about a CTRS $\mathcal{C} := (\mathbf{C}, \mathsf{F}, \mathbb{N}, \to)$:

- any critical data (r_1, u, r_2) of $\mathcal{C}$ is such that $u = \epsilon$. In other words, left members of rules of $\mathcal{C}$ can overlap only at the root;
- Any $\mathbf{C}, \mathsf{V}$-term is a normal form of $\mathcal{C}$. The converse is false.

Definition

A *recursive program scheme* (*RPS*) is a CTRS $\mathcal{C} := (\mathbf{C}, \mathsf{F}, \mathbb{N}, \to)$ such that

1. for any rewrite rule $(\mathrm{t}, \mathrm{t}')$ of $\mathcal{C}$, $\mathrm{t} = f\, v_1 \dots v_n$ where $f \in \mathsf{F}{\cdot}n$, $n \in \mathbb{N}$, $v_1, \dots, v_n \in \mathsf{V}$, and $v_i = v_{i'}$ implies $i = i'$;
2. for any $f \in \mathsf{F}{\cdot}n$, $n \in \mathbb{N}$, there is exactly one rewrite rule of $\mathcal{C}$ of the form of 1..

Example

The CTRS $\mathcal{C} := (\mathbf{C}, \mathsf{F}, \mathbb{N}, \to)$ such that $\mathbf{C}$ is the signature containing the nullary constant $\mathbf{l}$ and the binary constant $\mathbf{n}$, F is the signature containing a binary constant f and a ternary constant g, and $\to$ is defined by $\mathsf{f}12 \to \mathbf{n}21$ and $\mathsf{g}123 \to \mathbf{n}\lfloor\underline{\mathbf{n}12}\rfloor\lfloor\underline{\mathbf{n}23}\rfloor$.

It is immediate that any RPS is orthogonal.

Hence, by Theorem [Confluence of weakly orthogonal TRSs], any RPS is confluent.

Theorem [CTRSs and universality of computation]

For any computable partial function f between two sets X and Y, there exist a CTRS $\mathcal{C}_f := (\mathbf{C}, \mathsf{F}, \mathsf{V}, \to)$ and two coding functions $c_X : X \to \mathfrak{T}_{\lfloor \mathbf{C} \sqcup \mathsf{F} \rfloor, \emptyset}$ and $c_Y : Y \to \mathfrak{T}_{\lfloor \mathbf{C} \sqcup \mathsf{F} \rfloor, \emptyset}$ such that, for any $x \in X$,

- if $f(x)$ is defined, then $c_X(x)$ has $c_Y(f(x))$ as a normal form in $\mathcal{C}_f$;
- otherwise, $c_X(x)$ has no normal form in $\mathcal{C}_f$.

Theorem [CTRSs and universality of computation] says that **CTRSs form a Turing-complete programming language**.

Theorem [Decidability of normalization in RPSs]

The problem of deciding, given an RPS $\mathcal{C} := (\mathbf{C}, \mathsf{F}, \mathsf{V}, \to)$ and a $\lfloor \mathbf{C} \sqcup \mathsf{F} \rfloor, \mathsf{V}$-term t, whether t admits a normal form in $\mathcal{C}$ is decidable.

Theorem [Decidability of normalization in RPSs] appears in [Z. Khasidashvili, Optimal normalization in orthogonal term rewriting systems, 1993].

Since normalization on a given input is undecidable for any Turing-complete programming language, Theorem [Decidability of normalization in RPSs] implies that **RPSs are not Turing-complete**.

Exercise ●●○○○

Define a CTRS which represents natural numbers (as Peano integers), their addition, multiplication, and exponentiation.

Exercise ●●○○○

1. Define a CTRS which is not an RPS and is not terminating.
2. Define a CTRS which is not an RPS and is is not confluent.
3. Define an RPS which is not terminating.
4. Define an RPS which is not confluent.

Exercise ●●●○○

Define a CTRS able to represent, in particular, the OCaml expression

```
if x = y then e else e'
```

where `x` and `y` are any natural numbers, and `e` and `e'` are any expressions.

11.3. Applicative systems

A signature $\mathcal{S}$ is *applicative* if $\mathcal{S}{\cdot}2$ is a singleton and for any $n \in \mathbb{N} \setminus \{0, 2\}$, $\mathcal{S}{\cdot}n = \emptyset$.

The unique element of arity 2 of an applicative signature is denoted by $\bullet$. This constant is the *application constant*.

For any set C, let $[C]$ be the applicative signature such that $[C]{\cdot}0 = C$.

Example

The applicative signature $[\{a, b, c\}]$ is such that $[\{a, b, c\}]{\cdot}0 = \{a, b, c\}$, $[\{a, b, c\}]{\cdot}1 = \emptyset$, $[\{a, b, c\}]{\cdot}2 = \{\bullet\}$, and $[\{a, b, c\}]{\cdot}n = \emptyset$ for any $n \geqslant 3$.

Given a signature $\mathcal{S}$ and a set of variables V, an $\mathcal{S}, \mathsf{V}$-term is *applicative* if $\mathcal{S}$ is applicative.

Given an applicative $\mathcal{S}, \mathsf{V}$-term t, the *concise notation* of t is obtained by deleting all symbols $\bullet$ from the applicative notation of t.

Example

Let us consider the previous applicative signature. The labeled $\mathcal{S}$-term $\bullet\bullet\underline{\lfloor a \lfloor \bullet b a \rfloor 1 \rfloor}\underline{\lfloor \bullet b \lfloor \bullet 2 1 \rfloor \rfloor}$ admits $a\underline{\lfloor b a \rfloor} 1 \underline{\lfloor b \lfloor 2 1 \rfloor \rfloor}$ as concise notation.

Let $\mathcal{S}$ be a signature and V be a set of variables.

The *curried* version of $\mathcal{S}$ is the applicative signature $\mathrm{cur}{\cdot}\mathcal{S}$ such that $\mathrm{cur}{\cdot}\mathcal{S}{\cdot}0 = \bigsqcup_{n\in\mathbb{N}} \mathcal{S}{\cdot}n$.

The *curried* version of an $\mathcal{S},\mathsf{V}$-term t is the $\mathrm{cur}{\cdot}\mathcal{S},\mathsf{V}$-term $\mathrm{cur}{\cdot}\mathrm{t}$ defined by

$$\mathrm{cur}{\cdot}\mathrm{t} := \begin{cases} v & \text{if } \mathrm{t} = v \text{ and } v \in \mathsf{V}, \\ \bullet \underbrace{\dots \bullet c \lfloor \mathrm{cur}{\cdot}\mathrm{t}_1 \rfloor \dots \lfloor \mathrm{cur}{\cdot}\mathrm{t}_n \rfloor}_{} & \text{otherwise, where } \mathrm{t} = c\,\mathrm{t}_1 \dots \mathrm{t}_n, c \in \mathcal{S}{\cdot}n, n \in \mathbb{N}, \mathrm{t}_1, \dots, \mathrm{t}_n \in \mathfrak{T}{\cdot}\mathcal{S}{\cdot}\mathsf{V}. \end{cases}$$

Example

Let the labeled $\mathcal{S}_{\mathbb{N}^2}$-term

$$\mathrm{t} := \mathsf{c}_3 1 \lfloor \mathsf{c}_2 \mathsf{c}_0 \lfloor \mathsf{c}_1 2 \rfloor \rfloor \lfloor \mathsf{c}_3 2 \mathsf{c}_0 3 \rfloor.$$

The curried version of t is the labeled $\mathrm{cur}{\cdot}\mathcal{S}_{\mathbb{N}^2}$-term

$$\mathrm{cur}{\cdot}\mathrm{t} = \bullet \bullet \underbrace{\bullet \mathsf{c}_3 1 \rfloor \bullet \lfloor \mathsf{c}_2 \mathsf{c}_0 \lfloor \mathsf{c}_1 2 \rfloor \rfloor}_{} \bullet \underbrace{\bullet \lfloor \bullet \mathsf{c}_3 2 \rfloor \mathsf{c}_0 \rfloor 3}_{}.$$

Note that the applicative notation of an $\mathcal{S},\mathsf{V}$-term t and the concise notation of $\mathrm{cur}{\cdot}\mathrm{t}$ are the same strings.

A TRS $\mathcal{T}$ is *applicative* if its underlying signature is applicative.

Given a TRS $\mathcal{T} := (\mathcal{S}, \mathsf{V}, \to)$, the *curried* version of $\mathcal{T}$ is the TRS $\mathrm{cur}{\cdot}\mathcal{T} := (\mathrm{cur}{\cdot}\mathcal{S}, \mathsf{V}, \mathrm{cur}{\cdot} \to)$ such that $\mathrm{cur}{\cdot} \to$ is the elementary rewrite relation defined by $(\mathrm{cur}{\cdot}\mathfrak{t}, \mathrm{cur}{\cdot}\mathfrak{t}') \in \mathrm{cur}{\cdot} \to$ if $\mathfrak{t}$ and $\mathfrak{t}'$ are two $\mathcal{S}, \mathsf{V}$-terms such that $\mathfrak{t} \to \mathfrak{t}'$.

Example

Let the TRS $\mathsf{NatAdd} := (\mathcal{S}, \mathbb{N}, \to)$ such that $\mathcal{S}$ is the signature containing one nullary constant z, one unary constant s, and one binary constant a, and such that $a1z \to 1$ and $a1\lfloor s2 \rfloor \to s\lfloor a12 \rfloor$.

In $\mathrm{cur}{\cdot}\mathsf{NatAdd}$, the underlying applicative signature contains z, s, and a as three nullary constants. Moreover, the elementary rewrite relation of this TRS is defined by

$$\bullet\lfloor \bullet a1 \rfloor z \;\; \mathrm{cur}{\cdot} \to \;\; 1$$

and

$$\bullet\lfloor \bullet a1 \rfloor \lfloor \bullet s2 \rfloor \;\; \mathrm{cur}{\cdot} \to \;\; \bullet s\lfloor \bullet\lfloor \bullet a1 \rfloor 2 \rfloor.$$

By construction, the curried version of a TRS is an applicative TRS.

Theorem [Curried version of TRS and termination preservation]

Let $\mathcal{T}$ be a TRS. If $\mathcal{T}$ is terminating, then $\mathrm{cur}{\cdot}\mathcal{T}$ is terminating.

This result appears in [J. R. Kennaway, J. W. Klop, M. R. Sleep, F. J. de Vries, Transfinite reductions in orthogonal term rewriting systems, 1995].

Theorem [Curried version of TRS and confluence preservation]

Let $\mathcal{T}$ be a TRS. If $\mathcal{T}$ is confluent, then $\mathrm{cur}{\cdot}\mathcal{T}$ is confluent.

This result appears in [S. Kahrs, Confluence of curried term-rewriting systems, 1995].

11.4. Combinatory logic

A *basic combinator* is a triple $b := (c, n_c, \mathrm{t}_c)$ such that

- c is a symbol, the *constant* of b;
- $n_c \in \mathbb{N} \setminus \{0\}$ is the *order* of b;
- t_c is a labeled $\{\bullet\}$-term such that $\mathrm{rk_v}{\cdot}\mathrm{t}_c \leqslant n_c$, called the *right member* of b.

Definition

A *combinatory logic system* (*CLS*) is an applicative TRS $\mathcal{C} := (\mathcal{S}, \mathbb{N} \setminus \{0\}, \to)$ such that for any $c \in \mathcal{S}{\cdot}0$, there is a unique basic combinator (c, n_c, t_c) such that

$$c\,1 \ldots n_c \to \mathrm{t}_c,$$

and all rewrite rules of $\mathcal{C}$ are of this form.

Example

Let $\mathcal{S}$ be the applicative signature $[\{a, b\}]$. Let the basic combinators $(a, 4, 11\underline{\lfloor 213 \rfloor}2)$ and $(b, 3, 21\underline{\lfloor 33 \rfloor\lfloor 22 \rfloor})$, and let us denote by $\mathcal{C}$ the CLS specified by $\mathcal{S}$ and these two basic combinators.

The elementary rewrite relation $\Rightarrow$ of $\mathcal{C}$ satisfies $a1234 \to 11\underline{\lfloor 213 \rfloor}2$ and $b123 \to 21\underline{\lfloor 33 \rfloor\lfloor 22 \rfloor}$.

A *combinator* of a CLS $\mathcal{C}$ having $\mathcal{S}$ as underlying applicative signature is a ground labeled $\mathcal{S}$-term.

Here are some basic combinators:

- *Idiot Bird*: $(\mathbf{I}, 1, \mathbf{1})$;
- *Mockingbird*: $(\mathbf{M}, 1, 11)$;
- *Kestrel*: $(\mathbf{K}, 2, 1)$;
- *Thrush*: $(\mathbf{T}, 2, 21)$;
- *Warbler*: $(\mathbf{W}, 2, 112)$;
- *Lark*: $(\mathbf{L}, 2, 1\lfloor 22 \rfloor)$;
- *Owl*: $(\mathbf{O}, 2, 2\lfloor 12 \rfloor)$;
- *Turing Bird*: $(\mathbf{U}, 2, 2\lfloor 112 \rfloor)$;
- *Cardinal*: $(\mathbf{C}, 3, 132)$;
- *Vireo*: $(\mathbf{V}, 3, 312)$;
- *Bluebird*: $(\mathbf{B}, 3, 1\lfloor 23 \rfloor)$;
- *Starling*: $(\mathbf{S}, 3, 13\lfloor 23 \rfloor)$.

Most of these appear in [R. Smullyan, To Mock a Mockingbird, 1985].

Some of these are very concrete:

- The Idiot Bird, satisfying for any x, $\mathbf{I}x \to x$, is the **identity function**;
- The Kestrel, satisfying for any α and x, $\mathbf{K}\alpha x \to \alpha$, allows us to build constant functions. Indeed, $\mathbf{K}\alpha$ is the function sending any x to α;
- The Thrush, satisfying for any f and x, $\mathbf{T}xf \to fx$ is the reverse application function. In OCaml, this function is denoted by `(|>)`;
- The Bluebird, satisfying for any f_1, f_2, and x, $\mathbf{B}f_1f_2x \to f_1\lfloor f_2x \rfloor$ allows us to compose functions. Indeed, $\mathbf{B}f_1f_2$ is the function sending any x to the application of f_1 on the application of f_2 on x.

Proposition [Confluence of CLSs]

Any CLS is confluent.

This is a consequence of the fact that any CLS is orthogonal and Theorem [Confluence of weakly orthogonal TRSs]. Indeed, the leftmost symbol of any left-hand side of a rewrite rule of a CLS is a unique constant of arity 0 of its underlying signature.

Nevertheless, CLSs are in general not terminating.

Example

Let $\mathcal{C}$ be the CLS on the two basic combinators $\mathbf{I}$ and $\mathbf{S}$.
Let $\mathrm{t} := \mathbf{S}\lfloor\underline{\mathbf{SII}}\rfloor\mathbf{I}$.
Observe that for any variable v,

$$\mathrm{t}v = \mathbf{S}\lfloor\underline{\mathbf{SII}}\rfloor\mathbf{I}v \Rightarrow \mathbf{SII}v\lfloor\underline{\mathbf{I}v}\rfloor \Rightarrow \mathbf{I}v\lfloor\underline{\mathbf{I}v}\rfloor\lfloor\underline{\mathbf{I}v}\rfloor \Rightarrow v\lfloor\underline{\mathbf{I}v}\rfloor\lfloor\underline{\mathbf{I}v}\rfloor \Rightarrow vv\lfloor\underline{\mathbf{I}v}\rfloor \Rightarrow vvv.$$

As a consequence,

$$\mathrm{tt} \Rightarrow^* \mathrm{ttt} \Rightarrow^* \mathrm{tttt} \Rightarrow^* \cdots$$

is an infinite rewrite sequence in $\mathcal{C}$.

Let $\mathcal{C} := (\mathcal{S}, \mathbb{N} \setminus \{0\}, \to)$ be a CLS.

Any labeled $\mathcal{S}$-term t is seen as a **function**: given a sequence $\mathrm{t}_1, \ldots, \mathrm{t}_n$, $n \in \mathbb{N}$, of labeled $\mathcal{S}$-terms, the iterated application $\mathrm{t}\,\mathrm{t}_1 \ldots \mathrm{t}_n$ is the **application** of t to the **arguments** t_1, ..., t_n.

Let $\equiv_\mathrm{e}$ be the *extensional relation*, defined as the smallest equivalence relation on $\mathfrak{T}\cdot\mathcal{S}\cdot\underline{\mathbb{N} \setminus \{0\}}$ such that, for any labeled $\mathcal{S}$-terms t and t',

- if $\mathrm{t} \equiv \mathrm{t}'$, then $\mathrm{t} \equiv_\mathrm{e} \mathrm{t}'$;
- if $\mathrm{t}v \equiv_\mathrm{e} \mathrm{t}'v$, where v is any variable having no occurrence in t and t', then $\mathrm{t} \equiv_\mathrm{e} \mathrm{t}'$.

Intuitively, when $\mathrm{t} \equiv_\mathrm{e} \mathrm{t}'$, the labeled $\mathcal{S}$-terms t and t', seen as functions, have the same behavior on generic arguments.

Example

Let $\mathcal{C}$ be the CLS on the three basic combinators $\mathbf{I}$, $\mathbf{K}$, and $\mathbf{S}$.
Let $\mathrm{t} := \mathbf{SK}$ and $\mathrm{t}' := \mathbf{KI}$. Since

$$\mathrm{t}12 = \mathbf{SK}12 \Rightarrow \mathbf{K}2\underline{12} \Rightarrow 2$$

and

$$\mathrm{t}'12 = \mathbf{KI}12 \Rightarrow \mathbf{I}2 \Rightarrow 2,$$

we have $\mathrm{t} \equiv_\mathrm{e} \mathrm{t}'$.

Let $\mathcal{C} := (\mathcal{S}, \mathbb{N} \setminus \{0\}, \to)$ be a CLS.

Let $\mathfrak{s}$ be a labeled $\{\bullet\}$-term such that $\mathrm{rk_v}{\cdot}\mathfrak{s} = n$, $n \geqslant 1$. When there is a **combinator** $\mathfrak{t}$ of $\mathcal{C}$ such that

$$\mathfrak{t}\,1 \ldots n \Rightarrow^* \mathfrak{s},$$

the combinator $\mathfrak{t}$ *computes* $\mathfrak{s}$, and $\mathfrak{s}$ is *computable* in $\mathcal{C}$.

Example

Let $\mathcal{C}$ be the CLS on the two basic combinators $\mathbf{K}$ and $\mathbf{B}$.
Let the labeled $\{\bullet\}$-term $\mathfrak{s} := 12\lfloor 345 \rfloor$. The combinator $\mathfrak{t} := \mathbf{KBB}\lfloor \mathbf{BBB} \rfloor$ computes $\mathfrak{s}$. Indeed,

$$\mathfrak{t}12345 = \mathbf{KBB}\lfloor \mathbf{BBB} \rfloor 12345 \Rightarrow \mathbf{B}\lfloor \mathbf{BBB} \rfloor 12345 \Rightarrow \lfloor \mathbf{BBB} \rfloor \lfloor 12 \rfloor 345 \Rightarrow \mathbf{B}\lfloor \mathbf{B}\lfloor 12 \rfloor \rfloor 345 \Rightarrow \mathbf{B}\lfloor 12 \rfloor \lfloor 34 \rfloor 5 \Rightarrow 12\lfloor 345 \rfloor = \mathfrak{s}.$$

Definition

A CLS $\mathcal{C}$ is *combinatory complete* if all labeled $\{\bullet\}$-terms are computable in $\mathcal{C}$.

Theorem [Turing completeness of combinatory complete CLSs]

Let $\mathcal{C} := (\mathcal{S}, \mathbb{N} \setminus \{0\}, \rightarrow)$ be a combinatory complete CLS. There exists a translation function $t \mapsto \langle t \rangle$ from λ-terms to labeled $\mathcal{S}$-terms such that, for any λ-terms t and t', if t reduces to t' in the λ-calculus, then $\langle t \rangle \Rightarrow^* \langle t' \rangle$.

Theorem [Combinatory completeness of the CLS on $\mathbf{K}$ and $\mathbf{S}$]

The CLS on the two basic combinators $\mathbf{K}$ and $\mathbf{S}$ is combinatory complete.

This result appears in [H. B. Curry, R. Feys, Combinatory Logic, Vol. I, 1958]. See also [M. Schönfinkel, Über die Bausteine der mathematischen Logik, 1924].

There are other known combinatory complete CLSs. For instance, the CLS on the four basic combinators $\mathbf{B}$, $\mathbf{C}$, $\mathbf{K}$, and $\mathbf{W}$ is one of these [H. B. Curry, Grundlagen der kombinatorischen Logik, 1930].

Theorem [Minimal order of basic combinators of complete CLSs]

Any combinatory complete CLS has at least one basic combinator of order 3 or more.

This result comes from [R. Legrand, A Basis Result in Combinatory Logic, 1988].