(An algebraic and combinatorial point of view of)
Term rewrite systems

INF889K

Samuele Giraudo

Département d'informatique, Université du Québec à Montréal

giraudo.samuele@uqam.ca

Sujets spéciaux en informatique

Hiver 2026

1. General information

1.1. Administrative information

Classes: 15 class meetings, on Mondays, from 13:30 to 16:30.

Evaluation:

- written presentation of an existing research result, 35%;
- oral presentation of an existing research result, 25%;
- exam, 40%.

Each item is scored from 0 to 100.

Passing grade: weighted total score of 50 or higher.

Timetable:

1. **Week 7**: each student must prepare a list of 3 known results on the topic, including a short bibliography.
2. **Week 8**: the professor assigns a result to each student (coming from their list).
3. **Week 13**: each student must write a text presenting the result in LaTeX, 4--16 pages, including the context, proofs, and bibliography.
4. **Week 14**: each student gives an oral presentation of the result, lasting 20 minutes.
5. **Week 15**: exam.

1.2. Content

Objectives:

- Provide an introduction about term rewrite systems;
- Adopt a combinatorial point of view of the topic;
- Apply rewriting techniques for algebraic problems;
- Read, understand, and present research results on the topic.

This course **is not designed to**

- provide an in-depth algorithmic treatment of term rewrite systems;
- cover the categorical viewpoint of the topic.

Content:

1. Abstract rewrite systems (binary relations, first general results);
2. Combinatorics of terms (terms, substitutions);
3. Term series (formal series, products on series, enumeration);
4. Term rewrite systems (matchings, patterns, general definition, main properties);
5. Termination and confluence (reduction orders, polynomial interpretations, critical pairs, completion);
6. Universal algebra and clones (varieties, word problem, Tietze transformations, clones);
7. Programming with term rewriting (applicative systems, currying, combinatory logic).

Bibliography:

1. F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

2. Terese, Term Rewriting Systems, Cambridge University Press, 2003.

3. J. W. Klop, Term Rewriting Systems, Handbook of Logic in Computer Science, Vol. 2, Oxford University Press, 1992.

4. J. R. Hindley, J. P. Seldin, Lambda-Calculus and Combinators, an Introduction, Cambridge University Press, 2008.

5. P. M. Cohn, Universal Algebra, Springer Dordrecht, 1981.

Exercises are classified according to a **difficulty level**:

- ○○○○○ : extremely easy exercise---almost immediate to solve once the question is understood;
- ●○○○○ : very easy exercise---a few minutes to solve and write completely;
- ●●○○○ : easy exercise applying directly the concepts of the lecture---about ten minutes to solve and write completely;
- ●●●○○ : moderate question applying several concepts of the lecture---on the order of an hour to solve and write completely;
- ●●●●○ : difficult question requiring careful consideration---on the order of several hours to solve and write completely;
- ●●●●● : research question requiring a complete exploration---on the order of several days to be considered. In some cases, research questions may still be quite approachable.

1.3. General conventions and notations

Functions are written in **curried form**: given a function

$$f : A_1 \times \cdots \times A_n \to A,$$

we implicitly identify f with its curried form, so that

$$f : A_1 \to \cdots \to A_n \to A$$

where $\to$ is right-associative.

For any $a_1 \in A_1$, the **partial application** $f{\cdot}a_1$ is the function of type $A_2 \to \cdots \to A_n \to A$ obtained by specializing the first argument of f as a_1. Hence, by iterating this, we write $f{\cdot}a_1{\cdot}\cdots{\cdot}a_n$ rather than $f(a_1, \ldots, a_n)$.

We will sometimes use **underlining** rather than parentheses to enclose sub-expressions.

Example

Let $f : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be the function defined by $f(x_1, x_2, x_3) := x_1x_2 + x_1x_3 + x_2x_3$.

Under the above identification, the type of f is $\mathbb{N} \to \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ and

$$f{\cdot}\underline{x + y}{\cdot}x{\cdot}\underline{z + t} = (x + y)x + (x + y)(z + t) + x(z + t).$$

The partial application $f{\cdot}1$ is the function $g : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ satisfying $g{\cdot}x_2{\cdot}x_3 = x_2 + x_3 + x_2x_3$.

The *Iverson bracket* is defined as follows. For any statement P,

$$[P] := \begin{cases} 1 & \text{if } P, \\ 0 & \text{otherwise.} \end{cases}$$

Let us define the following **sets of integers**:

- for any $i, j \in \mathbb{Z}$, $[i, j] := \{z \in \mathbb{Z} : i \leqslant z \leqslant j\}$;
- for any $n \in \mathbb{N}$, $[n] := [1, n]$;
- for any $n \in \mathbb{N}$, $[\![n]\!] := \{0\} \cup [n]$.

Some definitions about **words**:

- for any set A, A^* is the set of words on A;
- the **empty word** is denoted by ϵ;
- for any $w \in A^*$, $\ell{\cdot}w$ is the **length** of w;
- for any $w \in A^*$ and $i \in [\ell{\cdot}w]$, $w{\cdot}i$ is the i-th letter of w, where letters are indexed from 1;
- for any $w \in A^*$ and $a \in A$, let $\ell_a{\cdot}w := \#\{i \in [\ell{\cdot}w] : w{\cdot}i = a\}$;
- for any $w, w' \in A^*$, the **concatenation** of w and w' is $w.w'$.

2. Introduction

2.1. Solving the Coffee Can Problem

Consider the following **game**, called the *Coffee Can Problem* [D. Gries, The Science of Programming, 1981]:

- let a coffee can containing some white beans ○ and some black beans ●;
- keep an unlimited supply of black beans aside;
- consider the action consisting in randomly picking two beans from the can and
  - if they have the same color, then throw them both and put a black bean into the can;
  - otherwise, throw the black bean and return the white bean into the can;
- repeat this action as long as possible.

Example

Consider the can ○○●●●●○●○.

Pick the 2-nd bean and the 5-th bean. Since they have different colors, the can becomes ○○●●●○●○.

Now, pick the 3-rd and the 4-th bean. Since they have the same color, the can becomes ○○●●○●○.

Some questions about the described process:

- prove that this process always terminates;
- prove that all ways to execute the process lead to the same remaining bean color;
- predict the remaining bean color in terms of the initial can state.

First, we **formalize the problem** by encoding the state of the can as a pair $(i, j) \in \mathbb{N}^2$ where i is the number of $\circ$ and j is the number of $\bullet$ in the can.

Then, we define a **transformation rule** $\Rightarrow$, called *rewrite relation*, which encodes the action:

- when the picked beans are two $\circ$:

$$(i, j) \Rightarrow (i - 2, j + 1);$$

- when the picked beans are two $\bullet$:

$$(i, j) \Rightarrow (i, j - 1);$$

- when the picked beans are of different colors:

$$(i, j) \Rightarrow (i, j - 1).$$

The set of states and of the rewrite relation form a *rewrite system*.

The process terminates because if $(i, j) \Rightarrow (i', j')$, then $i + j > i' + j'$ and, as a state is an element of $\mathbb{N}^2$, it is not possible to perform an infinite sequence of actions from a state.

For this reason, we say that this rewrite system is *terminating*.

To prove that all ways to execute the process lead to the same remaining bean color, we first prove an important property call *local confluence*.

This property holds when, given a state (i, j), if we have two states (i_1, j_1) and (i_2, j_2) such that $(i, j) \Rightarrow (i_1, j_1)$ and $(i, j) \Rightarrow (i_2, j_2)$, there is a state (i', j') reachable from both (i_1, j_1) and (i_2, j_2).

This is the case since we have the commuting square

$$\begin{array}{ccc} (i, j) & \Rightarrow & (i, j-1) \\ \Downarrow & & \Downarrow \\ (i-2, j+1) & \Rightarrow & (i-2, j) \end{array}.$$

This shows that the rewrite system is *locally confluent*.

Since the process is, as shown previously, terminating, by **Newman's Lemma** [M. H. A. Newman, On Theories with a Combinatorial Definition of ''Equivalence'', 1942], the rewrite system is also confluent.

This says exactly that all ways to execute the process lead to the same remaining bean color.

The two possible **final outcomes** of the process are $(1,0)$ and $(0,1)$. They are called *normal forms*.

To predict the final outcome, let us consider the map $\theta : \mathbb{N}^2 \to \mathbb{N}$ such that $\theta{\cdot}(i,j) := i \mod 2$.

Observe that

$$\theta{\cdot}(i,j) = \theta{\cdot}(i,j-1) = i \mod 2$$

and

$$\theta{\cdot}(i,j) = \theta{\cdot}(i-2,j+1) = i \mod 2.$$

From this **invariant**, we deduce that

$$(i,j) \Rightarrow^* \begin{cases} (0,1) & \text{if } i \mod 2 = 0, \\ (1,0) & \text{otherwise.} \end{cases}$$

This shows that the final outcome of the Coffee Can Problem is a white bean if the initial number of white beans in the can is odd, and a black bean otherwise, independently from the choice of each pair of beans at each step of the process.

2.2. A modelization of the Frogs and Toads Puzzle

Let us consider the following game, called the *Frogs and Toads Puzzle* [É. Lucas, 1883]:

- let a line of $n+1+n$ squares;
- let n frogs $\circ$ on the first n squares;
- let n toads $\bullet$ on the last n squares;
- the middle square is empty $\cdot$;
- the goal consists in exchanging the position of all $\circ$ and $\bullet$ by executing a sequence of moves where
  - a $\circ$ can be moved to its right adjacent square $\cdot$;
  - a $\bullet$ can be moved to its left adjacent square $\cdot$;
  - a $\circ$ can jump over the adjacent $\bullet$ on its right and land on the $\cdot$ on the right;
  - a $\bullet$ can jump over the adjacent $\circ$ on its left and land on the $\cdot$ on the left.

Example

For $n := 3$, we have this sequence of configurations, starting from the initial one:

$$\circ\circ\circ\cdot\bullet\bullet\bullet, \quad \circ\circ\cdot\circ\bullet\bullet\bullet, \quad \circ\circ\bullet\circ\cdot\bullet\bullet, \quad \circ\circ\bullet\circ\bullet\cdot\bullet, \quad \circ\circ\bullet\cdot\bullet\circ\bullet.$$

This puzzle can be formalized in the following way as a **rewrite system on words**.

A **state** is a word on the alphabet $\{\circ, \cdot, \bullet\}$ having n occurrences of $\circ$, 1 occurrence of $\cdot$, and n occurrences of $\bullet$.

Consider the **rewrite rule** $\to$ on such words, defined by

$$\circ\cdot \to \cdot\circ,$$

$$\cdot\bullet \to \bullet\cdot,$$

$$\circ\bullet\cdot \to \cdot\bullet\circ,$$

$$\cdot\circ\bullet \to \bullet\circ\cdot.$$

This rewrite rule $\to$ is extended as a **rewrite relation** $\Rightarrow$ by extending it to the context by

$$uxv \Rightarrow ux'v \text{ if } x \to x'$$

for any words $u, v \in \{\circ, \cdot, \bullet\}^*$.

The puzzle consists in finding a **rewrite sequence** of the form

$$\circ^n \cdot \bullet^n \Rightarrow \cdots \Rightarrow \bullet^n \cdot \circ^n.$$

Exercise ●●○○○

Let us consider the previous description of the Frog and Toads Puzzle as a rewrite system on words on $n \geqslant 1$ frogs and toads.

1. Describe the **normal forms** of the puzzle, that are, the configurations wherein no move can be performed.
2. Prove that the puzzle is **terminating**, that is, from the initial configuration, by playing a sequence of any moves, a normal form is reached.
3. Prove that the puzzle is **not confluent**, that is, from some configuration, it is possible to play two different moves such that the two resulting configurations lead to no common future configuration.
4. Prove that the required number of moves from the initial configuration to the goal configuration is always $n^2 + 2n$.
5. Provide a description of a sequence of moves from the initial configuration to the goal configuration.

2.3. Computing with natural numbers

Let us represent **expressions on natural numbers** and a way to **compute addition, multiplication, and factorial** on these.

Each natural number $n \in \mathbb{N}$ is represented in a *functional way* in the **unary numeral system** via zero and succ as $\mathsf{succ}^n(\mathsf{zero})$.

Example

The number 4 is represented by

$$\mathsf{succ}^4(\mathsf{zero}) = \mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{succ}(\mathsf{zero})))).$$

Each expression on natural numbers involving the operations $+$, $\times$, and $!$ is denoted in a functional way via add, mul, and fact.

Example

The expression $(2 + 3!) \times 5$ is represented functionally by

$$\mathsf{mul}\big(\mathsf{add}\big(\mathsf{succ}^2(\mathsf{zero}), \mathsf{fact}\big(\mathsf{succ}^3(\mathsf{zero})\big)\big), \mathsf{succ}^5(\mathsf{zero})\big).$$

To compute expressions involving addition, let us introduce the rewrite rules

$$\mathsf{add}(n, \mathsf{zero}) \to n,$$

$$\mathsf{add}(n_1, \mathsf{succ}(n_2)) \to \mathsf{succ}(\mathsf{add}(n_1, n_2)).$$

Let $\Rightarrow$ be the rewrite relation obtained by extending $\to$ on the context.

The rewrite relation $\Rightarrow$ is used to locally rewrite an expression into another, while possible, in order to get, from an expression on zero, succ, and add, an expression involving only zero and succ.

Example

We have this rewrite sequence:

$$\mathsf{add}\big(\mathsf{succ}^2(\mathsf{zero}), \mathsf{succ}^3(\mathsf{zero})\big) \Rightarrow \mathsf{succ}\big(\mathsf{add}\big(\mathsf{succ}^2(\mathsf{zero}), \mathsf{succ}^2(\mathsf{zero})\big)\big) \Rightarrow \mathsf{succ}\big(\mathsf{succ}\big(\mathsf{add}\big(\mathsf{succ}^2(\mathsf{zero}), \mathsf{succ}(\mathsf{zero})\big)\big)\big)$$

$$\Rightarrow \mathsf{succ}\big(\mathsf{succ}\big(\mathsf{succ}\big(\mathsf{add}\big(\mathsf{succ}^2(\mathsf{zero}), \mathsf{zero}\big)\big)\big)\big) \Rightarrow \mathsf{succ}\big(\mathsf{succ}\big(\mathsf{succ}\big(\mathsf{succ}^2(\mathsf{zero})\big)\big)\big) = \mathsf{succ}^5(\mathsf{zero}).$$

Exercise ●○○○○

Translate the expression $(1 + 1) + (2 + 1)$ as an expression involving zero, succ, and add, and apply the previous rewrite relation $\Rightarrow$ in order to transform it while possible.

In a similar way, we include the following rules to compute expression involving **multiplications**:

$$\mathsf{mul}(n, \mathsf{zero}) \to \mathsf{zero},$$

$$\mathsf{mul}(n_1, \mathsf{succ}(n_2)) \to \mathsf{add}(\mathsf{mul}(n_1, n_2), n_1),$$

and **factorials**:

$$\mathsf{fact}(\mathsf{zero}) \to \mathsf{succ}(\mathsf{zero}),$$

$$\mathsf{fact}(\mathsf{succ}(n)) \to \mathsf{mul}(\mathsf{succ}(n), \mathsf{fact}(n)).$$

Exercise ●○○○○

Translate the expression $3!$ as a functional expression involving zero, succ, and fact, and apply the six previous rules in order to transform it while possible.

We have defined a **term rewrite system**, where **terms** represent expressions on natural numbers, and the iterated application of the **rewrite relation** allows us to simplify a term into a term which cannot be simplified anymore.

Some questions in this context:

- Some expressions can be rewritten in several different ways. Do all these ways lead to the same result? In other words, is the rewrite system confluent?
- Does this rewrite process always terminate? In other words, is the rewrite system terminating?
- How many steps of rewrites an expression need until the rewriting process ends?
- Does there exist strategies to speed up the process?

2.4. Map operation on lists

Let us represent **expressions on lists** and a way to compute **map** on lists.

Each list is represented in an *applicative way* via nil and cons.

Example

The list $[x_1, x_2, x_3, x_4]$ is represented by

$$\mathsf{cons}\ x_1\ (\mathsf{cons}\ x_2\ (\mathsf{cons}\ x_3\ (\mathsf{cons}\ x_4\ \mathsf{nil}))).$$

Each expression on lists involving the map operation is denoted in an applicative way via map.

Example

The expression specifying the map operation on the list $[1, 2, 3]$ via the factorial function $!$ is represented by

$$\mathsf{map}\ !\ (\mathsf{cons}\ 1\ (\mathsf{cons}\ 2\ (\mathsf{cons}\ 3\ \mathsf{nil}))).$$

To compute such map operation on list, let us introduce the **rewrite rules**

$$\mathsf{map}\ f\ \mathsf{nil} \to \mathsf{nil},$$

$$\mathsf{map}\ f\ (\mathsf{cons}\ x\ l) \to \mathsf{cons}\ (f\ x)\ (\mathsf{map}\ f\ l).$$

Let $\Rightarrow$ be the **rewrite relation** obtained by extending $\to$ on the context.

Example

We have this rewrite sequence:

$$\mathsf{map}\ !\ (\mathsf{cons}\ 1\ (\mathsf{cons}\ 2\ (\mathsf{cons}\ 3\ \mathsf{nil}))) \Rightarrow \mathsf{cons}\ (!\ 1)\ (\mathsf{map}\ !\ (\mathsf{cons}\ 2\ (\mathsf{cons}\ 3\ \mathsf{nil})))$$

$$\Rightarrow \mathsf{cons}\ (!\ 1)\ (\mathsf{cons}\ (!\ 2)\ (\mathsf{map}\ !\ (\mathsf{cons}\ 3\ \mathsf{nil}))) \Rightarrow \mathsf{cons}\ (!\ 1)\ (\mathsf{cons}\ (!\ 2)\ (\mathsf{cons}\ (!\ 3)\ (\mathsf{map}\ !\ \mathsf{nil})))$$

$$\Rightarrow \mathsf{cons}\ (!\ 1)\ (\mathsf{cons}\ (!\ 2)\ (\mathsf{cons}\ (!\ 3)\ (\mathsf{nil}))).$$

Exercise ●●○○○

By using similar methods, propose rules in order to compute left fold on lists by mean of the symbol fold_left.

We have defined an **applicative term rewrite system** to represent lists and their map operation.

In addition to the questions of the previous example, we can ask the following questions:

- What really means ‘‘applicative’’?
- Why such an applicative system is important in this context, as opposed to a functional one?
- Can we represent in a similar way other data structures and their operations, as stacks, heaps, binary search trees, *etc.*?

2.5. Formal derivative of polynomials

The **formal derivative of polynomials** of $\mathbb{K}\langle x\rangle$, where $\mathbb{K}$ is any ring, can be described by the rules

$$\partial k \to 0, \qquad \text{for all } k \in \mathbb{K},$$

$$\partial x \to 1,$$

$$\partial(f_1 + f_2) \to \partial f_1 + \partial f_2,$$

$$\partial(f_1 \times f_2) \to f_1 \times \partial f_2 + \partial f_1 \times f_2.$$

Let $\Rightarrow$ be the **rewrite relation** obtained by extending $\to$ on the context.

Example

We have the following rewrite sequence:

$$\partial\left(x^2 + 2x\right) = \partial(x \times x + 2 \times x) \Rightarrow \partial(x \times x) + \partial(2 \times x) \Rightarrow x \times \partial x + \partial x \times x + \partial(2 \times x)$$

$$\Rightarrow x \times \partial x + \partial x \times x + 2 \times \partial x + \partial 2 \times x \Rightarrow x \times \partial x + \partial x \times x + 2 \times \partial x + 0 \times x$$

$$\Rightarrow x \times \partial x + \partial x \times x + 2 \times \partial x + 0 \times x \Rightarrow x \times 1 + \partial x \times x + 2 \times \partial x + 0 \times x$$

$$\Rightarrow x \times 1 + 1 \times x + 2 \times \partial x + 0 \times x \Rightarrow x \times 1 + 1 \times x + 2 \times 1 + 0 \times x = 2x + 2.$$

We have considered the formal derivative of polynomials as a **rewrite system** allowing us to compute it from any polynomial.

Some observations and questions in this context:

- Here also, the derivative of some polynomials can be executed in different ways. Do all these ways lead to the same result?
- Does this computation always terminate?
- On polynomials, the operations $+$ and $\times$ are associative. This has been assumed implicitly. How to take this property into account rigorously?
- The same question holds for the commutativity for $+$ and $\times$ when $\mathbb{K}$ is a commutative ring.

2.6. Deciding equivalence for groups

A **group** is set together with a binary operation $\star$ which is associative, a neutral element $\mathbb{1}$ w.r.t. $\star$, and an inverse operation $x \mapsto x^{-1}$ w.r.t. $\star$.

A *formal group expression* is an expression combining variables x_i, $i \geqslant 1$, $\star$, $\mathbb{1}$, and $^{-1}$.

Example

$$(x_1 \star x_2) \star \left(x_3 \star \left(x_1^{-1} \star x_3\right)\right)^{-1}$$

is a formal group expression.

A natural question concerns the decision of the equivalence of two formal group expressions.

Two formal group expressions e_1 and e_2 are **equivalent** if in any group, by quantifying universally on the variables appearing in e_1 and e_2, these two expressions compute the same thing.

To prove that two formal group expressions e_1 and e_2 are equivalent, we search for a transformation of e_1 into e_2 by using the group axioms of the *usual presentation of groups*:

$$\mathbb{1} \star x_1 \equiv x_1 \equiv x_1 \star \mathbb{1},$$

$$x_1^{-1} \star x_1 \equiv \mathbb{1} \equiv x_1 \star x_1^{-1},$$

$$(x_1 \star x_2) \star x_3 \equiv x_1 \star (x_2 \star x_3).$$

Example

Let the two formal group expressions $e_1 := (x_1 \star x_2)^{-1}$ and $e_2 := x_2^{-1} \star x_1^{-1}$.
We have

$$e_1 = (x_1 \star x_2)^{-1} \equiv \mathbb{1} \star (x_1 \star x_2)^{-1} \equiv \left(x_2^{-1} \star x_2\right) \star (x_1 \star x_2)^{-1} \equiv x_2^{-1} \star \left(x_2 \star (x_1 \star x_2)^{-1}\right)$$

$$\equiv x_2^{-1} \star \left(\left(x_1^{-1} \star x_1\right) \star \left(x_2 \star (x_1 \star x_2)^{-1}\right)\right) \equiv x_2^{-1} \star \left(x_1^{-1} \star \left(x_1 \star \left(x_2 \star (x_1 \star x_2)^{-1}\right)\right)\right)$$

$$\equiv x_2^{-1} \star \left(x_1^{-1} \star \left((x_1 \star x_2) \star (x_1 \star x_2)^{-1}\right)\right) \equiv x_2^{-1} \star \left(x_1^{-1} \star \mathbb{1}\right) \equiv x_2^{-1} \star x_1^{-1} = e_2,$$

so that e_1 and e_2 are equivalent.

This **does not provide a decision algorithm** since to transform e_1 into e_2, we need to consider the group axioms from left to right or right to left and this could cause an infinite process.

We can deduce from these group axioms the set of ten rules

$$\mathbb{1} \star x_1 \to x_1,$$

$$x_1^{-1} \star x_1 \to \mathbb{1},$$

$$(x_1 \star x_2) \star x_3 \to x_1 \star (x_2 \star x_3),$$

$$x_1^{-1} \star (x_1 \star x_2) \to x_2,$$

$$x_1 \star \mathbb{1} \to x_1,$$

$$\mathbb{1}^{-1} \to \mathbb{1},$$

$$\left((x_1)^{-1}\right)^{-1} \to x_1,$$

$$x_1 \star x_1^{-1} \to \mathbb{1},$$

$$x_1 \star \left(x_1^{-1} \star x_2\right) \to x_2,$$

$$(x_1 \star x_2)^{-1} \to x_2^{-1} \star x_1^{-1}.$$

Let $\Rightarrow$ be the **rewrite relation** obtained by extending $\to$ on the context.

If this rewrite system is **terminating** and **confluent**, then two formal group expressions e_1 and e_2 are equivalent iff their normal forms by $\Rightarrow$ coincide.

Example

Let the two formal group expressions $e_1 := \left(\left(x_1^{-1} \star x_3\right) \star x_3^{-1}\right) \star (x_1 \star x_2)$ and $e_2 := x_1^{-1} \star (x_1 \star x_2 \star \mathbb{1})$.
We have

$$e_1 = \left(\left(x_1^{-1} \star x_3\right) \star x_3^{-1}\right) \star (x_1 \star x_2) \Rightarrow \left(x_1^{-1} \star \left(x_3 \star x_3^{-1}\right)\right) \star (x_1 \star x_2) \Rightarrow \left(x_1^{-1} \star \mathbb{1}\right) \star (x_1 \star x_2) \Rightarrow x_1^{-1} \star (x_1 \star x_2) \Rightarrow x_2$$

and

$$e_2 = x_1^{-1} \star (x_1 \star x_2 \star \mathbb{1}) \Rightarrow x_1^{-1} \star ((x_1 \star x_2) \star \mathbb{1}) \Rightarrow x_1^{-1} \star (x_1 \star (x_2 \star \mathbb{1})) \Rightarrow x_1^{-1} \star (x_1 \star x_2) \Rightarrow x_2,$$

so that, as the same expression x_2 is obtained through the reduction process, e_1 and e_2 are equivalent.

We have described a way to decide the equivalence of expressions in the usual presentation of the group axioms. For this, we have used a **completion** process of these axioms [D. Knuth, P. Bendix, Simple Words Problems in Universal Algebras, 1970].

Some observations and questions in this context:

- How obtain the ten rules from the usual presentation of the group axioms?
- These rules come from [J.-M. Hullot, A catalogue of canonical term rewriting systems, 1980].
- Is this kind of completion possible for any other kinds of algebraic structures (like monoids, bands, lattices, *etc.*)? What are the conditions?

2.7. Alternative axiomatization of commutative groups

Consider commutative groups: these are groups such that binary operation $\star$ is commutative.

We can present these algebraic structures by the axioms

$$\mathbb{1} \star x_1 \equiv x_1 \equiv x_1 \star \mathbb{1},$$

$$x_1^{-1} \star x_1 \equiv \mathbb{1} \equiv x_1 \star x_1^{-1},$$

$$(x_1 \star x_2) \star x_3 \equiv x_1 \star (x_2 \star x_3),$$

$$x_1 \star x_2 \equiv x_2 \star x_1.$$

We have in this case

- three generating operations ($\star$, $\mathbb{1}$, and $^{-1}$);
- six axioms.

Is it possible to provide alternative presentations for commutative groups?

Let us define the **division** $/$ in commutative groups by

$$x_1/x_2 := x_1 \star x_2^{-1}.$$

We **recover** the usual operations in groups by mean of the division by

- $\mathbb{1} \equiv x_1/x_1$;
- $x_1^{-1} \equiv (x_1/x_1)/x_1$;
- $x_1 \star x_2 \equiv x_1/((x_1/x_1)/x_2)$.

Moreover, we can check that, from the usual group axioms and commutativity of $\star$, we have the **identity**

$$x_1/(x_2/(x_3/(x_1/x_2))) \equiv x_3.$$

The interesting point is that this single identity of which $/$ is subject implies all other axioms of the group in its usual presentation.

Therefore, we can axiomatize commutative groups by mean of a single binary operation and a **single axiom** [A. Tarski, Ein Beitrag zur Axiomatik der Abelschen Gruppen, 1938].

We have provided an **alternative description of commutative groups**, more minimalistic in some sense compared to the usual one.

Some observations and questions in this context:

- What is the theoretical framework to establish the discussed alternative presentation of commutative groups?
- Can we apply this framework to discover alternative presentations of other algebraic structures? What are the conditions for this to work?
- Can we describe some other algebraic structures only with a single axiom? To give some other interesting examples:
  - this exists for groups [G. Higman, B. H. Neumann, Groups as groupoids with one law, 1952];
  - this exists for lattices [W. McCune, R. Padmanabhan, R. Veroff, Yet another single law for lattices, 2003];
  - this exists for Boolean algebras [W. McCune, R. Veroff, B. Fitelson, K. Harris, A. Feist, L. Wos, Short single axioms for Boolean algebra, 2002];
  - this does not exist for semi-lattices [D. Potts, Axioms for semi-lattices, 1965];
  - this does not exist for distributive lattices [R. McKenzie, Equational Bases for Lattice Theories, 1970].

3. Abstract rewrite systems

3.1. Binary relations

Let X be a set. A *binary relation on* X is a subset $\mathcal{R}$ of X^2.

The property $(x, x') \in \mathcal{R}$ is denoted by $x \mathcal{R} x'$. The property $(x, x') \notin \mathcal{R}$ is denoted by $x \not\mathcal{R} x'$.

The *identity binary relation on* X is the binary relation

$$\mathcal{I}_X := \{(x, x) : x \in X\}.$$

As binary relations are sets, most of **set operations** can be used on binary relations on a same set X (union, intersection, complement, *etc.*).

The *composition* of two binary relations $\mathcal{R}_1$ and $\mathcal{R}_2$ on X is the binary relation

$$\mathcal{R}_1 \circ \mathcal{R}_2 := \left\{(x, x') \in X^2 : \text{there exists } y \in X \text{ such that } x \mathcal{R}_1 y \text{ and } y \mathcal{R}_2 x'\right\}.$$

This operation is **associative** and admits $\mathcal{I}_X$ as the **neutral element**.

Our convention for composition of binary relations is **left-to-right** (opposite to the usual convention for function composition).

Let $\mathcal{R}$ be a binary relation on X.

The *inverse* of $\mathcal{R}$ is the binary relation

$$\mathcal{R}^{-1} := \{(x, x') \in X^2 : x' \mathcal{R} x\}.$$

For any $k \in \mathbb{Z}$, the *k-th composition* of $\mathcal{R}$ is the binary relation

$$\mathcal{R}^k := \begin{cases} \mathcal{R}^{k-1} \circ \mathcal{R} & \text{if } k \geqslant 1, \\ \mathcal{I}_X & \text{if } k = 0, \\ \mathcal{R}^{-1} \circ \mathcal{R}^{k+1} & \text{otherwise } (k \leqslant -1). \end{cases}$$

Let us consider the following **closures**:

- the *reflexive closure* of $\mathcal{R}$, defined as $\mathcal{R}^0 \cup \mathcal{R}$;
- the *symmetric closure* of $\mathcal{R}$, defined as $\mathcal{R} \cup \mathcal{R}^{-1}$;
- the *transitive closure* of $\mathcal{R}$, defined as $\mathcal{R}^+ := \bigcup_{n \geqslant 1} \mathcal{R}^n$;
- the *reflexive and transitive closure* of $\mathcal{R}$, defined as $\mathcal{R}^* := \mathcal{R}^0 \cup \mathcal{R}^+$;
- the *reflexive, symmetric, and transitive closure* of $\mathcal{R}$, defined as $\mathcal{R}^\bullet := \left(\mathcal{R} \cup \mathcal{R}^{-1}\right)^*$.

3.2. Abstract rewrite systems

Definition

An *abstract rewrite system* (*ARS*) is a pair $(X, \Rightarrow)$ such that X is a set, called the *underlying set*, and $\Rightarrow$ is a binary relation on X, called the *rewrite relation*.

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

Let us introduce some notations:

- let $\Leftarrow$ be the **inverse** $\Rightarrow^{-1}$ of $\Rightarrow$.

  The ARS $(X, \Leftarrow)$ is the *dual* of $\mathcal{A}$;

- let $\Leftrightarrow$ be the **symmetric closure** of $\Rightarrow$;

- let $\equiv$ be the **reflexive, symmetric, and transitive closure** $\Rightarrow^{\bullet}$ of $\Rightarrow$.

  The **equivalence relation** $\equiv$ on X is the *convertibility relation* of $\mathcal{A}$.

  The **$\equiv$-equivalence class** of $x \in X$ is denoted by $[x]_{\equiv}$.

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

- The *set of successors* of $x \in X$ in $\mathcal{A}$ is the set $x^{\Rightarrow} := \{x' \in X : x \Rightarrow x'\}$.
  The map $x \mapsto x^{\Rightarrow} : X \to \mathcal{P}{\cdot}X$ is the *successor function* of $\mathcal{A}$.
- The *set of predecessors* of $x \in X$ in $\mathcal{A}$ is the set $x^{\Leftarrow}$ of successors of x in the dual of $\mathcal{A}$.
  The *predecessor function* of $\mathcal{A}$ is the successor function of the dual of $\mathcal{A}$.
- The *future* of $x \in X$ in $\mathcal{A}$ is the set $x^{\Rightarrow^*}$.
  The *future function* of $\mathcal{A}$ is the successor function of $(X, \Rightarrow^*)$.
- The *past* of $x \in X$ in $\mathcal{A}$ is the set $x^{\Leftarrow^*}$ of the future of x in the dual of $\mathcal{A}$.
  The *past function* of $\mathcal{A}$ is the future function of the dual of $\mathcal{A}$.

The *rewrite graph* of $\mathcal{A}$ is the **directed graph** having X as set of vertices and $\Rightarrow$ as set of arcs.

An ARS can be **specified equivalently** through its rewrite relation, through its successor function, or through its predecessor function.

Example

Let the ARS Succ $:= (\mathbb{N}, \Rightarrow)$ such that $n \Rightarrow n+1$ for any $n \in \mathbb{N}$. In Succ, we have for any $n \in \mathbb{N}$,

- $n^{\Rightarrow} = \{n+1\}$;
- $n^{\Rightarrow^*} = \{m \in \mathbb{N} : m \geqslant n\}$;
- $n^{\Leftarrow} = \{n-1\}$ if $n \geqslant 1$ and $0^{\Leftarrow} = \emptyset$;
- $n^{\Leftarrow^*} = \{m \in \mathbb{N} : m \leqslant n\}$.

Example

Let the ARS Pred defined as the dual of Succ. In Pred, we have for any $n \in \mathbb{N}$,

- $n^{\Rightarrow} = \{n-1\}$ if $n \geqslant 1$ and $0^{\Rightarrow} = \emptyset$;
- $n^{\Rightarrow^*} = \{m \in \mathbb{N} : m \leqslant n\}$;
- $n^{\Leftarrow} = \{n+1\}$;
- $n^{\Leftarrow^*} = \{m \in \mathbb{N} : m \geqslant n\}$.

Example

Let the ARS PredSucc $:= (\mathbb{N}, \Rightarrow)$ such that $n \Rightarrow n+1$ for any $n \in \mathbb{N}$ and $n \Rightarrow n-1$ for any $n \in \mathbb{N} \setminus \{0\}$. In PredSucc, we have for any $n \in \mathbb{N}$,

- $n^{\Rightarrow} = \{n-1, n+1\}$ if $n \geqslant 1$ and $0^{\Rightarrow} = \{1\}$;
- $n^{\Rightarrow^*} = \mathbb{N}$;
- $n^{\Leftarrow} = n^{\Rightarrow}$;
- $n^{\Leftarrow^*} = \mathbb{N}$.

Example

Let the ARS Factors $:= (\mathbb{N} \setminus \{0\}, \Rightarrow)$ such that $n \Rightarrow m$ if $m \in \mathbb{N}$ is a (proper or not) factor of $n \in \mathbb{N}$. In Factors, we have for any $n \in \mathbb{N}$,

- $n^{\Rightarrow} = \{m \in \mathbb{N} \setminus \{0\} : m \text{ divides } n\}$;
- $n^{\Rightarrow^*} = n^{\Rightarrow}$;
- $n^{\Leftarrow} = \{nm : m \in \mathbb{N} \setminus \{0\}\}$;
- $n^{\Leftarrow^*} = n^{\Leftarrow}$.

Example

Let for any $k \geqslant 1$ the ARS $\mathsf{Cycle}_k := (\llbracket k-1 \rrbracket, \Rightarrow)$ such that $n \Rightarrow n+1 \mod k$ for any $n \in \llbracket k-1 \rrbracket$. In Cycle_k, we have for any $n \in \llbracket k-1 \rrbracket$,

- $n^{\Rightarrow} = \{n+1 \mod k\}$;
- $n^{\Rightarrow^*} = \llbracket k-1 \rrbracket$;
- $n^{\Leftarrow} = \{n-1 \mod k\}$;
- $n^{\Leftarrow^*} = \llbracket k-1 \rrbracket$.

Example

Let the ARS $\mathsf{Grid} := \left(\mathbb{Z}^2, \Rightarrow\right)$ such that $(i,j) \Rightarrow (i+1,j)$ and $(i,j) \Rightarrow (i,j+1)$ for any $i,j \in \mathbb{Z}$. In Grid, we have for any $(i,j) \in \mathbb{Z}^2$,

- $(i,j)^{\Rightarrow} = \{(i+1,j),(i,j+1)\}$;
- $(i,j)^{\Rightarrow^*} = \left\{(i',j') \in \mathbb{Z}^2 : i' \geqslant i \text{ and } j' \geqslant j\right\}$;
- $(i,j)^{\Leftarrow} = \{(i-1,j),(i,j-1)\}$;
- $(i,j)^{\Leftarrow^*} = \left\{(i',j') \in \mathbb{Z}^2 : i' \leqslant i \text{ and } j' \leqslant j\right\}$.

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

In $\mathcal{A}$, $x \in X$ is

- *finitely branching* if $x^{\Rightarrow}$ is **finite**;
- *globally finite* if $x^{\Rightarrow^*}$ is **finite**;
- *acyclic* if $x \notin x^{\Rightarrow^+}$.

When all elements of X are finitely branching (resp. globally finite, acyclic), $\mathcal{A}$ is *finitely branching* (resp. *globally finite*, *acyclic*).

Examples

- The ARS Succ is finitely branching and not globally finite. Since for any $n \in \mathbb{N}$, $n^{\Rightarrow^+} = \{m \in \mathbb{N} : m > n\}$, Succ is acyclic.
- The ARS Pred is finitely branching and globally finite. Since for any $n \in \mathbb{N}$, $n^{\Rightarrow^+} = \{m \in \mathbb{N} : m < n\}$, Pred is acyclic.
- For any $k \geqslant 1$, the ARS Cycle_k is finitely branching and globally finite. Since, for any $n \in [\![k-1]\!]$, $n^{\Rightarrow^+} = [\![k-1]\!]$, Cycle_k is not acyclic.

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

Let I be a **nonempty initial interval** of $\mathbb{N}$ (possibly infinite).

A *rewrite sequence* in $\mathcal{A}$ is a sequence $u = (u_i)_{i \in I}$ on X such that for any $i \in I$, if $i+1 \in I$, then $u_i \Rightarrow u_{i+1}$.

If $u = (u_i)_{i \in I}$ is a rewrite sequence in $\mathcal{A}$:

- u *starts from* u_0;
- when I is finite, u *ends at* u_ℓ where ℓ is the greatest element of I;
- when I is finite, the *length* $\ell \cdot u$ of u is $\#I - 1$, that is, the greatest element of I.

Example

The sequence

$$(0,-1)\,(0,0)\,(0,1)\,(1,1)\,(2,1)\,(3,1)$$

is a rewrite sequence in Grid starting from $(0,-1)$, ending at $(3,1)$, and of length 5.

Moreover,

$$(1,1)\,(1,2)\,(1,3)\,(1,4)\ldots$$

is an infinite rewrite sequence in Grid starting from $(1,1)$.

Let $\mathcal{A} := (X, \Rightarrow)$ and $\mathcal{A}' := (X', \Rightarrow')$ be two ARSs.

1. If $X' \subseteq X$ and $\Rightarrow' \subseteq \Rightarrow \cap X'^2$, then $\mathcal{A}'$ is a *sub-ARS* of $\mathcal{A}$.
2. If $\mathcal{A}'$ is a sub-ARS of $\mathcal{A}$ and $\Rightarrow' = \Rightarrow \cap X'^2$, then $\mathcal{A}'$ is an *induced sub-ARS* of $\mathcal{A}$.
3. If $\mathcal{A}'$ is an induced sub-ARS of $\mathcal{A}$ and for any $x \in X$ and $x' \in X'$, $x' \Rightarrow x$ implies $x \in X'$, then $\mathcal{A}'$ is a *closed sub-ARS* of $\mathcal{A}$.

Examples

1. The ARS $\left(\mathbb{N}^2, \Rightarrow\right)$ such that $(i,j) \Rightarrow (i+1, j)$ for any $i, j \in \mathbb{N}$ is a sub-ARS of Grid.
2. The ARS $(\{(0,0), (1,0), (0,1), (1,1)\}, \Rightarrow)$ such that $(0,0) \Rightarrow (1,0)$, $(0,0) \Rightarrow (0,1)$, $(1,0) \Rightarrow (1,1)$, and $(0,1) \Rightarrow (1,1)$ is an induced sub-ARS of Grid.
3. The ARS $(\{(i,j) : i, j \geqslant 1\}, \Rightarrow)$ such that $(i,j) \Rightarrow (i+1, j)$ and $(i,j) \Rightarrow (i, j+1)$ for any $i, j \geqslant 1$ is a closed sub-ARS of Grid.