Termination orders for 3-dimensional rewriting

Yves Guiraud

16th November 2004 - Last modification: 30th August 2005

Termination orders for 3-dimensional rewriting

Yves Guiraud[1]

Abstract: This paper studies 3-polygraphs as a framework for rewriting on two-dimensional words. A translation of term rewriting systems into 3-polygraphs with explicit resource management is given, and the respective computational properties of each system are studied. Finally, a convergent 3-polygraph for the (commutative) theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces is given. In order to prove these results, it is explained how to craft a class of termination orders for 3-polygraphs.

Outline

This paper starts with the introductory section 1 on equational theories and term rewriting systems. It gives notations and graphical representations that are used in the sequel. Then, it focuses on one major restriction of term rewriting, namely the fact that it cannot provide convergent presentations for *commutative* equational theories: equational theories that contain a commutative binary operator.

Section 2 studies the resource management operations of permutation, erasure and duplication: they are implicit and global in term rewriting and it is sketched there how to make them explicit. However, the framework for rewriting in algebraic structures needs to be extended to include this change; section 3 proposes 3-polygraphs to fulfill this role. Here, these objects, introduced in [Burroni 1993], are used as equational presentations of a special case of 2-categories: MacLane's product categories, called PROs, for short, in [MacLane 1965].

These first three sections do not introduce new material, but focus on the notations, representations, terminology and philosophy of this paper. Then section 4 gives some relations between term rewriting systems and 3-polygraphs: a translation from the former to the latter is built and some properties are given. The main result of the section is the proof of a conjecture from [Lafont 2003]: any left-linear convergent term rewriting system can be translated into a convergent 3-polygraph.

To prove some of these results, one needs new tools, in adequation with the more complicated structure of polygraphs. In particular, section 5 introduces a recipe to build termination orders for them. Section 6 consists in the application of this technique to prove some termination results of section 4. Finally, section 7 applies the same technique to prove the termination of the 3-polygraph $\mathrm{L}(\mathbb{Z}_2)$ which was introduced in [Lafont 2003] and, since then, was already known to be a confluent presentation of the equational theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces. It is therefore the first known convergent presentation of a commutative equational theory.

[1]Institut de mathématiques de Luminy, Marseille, France - guiraud@iml.univ-mrs.fr

1 Equational theories and term rewriting systems

Universal algebra provides different types of objects in order to modelize algebraic structures. Among them are *equational theories*: these are presentations by generators (or *operators*) and relations (or *equations*, *equalities*). As an example, the equational theory of *monoids* is a pair (Σ, E_0) consisting of the *signature* Σ (a set of operators) and the family E_0 of equations given by:

$$\Sigma = \{\, \mu : 2 \to 1,\ \eta : 0 \to 1 \,\},$$
$$E_0 = \big(\, \mu(\mu(x,y),z) = \mu(x,\mu(y,z)),\ \mu(\eta,x) = x,\ \mu(x,\eta) = x \,\big).$$

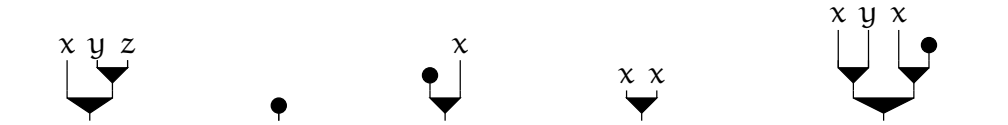
Each operator has a finite number of inputs and of outputs. When each one has exactly one output, which is the case here, the signature is said to be *algebraic*. The given equational theory (Σ, E_0) is said to be *the theory of monoids* since monoids are exactly sets endowed with a binary operation and a constant, such that the operation is associative and admits the constant as a left and right unit.

The formal operations one can form on any set with a binary operation and a constant are called the *terms* built from the signature Σ. There exist numerous ways to build the set $T\Sigma$ of such terms, and each one gives a different representation for them. Two are used here, a *syntactic* one and a *diagrammatic* one. For each one, a fixed countable set V is needed; its elements are called *variables*.

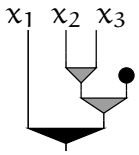
The classical representation of terms define them inductively with the following construction rules: the first one states that each variable is a term; furthermore, the constant η is a term; then, for any two terms u and v, the formal expression $\mu(u,v)$ is a term.

The diagrammatic representation starts with the assignment, for each operator with n inputs, of an arbitrarily chosen tree of height one with n leaves. For example, one can fix the following trees:

Then, the terms are all the trees one can build from these two generating trees and which leaves are labelled with variables. As an example, the following figure pictures terms built from the signature Σ, with the two representations for each one:

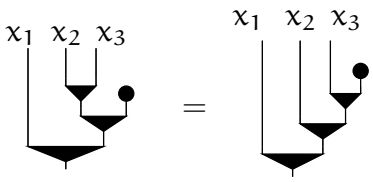


The equations from the theory of monoids generate equalities between terms that represent the same operation, through a *rewriting* process. Let us sketch how this works. For example, the following term contains the tree-part of the associativity rule left-member, which has been greyed out:



Hence, the associativity equation generates an equality between the chosen term and another. To determine which one, let us follow the following method, which consists of three steps: at first, the remaining (black) part of the term is copied; then, in the space left empty, the other member of the rule is placed; finally, the two parts obtained are joined (by dotted lines), according to the respective position of the variables in each member of the equation. Concerning our example, this process is pictured as follows:

Note that each variable appears once and in the same position in each member of the associativity rule, so that the links are direct. When the second term is compacted, the following equality holds and is said to be generated by the associativity equation:



In order to study the computational properties of these rewriting processes, *term rewriting systems* are useful; they can be defined as oriented equational theories. Indeed, such a rewriting system is defined from an equational theory by keeping the same operators and replacing each equation by a *rewrite rule*: it is an oriented version of the equation, which can only be used in one way. As an example, starting from the equational theory of monoids, one can form the term rewriting system (Σ, R_0), where Σ is still the same algebraic signature made of a product μ and a unit η and R_0 is the following set of three rules:

$$\mu(\mu(x,y),z) \to \mu(x,\mu(y,z)), \quad \mu(\eta,x) \to x, \quad \mu(x,\eta) \to x.$$

Rewrite rules generate *reductions* instead of equalities, and a graph containing terms as vertices and reductions as edges is called a *reduction graph*. Some geometrical properties of reduction graphs are of particular interest since they have consequences on computational properties of the rewriting process. Among these geometrical properties, three are particularly studied: *termination*, *confluence* and *convergence*.

A rewriting system *terminates* if it contains no infinite length reduction paths such as:

$$u_0 \to u_1 \to u_2 \to \ldots \to u_n \to u_{n+1} \to \ldots$$

Intuitively, this means that the rewriting calculus must end after a finite time, whatever the input is. This is formalized by the following consequence of termination: every term u has at least one *normal form* $\hat{u}$; this means that $\hat{u}$ is a term such that there exists a finite reduction path from u to $\hat{u}$ (denoted by $u \twoheadrightarrow \hat{u}$) and $\hat{u}$ is irreducible (no rule can apply on it).

A rewriting system is *confluent* if, whenever there exist three terms u, v and w such that $u \twoheadrightarrow v$ and $u \twoheadrightarrow w$, then there exists a fourth term t such that $v \twoheadrightarrow t$ and $w \twoheadrightarrow t$. Intuitively, this means that choices made between two rules that can transform the same term do not have any consequence on a potential final result; equivalently, this means that any term has at most one normal form.

Thus, one defines the last property: a rewriting system is *convergent* when it is both terminating and confluent. One immediate consequence is that any term has exactly one normal form. This property is very useful for several purposes.

One of the most known is the following usage: let us assume that (Σ, E) is an equational theory and that (Σ, R) is a rewriting system that is a *finite convergent presentation* of (Σ, R), which means that it is a convergent rewriting system with a finite number of rules and such that two terms are equal in the equational theory if and only if there exists a non oriented reduction path between these two terms in the rewriting system. Then there exists a decision procedure to check if two terms u and v are equal or not.

Indeed, one computes their unique normal forms $\hat{u}$ and $\hat{v}$. Note that this is where the finiteness condition is useful: it allows one to check if a term is a normal form. Then the two normal forms $\hat{u}$ and $\hat{v}$ are compared: u and v are equal in the equational theory if and only if $\hat{u}$ and $\hat{v}$ are (synctactically) equal.

However, term rewriting systems have a major restriction in this field: there is a large class of equational theories for which they cannot provide a convergent presentation. These are the commutative theories, fairly frequent in algebra, which are equational theories with a commutative binary operator. As an example, let us take a look at one of the simplest, namely the equational theory of commutative monoids. Its signature is still Σ; its set E_1 of equations is made of the same three as the ones for monoids (associativity and left and right units) plus the following one expressing the commutativity of the product:

$$\mu(x, y) = \mu(y, x).$$

From this theory, one can form a number of term rewriting systems, such as the one with Σ as signature and with the following choice R_1 of orientations for equations:

$$\mu(\mu(x, y), z) \to \mu(x, \mu(y, z)), \quad \mu(\eta, x) \to x, \quad \mu(x, \eta) \to x, \quad \mu(x, y) \to \mu(y, x).$$

Note that the last rule could have been chosen in the reverse direction, but it would not change the following fact: this rule generates infinite reduction paths. Indeed, for any two terms u and v, the commutativity rules generates:

$$\mu(u, v) \to \mu(v, u) \to \mu(u, v) \to \mu(v, u) \to \dots$$

The purpose of this paper is to provide a framework where some commutative equational theories admit convergent presentations: 3-polygraphs. Links between term rewriting systems and 3-polygraphs are studied and a new tool to prove termination is given and applied on some examples.

The equational theory that provides the main example here is the one of $\mathbb{Z}/2\mathbb{Z}$-vector spaces: it has the same operators as the previous ones (the binary product embodies the sum and the unit is the zero) and a set E_2 of five equations made of the four from E_1 (associativity, left and right units and commutativity) plus the following fifth equation:

$$\mu(x, x) = \eta.$$

It expresses the fact that, in a $\mathbb{Z}/2\mathbb{Z}$-vector space, any element is its own opposite. This theory is prefered to the theory of commutative monoids for two reasons. The first one is theoretical: any boolean algebra has an underlying $\mathbb{Z}/2\mathbb{Z}$-vector space, so that any convergent presentation for $\mathbb{Z}/2\mathbb{Z}$-vector spaces is a first step towards one for boolean circuits. The second one concerns the application range of the tools developped here: this fifth equation has some nasty computational effects and is thus important to encompass in the new framework, so that it can be used for other applications.

From the theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces, the term rewriting system (Σ, R_2) is built, where R_2 is the following choice of orientations:

$$\mu(\mu(x,y),z) \to \mu(x,\mu(y,z)), \quad \mu(\eta,x) \to x, \quad \mu(x,\eta) \to x, \quad \mu(x,y) \to \mu(y,x), \quad \mu(x,x) \to \eta.$$

Note that this rewriting system is neither terminating nor confluent but will serve as a starting point to build a convergent presentation. This transformation will start with the study of the so-called *resource management operations*. For further information on (term) rewriting systems, one can refer to [Baader Nipkow 1998].

2 Resource management operations

Let us recall the last step of the term rewriting process: one has to draw links between two parts of a term, according to the variables occuring in the corresponding rule. As mentionned earlier, the rewriting example in section 1 is the simpliest case: indeed, the variables occur once each and in the same order in each member of the associativity rule. However, if this is not the case, one has two use additional operations before links are drawn: these operations are called the *resource management operations* and there are three of the kind, *permutation*, *erasure* and *duplication*.

Permutation is used, for example, when the commutativity rule is applied. Indeed, when in this case, one has to use a permutation operation that will exchange the two grey subterms in any term such as the following generic one:

The second operation, erasure, is used in the following case, for example: let us consider a theory containing a binary operator and a constant which is a right absorbing element. The following figure displays a rule which expresses this property (on the right) together with a generic application of this rule (on the left); this requires an intermediate operation that erases the grey subterm:

Finally, the last operation, called duplication, can occur in the following case: let us consider a theory containing two binary operators, one of which is left-distributive with respect to the other. Then, when applied, a rule that expresses this property (such as the one pictured on the right) requires the use of an

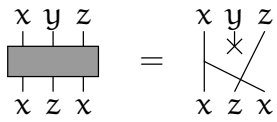
operation that can duplicate the greymost subterm (and exchange one of its copies with another subterm, but this is the already-encountered permutation):

Thus, in term rewriting, these three operations are both *implicit* (they are not specified by rules) and *global* (they act immediately on subterms of any size). We are now going to sketch how one can make them explicit and local: only the idea is given here, the full translation is postponed to section 4.

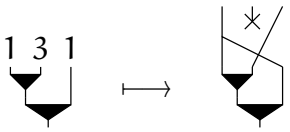
Let us start with the following observation: the use of the three resource management operations is specified both by the number of occurences and the order of appearance of each variable in each member of a rewrite rule. Thus, in order to make these operations explicit, variables will be replaced by some additional operators that will represent local permutations, erasers and duplicators; furthermore, rules will guarantee the global behaviour of these local operators.

In order to give an idea of how the translation works, let us start with the study of this term, which represents the operation $(x, y, z) \mapsto \mu(\mu(x, z), x)$:

Seen as an operation, it is the composite of $(x, y, z) \mapsto (x, z, x)$ followed by $(x, y, z) \mapsto \mu(\mu(x, y), z)$. The first operation can be pictured as the following diagram (a shunter), since its action is to tell where each of the three arguments goes in the term:

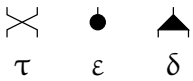


This diagram will be formalized as a composite of new operators and the term will be translated this way (with some explanations below):



Variables in the term have been replaced by ordinals; indeed, we have seen that variables are just labels corresponding to the first, second, third, etc. arguments taken by the corresponding operation. Hence, they will be replaced by ordinals whenever it makes the translation clearer. The second remark is also about variables, but in the translated diagram: they will always appear, after translation, in order: 1, 2, 3, etc. Thus, they have no purpose anymore; they will therefore vanish, as in the diagram.

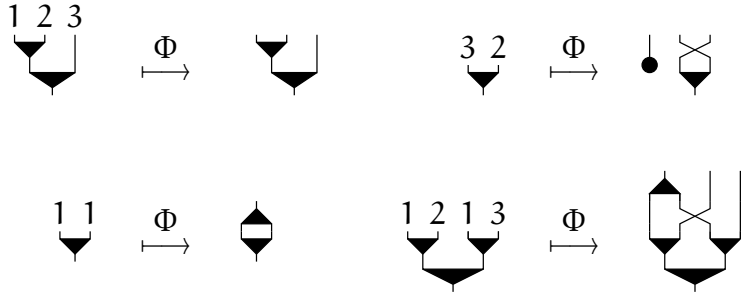
Finally, let us see what operators will be added to the signature and sketch how to translate terms and rules. One operator is added for each resource management operation: indeed, in order to formalize our previous diagram, one must be able to exchange two arguments, erase one or duplicate another one. Thus, we fix a (non-algebraic) signature Δ made of the following three *resource management* operators:



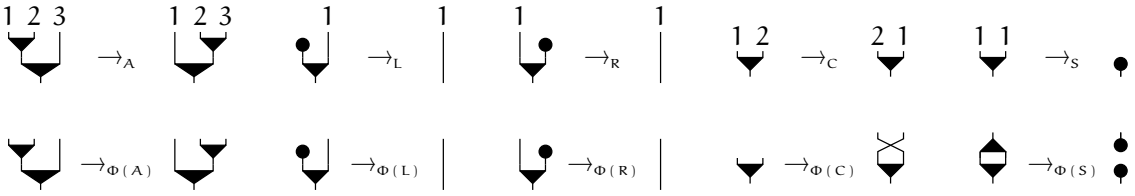
Each one has a representation that makes explicit the operation one wishes it to embody. Some rules will be added to ensure their global behaviour, but they will be given in section 4. For the moment, the only thing we need to know is that these rules give the following interpretations to these three operators:

$$\tau(x, y) = (y, x), \quad \varepsilon(x) = (\texttt{nothing}), \quad \delta(x) = (x, x).$$

Now, let us sketch how terms are translated: first, the tree-part is copied; then and progressively, resource management operators are added on the top of the copy, according to the variables that appear in the term. The following figure gives four sample translations (the translating map is denoted by Φ thereafter):



Then, let us see how to translate the five rules of our term rewriting system derived from the theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces. Each rule is pictured in order (associativity, left and right units, commutativity and self-inverse), has been given a name (A, L, R, C and S) and has its translation written just below:



Note that several cases may occur. For the first three rules, no resource management operator is added during translation: these three rules are *linear* (or *left-* and *right-linear*). When translated, the commutativity rule has one operator added on its right side and none on its left side: it is a left-linear but not right-linear rule. Finally, the self-inverse rule has one operator added on each of its members during translation: it is neither left- nor right-linear.

Some issues have now been arisen. The first one concerns the rules to be added in order both to describe the behaviour of our local permutation, eraser and duplicator and to ensure the global coherence of these local rules.

The next issue is about the respective computational properties of the starting term rewriting system and of the rewriting system one gets as a result of making the resource management operations explicit. These first two issues are adressed in section 4.

For the moment, we are concerned with a third issue: where does rewriting takes place now? Indeed, starting from a term rewriting system, we have crafted another rewriting system which is not a term one, and for two reasons. The first one is that its signature contains non-algebraic operators, that is operators that do not have exactly one output (the resource management operators have zero or two outputs). The second reason is that variables have been dropped to be replaced by these new operators: this is also a step outside term rewriting. Hence, our new object is not a term rewriting system and section 3 recalls a notion from [Burroni 1993] used to describe it.

3 Three-dimensional polygraphs

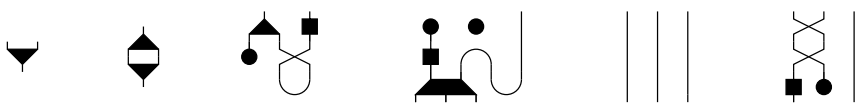
Like equational theories, *3-polygraphs* are useful objects in universal algebra, in the sense that they allow one to present algebraic structures by generators and relations. However, they are far more general than equational theories, and this has two consequences: on one hand, they can handle more general objects, like the rewriting system sketched in section 2, or the structure of quantum groups; but, on the other hand, their generality comes with an increase in the structural complexity: the development of new tools is mandatory to prove termination, for example.

Polygraphs are genuine categorical objects but we prefer a diagrammatic definition here. For this paper, a 3-polygraph is made of a signature, that is a set of operators with a finite number of inputs and a finite number of outputs, together with a family of rules: in fact, this is just a special case of 3-polygraph, one with only one 0-cell and one 1-cell. For the complete theory of n-polygraphs, the interested reader should check [Burroni 1993].

The operators are once again represented by fixed diagrams of size one, with as many free edges at the top as the operator inputs and as many free edges at the bottom as the operator outputs. For example, some usual diagram shapes are pictured here:

Some of them have already been encountered, some of the others are less algebraic: one has zero input and output - it is usefull to describe Petri nets, see [Guiraud 2004] -, one has two inputs and zero output - it is used together with its dual with zero input and two outputs to represent knots and tangles.

Here, the "terms" one considers are all the circuits one can build with all these elementary diagrams: these are the *Penrose diagrams* (or *circuits*) one can build with the size one diagrams representing the operators, such as:



Each of these circuits has a finite number of inputs (on the top) and of outputs (on the bottom) but has no variable. Furthermore, they need not be connected, as the three-inputs and three-outputs wire-only one.

These circuits, which are also called diagrams or arrows, have an algebraic structure. To explain it, let us use the notation $f : m \to n$ to express that f is a circuit with m inputs and n outputs. For any circuit f, $s(f)$ is its number of inputs and $t(f)$ its number of outputs. The following constructions and properties are valid for circuits:

- Let $f : m \to n$ and $g : n \to p$. Then, one can connect each output of f with the corresponding input of g, in the same order, to form a new circuit with m inputs and p outputs denoted by $g \circ f$.
- This composition operation admits local units: a circuit $f : m \to n$ satisfies $f \circ m = f$ and $n \circ f = f$, where p is the wire-only circuit with p inputs and p outputs.
- Let $f : m \to n$ and $g : p \to q$. Then, one can put f and g side by side to form a new circuit with $m + p$ inputs and $n + q$ outputs, denoted by $f \otimes g$.
- This product operation admits a bilateral neutral element: the empty circuit 0 with no input nor output, represented by an empty diagram.
- Finally, the composition and product are related by the *exchange relations*. They are given by the following equality, that is required to hold for any two circuits $f : m \to n$ and $g : p \to q$:

$$(t(f) \otimes g) \circ (f \otimes s(g)) = f \otimes g = (f \otimes t(g)) \circ (s(f) \otimes g).$$

Definition 3.1. A family $\mathcal{C}$ of circuits endowed with this structure $\otimes$ and $\circ$, satisfying the aforegiven unit and exchange relations, is called a *product category*; the subset of circuits with m inputs and n outputs is denoted by $\mathcal{C}(m, n)$. When the circuits of $\mathcal{C}$ are freely built from a signature Σ, this object is the *free product category generated by* Σ, denoted by $\langle \Sigma \rangle$. ♦

Remark 3.2. Product categories, or PROs, were defined in [MacLane 1965]. An alternative definition is: a product category is a strict monoidal category whose underlying monoid of objects is $(\mathbb{N}, +, 0)$, the one of natural numbers with addition and zero. In [Guiraud 2004], such a category was called a *(monochromatic) operad*, for this structure is a common generalization of many universal algebra objects: May's operads, Lawvere's algebraic theories and MacLane's PROs and PROPs.

Product categories are also a special case of *2-monoids* or *2-categories with only one 0-cell*. A generalization of this paper results should be possible, since circuit-like diagrams extend to general 2-cells. For this paper, we stick to MacLane's product categories, but all this terminology will be made clear in subsequent work.

A *rewrite rule* on a product category $\mathcal{C}$ is a pair $f \to g$ of *parallel* arrows (they have the same number of inputs and the same number of outputs). Such a rule generates *reductions* on circuits: whenever an arrow h contains f, the rule generates a reduction from h to k, where k is the same as h, except that f has been replaced by g. The fact that f and g have the same number of inputs and the same number of outputs ensures that one can connect the unchanged part of the circuit with the changed part, without using implicit operations before.

Definition 3.3. A *3-polygraph* is a pair (Σ, R) where Σ is a signature and R is a family of rewrite rules on $\langle\Sigma\rangle$.

One way to formalize the reduction relation generated by rules on a free product category $\langle\Sigma\rangle$ is to define *contexts*. We just explain here what they are, avoiding to dig further into the technical aspects, developped in [Guiraud 2004]. Let Σ be a signature. Then, a *context* on $\langle\Sigma\rangle$ is a circuit c with a "hole" inside: this hole has a finite number of inputs and of outputs where on can paste a circuit f with correponding numbers of inputs and outputs; this pasting operation results in a circuit denoted by $c[f]$. Then, a rule $f \to g$ generates a reduction from each circuit $c[f]$, with c any context, to the circuit $c[g]$.

Finally, given two product categories $\mathcal{C}$ and $\mathcal{D}$, a *product category functor from $\mathcal{C}$ to $\mathcal{D}$* is a map which sends each circuit of $\mathcal{C}$ onto a circuit of $\mathcal{D}$ with the same number of inputs and of outputs, and which preserves identities, products and compositions. When $\mathcal{C}$ is the free product category $\langle\Sigma\rangle$, then a classical categorical argument tells us that any product category functor $F : \langle\Sigma\rangle \to \mathcal{D}$ is entirely and uniquely given by the circuits $F(\varphi)$ in $\mathcal{D}$, for every operator φ in Σ.

4 From term rewriting to 3-polygraphs

This section uses results from [Burroni 1993], presented in a slightly different way, in order to prove a conjecture from [Lafont 2003]: this is theorem 4.6. This is the result that allows the definition 4.8 of a translation Φ from any term rewriting system into a 3-polygraph. Proposition 4.11 and theorem 4.12 give the respective computational properties of the term rewriting system and the 3-polygraph.

In section 2, a 3-polygraph has been built from the term rewriting system (Σ, R_2), which presents the equational theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces. Its signature, denoted by Σ^c, is the one built from Σ by addition of the three resource managment operators τ, δ and ε from Δ. Its family of rules, denoted by $\Phi(R_2)$, consists of the translations $\Phi(A)$, $\Phi(L)$, $\Phi(R)$, $\Phi(C)$ and $\Phi(S)$ of the five rules from the original term rewriting system. This construction can be generalized to any term rewriting system but is still incomplete for the moment. It lacks two families of rules and this section starts with their description.

Let us fix an algebraic signature Σ. The set of terms built on the signature Σ and on some fixed countable set V of variables is denoted by $T\Sigma$. Let us assume that the set V is endowed with a total order (given by a bijection with $\mathbb{N}$), so that the variables can be written x_1, x_2, x_3, etc. For any term u, the notation $\sharp u$ is used for the greatest natural number i such that x_i appears in u. Then, we define $\mathbb{T}\Sigma(m, n)$ to be the set of families $(u_1, \dots, u_n)$ of n terms such that $\sharp u_i \leq m$ for every i. Note that the set $\mathbb{T}\Sigma(m, 0)$ has only one element, denoted by $*(m)$. The following operations provide the set $\mathbb{T}\Sigma$ with a product category structure:

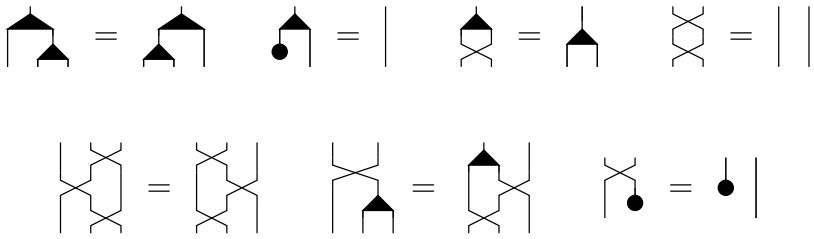
- If $u = (u_1, \dots, u_n)$ is in $\mathbb{T}\Sigma(m, n)$ and $v = (v_1, \dots, v_p)$ is in $\mathbb{T}\Sigma(n, p)$, then their composite $v \circ u$ is the family $(w_1, \dots, w_p)$ where each w_i is built from v_i by replacing each x_j with u_j.
- The identity of n, for any natural number n, is the family $(x_1, \dots, x_n)$.
- The product $u \otimes v$ of $u = (u_1, \dots, u_n)$ in $\mathbb{T}\Sigma(m, n)$ and of $v = (v_1, \dots, v_q)$ in $\mathbb{T}\Sigma(p, q)$ is the family $(w_1, \dots, w_{n+q})$ built that way: if i lies between 1 and n, then w_i is u_i; otherwise, w_{i+n} is v_i where each x_j has been replaced by x_{j+m}.

Furthermore, this product category satisfies some additional properties. The first one is that $\mathbb{T}\Sigma$ is a *cartesian* category: seen as a strict monoidal category, the monoidal product $\otimes$ is the functorial part of a cartesian product. In our case and informally, this means that every circuit $f : m \to n$ is entirely and uniquely determined by n circuits $m \to 1$, in the same way that any function $f : X^m \to X^n$, where X is a set, is entirely and uniquely determined by n functions $X^m \to X$: its components. To check that $\mathbb{T}\Sigma$ is indeed cartesian, one uses a result from [Burroni 1993], restricted to our setting:

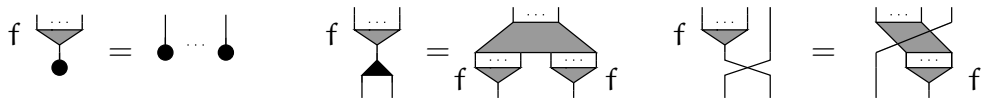
Theorem 4.1 (Burroni). *A product category $\mathcal{C}$ is cartesian if and only if it contains three arrows:*

Such that the two following families of equations hold:

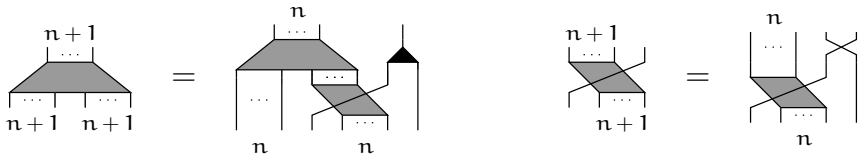
1. *The family E_Δ, made of the following seven equations:*



2. *The family E_Σ, made of three equations for each integer n and each arrow $f : n \to 1$ in $\mathcal{C}$:*



The following recursively defined arrows families $(\delta_n)_{n\in\mathbb{N}}$ and $(\tau_{n,1})_{n\in\mathbb{N}}$ have been used:



with the initial values $\delta_0 = 0$ and $\tau_{0,1} = 1$.

Note that the following convention is now used in diagrams: generating operators are drawn with black diagrams, while composite arrows are grey. The union of the two families E_Δ and E_Σ is denoted by $E_{\Delta\Sigma}$. Theorem 4.1 is not mandatory to get the following proposition but yields an easy proof of it:

Proposition 4.2. *The product category $\mathbb{T}\Sigma$ is cartesian.*

Proof. Let us start with the definition of the three arrows from theorem 4.1: the arrow τ is the pair (x_2, x_1) of terms; the arrow δ is (x_1, x_1); finally, the arrow ε is the empty family $*(1)$. Computations to check the equations of theorem 4.1 are straightforward. ♢

The next step consists in the proof that $\mathbb{T}\Sigma$ is the *free* cartesian category generated by the algebraic signature Σ. In order to prove this fact, one starts with another use of theorem 4.1:

Corollary 4.3 (of theorem 4.1). *For every algebraic signature Σ, the category $\langle \Sigma^c \rangle / E_{\Delta\Sigma}$ is the free cartesian category generated by Σ.*

Hence, in order to prove that $\mathbb{T}\Sigma$ is another version of the free cartesian category generated by Σ, it is sufficient to prove that there exists an isomorphism $\hat{\Phi} : \mathbb{T}\Sigma \to \langle \Sigma^c \rangle / E_{\Delta\Sigma}$.

The signature Σ is contained in $\mathbb{T}\Sigma$: one defines an inclusion i which sends each $\varphi : n \to 1$ from Σ onto the term $\varphi(x_1, \ldots, x_n)$. Hence, corollary 4.3 extends i into a cartesian functor F from $\langle \Sigma^c \rangle / E_{\Delta\Sigma}$ to $\mathbb{T}\Sigma$: this functor sends each φ from Σ onto $i(\varphi)$ and τ, δ and ε respectively onto (x_2, x_1), (x_1, x_1) and $*(1)$.

Conversely, let us consider an arrow $f = (u_1, \ldots, u_n)$ in $\mathbb{T}\Sigma(m, n)$. Each term u_i can be written $u_i = f_i(y_1^i, \ldots, y_{k_i}^i)$, with k_i an integer, f_i an arrow in $\langle \Sigma \rangle(k_i, 1)$ and each y_j^i a variable from $\{x_1, \ldots, x_m\}$. Furthermore, this decomposition of terms is unique. Thus, the arrow f uniquely decomposes into:

$$f = (f_1 \otimes \ldots \otimes f_n) \circ (y_1^1, \ldots, y_{k_n}^n).$$

There remains to prove that every family $(y_1, \ldots, y_k)$ of variables in $\{x_1, \ldots, x_m\}$ can be uniquely written (*modulo* E_Δ) with the three arrows $i(\tau)$, $i(\delta)$ and $i(\varepsilon)$. This can be done in two steps.

Let us define the sub-product category $\mathbb{V}$ of $\mathbb{T}\Sigma$ by restricting ourselves to families of variables: this is $\mathbb{T}\emptyset$, where $\emptyset$ denotes the signature with no operator. One also defines the cartesian category $\mathbb{F}^o$ of *finite sets* with: the arrows of $\mathbb{F}^o(m, n)$ are in bijective correspondance with the functions from the finite set $[n] = \{1, \ldots, n\}$ to $[m]$. Then:

Lemma 4.4. *The cartesian categories $\mathbb{V}$ and $\mathbb{F}^o$ are isomorphic.*

Proof. Let $(y_1, \ldots, y_n)$ be a family of variables taken in $\{x_1, \ldots, x_m\}$. Then, there exists an unique function f^* from $[n]$ to $[m]$ such that $y_i = x_{f^*(i)}$ for each i. Let us fix $\theta(y_1, \ldots, y_n)$ as the arrow f in $\mathbb{F}^o$ that corresponds to f^*. Conversely, if f is an arrow in $\mathbb{F}^o(m, n)$: let us denote by f^* the corresponding function from $[n]$ to $[m]$. Then one defines $\omega(f) = (x_{f^*(1)}, \ldots, x_{f^*(n)})$. There remains to check that θ and ω are cartesian functors which are inverse one another, which is straightforward. ♢

The second step uses another result from [Burroni 1993]:

Theorem 4.5 (Burroni). *The cartesian categories $\mathbb{F}^o$ and $\langle \Delta \rangle / E_\Delta$ are isomorphic.*

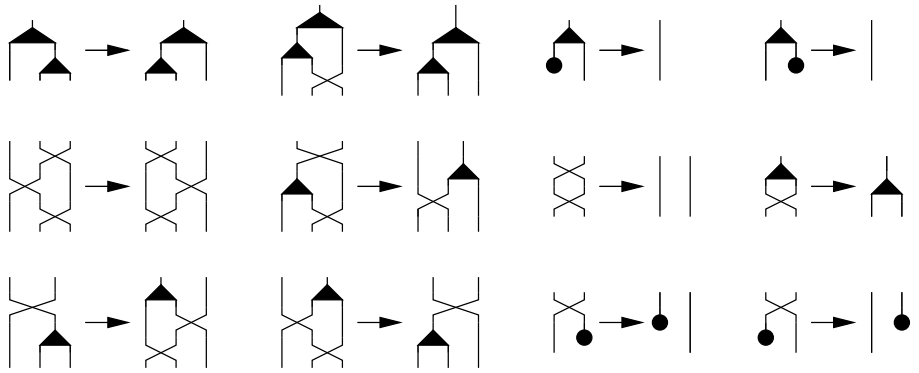
Hence, the cartesian categories $\mathbb{V}$ and $\langle \Delta \rangle / E_\Delta$ are isomorphic. Consequently, each family $(y_1, \ldots, y_k)$ of variables taken in $\{x_1, \ldots, x_n\}$ corresponds to a unique arrow in $\langle \Delta \rangle / E_\Delta$. Furthermore, each arrow f in $\mathbb{T}\Sigma(m, n)$ admits a unique decomposition $f = f_\Sigma \circ f_\Delta$ with f_Σ in $\langle \Sigma \rangle$ and f_Δ in $\mathbb{V}$.

Finally, one gets that the cartesian functor F from $\langle \Sigma^c \rangle / E_{\Delta\Sigma}$ to $\mathbb{T}\Sigma$ is an isomorphism. However, we want an map from $\mathbb{T}\Sigma$ to $\langle \Sigma^c \rangle$: let us find a convergent 3-polygraph $(\Sigma^c, R_{\Delta\Sigma})$ such that $\langle \Sigma^c \rangle / R_{\Delta\Sigma}$ is isomorphic to $\langle \Sigma^c \rangle / E_{\Delta\Sigma}$ and use the unique normal form property.

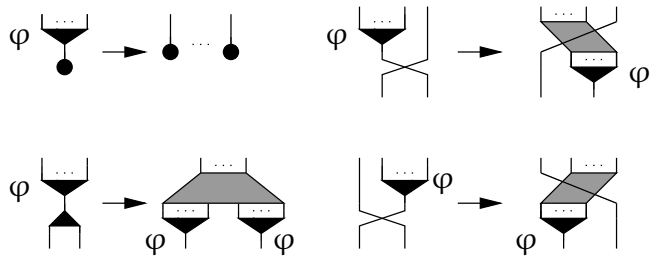
A conjecture from [Lafont 2003] is proved:

Theorem 4.6. *For any algebraic signature Σ, the 3-polygraph $(\Sigma^c, R_{\Delta\Sigma})$ is convergent and $\langle \Sigma^c \rangle / R_{\Delta\Sigma}$ is isomorphic to the free cartesian category $\langle \Sigma^c \rangle / E_{\Delta\Sigma}$ generated by Σ, where the family of rules $R_{\Delta\Sigma}$ is made of the following two subfamilies:*

1. *The family* R_Δ*:*



2. *The family* R_Σ *given, for each integer* n *and each operator* φ *in* $\Sigma(n,1)$*, by:*



Remark 4.7. Three families of verifications need to be done. The first one consists in checking that the new rules are derivable from $E_{\Delta\Sigma}$, which is straightforward.

The second one is much more complicated: one needs to check that the 3-polygraph terminates. However, the structural complexity of polygraphs requires new techniques since the usual ones used in rewriting do not work. One way to craft reduction orders for 3-polygraphs is made explicit in section 5 and used in section 6 in order to prove the termination of $(\Sigma^c, R_{\Delta\Sigma})$.

Finally, one needs to check that this 3-polygraph is confluent. Here, this is equivalent to computing all of its critical pairs and check that each one is confluent. Once again, the structural complexity of polygraphs generates problems unknown with other kinds of rewriting theories. For example, a finite 3-polygraph can produce an infinite number of critical pairs; this is the case here. However, among these critical pairs, some have properties that allow us to finally have only a finite number of computations to do. Critical pairs of 3-polygraphs need to be further studied and classified according to properties of this kind; this will be addressed in subsequent work.

The present case is discussed in section 6 and fully studied in [Guiraud 2004].

From theorem 4.6, one concludes the existence of a map Φ from $\mathbb{T}\Sigma$ to $\langle\Sigma^c\rangle$. Indeed, if f is an arrow in the cartesian category $\mathbb{T}\Sigma$, then $\Phi(f)$ will be the $R_{\Delta\Sigma}$-normal form of any representant in $\langle\Sigma^c\rangle$ of the arrow $F(f)$ in the product category $\langle\Sigma^c\rangle/E_{\Delta\Sigma}$. This map Φ, which could not be proved to exist until theorem 4.6, allows the formal definition of the translation of terms into circuits.

Definition 4.8. For every term u in $T\Sigma$ and for every integer $n \geq \sharp u$, the term u can be seen as an arrow u_n in $\mathbb{T}\Sigma(n,1)$. One denotes by $\Phi^n(u)$ the arrow $\Phi(u_n)$ of $\langle \Sigma^c \rangle$ and by $\Phi(u)$ the particular case $\Phi^{\sharp u}(u)$. If $\alpha = (u,v)$ is a rewrite rule on $T\Sigma$, the notation $\Phi(\alpha)$ is used for the rewrite rule $(\Phi(u), \Phi^{\sharp u}(v))$ on $\langle \Sigma^c \rangle$.

As an immediate consequence of the definition, one gets:

Lemma 4.9. *For any algebraic signature Σ, any term u in $T\Sigma$ and any integer $n \geq \sharp u$, the arrow $\Phi^n(u)$ is a normal form for the resource management rules $R_{\Delta\Sigma}$.*

The rest of this section is devoted to the comparison of a term rewriting system (Σ, R) with the 3-polygraph (Σ^c, R^c), where R^c is the union of the family $R_{\Delta\Sigma}$ of resource management rules and of the family $\Phi(R)$ made of the translations by Φ of the rules R.

Remark 4.10. Before stating the result, let us qualify by *uniformized* a rule (u,v) on $T\Sigma$ such that $u = f(y_1, \dots, y_k)$ with f an arrow in $\langle \Sigma \rangle$ and $(y_1, \dots, y_k)$ a family of variables with the following property: y_1 is x_1; then, for each i in $\{1, \dots, k-1\}$, the variable y_{i+1} is either in $\{y_1, \dots, y_i\}$, or y_{i+1} is x_{p+1} if $\{y_1, \dots, y_i\} = \{x_1, \dots, x_p\}$.

Note that any rule on $T\Sigma$ can be replaced by a uniquely defined uniformized rule that generates the same reduction relation. Furthermore, if a left-linear rule is replaced by its uniformized rule, this one is also left-linear.

Hence, for what follows, (left-linear) term rewriting systems can always be considered uniformized: if they are not, they are replaced by their uniformized equivalent version, with no consequence on the results.

This choice simplifies the translations: a rule (u,v) that is both left-linear and uniformized satisfies $u = f(x_1, \dots, x_{\sharp u})$, with f an arrow in $\langle \Sigma \rangle$, uniquely defined; hence, the translation by Φ of such a u is f and thus is an arrow of $\langle \Sigma \rangle$.

Proposition 4.11. *If (Σ, R) is a term rewriting system, then:*

1. *If the term rewriting system (Σ, R) terminates, so does the 3-polygraph (Σ^c, R^c).*
2. *The translation Φ preserves the reduction steps generated by any left-linear rule α, that is: for any pair (u,v) of terms such that $u \to_\alpha v$ and any integer $n \geq \sharp u$, there exists an arrow f in $\langle \Sigma^c \rangle$ such that*
$$\Phi^n(u) \to_{\Phi(\alpha)} f \twoheadrightarrow_{R_{\Delta\Sigma}} \Phi^n(v).$$

Proof. Point 1 uses the technique to be introduced in section 5. Its proof is thus postponed until section 6. Point 2 requires lengthy and cumbersome though intuitively simple computations that can be found in [Guiraud 2004]. ♢

Theorem 4.12. *A left-linear term rewriting system (Σ, R) terminates (resp. is confluent) if and only if its associated 3-polygraph (Σ^c, R^c) terminates (resp. is confluent).*

Proof. Let us assume that the 3-polygraph (Σ^c, R^c) terminates while the term rewriting system (Σ, R) does not. Consequently, there exists some sequence $(u_n)_{n\in\mathbb{N}}$ of terms in $T\Sigma$ such that $u_n \to_R u_{n+1}$ for every n. From 4.11, since every rule in R is left-linear, one concludes that, for every $k \geq \sharp u_0$:
$$\Phi^k(u_0) \twoheadrightarrow^+_{R^c} \Phi^k(u_1) \twoheadrightarrow^+_{R^c} \cdots \twoheadrightarrow^+_{R^c} \Phi^k(u_n) \twoheadrightarrow^+_{R^c} \Phi^k(u_{n+1}) \twoheadrightarrow^+_{R^c} \cdots$$

where the notation $\twoheadrightarrow^+_{R^c}$ stands for a *non-empty* R^c-reduction path. Such an infinite reduction path existence is denied by the termination of the 3-polygraph (Σ^c, R^c), thus giving this property for (Σ, R). The converse, which is true even if the term rewriting system is not left-linear, is still postponed to section 6.

Now, let us assume that the term rewriting system (Σ, R) is confluent. Let us consider a branching (f, g, h) of (Σ^c, R^c): the arrows f, g and h have the same finite number of inputs, say m, and the same finite number of outputs, say n, and satisfy $f \twoheadrightarrow_{R^c} g$ and $f \twoheadrightarrow_{R^c} h$. Let us denote by π the canonical projection of $\langle \Sigma^c \rangle$ onto $\mathbb{T}\Sigma$. Then, π sends each of f, g and h on families $(f_1, \dots, f_n)$, $(g_1, \dots, g_n)$ and $(h_1, \dots, h_n)$ of terms such that each one has variables in $\{x_1, \dots, x_m\}$. Moreover, for each i, one gets that the triple (f_i, g_i, h_i) is a branching of (Σ, R). From confluence of this rewriting system, one concludes the existence of a arrow k_i that closes this branching. Let us define k as the translation, by Φ^m, in $\langle \Sigma^c \rangle(m, n)$, of the family $(k_1, \dots, k_n)$ of terms. Since (Σ, R) is left-linear, proposition 4.11 ensures that this arrow k closes the branching (f, g, h).

Conversely, let us assume that the 3-polygraph (Σ^c, R^c) is confluent. Let us consider a branching (u, v, w) in (Σ, R); since this rewriting system is left-linear, this branching translates to a branching $(\Phi^n(u), \Phi^n(v), \Phi^n(w))$ in (Σ^c, R^c) for any $n \geq \sharp u$. Since the 3-polygraph is confluent, there exists some arrow f in $\langle \Sigma^c \rangle(n, 1)$ closing this branching. The projection $\pi(f)$ is an arrow in $\mathbb{T}\Sigma(n, 1)$ and thus corresponds to a term that closes the initial branching (u, v, w). ♢

Before considering what this result allows (or rather does not allow) us to conclude about our term rewriting system (Σ, R_2) presenting the theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces, there remains some termination results to prove in the elapsed section. However, the intrinsic complexity of the polygraph structure prevents the use of classical techniques; rather, the incoming section presents an adaptation to the particular case of 3-polygraph we consider of classical interpretation techniques used to craft termination orders for terms.

5 Termination orders for 3-polygraphs

In rewriting, one of the most used technique to prove termination is the following one: build a *reduction order*, which is a terminating strict order that is compatible with the term structure; then prove that this order contains the rules. Hence any reduction path in the corresponding rewriting system yields a strictly decreasing family for the reduction order: the fact that such families cannot be infinite ensures that there cannot exist any infinite reduction path or, equivalently, that the rewriting system terminates.

In term rewriting, one easy way to build reduction orders is by means of an *interpretation*. The simpliest ones are: each term u such that $\sharp u = n$ is sent to a function u_* from $\mathbb{N}^n$ to $\mathbb{N}$ (or any set equipped with a terminating strict order). Then, one says that $u > v$ if each n-uple of integers is sent to a strictly greater integer by u_* than by v_*. One easy way to compute u_* for each term u is to fix φ_* for each operator φ in the considered signature and to extend these values functorially. If one can prove that each φ_* is a strictly monotone map and that $f > g$ for each rule $f \to g$, then $u > v$ whenever there is a reduction from u to v. Since the order on $\mathbb{N}$ is terminating, so is the order on functions: hence, the considered term rewriting system terminates.

However, in the case of 3-polygraphs, this classical interpretation technique does not yield reduction orders in general. Indeed, it is not always possible to send each operator φ of the signature onto a strictly monotone map: for example, the erasure operator ε will be sent to an function from $\mathbb{N}$ to $\mathbb{N}^0$, that is to a single-element set: this function is unique and monotone, but not strictly. Consequently: even if a rule

$f \to g$ satisfies $f_* > g_*$, then $(\varepsilon^n \circ f)_* = (\varepsilon^n \circ g)_*$, with n the number of outputs of both f and g.

One could also consider contravariant interpretations: hence, ε would be sent to a constant natural number ε^*. But, in the most interesting 3-polygraphs, such as the ones we are concerned with, there is a constant operator η which cannot be contravariantly sent to a strictly monotone map. The interpretation technique must be adaptated to the polygraph structure in order to yield termination orders.

Here we are in front of a choice between two possible directions: the first one consists in interpreting arrows into functions between objects equipped with a monoidal product, rather than a cartesian one, such as vector spaces. But, when examined, this has led to horrendous computations that did not produce any reduction order. Nonetheless, this trail is not to be forgotten and shall be reexamined when there is a computational tool, adaptated to polygraphs.

The other path consists in using classical interpretations, both covariant and contravariant, as tools to build a third interpretation: this one will give the desired reduction orders. Let us present images that describe the intuition beneath the formalism. Each arrow in the considered product category is seen as an electrical circuit whose elementary components are the operators it is built from, such as suggested by the diagrammatic representation used. Then, a heat production value is associated to each circuit: each of its inputs and outputs receives a current with a fixed intensity; hence there are two types of currents: some are descending (they come from the inputs and propagate downwards to the outputs) and some are ascending (they propagate upwards, from the outputs to the inputs).

The heat produced by a fixed circuit is calculated this way: an operator is arbitrarily chosen. Then, currents are propagated through the other operators to the chosen one. This requires that choices have been made for each operator: for each one, one must be able to compute the intensities of descending currents transmitted when he knows the intensities of incoming descending current, and similarly with ascending currents. When one knows the intensities of each current coming into the chosen operator, one computes the heat it produces, according to values fixed in advance. Then, one repeats the same procedure for each operator, and sums the results to get the heat produced by the considered circuit, for the chosen current intensities.

Two circuits with the same number of inputs and the same number of outputs are compared this way: if, for each family of (ascending and descending) current intensities, one produces more heat than the other one, then the first one is said to be greater. The goal of this section is twofolds: firstly, to formalize the objects required to compute such an order; secondly, to obtain sufficient conditions for this order to be a reduction order.

Let us describe the required materials. The first one is the object where the interpretations take their values: this will be a product category equipped with a strict order. In order to build it, one considers (non-empty) ordered sets X and Y to express the current intensities, one for descending currents, one for ascending currents (for one of the applications to be described, two different sets of values are needed). Then, a commutative monoid M will contain the possible values of heats; moreover, it is supposed to be equipped with an order such that the addition is strictly monotone in both variables.

From the data X, Y and M, one builds a somewhat weird product category $\mathcal{O}(X, Y, M)$ this way: an arrow from m to n in $\mathcal{O}(X, Y, M)$ is a triple $f = (f_*, f^*, [f])$ consisting of three *monotone* functions

$$f_* : X^m \to X^n, \quad f^* : Y^n \to Y^m, \quad [f] : X^m \times Y^n \to M.$$

The identity of n is the triple $n = (X^n, Y^n, 0)$ made from the identities of X^n and Y^n and the constant

zero-function from $X^n \times Y^n$ to M. Two arrows $f : m \to n$ and $g : n \to p$ compose this way: $(g \circ f)_*$ and $(g \circ f)^*$ are respectively the composites $g_* \circ f_*$ and $f^* \circ g^*$; for elements $\vec{x}$ in X^m and $\vec{y}$ in Y^p, the function $[g \circ f]$ is given by:

$$[g \circ f](\vec{x}, \vec{y}) = [f](\vec{x}, g^*(\vec{y})) + [g](f_*(\vec{x}), \vec{y}).$$

If $f : m \to n$ and $g : p \to q$ are two arrows in $\mathcal{O}(X, Y, M)$, then their product is given by: $(f \otimes g)_*$ and $(f \otimes g)^*$ are respectively $f_* \otimes g_*$ and $f^* \otimes g^*$; if $\vec{x}$, $\vec{y}$, $\vec{x}'$ and $\vec{y}\,'$ are respectively elements of X^m, Y^n, X^p and Y^q, then $[f \otimes g]$ is given by:

$$[f \otimes g](\vec{x}, \vec{x}', \vec{y}, \vec{y}\,') = [f](\vec{x}, \vec{y}) + [g](\vec{x}', \vec{y}\,').$$

Then one checks that these operations return monotone functions and that they satisfy the required equations, in order to get:

Lemma 5.1. *The aforedefined object $\mathcal{O}(X, Y, M)$ is a product category.*

On top of this product category structure, a strict order relation $\succ$ is defined on parallel arrows of $\mathcal{O}(X, Y, M)$. If f and g are two arrows from m to n, then $f \succ g$ if, for any $\vec{x}$ in X^m and $\vec{y}$ in Y^n, the following three inequalities hold:

$$f_*(\vec{x}) \geq g_*(\vec{x}), \quad f^*(\vec{y}) \geq g^*(\vec{y}), \quad [f](\vec{x}, \vec{y}) > [g](\vec{x}, \vec{y}).$$

Now, let us consider a signature Σ. Let us asume that each operator $\varphi : m \to n$ in Σ is associated with an arrow $(\varphi_*, \varphi^*, [\varphi]) : m \to n$ in $\mathcal{O}(X, Y, M)$: this is the interpretation. For any φ, the monotone functions φ_*, φ^* and $[\varphi]$ respectively express how the operator transmits descending and ascending currents and how much heat it produces, according to the current intensities it receives.

Since $\langle\Sigma\rangle$ is the free product category generated by the signature Σ, the map sending each φ in Σ to the triple $(\varphi_*, \varphi^*, [\varphi])$ uniquely extends to a product category functor F from $\langle\Sigma\rangle$ to $\mathcal{O}(X, Y, M)$. This means that one can compute f_*, f^* and $[f]$ for any circuit f in $\langle\Sigma\rangle$, from the values φ_*, φ^* and $[\varphi]$ given for each operator φ in Σ and using the formulas for composition and product in $\mathcal{O}(X, Y, M)$.

The last step consists in using F to get the order $\succ$ back from $\mathcal{O}(X, Y, M)$ on $\langle\Sigma\rangle$: for any two parallel arrows f and g in $\langle\Sigma\rangle$, then $f \succ g$ is $F(f) \succ F(g)$.

Theorem 5.2. *With the aforegiven notations and if the strict part of the order on M is terminating, then the strict order $\succ$ constructed on $\langle\Sigma\rangle$ is a reduction order.*

Proof. One must check that the binary relation $\succ$ built on $\langle\Sigma\rangle$ is antireflexive, transitive, terminating and compatible with the product category structure. Let us assume that f is an arrow in $\langle\Sigma\rangle(m, n)$ such that $f \succ f$; let us fix any elements $\vec{x}$ and $\vec{y}$ respectively in the non-empty sets X^m and Y^n; then, by definition of $\succ$, one gets the following strict inequality in M:

$$[F(f)](\vec{x}, \vec{y}) > [F(f)](\vec{x}, \vec{y}).$$

However, this inequality cannot hold in M since $>$ is the strict part of an order relation. The termination is proved by a similar argument: any infinite and strictly decreasing sequence in $\langle\Sigma\rangle$ yields, through the non-emptyness of X and Y, at least one infinite strictly decreasing sequence in M, which existence is

denied by the assumed termination of the strict part of its order. The transitivity comes from the ones of the orders on X, Y and M. Finally, compatibility with the product category structure is checked through computations which use the monotone quality of each f_*, f^* and $[f]$ in $\mathcal{O}(X, Y, M)$, together with the facts that M is a commutative monoid and F is an product category functor. ♢

For concrete applications, presented in the next two sections, the following corollary will be used instead of theorem 5.2:

Corollary 5.3. *Let us consider a* 3*-polygraph* (Σ, R)*. Let us assume that there exist:*

1. *Two non-empty ordered sets* X *and* Y*.*
2. *A commutative monoid* M *equipped with an order such that its strict part is* terminating *and such that the sum is strictly monotone in both variables.*
3. *For each operator* φ *in* $\Sigma(m, n)$*, three* monotone *functions:*

$$\varphi_* : X^m \to X^n, \quad \varphi^* : Y^n \to Y^m, \quad [\varphi] : X^m \times Y^n \to M.$$

If the strict order $\succ$ *on arrows of* $\langle\Sigma\rangle$ *built from these data, in the aforegiven manner, satisfies* $f \succ g$ *for every rule* $f \to g$ *in* R*, then the* 3*-polygraph* (Σ, R) *terminates.*

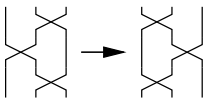
6 Application 1: explicit resource management polygraphs

This section is devoted to the remaining unproved results from section 4. Let us fix a term rewriting system (Σ, R) for the whole section.

6.1 Convergence of the 3-polygraph of explicit resource management

The first result to prove is theorem 4.6: the 3-polygraph $(\Sigma^c, R_{\Delta\Sigma})$ is convergent, where we recall from section 4 that Σ^c is the signature made of the algebraic signature Σ and the resource management signature Δ, while $R_{\Delta\Sigma}$ is the family of resource management rules.

The proof is divided in three steps: the first one consists in proving its termination; then, we recall from [Guiraud 2004] that this 3-polygraph is locally confluent; finally, Newman's lemma is applied to get its convergence. Let us start with termination: we use the technique developped in section 5. However, the considered polygraph is rather complex and needs two applications of the technique. For the rest of this paragraph, let us fix some notations. Let us denote by α the following rule:



We denote by $\mathbb{N}^*$ the set of non-zero natural numbers with its natural order relation. The commutative monoid freely generated by $\mathbb{N}^*$ is denoted by $[\mathbb{N}^*]$ and is considered equipped by the *multiset order* generated by the usual order relation on natural numbers. The elements of $[\mathbb{N}^*]$ are all the finite formal sums of non-zero natural numbers; a natural number n, seen as a generator of $[\mathbb{N}^*]$, is denoted by $\underline{n}$.

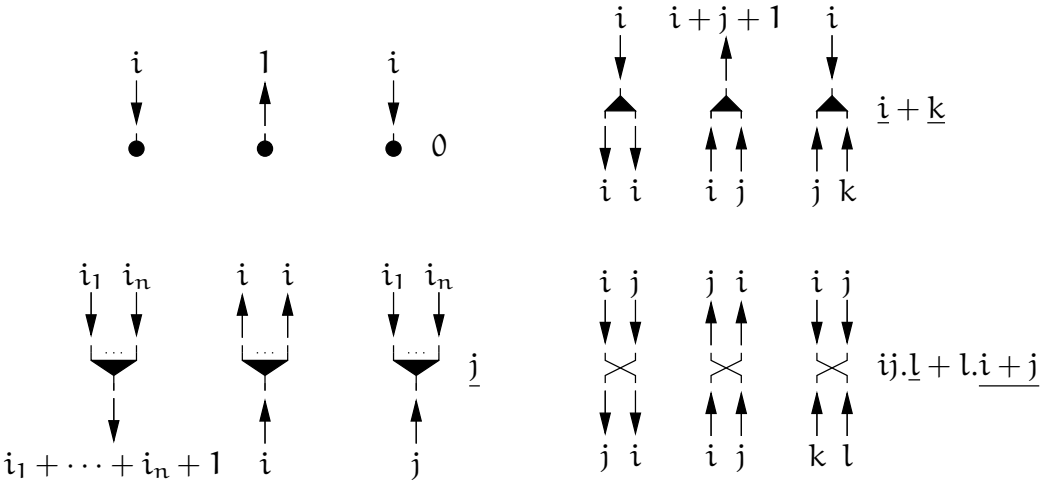
The multiset order is defined in two steps: for the first one, one says that any sum $a = \sum_i k_i.\underline{n_i}$ satisfies the inequality $\underline{n} > a$ if $n > n_i$ for each i; then, the multiset order is taken as the reflexive and structure-compatible closure of this relation.

This implies that the addition is strictly monotone in both variables; furthermore, since the strict order $>$ on $\mathbb{N}^*$ terminates, so does the strict part of the multiset order. Here is an example of some strict inequalities that hold in $[\mathbb{N}^*]$:

$$0 < 127.\underline{1} < \underline{2} < 4.\underline{1} + 2.\underline{3} < \underline{4}\,.$$

Lemma 6.1.1. *The 3-polygraph* $(\Sigma^c, R_{\Delta\Sigma})$ *terminates if and only if the 3-polygraph* $(\Sigma^c, \{\alpha\})$ *terminates.*

Proof. Let us consider the product category $\mathcal{O}(\mathbb{N}^*, \mathbb{N}^*, [\mathbb{N}^*])$ together with the termination order $\succ$ as defined in section 5. Let us denote by F the product category functor from $\langle \Sigma^c \rangle$ into $\mathcal{O}(\mathbb{N}^*, \mathbb{N}^*, [\mathbb{N}^*])$ defined by the following values on the operators of Σ^c:



Three diagrams are given for each operator φ: two represent the functions φ_* and φ^* (how φ transmits the current intensities) and one represents $[\varphi]$ (the heat φ produces). Now, it is checked that, for every rule $f \to g$ in $R_{\Delta\Sigma}$, the inequality $F(f) \succ F(g)$ holds, except for the rule $\alpha : s\alpha \to t\alpha$, for which $F(s\alpha) = F(t\alpha)$. Let us check the (in)equalities for three sample rules. The complete computations are in [Guiraud 2004]. Let us start with the coassociativity rule for δ:

One checks that the first two non-strict inequalities are satisfied:

$$\begin{cases} ((1 \otimes \delta) \circ \delta)_*(i) = (i,i,i) = ((\delta \otimes 1) \circ \delta)_*(i) \\ ((1 \otimes \delta) \circ \delta)^*(i,j,k) = i + j + k + 2 = ((\delta \otimes 1) \circ \delta)^*(i,j,k). \end{cases}$$

Moreover: $\begin{cases} [(1 \otimes \delta) \circ \delta](i,j,k,l) = 2.\underline{i} + \underline{l} + \underline{k + l + 2} \\ [(\delta \otimes 1) \circ \delta](i,j,k,l) = 2.\underline{i} + \underline{l} + \underline{k}. \end{cases}$

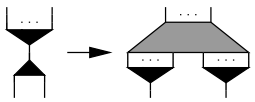
Since $l+2>0$, one gets $\underline{k+l+2}>\underline{k}$ and the required strict inequality. Then, consider the rule α for which the chosen values do not work. One gets the two following equalities:

$$\begin{cases}((1\otimes\tau)\circ(\tau\otimes 1)\circ(1\otimes\tau))_*(i,j,k)=(k,j,i)=((\tau\otimes 1)\circ(1\otimes\tau)\circ(\tau\otimes 1))_*(i,j,k)\\ ((1\otimes\tau)\circ(\tau\otimes 1)\circ(1\otimes\tau))^*(i,j,k)=(k,j,i)=((\tau\otimes 1)\circ(1\otimes\tau)\circ(\tau\otimes 1))^*(i,j,k).\end{cases}$$

And also this equality:

$$\begin{aligned}&[(1\otimes\tau)\circ(\tau\otimes 1)\circ(1\otimes\tau)](i,j,k,l,m,n)\\ =\;& jk.\underline{m}+m.\underline{j+k}+i.(j+k).\underline{n}+n.(\underline{i+j}+\underline{j+k})\\ =\;& [(\tau\otimes 1)\circ(1\otimes\tau)\circ(\tau\otimes 1)](i,j,k,l,m,n).\end{aligned}$$

To finish with our examples, let us consider the most complicated rule of this presentation, namely the local duplication rule:



This is this rule that motivates the use of the rather complicated product category $\mathcal{O}(\mathbb{N}^*,\mathbb{N}^*,[\mathbb{N}^*])$ to interpret $\langle\Sigma^c\rangle$. In order to make the computations for this rule, one must start by proving the following equations, which is done by iteration on the integer n:

$$\begin{cases}(\delta_n)_*(i_1,\dots,i_n)=(i_1,\dots,i_n,i_1,\dots,i_n)\\ \delta_n^*(i_1,\dots,i_n,j_1,\dots,j_n)=(i_1+j_1+1,\dots,i_n+j_n+1)\\ [\delta_n](i_1,\dots,i_n,j_1,\dots,j_n,k_1,\dots,k_n)\\ \quad=\sum_{1\le u\le n}(\underline{i_u}+\underline{k_u})+\sum_{1\le u<v\le n}(i_ui_v.\underline{k_u}+k_u.\underline{i_u+i_v}).\end{cases}$$

Then one gets these two equalities:

$$\begin{cases}(\delta\circ\varphi)_*(i_1,\dots,i_n)=(i_1+\cdots+i_n+1,i_1+\cdots+i_n+1)=((\varphi\otimes\varphi)\circ\delta_n)_*\\ (\delta\circ\varphi)^*(i,j)=(i+j+1,\dots,i+j+1)=((\varphi\otimes\varphi)\circ\delta_n)^*.\end{cases}$$

For the strict inequality to be checked:

$$\begin{cases}[\delta\circ\varphi](i_1,\dots,i_n,j,k)=\underline{j+k+1}+\underline{i_1+\cdots+i_n+1}+\underline{k}\\ [(\varphi\otimes\varphi)\circ\delta_n](i_1,\dots,i_n,j,k)\\ \quad=\underline{j}+(n+1+\sum_{1\le u<v\le n}i_ui_v).\underline{k}+\sum_{1\le u<v\le n}\underline{i_u}+k.\sum_{1\le u<v\le n}\underline{i_u+i_v}.\end{cases}$$

The multiset order properties allow the conclusion: the left member of this rule is strictly greater than its right member. Indeed, it is a consequence from the following strict inequalities that hold in $[\mathbb{N}^*]$:

$$\begin{cases}\underline{j+k+1}>\underline{j}\\ \underline{j+k+1}>\underline{k}\\ \underline{i_1+\cdots+i_n+1}>\underline{i_u} & \text{for every } u\\ \underline{i_1+\cdots+i_n+1}>\underline{i_u+i_v} & \text{for every } u \text{ and } v.\end{cases}$$

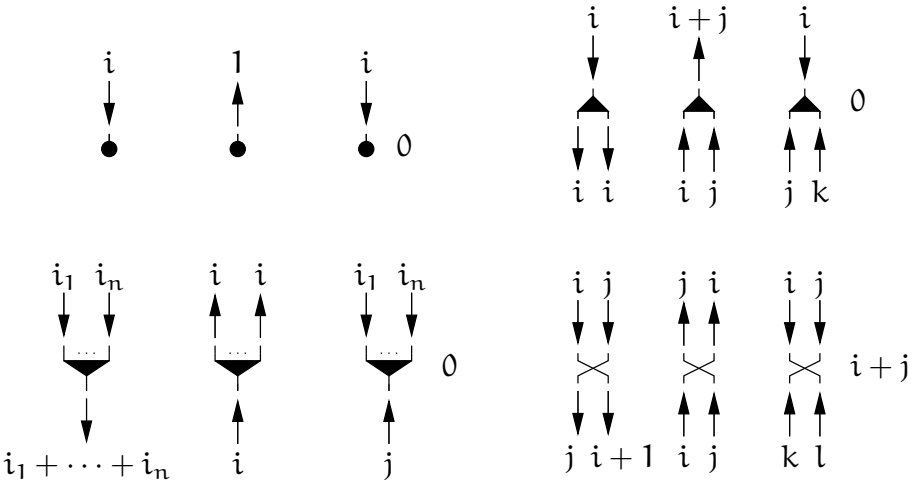
The computations for the other rules are handled similarly, albeit more easily. Now, let us check the equivalence between termination of the 3-polygraphs $(\Sigma^c, R_{\Delta\Sigma})$ and $(\Sigma^c, \{\alpha\})$. Since α is a rule of $R_{\Delta\Sigma}$, one concludes immediately that the termination of $(\Sigma^c, R_{\Delta\Sigma})$ implies the termination of $(\Sigma^c, \{\alpha\})$: any infinite reduction path generated by the latter would also be an infinite reduction path in the former.

Conversely, let us assume that $(\Sigma^c, \{\alpha\})$ terminates and that there exists an infinite reduction path $(f_n)_{n\in\mathbb{N}}$ in $(\Sigma^c, R_{\Delta\Sigma})$. This path yields an infinite decreasing sequence $(F(f_n))_n$ in $\mathcal{O}(\mathbb{N}^*, \mathbb{N}^*, [\mathbb{N}^*])$, equipped with the order $\succ$. Since this order terminates, the sequence is stationary, which means that there exists some natural number n_0 such that $F(f_n) = F(f_{n+1})$ whenever $n \geq n_0$. However, as proved earlier, one can have both $f \to_{R_{\Delta\Sigma}} g$ and $F(f) = F(g)$ only if $f \to_\alpha g$. This implies that the sequence $(f_n)_{n\geq n_0}$ is an infinite reduction path in $(\Sigma^c, \{\alpha\})$. However, the existence of such an infinite reduction path is prevented by the termination of $(\Sigma^c, \{\alpha\})$. ♢

Now, there remains to prove that:

Lemma 6.1.2. *The 3-polygraph $(\Sigma^c, \{\alpha\})$ terminates.*

Proof. This is done using the technique from section 5. The product category considered for the interpretations is $\mathcal{O}(\mathbb{N}, \mathbb{N}, \mathbb{N})$, where $\mathbb{N}$ is the set (or commutative monoid) of natural numbers, equipped with its natural order. We denote by G the product category functor from $\langle\Sigma^c\rangle$ to $\mathcal{O}(\mathbb{N}, \mathbb{N}, \mathbb{N})$ defined by the following values on the operators of Σ^c:



We must check that $\alpha : s\alpha \to t\alpha$ satisfies $F(s\alpha) \succ F(t\alpha)$. The computations give, on one hand, the two equalities:

$$\begin{cases} ((1\otimes\tau)\circ(\tau\otimes 1)\circ(1\otimes\tau))_*(i,j,k) = (k, j+1, i+2) = ((\tau\otimes 1)\circ(1\otimes\tau)\circ(\tau\otimes 1))_*(i,j,k) \\ ((1\otimes\tau)\circ(\tau\otimes 1)\circ(1\otimes\tau))^*(i,j,k) = (k,j,i) = ((\tau\otimes 1)\circ(1\otimes\tau)\circ(\tau\otimes 1))^*(i,j,k). \end{cases}$$

On the other hand, one gets:

$$\begin{cases} [(1\otimes\tau)\circ(\tau\otimes 1)\circ(1\otimes\tau)](i,j,k,l,m,n) = 2i+2j+2k+2 \\ [(\tau\otimes 1)\circ(1\otimes\tau)\circ(\tau\otimes 1)](i,j,k,l,m,n) = 2i+2j+2k+1. \end{cases}$$

By corollary 5.3, this gives the result. ♢

Thus, one gets, as a corollary of lemmas 6.1.1 and 6.1.2:

Proposition 6.1.3. *The 3-polygraph* $(\Sigma^c, \mathrm{R}_{\Delta\Sigma})$ *terminates.*

We recall the following result from [Guiraud 2004, proposition 5.31]:

Proposition 6.1.4. *The 3-polygraph* $(\Sigma^c, \mathrm{R}_{\Delta\Sigma})$ *is locally confluent.*

Finally, Newman's lemma [Baader Nipkow 1998] is applied to get theorem 4.6.

6.2 Termination of 3-polygraph built from a terminating rewriting system

This paragraph contains the proof of theorem 4.11, point 1: if a term rewriting system (Σ, R) terminates, then so does its associated 3-polygraph (Σ^c, R^c). The proof once again uses a termination order obtained with theorem 5.2. However, integer values cannot be used here, since rules in R are unknown. To handle this issue, the following classical result - see [Baader Nipkow 1998] - is used:

Theorem 6.2.1. *A term rewriting system terminates if and only if there exists some mapping* $|\cdot|$ *from the set of terms* $\mathrm{T}\Sigma$ *to* $\mathbb{N}$ *such that* $|u| > |v|$ *whenever* u *is a term that reduces into another term* v*. Moreover, in that case, the mapping* $|\cdot|$ *can be chosen such that* $|u| \geq |u'|$ *whenever* u' *is a subterm of* u*; the mapping can also be chosen so that it takes its values in any countable set.*

Proof. If (Σ, R) terminates, one can choose the mapping $|\cdot|$ to send each term u onto the length of the longest reduction path starting from u; this mapping satisfies $|u| \geq |u'|$ if u' is a subterm of u, since every reduction path from u' yields a reduction path of the same length from u. Conversely, if such a mapping exists, an infinite reduction path $(u_n)_{n\in\mathbb{N}}$ in (Σ, R) would generate a strictly decreasing infinite sequence $(|u_n|)_{n\in\mathbb{N}}$ in $\mathbb{N}$, which cannot exist; hence the term rewriting system (Σ, R) terminates. If this is the case, the mapping $|\cdot|$ can be composed with any bijection $\sigma : \mathbb{N} \to \mathrm{E}$, where E is any countable set. ♢

Hence, from our terminating term rewriting system (Σ, R), a mapping $|\cdot| : \mathrm{T}\Sigma \to \mathbb{N}^*$ is assumed to be chosen such that $|u| > |v|$ whenever u reduces in v and $|u| \geq |u'|$ whenever u' is a subterm of u. From this mapping, one defines a binary relation $>$ on $\mathrm{T}\Sigma$ by $u > v$ if, for every term context c, the inequality $|c[u]| > |c[v]|$ holds. From the fact that the usual order $>$ on $\mathbb{N}^*$ is a terminating strict order, this binary relation is proved to satisfy:

Lemma 6.2.2. *The aforedefined binary relation* $>$ *on* $\mathrm{T}\Sigma$ *is a terminating strict order.*

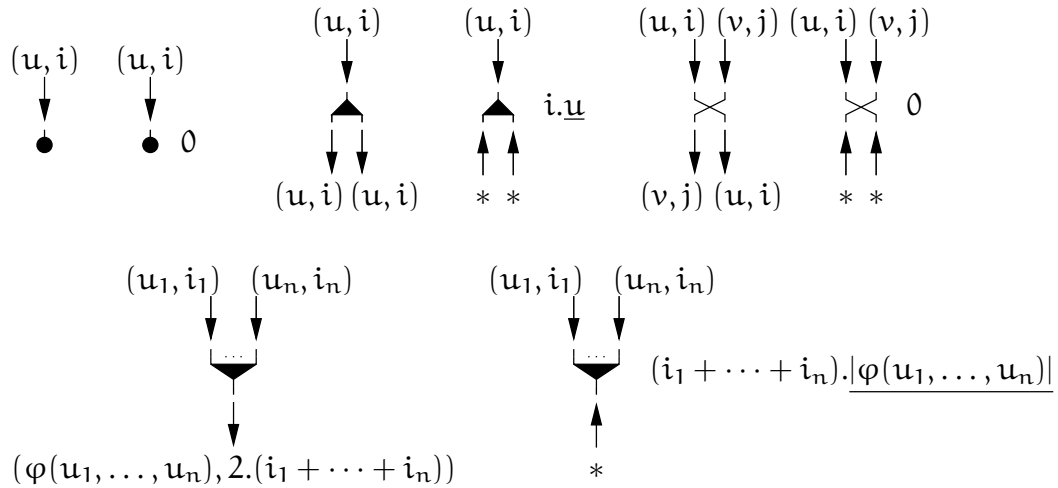
Then, one builds the lexicographical order $\geq$ on $\mathrm{T}\Sigma \times \mathbb{N}^*$: for this order, $(u, i) \geq (v, j)$ if $u > v$ or if $u = v$ and $i \geq j$. This order satisfies:

Lemma 6.2.3. *This relation* $\geq$ *is an order on* $\mathrm{T}\Sigma \times \mathbb{N}^*$*. Moreover, its strict part* $>$ *is a terminating strict order on* $\mathrm{T}\Sigma \times \mathbb{N}^*$*.*

The set $\mathrm{T}\Sigma \times \mathbb{N}^*$, together with the aforedefined order, is taken as the first set used in the interpretation. The second one is a one-element set $\{*\}$ with the only possible order. Finally, the commutative monoid is once again $[\mathbb{N}^*]$ with its already-used multiset order. The product category $\mathcal{O}(\mathrm{T}\Sigma \times \mathbb{N}^*, \{*\}, [\mathbb{N}^*])$ is denoted by $\mathcal{O}$.

Sometimes, the two elements $((u_1, i_1), \dots, (u_n, i_n))$ of $(T\Sigma \times \mathbb{N}^*)^n$ and $(u_1, \dots, u_n; i_1, \dots, i_n)$ of $T^n \times (\mathbb{N}^*)^n$ are identified.

The considered product category functor F from $\langle \Sigma^c \rangle$ to $\mathcal{O}$ is given by the following values (only two are given for each operator since the contravariant interpretation is trivial):



There are two steps to check the conditions given in corollary 5.3: the first one consists in ensuring that each given operation is monotone; the second part is about computing if $F(f) > F(g)$ holds for every rule $f \to g$ in R^c.

For the first part, consider, for example, the functions φ_* and $[\varphi]$ for some fixed operator φ in $\Sigma(n, 1)$, $n \geq 1$. Let us consider terms $u_1, \dots, u_n, v_1, \dots, v_n$ and non-zero integers $i_1, \dots, i_n, j_1, \dots, j_n$. Let us assume that $(u_k, i_k) \geq (v_k, j_k)$ for every k. In order to prove that φ_* is monotone, one must check that either $\varphi(u_1, \dots, u_n) > \varphi(v_1, \dots, v_n)$ or both are equal and $i_1 + \cdots + i_n \geq j_1 + \cdots + j_n$. Let c be a context. Since, for every k, $u_k \geq v_k$ and $c \circ \varphi(v_1, \dots, v_{k-1}, \Box, u_{k+1}, \dots, u_n)$ is a context, one gets the following inequality:

$$|c[\varphi(v_1, \dots, v_{k-1}, u_k, \dots, u_n)]| \geq |c[\varphi(v_1, \dots, v_k, u_{k+1}, \dots, u_n)]|.$$

Furthermore, if $u_k > v_k$ for some k, then this inequality is strict for the same k; in this case:

$$|c[\varphi(u_1, \dots, u_n)]| > |c[\varphi(v_1, \dots, v_n)]|.$$

Consequently, $\varphi(u_1, \dots, u_n) > \varphi(v_1, \dots, v_n)$. Otherwise, if $u_k = v_k$ and $i_k \geq j_k$ for every k, then:

$$\begin{cases} |c[\varphi(u_1, \dots, u_n)]| = |c[\varphi(v_1, \dots, v_n)]| \\ i_1 + \cdots + i_n \geq j_1 + \cdots + j_n. \end{cases}$$

Thus, in both cases:

$$(\varphi(u_1, \dots, u_n), 2.(i_1 + \cdots + i_n)) \geq (\varphi(v_1, \dots, v_n), 2.(j_1 + \cdots + j_n)).$$

In order to prove that $[\varphi]$ is monotone, let us fix some k in $[n]$. Then, either $u_k > v_k$ or $u_k = v_k$ and $i_k \geq j_k$. In the first case:

$$|\varphi(v_1, \ldots, v_{k-1}, u_k, \ldots, u_n)| > |\varphi(v_1, \ldots, v_k, u_{k+1}, \ldots, u_n)|.$$

Thus, by definition of the multiset order on $[\mathbb{N}^*]$:

$$\begin{aligned} & (j_1 + \cdots + j_{k-1} + i_k + \cdots + i_n).\underline{|\varphi(v_1, \ldots, v_{k-1}, u_k, \ldots, u_n)|} \\ > \quad & (j_1 + \cdots + j_k + i_{k+1} + \cdots + i_n).\underline{|\varphi(v_1, \ldots, v_k, u_{k+1}, \ldots, u_n)|}. \end{aligned}$$

In the second case, where $u_k = v_k$ and $i_k \geq j_k$:

$$\begin{aligned} & (j_1 + \cdots + j_{k-1} + i_k + \cdots + i_n).\underline{|\varphi(v_1, \ldots, v_{k-1}, u_k, \ldots, u_n)|} \\ \geq \quad & (j_1 + \cdots + j_k + i_{k+1} + \cdots + i_n).\underline{|\varphi(v_1, \ldots, v_k, u_{k+1}, \ldots, u_n)|}. \end{aligned}$$

Finally:

$$(i_1 + \cdots + i_n).\underline{|\varphi(u_1, \ldots, u_n)|} \geq (j_1 + \cdots + j_n).\underline{|\varphi(v_1, \ldots, v_n)|}.$$

If γ is a constant in $\Sigma(0,1)$ or for operators in Δ, proofs are direct. Furthermore, for each operator φ in either Σ or Δ, the operation φ^* is the only map from $\{*\}$ to itself, and it is monotone, so that:

Lemma 6.2.4. *For every operator φ in Σ^c, the aforegiven functions φ_*, φ^* and $[\varphi]$ are monotone.*

Then, we must check if $F(f) \succ F(g)$ for every rule $f \to g$ in R^c. Let us recall that this family of rules consists of three subfamilies: R_Δ, R_Σ and $\Phi(R)$. For any rule $f \to g$ in the first family R_Δ, one gets $F(f) = F(g)$, except for left and right counit rules, where $F(f) \succ F(g)$. Computations for rules in R_Σ are more complicated; let us examine, for example, the rule for local duplication and one of the rules for local permutation:

Let us fix some natural number $n \geq 1$ and some φ in $\Sigma(n,1)$; for constants in $\Sigma(0,1)$, computations are direct. By iteration on n, the following equalities are proved:

$$(\delta_n)_*(\vec{u}, \vec{\imath}) = (\vec{u}, \vec{u}; \vec{\imath}, \vec{\imath}) \quad \text{and} \quad [\delta_n](\vec{u}; \vec{\imath}) = i_1.\underline{u_1} + \cdots + i_n.\underline{u_n}.$$

This gives, at first:

$$\begin{aligned} (\delta \circ \varphi)_*(\vec{u}, \vec{\imath}) &= (\varphi(\vec{u}), \varphi(\vec{u}); 2.(i_1 + \cdots + i_n), 2.(i_1 + \cdots + i_n)) \\ &= ((\varphi \otimes \varphi) \circ \delta_n)_*(\vec{u}, \vec{\imath}). \end{aligned}$$

Then: $[\delta \circ \varphi](\vec{u}, \vec{\imath}) = 3.(i_1 + \cdots + i_n).\underline{|\varphi(\vec{u})|}$. To be compared with:

$$[(\varphi \otimes \varphi) \circ \delta_n](\vec{u}, \vec{\imath}) = 2.(i_1 + \cdots + i_n).\underline{|\varphi(\vec{u})|} + i_1.\underline{u_1} + \cdots + i_n.\underline{u_n}.$$

Since u_k is a subterm of $\varphi(\vec{u})$ for every k, and by asumption on $|\cdot|$, the inequality $|\varphi(\vec{u})| \geq |u_k|$ holds. Hence, for every k:

$$i_k.\underline{|\varphi(\vec{u})|} \geq i_k.\underline{u_k}.$$

Finally: $(i_1+\cdots+i_n).\underline{|\varphi(\vec{u})|} \geq i_1.\underline{u_1}+\cdots+i_n.\underline{u_n}$. This gives the inequality $[\delta\circ\varphi] \geq [(\varphi\otimes\varphi)\circ\delta_n]$. Now, let us consider the first rule for local permutation; the first step is to prove, by iteration on n:

$$(\tau_{n,1})_*(\vec{u},v;\vec{\imath},j) = (v,\vec{u};j,\vec{\imath}) \quad \text{and} \quad [\tau_{n,1}](\vec{u},v;\vec{\imath},j) = 0.$$

Then: $\quad (\tau\circ(\varphi\otimes 1))_*(\vec{u},v;\vec{i},j) \;=\; (v,\varphi(\vec{u});j,2.(i_1+\cdots+i_n)) \;=\; ((1\otimes\varphi)\circ\tau_{n,1})_*(\vec{u},v;\vec{i},j).$

And: $\quad [\tau\circ(\varphi\otimes 1)](\vec{u},v;\vec{i},j) \;=\; (i_1+\cdots+i_n).\underline{|\varphi(\vec{u})|} \;=\; [(1\otimes\varphi)\circ\tau_{n,1}](\vec{u},v;\vec{i},j).$

The other rules in R_Σ are similarly handled and give similar results: for every rule $f \to g$ in R_Σ, the inequality $F(f) \succeq F(g)$ holds in $\mathcal{O}$. The final part concerns the family $\Phi(R)$ of rules. Let us assume that $\alpha : f \to g$ is a rule in R; its translation by Φ is the rule $\Phi(\alpha) : \Phi(f) \to \Phi^{\sharp f}(g)$. Let us prove that $F\circ\Phi(f) \succ F\circ\Phi^{\sharp f}(g)$. The first step is to prove, by iteration on the degree of terms in $T\Sigma$, the following lemma:

Lemma 6.2.5. *Let u be a term in $T\Sigma$, n be an integer such that $n \geq \sharp u$, $\vec{v}$ a family of n terms in $T\Sigma$ and $\vec{\imath}$ a family of n non-zero natural numbers. Let us denote by $\sigma_{\vec{v}}$ the substitution defined by $x_k\cdot\sigma_v = v_k$ if $k \leq n$ and x_k otherwise. Then:*

1. *There exists some non-zero integer k such that $(\Phi^n(u))_*(u\cdot\sigma_{\vec{v}},k)$.*
2. *The inequality $[\Phi^n(u)](\vec{v},\vec{\imath}) < \underline{|u\cdot\sigma_{\vec{v}}+1|}$ holds in $[\mathbb{N}^*]$.*
3. *If u is not a variable, then the inequality $[\Phi^n(u)](\vec{v},\vec{\imath}) \geq \underline{|u\cdot\sigma_{\vec{v}}|}$ also holds in $[\mathbb{N}^*]$.*

Point 1 gives, when applied to f and g with $n = \sharp f$, the existence of non-zero natural numbers k and k' such that $\Phi(f)_*(\vec{u},\vec{\imath}) = (f\cdot\sigma_{\vec{u}},k)$ and $\Phi^n(g)_*(\vec{u},\vec{\imath}) = (g\cdot\sigma_{\vec{u}},k')$. Let us consider some context c. By definition of the reduction relation $\to_\alpha$ generated by the rule α, one gets $c[f\cdot\sigma_{\vec{u}}] \to_\alpha c[g\cdot\sigma_{\vec{u}}]$. Consequently, the properties of $|\cdot|$ give $|c[f\cdot\sigma_{\vec{u}}]| > |c[g\cdot\sigma_{\vec{u}}]|$. This holds for any context thus, by definition of $>$ on $T\Sigma$, one gets $f\cdot\sigma_{\vec{u}} > g\cdot\sigma_{\vec{u}}$. Finally, using the definition of $>$ on $T\Sigma\times\mathbb{N}^*$:

$$\Phi_*(f) > \Phi_*(g).$$

Let us prove now that $[\Phi(f)] > [\Phi^n(g)]$. Since α is a term rewrite rule, its source f is a non-variable term. Hence, point 3 of the previous lemma gives the inequality $[\Phi(f)](\vec{u},\vec{\imath}) \geq \underline{|f\cdot\sigma_{\vec{u}}|}$. Moreover, point 2 gives $[\Phi^n(g)](\vec{u},\vec{\imath}) < \underline{|g\cdot\sigma_{\vec{u}}+1|}$. Finally, since the reduction $f\cdot\sigma_{\vec{u}} \to_\alpha g\cdot\sigma_{\vec{u}}$ holds in (Σ,R) and by properties of $|\cdot|$: $|f\cdot\sigma_{\vec{u}}| > |g\cdot\sigma_{\vec{v}}|$. There remains to concatenate these three inequalities to get $[\Phi(f)] > [\Phi^n(g)]$ and, as a consequence $F\circ\Phi(f) \succ F\circ\Phi^n(g)$. The product category functor F from $\langle\Sigma^c\rangle$ to $\mathcal{O}$ gives us $F(f) \succ F(g)$ for every rule $f\to g$ in $\Phi(R)$ and $F(f)\succeq F(g)$ for every rule $f\to g$ in $R_{\Delta\Sigma}$. This yields the following result:

Proposition 6.2.6. *If the term rewriting system (Σ,R) terminates, then termination of the 3-polygraph (Σ^c,R^c) is equivalent to termination of $(\Sigma^c,R_{\Delta\Sigma})$.*

Since we already know that $(\Sigma^c,R_{\Delta\Sigma})$ always terminates, this concludes the proof of theorem 4.12.

7 Application 2: a convergent 3-polygraph for a commutative equational theory

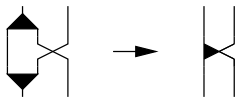
This final section is devoted to give a convergent presentation of the equational theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces, which is, as mentionned before, a commutative equational theory and thus do not have any convergent presentation by a term rewriting system.

In section 1, we have considered three term rewriting systems (Σ, R_0), (Σ, R_1) and (Σ, R_2) that repectively present the equational theories of monoids, of commutative monoids and of $\mathbb{Z}/2\mathbb{Z}$-vector spaces. All three have two operators, a product and a unit, and they have respectively three, four and five rules. Thus, their associated 3-polygraphs have five operators together with twenty-three rules for (Σ^c, R_0^c), twenty-four for (Σ^c, R_1^c) and twenty-five for (Σ^c, R_2^c).

Since (Σ, R_0) is a left-linear convergent term rewriting system, theorem 4.12 ensures, in particular, that (Σ^c, R_0^c) is a convergent presentation of the theory of monoids, *with explicit resource management*. The term rewriting system (Σ, R_1) is left-linear, non-terminating (due to the commutativity rule) and non-confluent (though it could be completed to get a confluent rewriting system), hence theorem 4.12 gives us that (Σ^c, R_1^c) is a non-terminating and non-confluent presentation of the equational theory of commutative monoids, with explicit resource management. Finally, the term rewriting system (Σ, R_2) is a non-left-linear, non-terminating and non-confluent term rewriting system: non-left-linearity denies us any information coming from theorem 4.12 about this presentation.

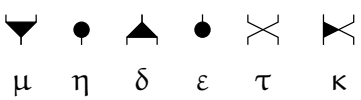
However, there is, in [Lafont 2003], an equivalent 3-polygraph called $L(\mathbb{Z}_2)$. Its signature contains a sixth operator, called κ and pictured this way:

This new operator is said to be *superfluous* since it represents, in a $\mathbb{Z}/2\mathbb{Z}$-vector space, the concrete operation $\kappa(x, y) = (\mu(x, y), x)$ that can be expressed in terms of μ, δ and τ. In the presentation, this relation is enforced by means of the following extra rule:

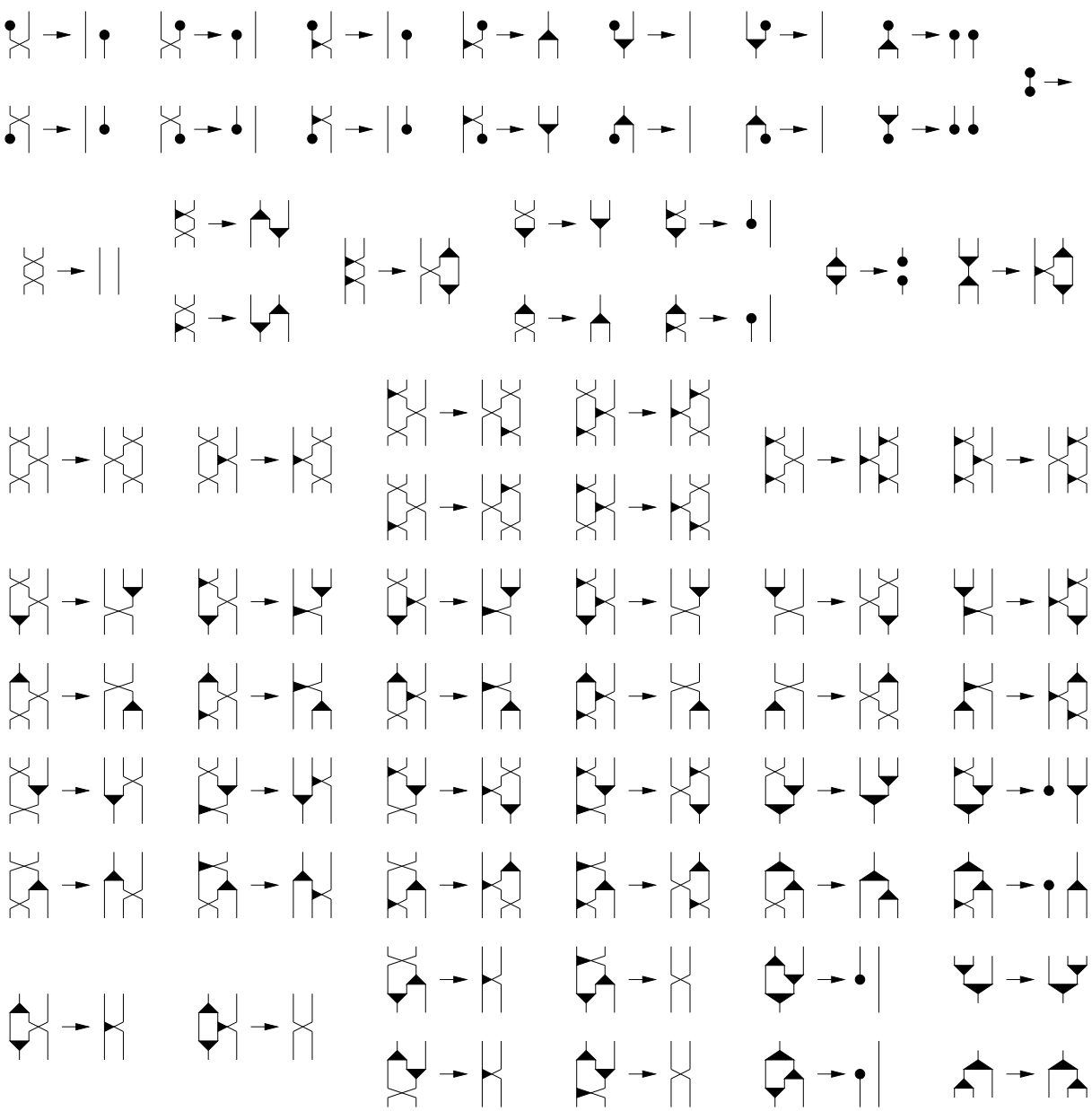


The main objective of these new operator and rule is to make proof of termination easier (if not just possible). Then, one has to add a certain amount of rules in order to complete the presentation, to finally obtain the 3-polygraph $L(\mathbb{Z}_2)$, discovered and baptized in [Lafont 2003].

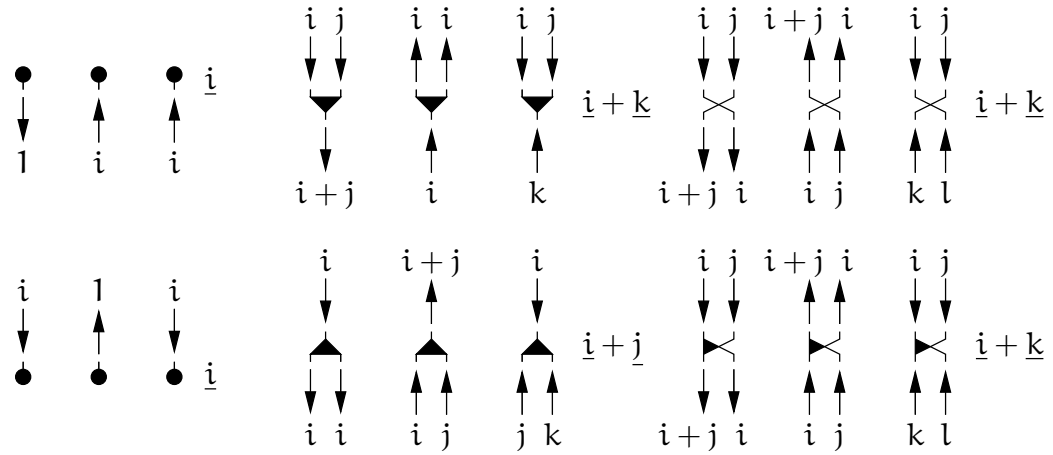
This polygraph has six operators:



And sixty-seven rules:



From [Lafont 2003], we already know that this presentation is confluent but termination was still a conjecture. The technique presented in section 5 now allows us to prove that it is also terminating, hence convergent. The interpretation product category we use is $\mathcal{O}(\mathbb{N}^*, \mathbb{N}^*, [\mathbb{N}^*])$, once again denoted by $\mathcal{O}$. The interpretation functor F is given by the following values on generating operators:



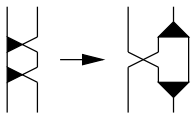
The chosen values simplify the computations greatly. Indeed, normally, there are three inequalities to check for each rule: hence, there should be 201 inequalities to check here. The first reduction comes from the fact that F identifies τ and κ: there are 24 rules that can be dropped since, for each one, there is another rule that is sent to the same image. Thus there remains 43 rules and 129 inequalities to check.

Moreover, the rules of $L(\mathbb{Z}_2)$ have some interesting symmetries that one can exploit: indeed, whenever $f \to g$ is a rule of $L(\mathbb{Z}_2)$, then $f^o \to g^o$ is also a rule of $L(\mathbb{Z}_2)$, where the duality $(\cdot)^o$ is the involution defined by:

$$\mu^o = \delta, \quad \eta^o = \varepsilon, \quad \tau^o = \tau, \quad \kappa^o = \kappa, \quad n^o = n, \quad (g \circ f)^o = f^o \circ g^o, \quad (f \otimes g)^o = f^o \otimes g^o.$$

Another way to define this duality is by its action on diagrams: there, it is the top-down symmetry. Furthermore, the functor F is compatible with this symmetry, in the sense that, for every arrow f, the functor F sends f^o onto $F(f)^o$, where the duality on $\mathcal{O}$ is defined that way: $(f_*, f^*, [f])^o = (f^*, f_*, [f]^o)$, with $[f]^o(\vec{x}, \vec{x}') = [f](\vec{x}', \vec{x})$. Note that this only have a meaning because the two sets X and Y are the same here (both equal to $\mathbb{N}^*$).

Thus, if some rule $f \to g$ in $L(\mathbb{Z}_2)$ satisfies $F(f) > F(g)$, then so does $f^o \to g^o$. As a consequence, this reduces the number of rules to study: 18 of the remaining rules have a distinct dual, hence only 25 rules need to be studied (75 inequalities). Furthermore, when a rule $f \to g$ is self-dual, the inequality $F(f)^* \geq F(g)^*$ holds if and only if $F(f)_* \geq F(g)_*$ holds: 8 of the remaining rules are in that case, which means there still are 67 inequalities from the former 201 to check. Computations do not rise any difficulty. For example, let us study the following (self-dual) rule:



One computes $\begin{cases} (\kappa \circ \kappa)_*(i,j) = (2i+j, i+j) \\ ((1 \otimes \mu) \circ (\tau \otimes 1) \circ (1 \otimes \delta))_*(i,j) = (i+j, i+j). \end{cases}$

Since i and j are non-zero natural numbers, the following inequality holds:

$$(\kappa \circ \kappa)_* > ((1 \otimes \mu) \circ (\tau \otimes 1) \circ (1 \otimes \delta))_*.$$

Then $\begin{cases} [\kappa \circ \kappa](i,j,k,l) = \underline{i} + \underline{i+j} + \underline{k} + \underline{k+l} \\ [(1 \otimes \mu) \circ (\tau \otimes 1) \circ (1 \otimes \delta)](i,j,k,l) = 2\underline{i} + \underline{j} + 2\underline{k} + \underline{l}. \end{cases}$

Since i and j are non-zero natural numbers, the inequalities $i + j > i$ and $i + j > j$ always hold. Thus, by property of the multiset order on $[\mathbb{N}^*]$, the inequality $\underline{i+j} > \underline{i} + \underline{j}$ always holds. Similarly, so does $\underline{k+l} > \underline{k} + \underline{l}$. Finally, the multiset order on $[\mathbb{N}^*]$ is compatible with addition, yielding:

$$[\kappa \circ \kappa] > [(1 \otimes \mu) \circ (\tau \otimes 1) \circ (1 \otimes \delta)].$$

The other rules are studied in a similar way [Guiraud 2004], which leads to the following result, proving that commutative equational theories can admit *polygraphic* convergent presentations:

Theorem 7.1. *The 3-polygraph $\mathrm{L}(\mathbb{Z}_2)$ is a convergent presentation of the equational theory of $\mathbb{Z}/2\mathbb{Z}$-vector spaces, with explicit resource management.*

Comments and future directions

The study of (3-)polygraphs has been started by Albert Burroni and Yves Lafont, as an algebraic model for 3-dimensional calculus on 2-dimensional objects. Foundations were laid in [Lafont 1992], [Burroni 1993] and [Lafont 1995]. In [Lafont 2003], rewriting systems generated by 3-polygraphs were considered and many known equational presentations are studied in order to be completed into convergent rewriting systems (or, at least, rewriting systems with the unique normal form property). Discussions with Albert Burroni, Yves Lafont and Philippe Malbos have been essential in order to achieve the results presented here. Comments from the referee were of great help to make this paper clearer.

There exist many research paths concerning polygraph. The first one is about confluence: as mentionned earlier, there exist theoretical issues with critical pairs of 3-polygraphs; exploration and classification are mandatory in order to achieve some automated completion procedure for these objects. Such a tool (which implementation in Caml has already started) would be very useful since, starting from an equational theory, one could use the constructions described in section 4 in order to obtain a 3-polygraph; then a completion procedure could be applied to correct termination and confluence issues. Suggested by Pierre Lescanne, other usual techniques for building reduction orders in term rewriting could also be examined, in order to see if they could also be adaptated to polygraphs. Among the most useful results to be studied are the ones concerning path orders, see [Baader Nipkow 1998], and dependency pairs, see [Arts Giesl 2000].

A second theme to be explored is the study of higher-dimensional polygraphs. For an example of application, 4-polygraphs provide a categorical framework for proof transformations in the *calculus of structures* [Guglielmi Straßburger 2001]. Such an approach could yield results such as proof decompositions or normal forms, given by a convergent 4-polygraph. At least, it suggests that formulas are 2-dimensional objects, proofs are 3-dimensional and computation on them (such as cut elimination) lives in dimension 4. This point of view is conjectured to yield a new class of objects describing formal proofs, giving a different, categorical and geometrical way to approach proof theory.

Theoretical studies can also be directed at pursuing the synthesis started in [Guiraud 2004] on rewriting systems: one of the main goals is to have a framework where one can compare two rewriting systems, regardless of the algebraic structure of their terms. The reduction space associated to each rewriting system is an algebraico-geometric object (a cubical object in some category of algebras) and one could use

the underlying cubical sets of these objects to compare rewriting systems, geometrically. Notions of (co)fibrations from Quillen model categories - see [Hovey 1999] - theory could be useful for a better understanding of results such as the ones of section 4; since many rewriting systems are special cases of polygraphs, this study will start with the construction of homotopical tools for these objects.

Still another question is the following: is there some n for which there exists a finite n-polygraph yielding a calculus with both explicit substitutions and explicit resource management for the λ-calculus. When $n = 3$, the answer seems to be negative, since theoretical results deny the existence of any non-trivial product category that is both cartesian (for resource management) and sovereign (for substitutions). An equational description of the structure of closed category (such as the one Albert Burroni has given for cartesian categories) should be the first step of this work. Another possibility is to use a 3-dimensional interpretation of proofs, together with the links between λ-terms and proofs.

Finally, 3-polygraphs have the interesting property to modelize computational circuits. Indeed, both classical and quantum algorithms accept representations as circuits which are, albeit not in their usual presentation, genuine operators of a 3-polygraph. Furthermore, equational presentations are known for both kinds of circuits. Questions that can be studied with this point of view concern the existence of convergent 3-polygraphs for classical or quantum circuits, thus leading to canonical representations of programs. One can take a look at [Kitaev Shen Vyalyi 2002] for more information on circuits and [Lafont 2003] for their links with polygraphs.

References

T. ARTS and J. GIESL, *Termination of term rewriting using dependency pairs.* Theoretical Computer Science 236, 133-178, 2000.

F. BAADER and T. NIPKOW, *Term rewriting and all that.* Cambridge University Press, 1998.

A. BURRONI, *Higher-dimensional word problems with applications to equational logic.* Theoretical Computer Science 115(1), 46-62, 1993.

A. GUGLIELMI AND L. STRASSBURGER, *Non-commutativity and MELL in the calculus of structures.* Lecture Notes in Computer Science 2142, 54-68, 2001.

Y. GUIRAUD, *Présentations d'opérades et systèmes de réécriture.* Thèse de doctorat, Montpellier, 2004.

M. HOVEY, *Model categories.* Mathematical Surveys and Monographs 63, 1999.

A. KITAEV, A. SHEN and M. VYALYI, *Classical and quantum computation.* Graduate Studies in Mathematics 47, 2002.

Y. LAFONT, *Penrose diagrams and 2-dimensional rewriting.* London Mathematical Society Lecture Notes Series 177, 191-201, 1992.

—, *Equational reasoning with 2-dimensional diagrams.* Lecture Notes in Computer Science 909, 170-195, 1995.

—, *Towards an algebraic theory of boolean circuits.* Journal of Pure and Applied Algebra 184, 257-310, 2003.

S. MACLANE, *Categorical algebra.* Bulletin of the American Mathematical Society 71, 40-106, 1965.

—, *Categories for the working mathematician.* Springer, 1998.