Polygraphs for termination of left-linear term rewriting systems

Yves Guiraud

POLYGRAPHS FOR TERMINATION OF LEFT-LINEAR TERM REWRITING SYSTEMS

February 2, 2007

Yves GUIRAUD

INRIA Lorraine – LORIA – yves.guiraud@loria.fr

Abstract – *We present a methodology for proving termination of left-linear term rewriting systems (TRSs) by using Albert Burroni's polygraphs, a kind of rewriting systems on algebraic circuits. We translate the considered TRS into a polygraph of minimal size whose termination is proven with a polygraphic interpretation, then we get back the property on the TRS. We recall Yves Lafont's general translation of TRSs into polygraphs and known links between their termination properties. We give several conditions on the original TRS, including being a first-order functional program, that ensure that we can reduce the size of the polygraphic translation. We also prove sufficient conditions on the polygraphic interpretations of a minimal translation to imply termination of the original TRS. Examples are given to compare this method with usual polynomial interpretations.*

1 Introduction

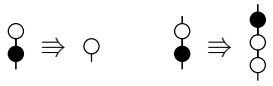
Termination is a fundamental property of rewriting systems, since it ensures that the rule-based computations they define end with a result [2]. Even if this is an undecidable property for a general term rewriting system (TRS), many different techniques have been developped for this purpose. Among them, we are particularly interested into polynomial interpretations [16]: indeed, when one can prove termination of a (first-order) functional program with a polynomial interpretation, there are many cases where one can deduce an implicit complexity bound for the function that the program computes [6, 3]. However, polynomial interpretations on TRSs have limits and here we address two of them.

First, as explained in [4], the interpretation of a term conveys a mixed up information, containing a common bound on the size of the values to be computed and the size of the computation itself. Thus, the idea is that the coefficients one computes are higher than necessary: this increases the time to find a correct polynomial interpretation and decreases the precision of the computational complexity bound. For example, let us consider the functional program $\{\mathsf{D}(0) \to 0,\ \mathsf{D}(\mathsf{s}(x)) \to \mathsf{s}(\mathsf{s}(\mathsf{D}(x)))\}$ computing the "double" function on natural numbers. Then one can prove that the lower polynomial interpretation yielding its termination takes D to $\mathsf{P}(\mathsf{D})(X) = 3X$: this is the sum of the size $2X$ of the computed value and the number X of rewriting steps required to reach it on an input of size X. With the help of dependency pairs [1], one can lower the interpretation of D to $2X$. It is possible that, by application of several other methods, one could prove that D can be interpreted to the polynomial X. But, atop of the complication of the process, we are not sure that theoretical results exist to state that this is an implicit complexity bound for the double function.

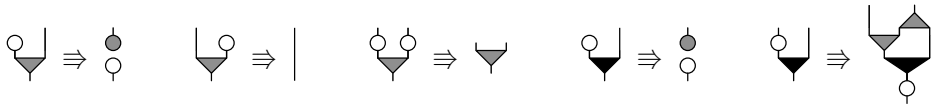
The second limit we consider comes with TRSs that do not admit simplification orders. The functional program $\{\mathsf{M}(0,x) \to 0,\ \mathsf{M}(x,0) \to x,\ \mathsf{M}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y),\ \mathsf{Q}(0,x) \to 0,\ \mathsf{Q}(\mathsf{s}(x),y) \to \mathsf{s}(\mathsf{Q}(\mathsf{M}(x,y),y))\}$, computing division on natural numbers, is an example of this class. Indeed, in the last rule, if one replaces y by $\mathsf{s}(x)$, the left-hand side $\mathsf{l}(x,\mathsf{s}(x))$ can be embedded into the right-hand side $\mathsf{r}(x,\mathsf{s}(x))$: hence, for any simplification order $>$, we have $\mathsf{r}(x,\mathsf{s}(x)) > \mathsf{l}(x,\mathsf{s}(x))$ while proving termination with this strict order would require the reverse strict inequality. Nonetheless, this TRS is

terminating and this can be proved, for example, by using dependency pairs, then semantic labelling and, finally, some simplification order. As above, this means that it is more complicated to prove termination of these systems and that we do not know if we can deduce an implicit complexity bound from this proof.

In order to solve these problems, we propose to use another formalism for expressing rewriting-based computations: *higher-dimensional rewriting* [13], a Turing-complete model [4], based on Albert Burroni's polygraphs [5], which could be described, at first glance, as an algebraic description of term graph rewriting systems [19] and interaction nets [11]. Let us give the polygraphic versions of the two programs we have seen. For the double function, terms are replaced by 2-dimensional algebraic circuits built upon the elementary gates $\mathsf{0}$ for 0, s for s and D for D. The rewriting rules are replaced by the following ones:



Concerning the division on natural numbers, we still use the gates $\mathsf{0}$ and s, plus M for M and Q for Q. We also need two extra gates δ and ε which are central in the present study and will be discussed later. The five rewriting rules are translated as follows:



Polygraphs are higher-dimensional categories which are free in every dimensions. They have been introduced by Albert Burroni in order to provide a unified algebraic structure to many objects from theoretical computer science with an emphasis on rewriting systems. Yves Lafont has started the study of the computational properties of polygraphs [12, 13]. Until now, polygraphs have been proved to unify several objects such as abstract, word and term rewriting systems [8], Petri nets [10] or formal proofs of propositional logics [9].

We think that polygraphs are particularly suited for proving termination of TRSs using adaptated polynomial interpretations and, particularly, for functional programs. Indeed, we can see on the examples that, given a TRS, its associated polygraph is a quite direct translation, so that programs are written as polygraphs in a natural way. Furthermore, we have proved that the termination of the polygraph implies the termination of the rewriting system, provided it is left-linear [8]. Finally, we have developped a tool called polygraphic interpretations [8], giving, on examples, some implicit complexity information which is much finer than the one we get on terms [4]: in the case of the double function, we get the X bound we have discussed. The reason comes from the ability of polygraphic interpretations to differentiate functions from constructors in functional programs, as does the dependency pairs method.

However, the standard translation of a TRS into a polygraph generates a huge object, with many more rewriting rules: indeed, in the polygraphic framework, one has to explicitely handle duplications δ, erasures ε and even permutations σ, which means that one needs to add all the rules to compute these operations. This may have some advantages: for example, in the polygraphic setting, commutativity equations can be directed in a terminating way [13, 8]. But, for the moment, this expliciteness also has practical drawbacks. Indeed, there is only one result linking the termination of a TRS to the one of its standard polygraphic translation and this requires to consider all the rules of the polygraph, including the extra ones. And there can be many of them: for a term rewriting system with $\mathfrak{m}$ sorts, $\mathfrak{n}$ operations and $\mathfrak{p}$

rules, the standard polygraphic translation has $p+2n(m+1)+m(m^2+6m+5)$ rules. Even if we know that these extra rules have nice computational properties, including termination and confluence [8], we could not, until now, set aside some or all of these rules, thus making any practical use of the polygraphic method really hard, at best.

In order to correct this problem, we propose in this study several results that allow us to discard some or all of these extra rules in order to alleviate the computational burden they otherwise generate. We think that the results we prove here make it possible to automatically prove termination of TRSs by polygraphic interpretations and, for some functional programs, to give an implicit complexity bound at the same time. We plan to test such a prover-bounder on the Termination Problems DataBase (*http://www.lri.fr/~marche/tpdb*): this will give essential information on the possible efficiency of our method compared to other ones, together with a general view on which systems it is most suited at and for which systems it can be improved.

This paper is organized into two main sections, apart from this introduction and the conclusion. In section 2, we recall the special case of polygraphs we consider (2.1), then we explain the method of polygraphic interpretation (2.2) and how to translate a TRS into a polygraph (2.3). Section 3 contains the conditions for reducing the size of the polygraph one has to consider to prove termination of a left-linear TRS: theorem 3.1.2 can always be applied to discard a family of extra rules, theorem 3.2.1 is a special case that allows one to consider no extra rules, theorem 3.3.2 is dedicated to the case of first-order functional programs and, finally, theorems 3.4.1 and 3.4.3 give sufficient conditions on a proof by polygraphic interpretation to discard several or all of the extra rules.

For some basic notions of rewriting we do not recall, the reader can consult [2]. The author wishes to thank Frédéric Blanqui, Guillaume Bonfante, Yves Lafont and Philippe Malbos for many valuable discussions about polygraphic interpretations.

2 Polygraphs, interpretations and term rewriting systems translations

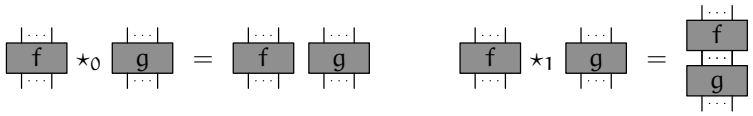
2.1 Polygraphs

The general definition of polygraph can be found in documents by Albert Burroni, Yves Lafont and François Métayer [5, 13, 18, 14, 15]. Here we give a rewriting-minded presentation of a special case of polygraphs, seeing them as rewriting systems on algebraic circuits.

Definition 2.1.1. A *monoidal 3-polygraph* is a composite object consisting of *cells*, *paths* and *compositions* organized into *dimensions*.

Dimension 1 contains elementary sorts called *1-cells* and represented by wires. Their concatenation $\star_0$ yields product types called *1-paths* and pictured as juxtaposed vertical wires. The empty product $*$ is also a 1-path, represented by the empty diagram.

Dimension 2 is made of operations called *2-cells*, with a finite number of typed inputs and outputs. They are pictured as circuit gates, with inputs at the top and outputs at the bottom. Using all the 1-cells and 2-cells as generators, one builds circuits called *2-paths*, using the following two compositions:



The constructions are considered *modulo* some relations, including topological deformation: one can stretch or contract wires freely, move 2-cells, provided one does not create crossings or break wires. Each 2-cell and each 2-path f has a 1-path $s_1(f)$ as input, its *1-source*, and a 1-path $t_1(f)$ as output, its *1-target*. The compact notation $f : s_1(f) \Rightarrow t_1(f)$ summarizes these facts.

Dimension 3 contains rewriting rules called *3-cells*. They always transform a 2-path into another one with the same 1-source and the same 1-target. Using all the 1-cells, 2-cells and 3-cells as generators, one can build reductions paths called *3-paths*, by application of the following three compositions, defined for F going from f to f' and G going from g to g': $F \star_0 G$ goes from $f \star_0 g$ to $f' \star_0 g'$; when $t_1(f) = s_1(g)$, then $F \star_1 G$ goes from $f \star_1 g$ to $f' \star_1 g'$; when $f' = g$, then $F \star_2 G$ goes from f to g'. These constructions are identified *modulo* some relations, given in [9], where their 3-dimensional nature was explained. The relations allow one to freely deform the constructions in a reasonable way: in particular, they identify paths that only differ by the order of application of the same 3-cells on non-overlapping parts of a 2-path. A 3-path is *elementary* when it contains exactly one 3-cell. Each 3-cell and each 3-path F has a 2-path $s_2(F)$ as left-hand side, its *2-source*, and a 2-path $t_2(F)$ as right-hand side, its *2-target*. The notation $F : s_2(F) \Rrightarrow t_2(F)$ stands for these facts.

In this study, *polygraph* always means *monoidal 3-polygraph*. For polygraphs, rewriting notions are defined in a similar way as for TRSs, with terms replaced by 2-paths, reduction steps by elementary 3-paths and reduction paths by 3-paths [8]. Hence, a *normal form* in a polygraph $\mathcal{P}$ is a 2-path f which is the 2-source of no elementary 3-path. The polygraph $\mathcal{P}$ *terminates* when it does not contain infinite families $(F_n)_{n\in\mathbb{N}}$ of elementary 3-paths such that $t_2(F_n) = s_2(F_{n+1})$ for all n. Other rewriting properties, such as *confluence* or *convergence* are also defined in an intuitive way. If X is a family of i-cells in $\mathcal{P}$, we denote by $\langle X\rangle$ the i-paths of $\mathcal{P}$ whose generating i-cells are all in X. If $i = 3$ and if there exists a 3-path in $\langle X\rangle$ from f to g, we use the notation $f \Rrightarrow_X g$.When this path is elementary, we write $f \Rrightarrow^1_X g$. If $X = \{\alpha\}$, we write $f \Rrightarrow_\alpha g$.

Example 2.1.2. We have already seen two examples of polygraphs in the introduction. The following one computes the addition and the multiplication on natural numbers $\langle$, $\rangle$, provided one adds the rules for the computation of and , as we explain in 2.3:

Note that we only give the 3-cells, since the 1-cells and 2-cells can be deduced from them. This polygraph is used in [4] to compute polynomials. The same document contains another example of polygraph, computing the fusion sort function on lists of natural numbers, which does not come from a TRS.

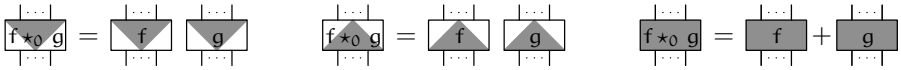
2.2 Polygraphic interpretations

In order to prove that a polygraph terminates, we have developped a notion of polygraphic interpretation [8]. Intuitively, we consider the 2-paths as circuits crossed by currents. Each 2-cell of a 2-path produces some heat according to the intensity of the currents that reach it and the total heat produced by the generating 2-cells of a 2-path is used to compare it to other ones.

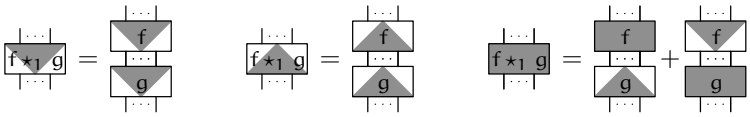
Definition 2.2.1. Let X and Y be non-empty ordered sets and M be a commutative monoid equipped with a strict, terminating order such that its addition is strictly monotone in both arguments. A *polygraphic*

interpretation of a polygraph $\mathcal{P}$ into (X, Y, M) consists into a mapping of each 2-path f with m inputs and n outputs onto three monotone maps $f_* = $  $: X^m \to X^n$, $f^* = $ $: Y^n \to Y^m$ and $[f] = $  $: X^m \times Y^n \to M$, such that the following conditions are satisfied:

- For every 1-path x of length n, we have $x_* = \mathrm{Id}_{X^n}$, $x^* = \mathrm{Id}_{Y^n}$ and $[x] = 0$.
- For every 2-paths f and g, the following three equalitities hold:



- For every 2-paths f and g such that $t_1(f) = s_1(g)$, the following three equalitities hold:



- For every 3-cell $\alpha : f \Rightarrow g$, we have $f \succ g$, which means that, for every possible x and y, the three inequalities $f_*(x) \geq g_*(x)$, $f^*(y) \geq g^*(y)$ and $[f](x, y) > [g](x, y)$ hold.

The first three conditions in the definition of polynomial interpretation ensure that, given a 2-path f, all the maps f_*, f^* and $[f]$ are uniquely determined by the maps φ_*, φ^* and $[\varphi]$ for all the 2-cells φ that f is made of. The following result was proved for polygraphs with exactly one 1-cell in [8]. In [9], it was explained, on an example with two 1-cells, how to extend the result to polygraphs with many 1-cells.

Theorem 2.2.2 ([8])**.** *If a 3-polygraph $\mathcal{P}$ admits a polygraphic interpretation, then it terminates.*

Example 2.2.3. Let us consider the polygraph consisting only of the four 3-cells for addition and multiplication given in example 2.1.2 and the following values:

-  $= 1$, $(i) = i + 1$,  $(i) = (i, i)$,  $(i, j) = i + j$,  $(i, j) = ij$;
- $\left[\right.$  $\left.\right] = \left[\right.$  $\left.\right](i) = \left[\right.$  $\left.\right](i) = \left[\right.$  $\left.\right](i) = 0$, $\left[\right.$  $\left.\right](i, j) = i$, $\left[\right.$$\left.\right](i, j) = (i + 1)j$.

To prove that this yields a polygraphic interpretation into $(\mathbb{N}, *, \mathbb{N})$, one makes computations such as:

$$\left[\right. \text{} \left.\right](i, j) \;=\; \left[\right. \text{} \left.\right](i, \text{}_*(j)) + \left[\right. \text{} \left.\right](j) \;=\; (i + 1)(j + 1).$$

One can find more examples of polygraphic interpretations and related computations in [8, 4]. In [4], it was proved that some polygraphic interpretations, such as the one we have built here, give more than termination: an information on the implicit complexity of the computed functions. Furthermore, the information we get here is divided into two parts: for every function , the current function $_*$ gives a bound on the size of the values computed by , while the heat function $\left[\right.$$\left.\right]$ limits the length of the computation. The heat function bound we get here is to be compared with the bounds that are found in [3] by using usual polynomial interpretations on a TRS: $(i + 1)j$ versus $(i + 1)(j + 1)$ for the multiplication and i versus $2i + j + 1$ for the addition.

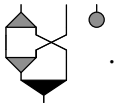
2.3 Polygraphic translations of term rewriting systems

This section recalls the standard translation of term rewriting systems into polygraphs. As a consequence of William Lawvere's work, term rewriting systems can be seen as presentations of algebraic theories [17]. Then, Albert Burroni has proved that an algebraic theory can be presented by a 3-polygraph [5]. Yves Lafont has given a standard translation of TRSs into 3-polygraphs [13]. We have proved that this translation preserves termination and, under the hypothesis of left-linearity, reflects it [8]. Because of size limitations, we only give here an informal construction of the polygraphic translation of a TRS, the formal results being in [13, 8].

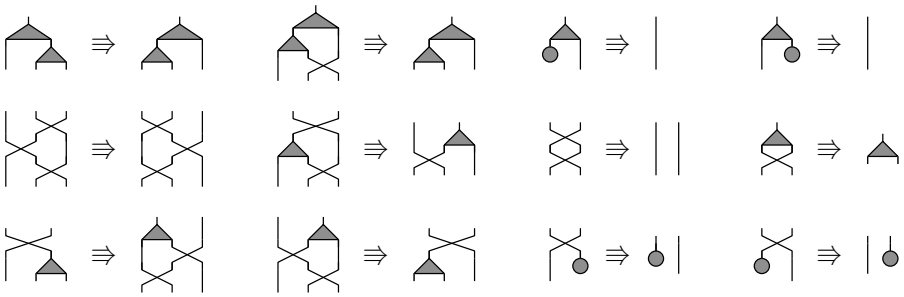
Let us fix a (many-sorted) TRS $\Sigma = (\Sigma_1, \Sigma_2, \Sigma_3)$, with elementary sorts in Σ_1, operations in Σ_2 and rewriting rules in Σ_3. The *standard polygraphic translation* of Σ is denoted by $\mathcal{P}(\Sigma)$ and is described thereafter, dimension after dimension:

Its 1-cells are the elements of Σ_1.

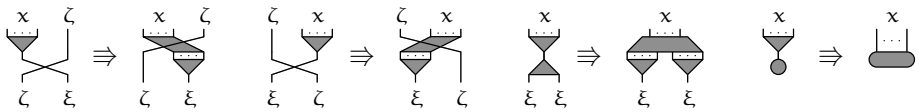
Its 2-cells are divided between *algebra 2-cells* and *structure 2-cells*. The algebra 2-cells are the elements of Σ_2; if $\varphi : \xi_1 \times \cdots \times \xi_n \to \xi$ is in Σ_2, then, as a 2-cell, $\varphi : \xi_1 \star_0 \cdots \star_0 \xi_n \Rightarrow \xi$. The family Δ_2 of *structure 2-cells* consists of one $\times : \xi \star_0 \zeta \Rightarrow \zeta \star_0 \xi$ for each pair (ξ, ζ) of 1-cells, plus one $\Delta : \xi \Rightarrow \xi \star_0 \xi$ and one $\bullet : \xi \Rightarrow *$ for each 1-cell ξ. Given a family $\vec{x}$ of distinct variables, any term u whose variables are in $\vec{x}$ admits a translation into a 2-path $u_{\vec{x}}$. This is formalized in [8] and suggested by the following examples, taken from the TRS for addition and multiplication of natural numbers:

$0_* =$  , $s(x)_{xy} =$, $A(x,x)_x =$  , $A(x,y)_{yx} =$  , $M(A(x,y),x)_{xyz} =$  .

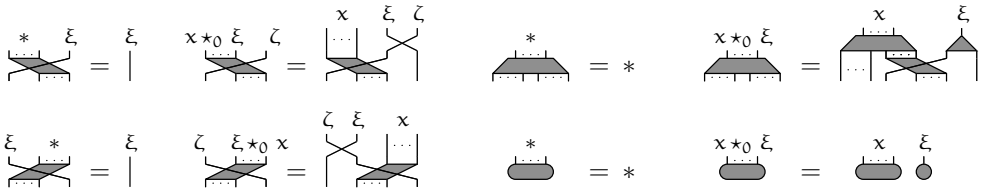
Its 3-cells are divided between *computation 3-cells* and *structure 3-cells*. The computation 3-cells are the elements of Σ_3; if $\alpha : u \to v$ is in Σ_3 and $\vec{x}$ is the family of distinct variables appearing in u from left to right, then $\alpha : u_{\vec{x}} \Rrightarrow v_{\vec{x}}$ when seen as a 3-cell. The family Δ_3 of structure 3-cells is divided into two subfamilies. The first subfamily Δ_3^1 depends only on the 1-cells and is given by the following diagrams with each wire coloured by any possible 1-cell:



The second subfamily Δ_3^2 depends on the algebra 2-cells and, for part, on the 1-cells. It is given, for any algebra 2-cell ∇ and 1-cell ζ, by:



The 2-targets of the 3-cells of Δ_3^2 use structure 2-paths built from the structure 2-cells by using the following structural induction rules:



Proposition 2.3.1 ([8])**.** *Let Σ be a term rewriting system. Then the following properties hold:*

1. *The three families Δ_3^1, Δ_3^2 and Δ_3 are convergent.*
2. *For any term u and any possible family $\vec{x}$ of variables, $u_{\vec{x}}$ is a Δ_3-normal form.*
3. *Any Δ_3^2 normal form is of the shape $f \star_1 g$ with f in $\langle \Delta_2 \rangle$ and g in $\langle \Sigma_2 \rangle$.*
4. *Any Δ_3-normal form is of the shape $f \star_1 g$ with f a Δ_3^1-normal form in $\langle \Delta_2 \rangle$ and g in $\langle \Sigma_2 \rangle$.*
5. *If Σ is left-linear, u and v are two terms and α is a rule such that $u \to_\alpha v$ holds, then, for any possible family $\vec{x}$ of variables, there exists a 2-path f such that $u_{\vec{x}} \Rrightarrow_\alpha^1 f \Rrightarrow_{\Delta_3} v_{\vec{x}}$ holds.*

In the following equivalence, the direct implication is true even without the hypothesis of left-linearity: it is proved by using a special polygraphic interpretation. The reverse direction is the one that is of interest for us in this study and is proved with the help of proposition 2.3.1 last point.

Theorem 2.3.2 ([8])**.** *A left-linear TRS Σ terminates if and only $\mathcal{P}(\Sigma)$ does.*

Example 2.3.3. Let us try to use theorem 2.3.2 to prove termination of the division program and see why and how we want to enhance it. We consider the polygraphic interpretation of the computation 3-cells into $(\mathbb{N}, *, \mathbb{N})$ generated by the following values:

- $\bigcirc_* = 1$, $\phi_*(i) = (i+2)$, $\times_*(i,j) = (j,i)$, $\triangle_*(i) = (i,i)$, $\triangledown_*(i,j) = \blacktriangledown_*(i,j) = i$;
- $[\bigcirc] = [\phi]\,(i) = [\times]\,(i,j) = [\triangle]\,(i) = [\bullet]\,(i) = 0$, $[\triangledown]\,(i,j) = j$, $[\blacktriangledown]\,(i,j) = ij$.

One can check that this yields an interpretation such that $s_2(\alpha) \succ t_2(\alpha)$ for any computation 3-cell α. For example, for the last computation 3-cell α, we have both $s_2(\alpha)_*(i,j)$ and $t_2(\alpha)_*(i,j)$ equal to $i+2$ and $[s_2(\alpha)](i,j) = ij + 2j$ while $[t_2(\alpha)](i,j) = ij + j$. But the 3-cells expressing how to duplicate $\triangledown$ and $\blacktriangledown$ satisfy the reverse strict inequality: we do not have a polygraphic interpretation of the standard polygraphic translation.

Thus, we do not have its termination and, consequently, no information on the implicit complexity of the division function, even if the interpretation we have considered satisfies the conditions of [4]. In section 3, we correct this problem with results that allow us, in particular, to conclude for the present example.

3 Reduced polygraphic translations for left-linear term rewriting systems

From now on, we assume that Σ is a left-linear term rewriting system.

3.1 The general case

Here we prove that there is no need to consider the family Δ_3^1 of structure 3-cells for proving termination of the original TRS: these 3-cells are only required to ensure confluence.

Lemma 3.1.1. *Let f be a 2-path and g its Δ_3-normal form. Then there exists a 2-path h in Δ_3^2-normal form such that $f \Rrightarrow_{\Delta_3^2} h \Rrightarrow_{\Delta_3^1} g$ holds.*

Proof. Let h be the Δ_3^2-normal form of f. We use proposition 2.3.1: since Δ_3 is confluent (first point), we know that the Δ_3-normal form of h is g. From the shape of h (third point), we deduce that only 3-cells from Δ_3^1 can be applied to h or any of its reduces. □

Theorem 3.1.2. *If $\mathcal{P}(\Sigma)$ terminates without the first family of structure 3-cells, then Σ terminates.*

Proof. Let us assume that $\mathcal{P}(\Sigma)$ terminates without Δ_3^1 but that Σ does not terminate. Then, there exists a sequence $(u_n)_{n\in\mathbb{N}}$ of terms and a sequence $(\alpha_n)_{n\in\mathbb{N}}$ such that, for every natural number n, we have $u_n \to_{\alpha_n} u_{n+1}$. Let us fix a family $\vec{x}$ of variables such that $(u_0)_{\vec{x}}$ is defined. Then, the last point of proposition 2.3.1 yields a family of 2-paths $(f_n)_{n\in\mathbb{N}}$ such that $(u_n)_{\vec{x}} \Rrightarrow^1_{\alpha_n} f_n \Rrightarrow_{\Delta_3} (u_{n+1})_{\vec{x}}$ holds for every n. Let us fix a natural number n. From proposition 2.3.1, we know that $(u_{n+1})_{\vec{x}}$ is a Δ_3-normal form (second point) and that Δ_3 is confluent (first point): hence $(u_{n+1})_{\vec{x}}$ is the Δ_3-normal form of the 2-path f_n. We apply lemma 3.1.1 and get a 2-path g_n in Δ_3^2-normal form satisfying:

$$(u_n)_{\vec{x}} \Rrightarrow^1_{\alpha_n} f_n \Rrightarrow_{\Delta_3} g_n \Rrightarrow_{\Delta_3^1} (u_{n+1})_{\vec{x}}.$$

With proposition 2.3.1, we know that $g_n = h_n \star_1 k_n$ (third point) and $(u_{n+1})_{\vec{x}} = h'_n \star_1 k_n$ (fourth point), with k_n in $\langle\Sigma_2\rangle$, h_n in $\langle\Delta_2\rangle$ and h'_n its Δ_3^1-normal form. But we have seen that $(u_{n+1})_{\vec{x}} \Rrightarrow^1_{\alpha_{n+1}} f_{n+1}$ holds and, since the TRS we consider is left-linear, the 2-source of α_{n+1} does not contain any structure 2-cell. This implies that $s_2(\alpha_{n+1})$ is entirely contained into k_n, so that f_{n+1} can be decomposed into $h'_n \star_1 k'_n$, with $k_n \Rrightarrow^1_{\alpha_{n+1}} k'_n$. We deduce from these facts the following reduction chain:

$$g_n = h_n \star_1 k_n \Rrightarrow^1_{\alpha_{n+1}} h_n \star_1 k'_n \Rrightarrow_{\Delta_3^1} h'_n \star_1 k'n = f_{n+1}.$$

We know that $(u_{n+2})_{\vec{x}}$ is the Δ_3-normal form of f_{n+1}. By confluence of Δ_3, we deduce that it is also the Δ_3-normal form of $h_n \star_1 k'_n$. Then lemma 3.1.1 gives the existence of a Δ_3^2-normal form g_{n+1} such that:

$$h_n \star_1 k'_n \Rrightarrow_{\Delta_3^2} g_{n+1} \Rrightarrow_{\Delta_3^1} (u_{n+2})_{\vec{x}}.$$

By induction on n, we conclude that the infinite reduction path $(u_n)_{n\in\mathbb{N}}$ in Σ generates an infinite reduction path in $\mathcal{P}(\Sigma)$ that only uses 3-cells of Σ_3 and Δ_3^2, the existence of which has been prohibited by assumption. Hence Σ terminates. □

Remark 3.1.3. Removing some structure 3-cells allows for a wider range of interpretations for the duplication, such as $\wedge_*(i) = (\lceil i/2 \rceil, \lfloor i/2 \rfloor)$ for the descending currents. This would be prohibited by the two structure 3-cells of Δ_3^1 involving both $\wedge$ and $\bullet$, since they require the inequality $\wedge_*(i) \geq (i, i)$.

3.2 The case of planar linear term rewriting systems

Let us recall that a term rewriting rule is usually called linear when no variable occur twice in its left member or in its right member. Here and in order to match the vocabulary from linear algebra and operadic theory, we call a rule *linear* when its two sides contain exactly the same variables, exactly once. Thus, if Σ is linear, which means that all of its rules are, the computation 3-cells of the polygraph $\mathcal{P}(\Sigma)$ do not use any  or  . We say that a term rewriting rule is *planar* when variables occur in the same order in its two sides. Then Σ is planar when all of its rules are and, in that case, the computation 3-cells of $\mathcal{P}(\Sigma)$ do not use any  .

Theorem 3.2.1. *Let us assume that Σ is both linear and planar. If $\mathcal{P}(\Sigma)$ terminates without the structure 3-cells, then Σ terminates.*

Proof. If Σ is both linear and planar, then the computation 3-cells of $\mathcal{P}(\Sigma)$ do not contain any structure 2-cell. Now, let us assume that $\mathcal{P}(\Sigma)$ terminates without the structure 3-cells. If Σ does not terminate, then there exists an infinite reduction path $u_0 \to_{\alpha_0} u_1 \to_{\alpha_1} u_2 \to_{\alpha_2} (\cdots)$. Let us fix a family $\vec{x}$ of distinct variables such that $(u_0)_{\vec{x}}$ is defined. By proposition 2.3.1, points two and four, each $(u_n)_{\vec{x}}$ decomposes into $g_n \star_1 h_n$ with g_n in $\langle \Delta_2 \rangle$ and h_n in $\langle \Sigma_2 \rangle$. Let us fix a natural number n. Then, point five of the same proposition yields a 2-path f_n such that:

$$g_n \star_1 h_n = (u_n)_{\vec{x}} \Rrightarrow^1_{\alpha_n} f_n \Rrightarrow_{\Delta_3} (u_{n+1})_{\vec{x}} = g_{n+1} \star_1 h_{n+1}.$$

Since α_n is left-linear, its 2-source does not contain any structure 2-cell, so that it is entirely contained into h_n. This means that there exists a 2-path h'_n such that $h_n \Rrightarrow^1_{\alpha_n} h'_n$ and $f_n = g_n \star_1 h'_n$. Since α_n is linear and planar, its 2-target does not contain any structure 2-cell either: this means that h'_n is in $\langle \Sigma_2 \rangle$. Hence, any 3-path of $\langle \Delta_3 \rangle$ starting at $f_n = g_n \star_1 h'_n$ only acts on g_n. So g_{n+1} is the Δ_3-normal form of g_n and $h'_n = h_{n+1}$. Thus, we have $h_n \Rrightarrow^1_{\alpha_n} h_{n+1}$. Since this is valid for any natural number n, we have an infinite reduction path in $\mathcal{P}(\Sigma)$ that does not use any structure 3-cell: this is prohibited by our hypothesis, so that Σ terminates. □

Remark 3.2.2. The proof of theorem 3.2.1 can be adaptated for TRSs whose computation 3-cell do not require any  or any  or any  . In that case, we do not discard all the srtucture 3-cells of Δ_3^2 but only the ones concerning the unused structure 2-cell(s).

Example 3.2.3. The TRS for the double function is both linear, in our sense, and planar. Hence, for proving its termination, we only have to prove the termination of the two computation 3-cells of its translation. We consider the interpretation into $(\mathbb{N}, *, \mathbb{N})$ generated by $\bigcirc_* = 1$, $\overset{|}{\bigcirc}_*(i) = i+1$, $\overset{|}{\bullet}_*(i) = 2i$, $\left[\bigcirc\right] = 0$, $\left[\overset{|}{\bigcirc}\right](i) = 0$, $\left[\overset{|}{\bullet}\right](i) = i$. On top of proving termination, this polygraphic interpretation satisfies the conditions given in [4] that allow us to conclude that the polynomials $\overset{|}{\bullet}_*(i) = 2i$ and $\left[\overset{|}{\bullet}\right](i) = i$ respectively bound the size of the computed values and the length of the computations, with respect to the size of the argument. This is to be compared with the polynomial bounds of $3i$ given by polynomial interpretations on terms and of $2i$ given by the same interpretations with a preprocessing using dependency pairs.

3.3 The case of first-order functional programs

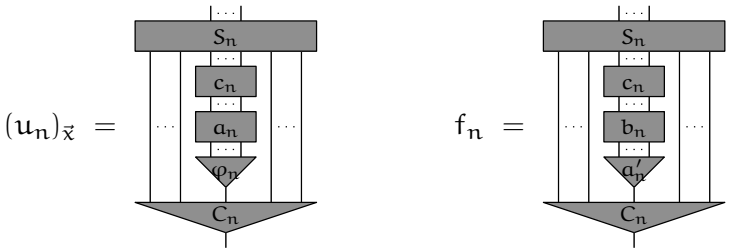
Definition 3.3.1. A *function* (or *defined symbol*) of Σ is an operation in Σ_2 that only appears as the root symbol of the left-hand side of rewriting rules of Σ_3. A *constructor* of Σ is an operation in Σ_2 that never appears as the root symbol of the left-hand side of rewriting rules of Σ_3. Two rules α and β are *weakly orthogonal* when all their critical pairs are of the form $u \rightrightarrows^{\alpha}_{\beta} v$: whenever one can apply α and β on overlapping parts of the same term, both reductions give the same result.

A *first-order functional program* is a left-linear term rewriting system Σ whose operations are either a function or a constructor and whose rewriting rules are pairwise weakly orthogonal. In that case, we denote by Σ_2^C the set of constructors and by Σ_2^F the one of functions. The *polygraphic program* associated to a first-order functional program Σ is the standard polygraphic translation $\mathcal{P}(\Sigma)$ without Δ_3^1 and Δ_3^F, the 3-cells of Δ_3^2 corresponding to functions. We denote by Δ_3^C the 3-cells of Δ_3^2 corresponding to constructors.

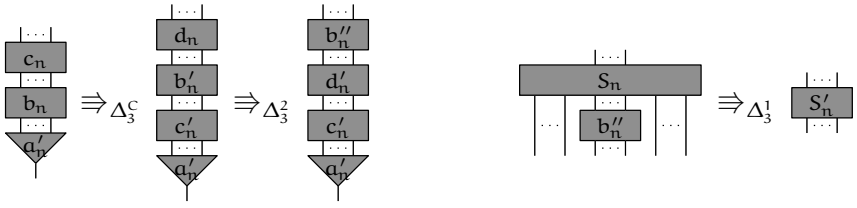
The notion of polygraphic program was introduced in [4]. This class of rewriting systems contains more than translations of first-order functional programs: polygraphic programs can compute functions with many outputs with a link between them, such as the list splitting function that is studied in [4]. A key argument for the following result is given in [7]: for a first-order functional program, termination and innermost termination are equivalent.

Theorem 3.3.2. *Let Σ be a first-order functional program. If the polygraphic program associated to Σ terminates, then so does Σ.*

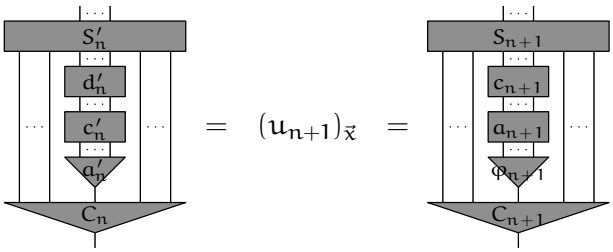
Proof. Let us assume that the polygraphic program associated to Σ terminates but that Σ does not. Hence, there exists an infinite innermost reduction sequence $u_0 \to^i_{\alpha_0} u_1 \to^i_{\alpha_1} u_2 \to^i_{\alpha_2} (\cdots)$ in Σ. Let $\vec{x}$ be a family of variables such that $(u_0)_{\vec{x}}$ is defined. Point five of proposition 2.3.1 tells us that the reduction sequence $(u_n)_{n\in\mathbb{N}}$ lifts up to $\mathcal{P}(\Sigma)$, yielding $(u_0)_{\vec{x}} \Rrightarrow^1_{\alpha_0} f_0 \Rrightarrow_{\Delta_3} (u_1)_{\vec{x}} \Rrightarrow^1_{\alpha_1} f_1 \Rrightarrow_{\Delta_3} (u_2)_{\vec{x}} \Rrightarrow^1_{\alpha_2} (\cdots)$. Since Σ is a functional program, there exist, for every n in $\mathbb{N}$, φ_n in Σ_2^F, a_n in $\langle\Sigma_2^C\rangle$, a'_n in $\langle\Sigma_2\rangle$ and b_n in $\langle\Delta_2\rangle$ such that $s_2(\alpha_n) = a_n \star_1 \varphi_n$ and $t_2(\alpha_n) = b_n \star_1 a'_n$. Now, since $(u_n)_{\vec{x}} \Rrightarrow^1_{\alpha_n} f_n$, there exist C_n and c_n in $\langle\Sigma_2\rangle$ and S_n in $\langle\Delta_2\rangle$ such that:



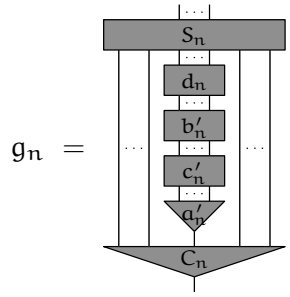
By examination of the shapes and properties of the structure 3-cells, there exist b'_n, b''_n and S'_n in $\langle\Delta_2\rangle$, c'_n in $\langle\Sigma_2^C\rangle$, d_n and d'_n in $\langle\Sigma_2\rangle$ such that the following three *normalizing* reductions hold:



Thus, we have two decompositions of $(u_{n+1})_{\vec{x}}$, expressed by the following equalities:



In the leftmost decomposition, the function 2-cell φ_{n+1} of the rightmost decomposition must appear in either d'_n, a'_n or C_n, since c'_n is in Σ_2^C and S'_n is in $\langle\Delta_2\rangle$. Let us assume that it is in d'_n. Since d'_n has been produced from d_n by the action of the structure 2-cells, any redex it contains is a copy of one that is already in d_n. This means that the reduction α_{n+1} can already be applied on u_n, in a proper subterm of the term where α_n is applied: this is in contradiction with the hypothesis that the reduction from u_n to u_{n+1} is innermost. Hence, φ_{n+1} is in C_n or in a'_n. Furthermore, a_{n+1} is only made of constructors: each 2-cell it contains was either already in C_n or a'_n or appear in c'_n as the result of the application of 3-cells of Δ_3^C. Hence, the reduction $(u_{n+1})_{\vec{x}} \Rrightarrow^1_{\alpha_{n+1}} f_{n+1}$ can be anticipated on the following 2-path g_n, which is the Δ_3^C-normal form of f_n:

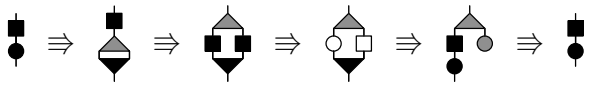


Let us denote by h_n the result of this α_{n+1}-reduction on g_n and by g_{n+1} the Δ_3^C-normal form of h_n. Then g_{n+1} can be normalized successively by Δ_3^2 and by Δ_3^1 to reach $(u_{n+2})_{\vec{x}}$. Using again the fact that the reductions on terms have been supposed to be innermost, we prove that the reduction acting on $(u_{n+2})_{\vec{x}}$ can also be anticipated on g_{n+1}, yielding h_{n+1} and so on. Thus, an induction on n gives an infinite reduction sequence $(u_0)_{\vec{x}} \Rrightarrow^1_{\alpha_0} f_0 \Rrightarrow_{\Delta_3^C} g_0 \Rrightarrow^1_{\alpha_1} h_0 \Rrightarrow_{\Delta_3^C} g_1 \Rrightarrow^1_{\alpha_2} h_1 \Rrightarrow_{\Delta_3^C} (\cdots)$, which cannot exist by termination of the polygraphic program associated to Σ. □

Example 3.3.3. Let us consider the interpretation we have built in example 2.3.3. Now, equipped with theorem 3.3.2, we can conclude that this interpretation proves the termination of the original term rewriting system for division. Moreover, using the results proved in [4], we conclude that the polynomials $\blacktriangledown_*(X,Y) = X$ and $\left[\blacktriangledown\right](X,Y) = XY$ respectively bound the spatial and temporal sizes of the computation of the division on two arguments with sizes X and Y.

Remark 3.3.4. The result is false if one removes the weak orthogonality assumption. Indeed, let us consider the polygraphic program whose computation 3-cells are:

We prove that it terminates with a mapping into $(\mathbb{N}, *, [\mathbb{N}])$, where $[\mathbb{N}]$ is the free commutative monoid generated by $\mathbb{N}$ with its natural multiset order, with $\{n\}$ standing for n seen as a generator of $[\mathbb{N}]$. We consider:  $= i$,  $= i + 1$,  $(i, j) = i + j$, $(i) = (\lceil i/2 \rceil, \lfloor i/2 \rfloor)$ and  $= 0$,  $= \{i\}$ and $= \{i + j\}$. One proves that this mapping satisfies $s_2(\alpha) \succ t_2(\alpha)$ if α is the second, third or fourth computation 3-cell and that $s_2(\beta) = t_2(\beta)$ is β is the first one or is in Δ_3^C. Then, we define the mapping counting the number of in a 2-path: this is the mapping into $(*, *, \mathbb{N})$ whose heat function sends any 2-cell to 0 except , sent to 1. Finally, we use the termination of Δ_3^C to get the one of the polygraphic program. However, if we add Δ_3^F and Δ_3^1, it does not terminate anymore, as proved by the following cycle:



However, the results in [7] seem to indicate that the weak orthogonality hypothesis is too strong and could be replaced by local confluence.

3.4 Special conditions on standard interpretations

Until now, we have seen conditions based solely on the properties of the original TRS. Here we assume that we use polygraphic interpretations and give conditions on them: the purpose of such results is to guide the automatic search of polygraphic interpretations. Let us recall that a TRS is *non-duplicating* when no right-hand side of a rewriting rule contains the same variable twice. Hence, the computation 3-cells of such a TRS polygraphic translation do not use any .

Theorem 3.4.1. *Let us assume that Σ is non-duplicating. If $\mathcal{P}(\Sigma)$, without the structure 3-cells, admits a polygraphic interpretation into some $(X, *, M)$ such that $(x, y) = (y, x)$, then Σ terminates.*

Proof. Since Σ is non-duplicating, we adapt the proof of 3.2.1, as mentionned in remark 3.2.2, to get that the termination of Σ can be deduced from the one of $\mathcal{P}(\Sigma)$, without Δ_3^1 and all the structure 3-cells of Δ_3^2 that concern ; for this proof, we denote by Δ_3^n the remaining structure 3-cells. Now, let us assume that $\mathcal{P}(\Sigma)$, without the structure 3-cells, admits a polygraphic interpretation $((\cdot)_*, (\cdot)^*, [\cdot])$ into $(X, *, M)$ such that $(i, j) = (j, i)$ holds. Then we define another mapping into $(X, *, M)$ with the same currents functions and with a heat function $\{\cdot\}$ defined as $[\cdot]$ except on the structure 2-cells, which it sends to 0. Since the addition of M is monotone in each argument and by properties of the heat functions, we have $[f] \geq \{f\}$ for any 2-path f, with equality when f is in $\langle \Sigma_2 \rangle$. Hence, by left-linearity of Σ, we get, for each computation 3-cell α, $\{s_2(\alpha)\} = [s_2(\alpha)] > [t_2(\alpha)] \geq \{t_2(\alpha)\}$. Now let us fix an algebra 2-cell and consider the structure 3-cells of Δ_3^n it is involved into. For , the current functions are equal on both sides and $(\vec{x}) = \left[\right] (\vec{x}) \geq 0 =$ $(\vec{x})$. For , we have equality between the current functions and the heat functions on both sides, thanks to the hypothesis $(x, y) = (y, x)$. Hence, the new map $\{\cdot\}$ generates a terminating order relation $\succ$ on $\mathcal{P}(\Sigma)$ such that, for every computation 3-cell, $s_2(\alpha) \succ t_2(\alpha)$ and, for every structure 3-cell β of Δ_3^n, $s_2(\beta) \succeq t_2(\beta)$. Thus, all these 3-cells, together, terminate if and only if Δ_3^n terminates, which is true. □

Notation 3.4.2. Let $\alpha : \varphi(u_1, \ldots, u_n) \to v$ be a rewrite rule in Σ_3. For $i \in \{1, \ldots, n\}$ we denote by $K_i(\alpha)$ the greatest of the number of occurences in v of each variable of u_i. For any function φ of

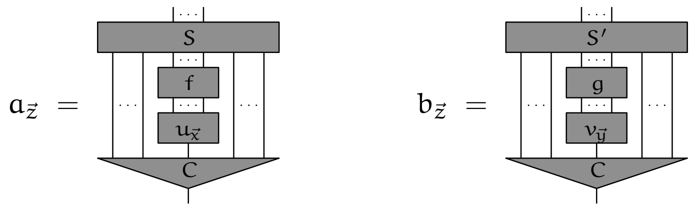
arity n in Σ_2 and any $i \in \{1, \ldots, n\}$, we denote by $K_i(\varphi)$ the greatest of the $K_i(\alpha)$ for all the rules α such that φ is the root symbol of the left-hand side of α.

Theorem 3.4.3. *Let X be a set equipped with a terminating strict order. Let us assume that $\mathcal{P}(\Sigma)$, without the structure 3-cells, admits a polygraphic interpretation into $(X, *, [X])$ such that the following conditions hold:*

- $\Delta(x) = (x, x)$, $\tau(x, y) = (y, x)$ *and* $[\sigma] = 0$ *when* $\sigma \in \{\tau, \Delta, \epsilon\}$.
- $\varphi_*(x_1, \ldots, x_n) \geq x_i$ *for all* φ *and all* i.
- $\{\varphi_*(\vec{x})\} > [\varphi](\vec{x})$ *for all* φ.
- $[\varphi](x_1, \ldots, x_n) > \{x_i\}$ *if* φ *is the root symbol of some rule and* $K_i(\varphi) \geq 2$.

Then Σ terminates.

Proof. Let a and b be two terms and α a rewriting rule such that $a \to_\alpha b$. We denote by $u = \varphi(u_1, \ldots, u_p)$ the source of α, by v its target and by $\vec{x}$ and $\vec{y}$ the respective families of distinct variables that appear in u and v from left to right. We fix a family $\vec{z}$ of distinct variables containing all the variables that appear in a. Then the 2-paths $a_{\vec{z}}$ and $b_{\vec{z}}$ decompose as:



$$a_{\vec{z}} = \ldots \qquad b_{\vec{z}} = \ldots$$

In these decompositions, S and S' are in $\langle \Delta_2 \rangle$ and f, g, C and C' in $\langle \Sigma_2 \rangle$. Let us write $f = f_1 \star_0 \cdots \star_0 f_m$ and $g = g_1 \star_0 \cdots \star_0 g_n$, where each f_i and g_j has a 1-cell as 1-target. We denote by $j(i)$ the element of $\{1, \ldots, p\}$ such that f_i appears inside $u_{j(i)}$ in a. Let c be the structure 2-path and d the algebra 2-path such that $v_{\vec{x}} = c \star_1 d$, given by proposition 2.3.1 points two and four. Using the same proposition, we know that there exists a structure 2-path c' such that $c' \star_1 g$ is the Δ_3^2-normal form of $f \star_1 c$. But the structure 3-cells of Δ_3^2 act in such a way that each g_j is exactly one f_i. Moreover, in the family $(g_1, \ldots, g_n)$, each f_i appears at most $K_{j(i)}(\varphi)$ times, by definition of $K_{j(i)}(\varphi)$.

Now, let us compute the interpretations of $a_{\vec{y}}$ and $b_{\vec{y}}$. By hypothesis on the interpretation, we have $(a_{\vec{y}})_* \geq (b_{\vec{y}})_*$. Concerning the heats, still using the assumptions on the interpretation, C receives at least the same currents in $a_{\vec{y}}$ than in $b_{\vec{y}}$ and $[C]$ is monotone: hence C produces at least the same heat in $a_{\vec{y}}$ than in $b_{\vec{y}}$. For the same reasons and since $[s_2(\alpha)] > [t_2(\alpha)]$, $u_{\vec{x}}$ produces strictly more heat in $a_{\vec{y}}$ than in $b_{\vec{y}}$. Furthermore f produces $\sum_{i=1}^{m} [f_i](\vec{k_i})$ while g produces $\sum_{i=1}^{m} K_{j(i)}(\varphi).[f_i](\vec{k_i})$ for some $\vec{k_i}$: indeed the currents received by each copy of f_i in g are the same as the currents received by the original f_i, by properties of the current map on structure 2-cells and by examination of each structure 3-cell. Finally, the structure 2-paths S and S' do not produce any heat.

Since we consider a multiset order on $[X]$, we can prove that $[a_{\vec{y}}] > [b_{\vec{y}}]$ by proving that $u_{\vec{y}}$ produces strictly more heat than each f_i such that $K_{j(i)}(\varphi) \geq 2$; by property of the heat function, it is even sufficient to prove that $[\varphi]$ produces more heat than each f_i such that $K_{j(i)} \geq 2$. In $a_{\vec{y}}$, the current received by $u_{\vec{y}}$

is $((f_1)_*(\vec{k}_1), \ldots, (f_m)_*(\vec{k})_m)$. Since each u_j transmits a current at least equal to the one it receives in any of its inputs, φ receives at least $(f_i)_*(\vec{k}_i)$ in its imput $j(i)$. By assumption, if $K_{j(i)}(\varphi) \geq 2$, then the heat produced by φ is strictly greater than $\{(f_i)_*(\vec{k}_i)\}$, which, in turn, is striclty greater than $[f_i](\vec{k}_i)$, once again since we consider a multiset order on $[X]$. Finally, we have $a_{\vec{y}} \succ b_{\vec{y}}$ whenever $a \to_\alpha b$: we deduce from this fact that Σ terminates. □

4 Conclusion

In this study, we have proved results that make easier the use of polygraphs and polygraphic interpretations for proving termination for TRSs and, when it comes to functional programs, finding an implicit complexity bound with some interpretations. We have seen, on some examples, that the method can give better results than polynomial interpretations on TRSs, mainly for functional programs: it gives better complexity bounds and can prove the termination and give bounds for TRSs that do not admit simplification orders.

The next step consists into a test of this method on the Termination Problems DataBase in order to get information on its efficiency and to formulate new conjectures. Among them, we want to examine the hypothesis of weak orthogonality used in theorem 3.3.2 and new ways to guide the construction of the polygraphic interpretation with respect to the shape of the rewriting rules, in the same spirit as in 3.4. We also plan to enhance the theoretical links between termination of TRSs and termination of polygraphic versions of them: indeed, we think that there are many known, finer results on term graph rewriting systems that can be adaptated to polygraphs [19]. The case of non left-linear TRSs may be examinated but we are not sure that the polygraphic translations can provide methods for them.

Finally, we need a better understanding of the mathematical structure behind the one of polygraphic interpretation. This will allow for new kinds of interpretations, extending the range of the method for a wider variety of functional programs. The main class we will focus on are ones with conditional rules or with the if-then-else construction, in order to solve some of the problems we have encountered in [4].

References

[1] Thomas Arts and Jürgen Giesl, *Termination of term rewriting using dependency pairs*, Theoretical Computer Science **236** (2000), no. 1-2, 133–178.

[2] Franz Baader and Tobias Nipkow, *Term rewriting and all that*, Cambridge University Press, 1998.

[3] Guillaume Bonfante, Adam Cichon, Jean-Yves Marion, and Hélène Touzet, *Algorithms with polynomial interpretation termination proofs*, Journal of Functional Programming **11** (2001), no. 1, 33–53.

[4] Guillaume Bonfante and Yves Guiraud, *Programs as polygraphs: computability and complexity*, Submitted, 2006.

[5] Albert Burroni, *Higher-dimensional word problems with applications to equational logic*, Theoretical Computer Science **115** (1993), no. 1, 43–62.

[6] Adam Cichon and Pierre Lescanne, *Polynomial interpretations and the complexity of algorithms*, Lecture Notes in Artificial Intelligence **607** (1992), 139–147.

[7] Bernhard Gramlich, *On modularity of termination and confluence properties of conditional rewrite systems*, Lecture Notes in Computer Science **859** (1994), 186–203.

[8] Yves Guiraud, *Termination orders for 3-dimensional rewriting*, Journal of Pure and Applied Algebra **207** (2006), no. 2, 341–371.

[9] ———, *The three dimensions of proofs*, Annals of Pure and Applied Logic **141** (2006), no. 1-2, 266–295.

[10] ———, *Two polygraphic presentations of petri nets*, Theoretical Computer Science **360** (2006), no. 1-3, 124–146.

[11] Yves Lafont, *Interaction nets*, Principles of Programming Languages, ACM Press, 1990, pp. 95–108.

[12] ———, *Equational reasoning for 2-dimensional diagrams*, Lecture Notes in Computer Science **909** (1995), 170–195.

[13] ———, *Towards an algebraic theory of boolean circuits*, Journal of Pure and Applied Algebra **184** (2003), no. 2-3, 257–310.

[14] ———, *Algebra and geometry of rewriting*, Preprint IML, 2006.

[15] Yves Lafont and François Métayer, *Polygraphic resolutions and homology of monoids*, Preprint IML, 2006.

[16] Dallas Lankford, *On proving term rewriting systems are noetherian*, Tech. report, Louisiana Tech University, 1979.

[17] Francis William Lawvere, *Functorial semantics of algebraic theories*, Reprints in Theory and Applications of Categories **5** (2004), 1–121.

[18] François Métayer, *Resolutions by polygraphs*, Theory and Applications of Categories **11** (2003), 148–184.

[19] Detlef Plump, *Term graph rewriting*, Handbook of Graph Grammars and Computing by Graph Transformation **2** (1999), 3–61.