Logical Methods in Computer Science
Vol. 5 (2:14) 2009, pp. 1–17
www.lmcs-online.org

Submitted Jan. 5, 2007
Published Jun. 3, 2009

POLYGRAPHIC PROGRAMS AND POLYNOMIAL-TIME FUNCTIONS

GUILLAUME BONFANTE AND YVES GUIRAUD

INRIA Nancy, 615 rue du Jardin Botanique, CS 20101, 54603 Villers-lès-Nancy, France
e-mail address: {guillaume.bonfante,yves.guiraud}@inria.fr

Abstract. We study the computational model of polygraphs. For that, we consider polygraphic programs, a subclass of these objects, as a formal description of first-order functional programs. We explain their semantics and prove that they form a Turing-complete computational model. Their algebraic structure is used by analysis tools, called polygraphic interpretations, for complexity analysis. In particular, we delineate a subclass of polygraphic programs that compute exactly the functions that are Turing-computable in polynomial time.

Introduction

Polygraphs as a computational model. Polygraphs (or computads) are presentations by "generators" and "relations" of some higher-dimensional categories [41, 12, 42, 43]. Albert Burroni has proved that they provide an algebraic structure to equational theories [12]. Yves Lafont and the second author have explored some of the computational properties of these objects, mainly termination, confluence and their links with term rewriting systems [27, 18]. The present study, extending notions and results presented earlier by the same authors [9], concerns the complexity analysis of polygraphs.

On a first approach, one can think of these objects as rewriting systems on algebraic circuits: instead of computing on syntactical terms, polygraphs make use of a net of cells, which individually behave according to some local transition rules, as do John von Neumann's cellular automata [46] and Yves Lafont's interaction nets [26].

Following Neil Jones' thesis that programming languages and semantics have strong connexions with complexity theory [24], we think that the syntactic features offered by polygraphs, with respect to terms, play an important role from the point of view of implicit computational complexity. As a running example, we consider the divide-and-conquer algorithm of fusion sort. It computes the function f taking a list l and returning the list made of the same elements, yet sorted according to some given order relation. For that, it uses a divide-and-conquer strategy: it splits l into two sublists l_1 and l_2 of equivalent sizes, then it recursively applies itself on each one to get $f(l_1)$ and $f(l_2)$ and, finally, it merges these

1998 ACM Subject Classification: F.1.1, F.4.

Key words and phrases: Polygraph; Polygraphic program; Polygraphic interpretation; Computability; Complexity; Polynomial time.

This work has been partially supported by ANR Inval project (ANR-05-BLAN-0267).

LOGICAL METHODS IN COMPUTER SCIENCE

DOI:10.2168/LMCS-5 (2:14) 2009

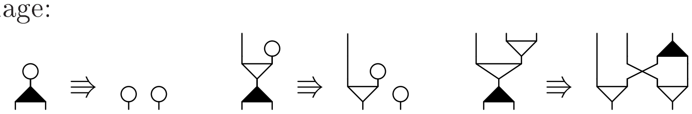
two results to produce $f(l)$. The following program, written in Caml [13], implements this algorithm:

```
let rec split = function
  | [] -> ([],[])
  | x::[] -> (x::[],[])
  | x::y::l -> let (l1,l2)=split(l) in (x::l1,y::l2)

let rec merge = function
  | ([],l) -> l
  | (l,[]) -> l
  | (x::l,y::m) -> if x<=y then x::merge(l,y::m) else y::merge(x::l,m)

let rec sort = function
  | [] -> []
  | x::[] -> x::[]
  | x::y::l -> let (l1,l2)=split(l) in merge(sort(x::l1),sort(y::l2))
```

In a polygraph, one can consider, at the same level as other operations, function symbols with many outputs. For example, the above definition of the split function becomes, in the polygraphic language:



With these rules, one can actually "see" how the computation is made, by "unzipping" lists. Also, one can internalize in polygraphs the sharing operation of termgraphs [39], described as an explicit and local duplication. As a consequence, the rules generating computations become linear: the operations for pointers management can be "seen" within the rules. Actually, in our analysis, we evaluate explicitly the number of structural steps of computation: allocations, deallocations and switches of pointers. In other words, we make explicit the design of a garbage collector.

The question of sharing has been widely studied for efficient implementations of functional programming languages and several solutions have been suggested: for instance, Dan Dougherty, Pierre Lescanne and Luigi Liquori proposed the formalism of addressed term rewriting systems [15]. Let us mention another approach for this kind of issues due to Martin Hofmann [23]: he developed a typing discipline, with a diamond type, for a functional language which allows a compilation into an imperative language such as C, without dynamic allocation.

The computational model of polygraphic programs, a subclass of polygraphs, is explained in the first part of this document, where we give their semantics and prove a completeness result: every Turing-computable function can be computed by a polygraphic program.

Complexity analysis of polygraphic programs. Here we use tools inspired by polynomial interpretations, which have been introduced by Dallas Lankford to prove termination of term rewriting systems [30]. They associate to each term a polynomial with natural numbers as coefficients, in a way that is naturally compatible with contexts and substitutions. When, for each rule, the interpretation of the left-hand side is greater than the one of the right-hand side, one gets a termination proof. For example, let us consider the following term rewriting system that computes the double function on natural numbers:

$$d(0) \to 0 \qquad\qquad d(s(x)) \to s(s(d(x))).$$

One proves its termination with the interpretation defined by $\varphi(0) = 1$, $\varphi(s(x)) = \varphi(x) + 1$ and $\varphi(d(x)) = 3\varphi(x)$. Indeed, one checks that the following inequalities hold:

$$\varphi(d(0)) = 3 > 1 = \varphi(0) \qquad \text{and} \qquad \varphi(d(s(x))) = 3\varphi(x) + 3 > 3\varphi(x) + 2 = \varphi(s(s(d(x)))).$$

Moreover, on top of termination results, polynomial interpretations can be used to study complexity. For instance, Dieter Hofbauer and Clemens Lautemann have established a doubly exponential bound on the derivation length of systems with polynomial interpretations [22]. Adam Cichon and Pierre Lescanne have considered more precisely the computational power of these systems [14]. Adam Cichon, Jean-Yves Marion and Hélène Touzet, with the first author, have identified complexity classes by means of restrictions on polynomial interpretations [7, 8].

Let us explain how this works on the example of the double function. The given interpretation sends the term $d(s^n(0))$ to the natural number $3n + 3$: since each rule application will strictly decrease this number, one knows that it takes at most $3n + 3$ steps to get from this term to its normal form $s^{2n}(0)$. Actually, the considered interpretation gives a polynomial bound, with respect to the size of the argument, on the time taken to compute the double function with this program.

In order to analyze polygraphs, we use algebraic tools called polygraphic interpretations, which have been introduced to prove termination of polygraphs [18]. Intuitively, one considers that circuits are crossed by electrical currents. Depending on the intensity of the currents that arrive to it, each circuit gate produces some heat. Then one compares circuits according to the total heat each one produces. Building a polygraphic interpretation amounts at fixing how currents are transmitted by each gate and how much heat each one emits.

The current part is called a functorial interpretation. Algebraically, it is similar to a polynomial interpretation of terms and we also use it as an estimation of the size of values, like quasi-interpretations [10]. The heat part is called a differential interpretation and it is specific to the algebraic structure of polygraphs. We use it to bound the number of computation steps remaining before reaching a result. Let us note that the distinction between these two parts makes it possible for polygraphic interpretations to cope with non-simplifying termination proofs, like Thomas Arts and Jürgen Giesl's dependency pairs [2].

However, some new difficulties arise with polygraphs. For example, since duplication and erasure are explicit in our model, we must show how to get rid of them for the interpretation. In our setting, the programmer focuses on computational steps (as opposed to structural steps) for which he has to give an interpretation. From this interpretation, we give a polynomial upper bound on the number of structural steps that will be performed.

In this work, we focus on polynomial-time computable functions or, shorter, FPTIME functions. The reason comes from Stephen Cook's thesis stating that this class corresponds to feasible computable functions. But it is strongly conjectured that the preliminary results developed in this paper can be used for other characterizations. In particular, the current interpretations can be seen as sup-interpretations, following [35]: this means that values have polynomial size.

Coming back to FPTIME, in the field of implicit computational complexity, the notion of stratification has shown to be a fundamental tool of the discipline. This has been developed by Daniel Leivant and Jean-Yves Marion [31, 32] and by Stephen Bellantoni and Stephen Cook [6] to delineate FPTIME. Other characterizations include Neil Jones' "Life without cons" WHILE programs [25] and Karl-Heinz Niggl and Henning Wunderlich's characterization of imperative programs [38]. There is also a logical approach to implicit computational complexity, based on a linear type discipline, in the seminal work of Jean-Yves Girard on light linear logic [16], Yves Lafont on soft linear logic [28] or Patrick Baillot and Kazushige Terui [5].

The second part of this document is devoted to general results about polygraphic interpretations of polygraphs. There, we explore the pieces of information they can give us about size issues. Then, in the third part, we apply these results to polygraphic programs. In particular, we identify a subclass **P** of these objects that compute exactly the functions that can be computed in polynomial-time by a Turing machine, or FPTIME functions for short.

General notations. Throughout this document, we use several notations that we prefer to group here for easier further reference.

If X is a set and p is a natural number, we denote by X^p the cartesian product of p copies of X. If X is an ordered set, we equip X^p with the product order, which is defined by $(x_1, \ldots, x_p) \leq (y_1, \ldots, y_p)$ whenever $x_i \leq y_i$ holds for every $i \in \{1, \ldots, p\}$.

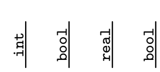
If $f : X \to X'$ and $g : Y \to Y'$ are maps, then $f \times g$ denotes the product map from $X \times X'$ to $Y \times Y'$. Let $f, g : X \to Y$ be two maps. If Y is equipped with a binary relation $\lhd$, then one compares f and g pointwise, which means that $f \lhd g$ holds when, for every $x \in X$, one has $f(x) \lhd g(x)$ in Y. Similarly, if Y is equipped with a binary operation $\diamond$, then one defines $f \diamond g$ as the map from X to Y sending each x of X to the element $f(x) \diamond g(x)$ in Y.

The sets $\mathbb{N}$ of natural numbers and $\mathbb{Z}$ of integers are always assumed to be equipped with their natural order. For every n in $\mathbb{N}$, we denote by μ_n the maximum map $\max\{x_1, \ldots, x_n\}$ and by $\mathbb{N}[x_1, \ldots, x_n]$ the set of polynomials over n variables and with coefficients in $\mathbb{N}$. If $f : \mathbb{N}^m \to \mathbb{N}^n$ is a map and if $k \in \mathbb{N}$, one denotes by kf the map sending $(x_1, \ldots, x_m)$ to $(ky_1, \ldots, ky_n)$, if $(y_1, \ldots, y_n)$ is $f(x_1, \ldots, x_m)$.

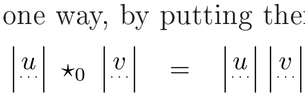
1. A COMPUTATIONAL MODEL BASED ON POLYGRAPHS

1.1. **A first glance at polygraphs.** On a first approach, one can consider polygraphs as rewriting systems on algebraic circuits, made of:

Types. They are the wires, called 1-cells. Each one conveys information of some elementary type. To represent product types, one uses several wires, in parallel, calling such a construction a 1-path. For example, the following 1-path represents the type of quadruples made of an integer, a boolean, a real number and a boolean:

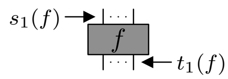


The 1-paths can be composed in one way, by putting them in parallel:

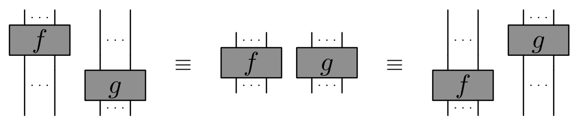


Operations. They are represented by circuits, called 2-paths. The gates used to build them are called 2-cells. The 2-paths can be composed in two ways, either by juxtaposition (parallel composition) or by connection (sequential composition):

Each 2-path (or 2-cell) has a finite number of typed inputs, a 1-path called its 1-source, and a finite number of typed outputs, a 1-path called its 1-target:



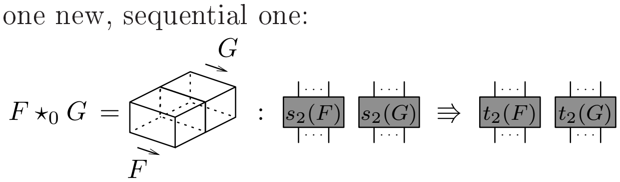
Several constructions represent the same operation. In particular, wires can be stretched or contracted, provided one does not cross them or break them. This can be written either graphically or algebraically:

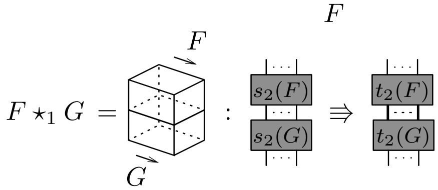


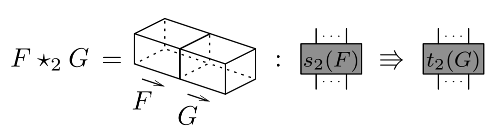
$$\big(f \star_0 s_1(g)\big) \star_1 \big(t_1(f) \star_0 g\big) \quad \equiv \quad f \star_0 g \quad \equiv \quad \big(s_1(f) \star_0 g\big) \star_1 \big(f \star_0 t_1(g)\big).$$

Computations. They are rewriting paths, called 3-paths, transforming a given 2-path, called its 2-source, into another one, called its 2-target. The 3-paths are generated by local rewriting rules, called 3-cells. The 2-source and the 2-target of a 3-cell or 3-path are required to have the same input and output, i.e., the same 1-source and the same 1-target. A 3-path is represented either as a reduction on 2-paths or as a genuine 3-dimensional object:

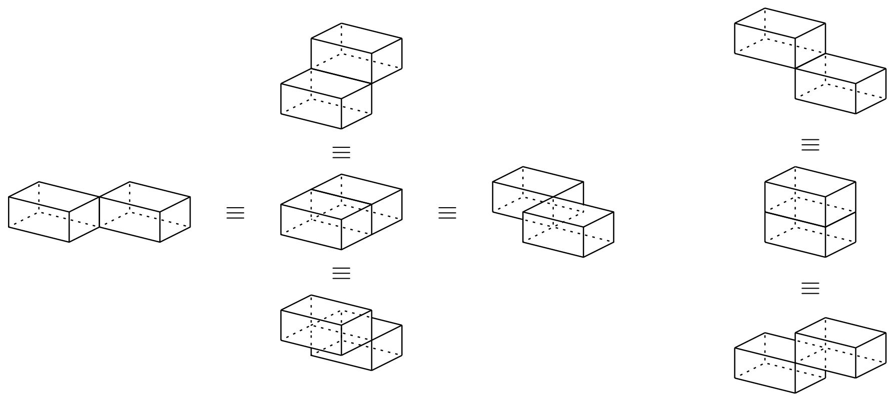
The 3-paths can be composed in three ways, two parallel ones coming from the structure of the 2-paths, plus one new, sequential one:







The 3-paths are identified modulo relations that include topological moves such as:



These graphical relations have an algebraic version given, for $0 \leq i < j \leq 2$, by:

$$\big(F \star_i s_j(G)\big) \star_j \big(t_j(F) \star_i G\big) \quad \equiv \quad F \star_i G \quad \equiv \quad \big(s_j(F) \star_i G\big) \star_j \big(F \star_i t_j(G)\big).$$

So far, we have described a special case of 3-polygraphs. A n-polygraph is a similar object, made of cells, paths, sources, targets and compositions in all dimensions up to n.

Remark 1.1. Polygraphs provide a uniform, algebraic and graphical description of objects coming from different domains: abstract, string and term rewriting systems [27, 17, 18], abstract algebraic structures [12, 17, 33], Feynman and Penrose diagrams [4], braids, knots and tangle diagrams equipped with Reidemeister moves [1, 17], Petri nets [20] and propositional proofs of classical and linear logics [19].

1.2. **Polygraphs.** On a first reading, one can skip the formal definition of polygraph and just keep in mind the graphical introduction. We define n-polygraphs by induction on the dimension n: given a definition of $(n-1)$-polygraphs, we define a n-polygraph as a base $(n-1)$-polygraph extended with a set of n-cells. Let us initiate the induction with 0-polygraphs and 1-polygraphs.

Definition 1.2. A *0-polygraph* is a set $\mathcal{P}$. Its *0-cells* and *0-paths* are its elements.

Definition 1.3. A *1-polygraph* is a data $\mathcal{P} = (\mathcal{B}, \mathcal{P}_1, s, t)$ made of a 0-polygraph $\mathcal{B}$, a set $\mathcal{P}_1$ and two maps s and t from $\mathcal{P}_1$ to $\mathcal{B}$. The *0-cells* and *0-paths* of $\mathcal{P}$ are the ones of $\mathcal{B}$. Its *1-cells* are the elements of $\mathcal{P}_1$. One inductively defines the set $\langle \mathcal{P}_1 \rangle$ of *1-paths* of $\mathcal{P}$, together with the *0-source map* s_0 and the *0-target map* t_0, both from 1-paths to 0-paths, as follows:

- Every 0-cell x is a 1-path, with $s_0(x) = t_0(x) = x$.
- Every 1-cell ξ is a 1-path, with $s_0(\xi) = s(\xi)$ and $t_0(\xi) = t(\xi)$.
- If u and v are 1-paths such that $t_0(u) = s_0(v)$, then $u \star_0 v$ is a 1-path called the *0-composition* of u and v. One defines $s_0(u \star_0 v) = s_0(u)$ and $t_0(u \star_0 v) = t_0(v)$.

The 1-paths are identified modulo the following relations:

- Associativity: $(u \star_0 v) \star_0 w = u \star_0 (v \star_0 w)$.
- Local units: $s_0(u) \star_0 u = u = u \star_0 t_0(u)$.

Example 1.4. A graph yields a 1-polygraph, with vertices as 0-cells and arrows as 1-cells. The 1-paths are the paths in the graph.

Example 1.5. A set X can be seen as a 1-polygraph, with one 0-cell and itself as set of 1-cells: in that case, the set $\langle X\rangle$ of 1-paths is exactly the free monoid generated by X or, equivalently, the set of words over the alphabet X.

Example 1.6. An abstract rewriting system is a binary relation R over a set X. Such an object yields a 1-polygraph $\mathcal{P}$ with $\mathcal{P}_0 = X$, $\mathcal{P}_1 = R$, $s_0(x,y) = x$ and $t_0(x,y) = y$. Then, the 1-paths of this 1-polygraph are in bijective correspondence with the rewriting paths generated by (X,R).

Now, let us fix a natural number $n \geq 2$ and assume that one has defined what a $(n-1)$-polygraph $\mathcal{P}$ is, how one builds its sets $\mathcal{P}_k$ of k-cells and $\langle\mathcal{P}_k\rangle$ of k-paths, $k \in \{0,\ldots,n\}$, and its j-source map s_j and j-target map t_j from $\langle\mathcal{P}_k\rangle$ to $\langle\mathcal{P}_j\rangle$, $j \in \{0,\ldots,k-1\}$.

Definition 1.7. An *n-polygraph* is a data $\mathcal{P} = (\mathcal{B}, \mathcal{P}_n, s, t)$ made of an $(n-1)$-polygraph $\mathcal{B}$, a set $\mathcal{P}_n$ and two maps s and t from $\mathcal{P}_n$ to $\langle\mathcal{B}_{n-1}\rangle$, such that the *globular relations* hold:

$$s_{n-2} \circ s = s_{n-2} \circ t \qquad \text{and} \qquad t_{n-2} \circ s = t_{n-2} \circ t.$$

For every k in $\{0,\ldots,n-1\}$, the *k-cells* and *k-paths* of $\mathcal{P}$ are the ones of $\mathcal{B}$. The *n-cells* of $\mathcal{P}$ are the elements of $\mathcal{P}_n$. One inductively defines the set $\langle\mathcal{P}_n\rangle$ of *n-paths* of $\mathcal{P}$, the *$(n-1)$-source map* s_{n-1}, the *$(n-1)$-target map* t_{n-1} and, for every $k \in \{0,\ldots,n-2\}$, extensions to n-paths of the k-source map s_k and the k-target map t_k of $\mathcal{B}$:

- For every $k \in \{0,\ldots,n-1\}$, every k-cell ξ is an n-path, with $s_{n-1}(\xi) = t_{n-1}(\xi) = \xi$. Values of other source and target maps do not change.
- Every n-cell φ is an n-path, with $s_{n-1}(\varphi) = s(\varphi)$ and $t_{n-1}(\varphi) = t(\varphi)$. If $k \in \{0,\ldots,n-2\}$, then s_k and t_k are respectively extended by $s_k(\varphi) = s_k \circ s_{n-1}(\varphi)$ and by $t_k(\varphi) = t_k \circ t_{n-1}(\varphi)$.
- If $k \in \{0,\ldots,n-1\}$ and if f and g are n-paths such that $t_k(f) = s_k(g)$ holds, then $f \star_k g$ is an n-path called the *k-composition* of f and g. For $j \in \{0,\ldots,n-2\}$, one defines:

$$s_j(f \star_k g) = \begin{cases} s_j(f) & \text{if } j \leq k \\ s_j(f) \star_k s_j(g) & \text{if } j > k \end{cases} \qquad \text{and} \qquad t_j(f \star_k g) = \begin{cases} t_j(g) & \text{if } j \leq k \\ t_j(f) \star_k t_j(g) & \text{if } j > k. \end{cases}$$

One does not distinguish two n-paths that only differ by the following relations:

- Associativity: $(f \star_k g) \star_k h = f \star_k (g \star_k h)$, for $0 \leq k \leq n-1$.
- Local units: $s_k(f) \star_k f = f = f \star_k t_k(f)$, for $0 \leq k \leq n-1$.
- Exchange: $(f_1 \star_j f_2) \star_k (g_1 \star_j g_2) = (f_1 \star_k g_1) \star_j (f_2 \star_k g_2)$, for $0 \leq j < k \leq n-1$.

Example 1.8. Let us consider a word rewriting system (X,R), made of set X and a binary relation R over $\langle X\rangle$. From it, one builds a 2-polygraph $\mathcal{P}$ with one 0-cell, $\mathcal{P}_1 = X$, $\mathcal{P}_2 = R$, $s_1(u,v) = u$ and $t_1(u,v) = v$. There is a bijection between the 2-paths of $\mathcal{P}$ and the rewriting paths generated by (X,R), considered modulo the commutation squares between two non-overlapping rule applications. Moreover the circuit-like pictures provide graphical representations for word rewriting: wires are letters, gates are applications of rewriting rules and circuits are traces of computations.

Example 1.9. Term rewriting systems generate 3-polygraphs, as explained by Albert Burroni [12], Yves Lafont [27] and the second author [18, 19]. The polygraphic programs one considers here are light versions of these [21].

Example 1.10. Petri nets correspond exactly to 3-polygraphs with one 0-cell and no 1-cell: one identifies places with 2-cells and transitions with 3-cells [20].

Definition 1.11. Let us fix a natural number n and an n-polygraph $\mathcal{P}$. The polygraph $\mathcal{P}$ is *finite* when it has a finite number of cells in every dimension. A family X of n-cells of $\mathcal{P}$ can be seen as an n-polygraph with the same cells as $\mathcal{P}$ up to dimension $n-1$.

If $0 \leq j < k \leq n$, two k-paths f and g are *j-composable* when $t_j(f) = s_j(g)$. They are *j-parallel* when $s_j(f) = s_j(g)$ and $t_j(f) = t_j(g)$. When $j = k-1$, one simply says *composable* and *parallel*. Similarly, the $(k-1)$-source and $(k-1)$-target of a k-path are simply called its *source* and *target*.

If $0 \leq k \leq n$, given a subset X of $\mathcal{P}_k$ and a k-path f, the *size of f with respect to X* is the natural number denoted by $||f||_X$ and defined as follows, by structural induction on f:

$$||f||_X \;=\; \begin{cases} 0 & \text{if } f \text{ is a cell and } f \notin X, \\ 1 & \text{if } f \in X, \\ ||g||_X + ||h||_X & \text{if } f = g \star_j h, \text{ for some } 0 \leq j < k. \end{cases}$$

When X is reduced to one cell φ, one writes $||f||_\varphi$ instead of $||f||_{\{\varphi\}}$. The *size of f* is its size with respect to $\mathcal{P}_k$, simply written $||f||$. A k-path is *degenerate* when it has size 0 and *elementary* when its size is 1.

Remark 1.12. One must check that the definition of the size of a k-path (with respect to a set of k-cells X) is correct. This is done by computing this map on both sides of the relations of associativity, local units and exchange and ensuring that both results are equal.

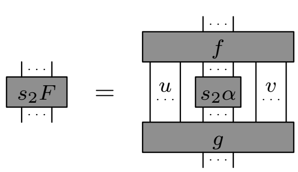
One proves that any non-degenerate k-path f of size p can be written

$$f \;=\; f_1 \star_{k-1} \cdots \star_{k-1} f_p,$$

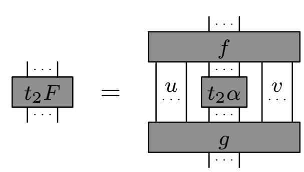
where each f_i is an elementary k-path. Moreover, if $k \geq 1$, then any elementary k-path f can be written as follows:

$$f \;=\; g_k \star_{k-1} \big(g_{k-1} \star_{k-2} \cdots \star_1 (g_1 \star_0 \varphi \star_0 h_1) \star_1 \cdots \star_{k-2} h_{k-1}\big) \star_{k-1} h_k,$$

where φ is a uniquely defined k-cell, while g_j and h_j are j-paths, for every $j \in \{1, \ldots, k\}$. For example, any elementary 3-path F can be decomposed as $F = f \star_1 (u \star_0 \alpha \star_0 v) \star_1 g$, where α is a uniquely determined 3-cell, f and g are 2-paths, u and v are 1-paths. As a consequence:



$$s_2F \;=\; f \star_1 (u \star_0 s_2\alpha \star_0 v) \star_1 g$$



$$t_2F \;=\; f \star_1 (u \star_0 t_2\alpha \star_0 v) \star_1 g$$

In order to study the computational properties of polygraphs, we use notions of higher-dimensional rewriting theory [18] that, in turn, make reference to abstract rewriting ones [3].

Definition 1.13. The *reduction graph* associated to an n-polygraph $\mathcal{P}$ is the graph with $(n-1)$-paths of $\mathcal{P}$ as objects and elementary n-paths of $\mathcal{P}$ as arrows. Rewriting notions of *normal forms*, *termination*, *(local) confluence*, *convergence*, etc. are defined on $\mathcal{P}$ by taking back the ones of its reduction graph.

Remark 1.14. One can check that, given two parallel $(n-1)$-paths f and g in an n-polygraph $\mathcal{P}$, there exists a path from f to g in the reduction graph of $\mathcal{P}$ if and only if there exists a non-degenerate n-path F with source f and target g in $\mathcal{P}$.

In what follows, we focus on 3-polygraphs and introduce some special notions and notations for them.

Definition 1.15. Let $\mathcal{P}$ be a 3-polygraph. The fact that f is a k-path of $\mathcal{P}$ with source x and target y is denoted by $f : x \to y$ when $k = 1$, by $f : x \Rightarrow y$ when $k = 2$, by $f : x \Rrightarrow y$ when $k = 3$. If f is a k-path of $\mathcal{P}$ and X a family of k-cells then, instead of $||f||_X$, one writes $|f|_X$ when $k = 1$ and $|||f|||_X$ when $k = 3$. When $f : x \Rightarrow y$, then $|x|$, $|y|$ and $(|x|, |y|)$ are respectively called the *arity*, the *coarity* and the *valence* of f.

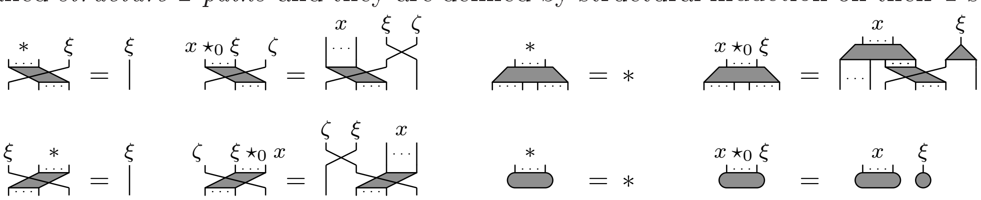
1.3. **Polygraphic programs.**

Definition 1.16. A *polygraphic program* is a finite 3-polygraph $\mathcal{P}$ with one 0-cell, thereafter denoted by $*$, and such that its sets of 2-cells and of 3-cells respectively decompose into $\mathcal{P}_2 = \mathcal{P}_2^S \amalg \mathcal{P}_2^C \amalg \mathcal{P}_2^F$ and $\mathcal{P}_3 = \mathcal{P}_3^S \amalg \mathcal{P}_3^R$, with the following conditions:

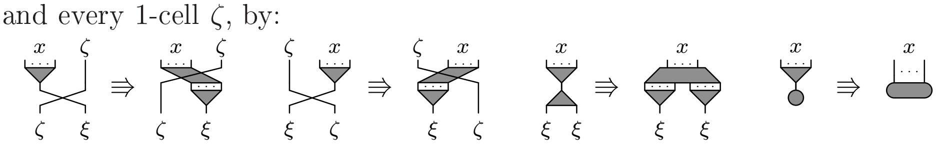
- The set $\mathcal{P}_2^S$ is made of the following elements, called *structure 2-cells*, where ξ and ζ range over the set of 1-cells of $\mathcal{P}$:

$$ {}_{\xi,\zeta} : \xi \star_0 \zeta \Rightarrow \zeta \star_0 \xi, \qquad {}_{\xi} : \xi \Rightarrow \xi \star_0 \xi, \qquad {}_{\xi} : \xi \Rightarrow *. $$

  When the context is clear, one simply writes , and . The following elements of $\langle \mathcal{P}_2^S \rangle$ are called *structure* 2*-paths* and they are defined by structural induction on their 1-source:



- The set $\mathcal{P}_2^C$ is made of 2-cells with coarity 1, i.e., of the shape , called *constructor 2-cells*.
- The elements of $\mathcal{P}_2^F$ are called *function 2-cells*.
- The elements of $\mathcal{P}_3^S$, called *structure 3-cells*, are defined, for every constructor 2-cell : $x \Rightarrow \xi$ and every 1-cell ζ, by:



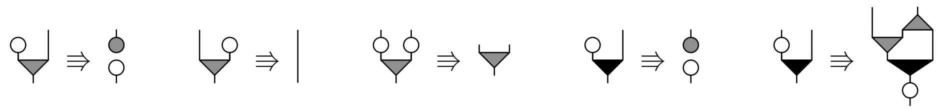
- The elements of $\mathcal{P}_3^R$ are called *computation 3-cells* and each one has a 2-source of the shape $t \star_1$, with $t \in \langle \mathcal{P}_2^C \rangle$ and $\in \mathcal{P}_2^F$.

Remark 1.17. In this study, we have decided to split structure cells from computation cells. From a traditional programming perspective, permutations, duplications and erasers are given for free in the syntax. With polygraphs, this is not the case. However, by putting these operations in a "special" sublayer, we show that the programmer has not to bother with structure cells: one can stay at the top-level, letting the sublevel work on its own.

Example 1.18. The following polygraphic program $\mathcal{D}$ computes the euclidean division on natural numbers (we formally define what this means later):

(1) It has one 1-cell `n`, standing for the type of natural numbers.

(2) Apart from the fixed three structure 2-cells, it has two constructor 2-cells, $: * \Rightarrow \mathtt{n}$ for zero and $: \mathtt{n} \Rightarrow \mathtt{n}$ for the successor operation, and two function 2-cells, $: \mathtt{n} \star_0 \mathtt{n} \Rightarrow \mathtt{n}$ for the minus function and $: \mathtt{n} \star_0 \mathtt{n} \Rightarrow \mathtt{n}$ for the division function.
(3) Its 3-cells are made of eight structure 3-cells, plus the following five computation 3-cells:



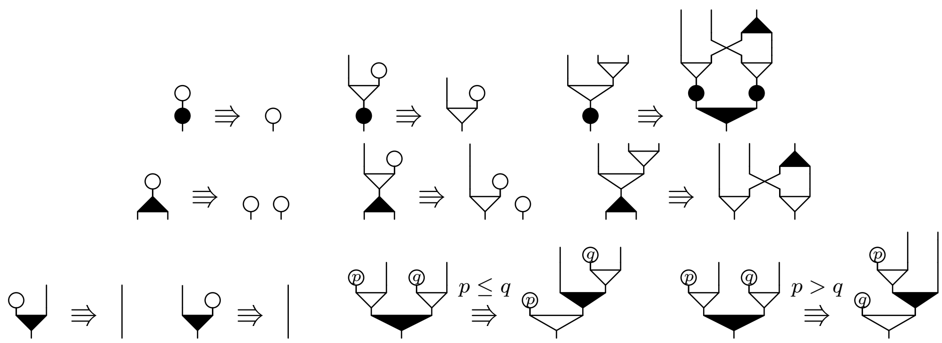
Example 1.19. The following program $\mathcal{F}$ computes the *fusion sort* function on lists of natural numbers lower or equal than some constant $N \in \mathbb{N}$:
(1) Its 1-cells are n, for natural numbers, and l, for lists of natural numbers.
(2) Its 2-cells are made of eight structure 2-cells, plus:
  (a) Constructor 2-cells, for the natural numbers 0, ..., N, the empty list and the list constructor:

$$\left(: * \Rightarrow \mathtt{n}\right)_{0 \leq n \leq N}, \qquad : * \Rightarrow \mathtt{l}, \qquad : \mathtt{n} \star_0 \mathtt{l} \Rightarrow \mathtt{l}.$$

  (b) Function 2-cells, respectively for the main sort and the two auxiliary split and merge:

$$: \mathtt{l} \Rightarrow \mathtt{l}, \qquad : \mathtt{l} \Rightarrow \mathtt{l} *_0 \mathtt{l}, \qquad : \mathtt{l} *_0 \mathtt{l} \Rightarrow \mathtt{l}.$$

(3) Its 3-cells are made of $6N + 18$ structure 3-cells, plus $N^2 + 2N + 8$ computation 3-cells:



Remark 1.20. One may object that sorting lists when the a priori bound N is known can be performed in a linear number of steps: one reads the list and counts the number of occurrences of each element, then produces the sorted list from this information. Nevertheless, the presented algorithm (up to the test $\leq$ on the natural numbers p and q) really mimics the "mechanics" of the fusion sort algorithm and, actually, we rediscover the complexity bound as given by Yiannis Moschovakis [36].

Why don't we internalize the comparison of numbers within the polygraphic program? This comes from the fact that the *if-then-else* construction implicitly involves an evaluation strategy: one first computes the test argument then, depending on this result, one computes *exactly one* of the other two arguments. As defined here, polygraphs algebraically describe the computation steps, but not the evaluation strategy. We let such a task for further research.

1.4. **Semantics of polygraphic programs.** One defines an interpretation $\llbracket\cdot\rrbracket$ of the elements of a polygraphic program into sets and maps, then one uses it to define the notion of function computed by such a program.

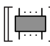
Definition 1.21. Let $\mathcal{P}$ be a polygraphic program. For a 1-path u, a *value of type* u is a 2-path in $\left\langle \mathcal{P}_2^C \right\rangle$ with source $*$ and target u; their set is denoted by $\llbracket u \rrbracket$. Given a 2-path $f : u \Rightarrow v$, one denotes by $\llbracket f \rrbracket$ the (partial) map from $\llbracket u \rrbracket$ to $\llbracket v \rrbracket$ defined as follows: if t is a value of type u and if $t \star_1 f$ has a unique normal form t' that is a value (of type v), then $\llbracket f \rrbracket(t)$ is t'; otherwise f is undefined on t.

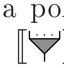
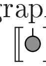
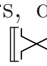
Among the following properties, the one for degenerate 2-paths explains the fact that $\llbracket u \rrbracket$ has two meanings: it is either the set of values of type u or the identity of this set.

Proposition 1.22. *Let $\mathcal{P}$ be a polygraphic program. The following properties hold on* 1-*paths:*
- *The set $\llbracket * \rrbracket$ is reduced to the* 0-*cell* $*$.
- *For every u and v, one has $\llbracket u \star_0 v \rrbracket = \llbracket u \rrbracket \times \llbracket v \rrbracket$.*

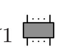
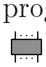
The following properties hold on 2-*paths:*
- *If u is degenerate then it is sent by $\llbracket\cdot\rrbracket$ to the identity of the set $\llbracket u \rrbracket$.*
- *For every f and g, one has $\llbracket f \star_0 g \rrbracket = \llbracket f \rrbracket \times \llbracket g \rrbracket$.*
- *If f and g are composable, then $\llbracket f \star_1 g \rrbracket = \llbracket g \rrbracket \circ \llbracket f \rrbracket$ holds.*

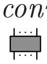
Finally, for every 3-*path F, the equality $\llbracket s_2 F \rrbracket = \llbracket t_2 F \rrbracket$ holds.* □

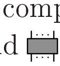
Definition 1.23. Let $\mathcal{P}$ be a polygraphic program. Let u, v be 1-paths and let f be a (partial) map from $\llbracket u \rrbracket$ to $\llbracket v \rrbracket$. One says that $\mathcal{P}$ *computes* f when there exists a 2-cell  such that $\llbracket$  $\rrbracket = f$.

Example 1.24. In a polygraphic program $\mathcal{P}$, every constructor 2-cell  with arity n satisfies the equality $\llbracket$  $\rrbracket (t_1, \ldots, t_n) = (t_1 \star_0 \cdots \star_0 t_n) \star_1$ . Since the right member is always a normal form, one can identify values of coarity 1 with the closed terms of a term algebra. Moreover, the polygraphic program $\mathcal{P}$ computes erasers, duplications and permutations on these terms, since $\llbracket$  $\rrbracket (t) = *$, $\llbracket$  $\rrbracket (t) = (t, t)$ and $\llbracket$  $\rrbracket (t, t') = (t', t)$ hold.

Thus, every polygraphic program computes one total map for each of its structure and constructor 2-cells. We give sufficient conditions to ensure that this is also the case on function 2-cells.

Definition 1.25. A polygraphic program $\mathcal{P}$ is *complete* if every 2-path of the form $t \star_1$  is reducible when t is a value and  is a function 2-cell.

Proposition 1.26. *Let $\mathcal{P}$ be a convergent and complete polygraphic program. Then, for every structure or function* 2-*cell*  $: u \Rightarrow v$, *the map* $\llbracket$  $\rrbracket : \llbracket u \rrbracket \to \llbracket v \rrbracket$ *is total.*

Proof. We start by recalling that the structure 3-cells, alone, are convergent [18, 19]. Furthermore, they are orthogonal to the computation 3-cells and every 2-path of the shape $t \star_1$  is reducible when t is a value and  is a structure 2-cell. Hence, as a polygraph, $\mathcal{P}$ is convergent and the 2-paths $* \Rightarrow x$ that are in normal form are exactly the values of type x. □

Example 1.27. Let us check that the polygraphic program $\mathcal{D}$ computes euclidean division. The set $[\![\mathtt{n}]\!]$ is equipotent to the set $\mathbb{N}$ of natural numbers through the bijection $\underline{0} = \circ$ and $\underline{n+1} = \underline{n} \star_1 \phi$. This polygraphic program is weakly orthogonal, hence locally confluent, and complete. We will also see later that it terminates. Thus it computes two maps from $[\![\mathtt{n} \star_0 \mathtt{n}]\!] \simeq \mathbb{N}^2$ to $[\![\mathtt{n}]\!] \simeq \mathbb{N}$, one for $\triangledown$ and one for $\blacktriangledown$. By induction on the arguments, one gets:

$$[\![\triangledown]\!]\,(\underline{m},\underline{n}) = \underline{\max\{0, m-n\}} \qquad \text{and} \qquad [\![\blacktriangledown]\!]\,(\underline{m},\underline{n}) = \underline{\lfloor m/(n+1) \rfloor}.$$

Example 1.28. In the polygraphic program $\mathcal{F}$, one has $[\![\mathtt{n}]\!] \simeq \{0,\ldots,N\}$ and $[\![\mathtt{l}]\!] \simeq \langle 0,\ldots,N \rangle$, thanks to the bijective correspondences $\underline{n} = ⓝ$, $\underline{[\,]} = \circ$ and $\underline{x :: l} = (\underline{x} \star_0 \underline{l}) \star_1 \triangledown$. This polygraphic program is weakly orthogonal, hence locally confluent, and complete. It is also terminating, as we shall see later. Thus, it computes one map for each of $\bullet$, $\blacktriangle$ and $\blacktriangledown$. For example, the map $[\![\bullet]\!]$ takes a list of natural numbers as input and returns the corresponding ordered list. Figure 1 gives an example of computation generated by this program, with explanations following.

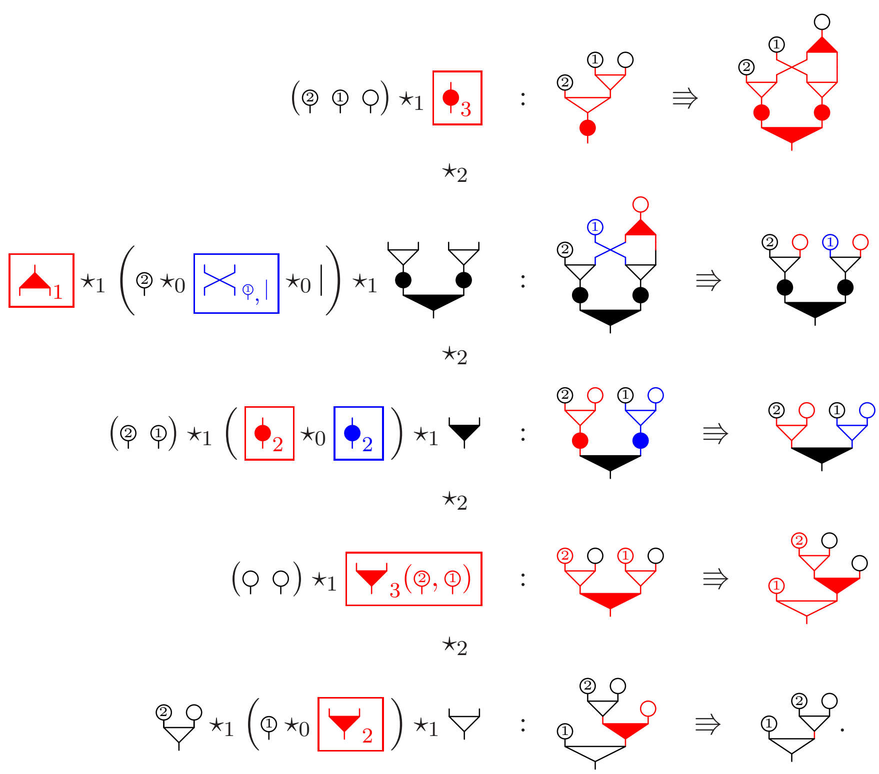


Figure 1: Normalizing 3-path in a polygraphic program

Let us consider the list $[2;1]$ of natural numbers and apply the fusion sort function $\bullet$ on it. The list is coded by the following value:

$$\underline{[2;1]} \;=\; (①\star_0 \circ) \star_1 (② \star_0 \triangledown) \star_1 \triangledown \;=\; ②①\circ\triangledown\triangledown.$$

The value $[\![\phi]\!]\left(\underline{[2;1]}\right)$ is, by definition, the unique normal form of the 2-path $\underline{[2;1]} \star_1 \phi$. Figure 1 presents a normalizing 3-path, obtained by $\star_2$-composition of smaller 3-paths, where we have given self-explanatory "names" to the involved 3-cells, without further explanations. After computation, one gets the expected $[\![\phi]\!]\left(\underline{[2;1]}\right) = \underline{[1;2]}$ as the target of this 3-path.

1.5. **Polygraphic programs are Turing-complete.** This completeness result is not a surprising one. Indeed, one could argue, for instance, that polygraphic programs simulate term rewriting systems, a Turing-complete model of computation. Our proof, similar to the one concerning interaction nets [26], prepares for the encoding of Turing machines with clocks, used for Theorem 3.27.

Definition 1.29. A *Turing machine* is a family $\mathcal{M} = (\Sigma, Q, q_0, q_f, \delta)$ made of:

- A finite set Σ, called the *alphabet*; one denotes by $\overline{\Sigma}$ its extension with a new element, denoted by $\sharp$ and called the *blank character*.
- A finite set Q, whose elements are called *states*, two distinguished elements q_0, the *initial state*, and q_f, the *final state*.
- A map $\delta : (Q - \{q_f\}) \times \overline{\Sigma} \to Q \times \overline{\Sigma} \times \{L, R\}$, called the *transition function*, where $\{L, R\}$ is any set with two elements.

A *configuration* of $\mathcal{M}$ is an element (q, a, w_l, w_r) of the product set $Q \times \overline{\Sigma} \times \langle\overline{\Sigma}\rangle \times \langle\overline{\Sigma}\rangle$: here q is the current state of the machine, a is the currently read symbol, w_l is the word at the left-hand side of a and w_r is the word at the right-hand side of a. For further convenience, the word w_l is written in reverse order, so that its first letter is the one that is immediately at the left of a.

The *transition relation* of $\mathcal{M}$ is the binary relation denoted by $\to_{\mathcal{M}}$ and defined on the set of configurations of $\mathcal{M}$ as follows, where e denotes the neutral element of $\langle\Sigma\rangle$:

- If $\delta(q_1, a) = (q_2, c, L)$ then $\begin{cases} (q_1, a, e, w_r) & \to_{\mathcal{M}} \quad (q_2, \sharp, e, cw_r)\,, \\ (q_1, a, bw_l, w_r) & \to_{\mathcal{M}} \quad (q_2, b, w_l, cw_r)\,. \end{cases}$
- If $\delta(q_1, a) = (q_2, c, R)$ then $\begin{cases} (q_1, a, w_l, e) & \to_{\mathcal{M}} \quad (q_2, \sharp, cw_l, e)\,, \\ (q_1, a, w_l, bw_r) & \to_{\mathcal{M}} \quad (q_2, b, cw_l, w_r)\,. \end{cases}$

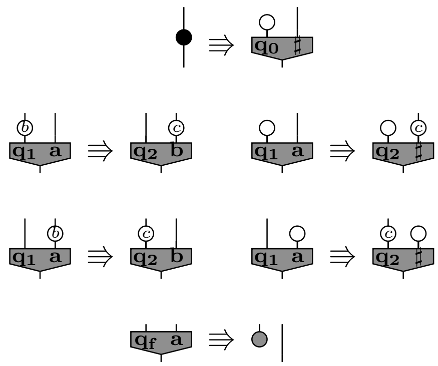
One denotes by $\to^*_{\mathcal{M}}$ the reflexive and transitive closure of $\to_{\mathcal{M}}$. Let $f : \langle\Sigma\rangle \to \langle\Sigma\rangle$ be a map. One says that $\mathcal{M}$ *computes* f when, for any w in $\langle\Sigma\rangle$, there exists a configuration of the shape $(q_f, a, v, f(w))$ such that $(q_0, \sharp, e, w) \to^*_{\mathcal{M}} (q_f, a, v, f(w))$ holds (in that case, this final configuration is unique).

Theorem 1.30. *Polygraphic programs form a Turing-complete model of computation.*

Proof. We fix a Turing machine $\mathcal{M} = (\Sigma, Q, q_0, q_f, \delta)$ and a map f computed by $\mathcal{M}$. From this Turing machine, we build the following polygraphic program $\mathcal{P}(\mathcal{M})$:

(1) It has one 1-cell $\mathtt{w}$, standing for the type of words over Σ.
(2) Apart from the three structure 2-cells, its 2-cells consist of:
   (a) Constructor 2-cells: $\circ : * \Rightarrow \mathtt{w}$, for the empty word, plus one $ⓐ : \mathtt{w} \Rightarrow \mathtt{w}$ for each a in Σ.
   (b) Function 2-cells: $\phi : \mathtt{w} \Rightarrow \mathtt{w}$, for the map f, plus one $\boxed{q\ a} : \mathtt{w} \star_0 \mathtt{w} \Rightarrow \mathtt{w}$ for each pair (q, a) in $Q \times \bar{\Sigma}$, for the behaviour of the Turing machine.

(3) Its 3-cells are the structure ones, plus the following computation 3-cells – the first one initializes the computation, the four subsequent families simulate the transitions of the Turing machine and the final cell starts the computation of the result:



both when $\delta(q_1, a) = (q_2, c, L)$

both when $\delta(q_1, a) = (q_2, c, R)$

One checks that $[\![\mathtt{W}]\!] \simeq \langle\Sigma\rangle$ through $\underline{e} = $ (circle) and $\underline{aw} = \underline{w} \star_1$ (a). Then, to every configuration (q, a, w_l, w_r), one associates the 2-path $\underline{(q, a, w_l, w_r)} = \big(\underline{w_l} \star_0 \underline{w_r}\big) \star_1$ (q a). The four cases in the definition of the transition relation of $\mathcal{M}$ are in one-to-one correspondence with the four middle families of 3-cells of the polygraph $\mathcal{P}(\mathcal{M})$. Hence the following equivalence holds:

$$(q, a, w_l, w_r) \rightarrow^*_{\mathcal{M}} (q', a', w'_l, w'_r) \quad \text{if and only if} \quad \underline{(q, a, w_l, w_r)} \Rrightarrow \underline{(q', a', w'_l, w'_r)}.$$

Finally, let us fix a w in $\langle\Sigma\rangle$. Since $\mathcal{M}$ computes f, there exists a unique configuration $(q_f, a, v, f(w))$, such that $(q_0, \sharp, e, w) \rightarrow^*_{\mathcal{M}} (q_f, a, v, f(w))$ holds. As a consequence, $\underline{w} \star_1$ (black dot) has a unique normal form, so that the following equalities hold, yielding $[\![$(black dot)$]\!] = f$:

$$[\![\bullet]\!]\,(\underline{w}) \;=\; [\![q_0\ \sharp]\!]\,(\circ \star_0 \underline{w}) \;=\; [\![q_f\ a]\!]\,\Big(\underline{v} \star_0 \underline{f(w)}\Big) \;=\; \underline{f(w)}. \qquad \square$$

2. Polygraphic interpretations

Here, we present general results about information that can be recovered from functorial and differential interpretations of 3-polygraphs.

2.1. Functorial interpretations.

Definition 2.1. A *functorial interpretation* of a 3-polygraph $\mathcal{P}$ is a pair $\varphi = (\varphi_1, \varphi_2)$ consisting of:

(1) a map φ_1 sending every 1-path u of size n to a non-empty part of $(\mathbb{N} - \{0\})^n$;
(2) a map φ_2 sending every 2-path $f : u \Rightarrow v$ to a monotone map from $\varphi_1(u)$ to $\varphi_1(v)$.

The following equalities, called *functorial relations*, must be satisfied:

- if u is a degenerate 2-path, then $\varphi_2(u)$ is the identity of $\varphi_1(u)$;
- if u and v are 0-composable 1-paths, then $\varphi_1(u \star_0 v) = \varphi_1(u) \times \varphi_1(v)$ holds;
- if f and g are 0-composable 2-paths, then $\varphi_2(f \star_0 g) = \varphi_2(f) \times \varphi_2(g)$ holds;
- if f and g are 1-composable 2-paths, then $\varphi_2(f \star_1 g) = \varphi_2(g) \circ \varphi_2(f)$ holds.

One simply writes φ for both φ_1 and φ_2. Intuitively, for every 2-cell (gray box), the map φ((gray box)) tells us how (gray box), seen as a circuit gate, transmits currents downwards. In practice, one computes the value of a current interpretation on a 2-path by computing it on the 2-cells it contains and assembling them in an intuitive way. The following result formalizes this fact.

Lemma 2.2. *A functorial interpretation of a 3-polygraph $\mathcal{P}$ is entirely and uniquely defined by its values on the 1-cells and 2-cells of $\mathcal{P}$.*

Proof. Using the functorial relations, one checks that a functorial interpretation takes the same values on both sides of the relations of associativity, local units and exchange on 2-paths: this property comes from the fact that set-theoretic maps satisfy these same relations. Then the functorial relations give the values of a current interpretation on 2-paths of size $n+1$ from its values on 2-paths of size $k \leq n$. □

A direct consequence of Lemma 2.2 is that, when one wants to introduce a functorial interpretation, one only has to give its values on the 1-cells and on the 2-cells.

Example 2.3. Let $\mathcal{P}$ be a polygraphic program with no constructor 2-cell and no function 2-cell. Then, given a non-empty part $\varphi(\xi)$ of $\mathbb{N} - \{0\}$ for every 1-cell ξ, the following values extend φ into a functorial interpretation of $\mathcal{P}$:

$$\varphi\left(\times_{\xi,\zeta}\right)(x,y) \;=\; (y,x) \qquad \text{and} \qquad \varphi\left(▲_{\xi}\right)(x) \;=\; (x,x).$$

Let us note that every functorial interpretation φ must send the 0-cell $*$ to some single-element part of $\mathbb{N} - \{0\}$. Hence, it must assign each $●_{\xi}$ to the only map from $\varphi(\xi)$ to $\varphi(*)$.

Example 2.4. The following values extend the ones of Example 2.3 into a functorial interpretation of the polygraphic program $\mathcal{D}$ of division:

$$\varphi(\mathtt{n}) = \mathbb{N} - \{0\}\,, \qquad \varphi(\underset{\circ}{○}) = 1, \qquad \varphi(\underset{\circ}{○})(x) = x+1,$$
$$\varphi(▼)(x,y) = \varphi(▼)(x,y) = x.$$

Example 2.5. For the polygraphic program $\mathcal{F}$ of fusion sort, we extend the functorial interpretation of Example 2.3 with the following values, where $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ stand for the rounding functions, respectively by excess and by default:

$$\varphi(\mathrm{n}) = \{1\}\,, \qquad \varphi(\mathrm{l}) = 2\mathbb{N}+1, \qquad \varphi(○) = \varphi(○) = 1, \qquad \varphi(▽)(x,y) = x+y+1,$$
$$\varphi(●)(x) = x, \qquad \varphi(▼)(x,y) = x+y-1, \qquad \varphi(▲)(2x+1) = \left(2\cdot\left\lceil\frac{x}{2}\right\rceil+1,\; 2\cdot\left\lfloor\frac{x}{2}\right\rfloor+1\right).$$

Example 2.6. Let $\mathcal{P}$ be a polygraphic program. One denotes by ν the functorial interpretation on the subpolygraph $\langle \mathcal{P}_2^C \rangle$ defined, for every 1-cell ξ, by $\nu(\xi) = \mathbb{N} - \{0\}$ and, for every constructor 2-cell ▼ with arity n, by:

$$\nu(▼)(x_1, \dots, x_n) \;=\; x_1 + \cdots + x_n + 1.$$

One checks that $\nu(t) = ||t||$ holds for every value t with coarity 1. Thus, given values t_1, $\dots$, t_n with coarity 1, the following equality holds in $\mathbb{N}^n$:

$$\nu(t_1 \star_0 \cdots \star_0 t_n) \;=\; \big(\,||t_1||\,, \dots, ||t_n||\,\big).$$

We use the functorial interpretation ν to describe the size of arguments of a function.

Lemma 2.7. *Let φ be a functorial interpretation of a 3-polygraph $\mathcal{P}$. Let f, g, h and k be 2-paths such that $\varphi(f) \leq \varphi(g)$ and $\varphi(h) \leq \varphi(k)$ hold. Then, for every $i \in \{0,1\}$ such that $f \star_i h$ is defined, the inequality $\varphi(f \star_i h) \leq \varphi(g \star_i k)$ is satisfied.*

Proof. One has:

$$\varphi(f \star_0 h) \;=\; \varphi(f) \times \varphi(h) \;\leq\; \varphi(g) \times \varphi(k) \;=\; \varphi(g \star_0 k).$$

Indeed, the two equalities are given by the functorial relations that φ satisfies, while the middle inequality comes from the hypotheses and the fact that one uses a product order. Then one has:

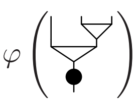
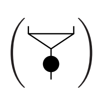
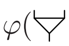
$$\varphi(f \star_1 h) \;=\; \varphi(h) \circ \varphi(f) \;\leq\; \varphi(h) \circ \varphi(g) \;\leq\; \varphi(k) \circ \varphi(g) \;=\; \varphi(g \star_1 k).$$

The equalities come from the functorial relations; the first inequality uses the hypothesis $\varphi(f) \leq \varphi(g)$ and the fact that $\varphi(h)$ is monotone; the second inequality uses $\varphi(h) \leq \varphi(k)$ and the fact that maps are compared pointwise. □

2.2. **Compatible functorial interpretations.**

Definition 2.8. Let φ be a functorial interpretation of a 3-polygraph $\mathcal{P}$. For every 3-cell α of $\mathcal{P}$, one says that φ is *compatible with* α when the inequality $\varphi(s_2\alpha) \geq \varphi(t_2\alpha)$ holds. One says that φ is *compatible* when it is compatible with every 3-cell of $\mathcal{P}$.

Example 2.9. The functorial interpretations given in Examples 2.4 and 2.5 are compatible with all the 3-cells of the corresponding 3-polygraph. We will see later that the values they take on structure 2-cells ensure that they are compatible with all the structure 3-cells. Concerning the computation 3-cells, let us consider, for example, the third one associated to the sort function 2-cell . For the source, one gets:

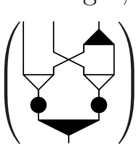
φ  $(1,1,2x+1) \;=\; \varphi$  $\big(1, \varphi($$)(1,2x+1)\big)$

$= \varphi($$) \circ \varphi($$)(1,2x+3)$

$= \varphi($$)(2x+5)$

$= 2x+5.$

Now, for the target, going quicker:

φ  $(1,1,2x+1) \;=\; \varphi($$)\big(2\cdot\lceil x/2\rceil+3,\; 2\cdot\lfloor x/2\rfloor+3\big) \;=\; 2x+5.$

Proposition 2.10. *Let φ be a compatible functorial interpretation of a polygraphic program. Then, for every 3-path F, the inequality $\varphi(s_2F) \geq \varphi(t_2F)$ holds.*

Proof. We proceed by induction on the size of 3-paths. If F is a degenerate 3-path, then $s_2F = t_2F$ holds and, thus, so does $\varphi(s_2F) = \varphi(t_2F)$.

Let us assume that F is an elementary 3-path. Then one decomposes s_2F and t_2F, using a 3-cell α, 2-paths f, g and 1-paths u, v, yielding:

$$\varphi(s_2F) \;=\; \varphi\big(f \star_1 (u \star_0 s_2\alpha \star_0 v) \star_1 g\big) \qquad \text{and} \qquad \varphi(t_2F) \;=\; \varphi\big(f \star_1 (u \star_0 t_2\alpha \star_0 v) \star_1 g\big).$$

The functorial interpretation φ is compatible with α, hence $\varphi(s_2\alpha) \geq \varphi(t_2\alpha)$ holds. Then one applies Lemma 2.7 four times to get $\varphi(s_2F) \geq \varphi(t_2F)$.

Now, let us fix a non-zero natural number N and assume that the property holds for every 3-path of size N. Let us consider a q3-path F of size $N+1$. Then one decomposes F

into $G \star_2 H$ where G is a 3-path of size N and H is an elementary 3-path. One concludes using the induction hypothesis on G and the previous case on H. □

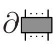
2.3. **Differential interpretations.** In this work, we use differential interpretations as an abstraction of "heats", but also, later, to define the property of conservativeness on "currents". For this reason, we introduce the following abstraction:

Definition 2.11. A *(strictly) ordered commutative monoid* is an ordered set $(M, \preceq)$ equipped with a commutative monoid structure $(+, 0)$ such that $+$ is (strictly) monotone in both arguments.

Example 2.12. Concretely, in what follows, we consider $\mathbb{N}$ equipped with its natural order and either the addition (strict case) or the maximum map (non-strict case), both with 0 as neutral element.

Definition 2.13. Let M be an ordered commutative monoid, let $\mathcal{P}$ be a 3-polygraph and let φ be a functorial interpretation of $\mathcal{P}$. A *differential interpretation of* $\mathcal{P}$ *over* φ *into* M is a map ∂ that sends each 2-path  of $\mathcal{P}$ with 1-source u to a monotone map ∂ from $\varphi(u)$ to M, such that the following conditions, called *differential relations*, are satisfied:

- If u is degenerate then $\partial u = 0$.
- If f and g are 0-composable then $\partial(f \star_0 g)(x, y) = \partial f(x) + \partial g(y)$ holds.
- If f and g are 1-composable then $\partial(f \star_1 g) = \partial f + \partial g \circ \varphi(f)$ holds.

Intuitively, given a 2-cell , the map ∂ tells us how much heat it produces, when seen as a circuit gate, depending on the intensities of incoming currents. In order to compute the heat produced by a 2-path, one determines the currents that its 2-cells propagate and, from those values, the heat each one produces; then one sums up all these heats.

Lemma 2.14. *A differential interpretation of a polygraph* $\mathcal{P}$ *is entirely and uniquely determined by its values on the* 2*-cells of* $\mathcal{P}$.

Proof. First, we prove that the differential relations imply that a differential interpretation takes the same values on each side of the relations of associativity, local units and exchange. For example, let us check this for the exchange relation. For that, let us fix 2-paths f, g, h and k such that both $t_1(f) = s_1(h)$ and $t_1(g) = s_1(k)$ are satisfied. We consider x in $\varphi(s_1(f))$ and y in $\varphi(s_1(g))$ and, using the functorial relations of φ and the differential relations of ∂, we compute each one of the following equalities in M:

$$\begin{aligned} \partial\big((f \star_0 g) \star_1 (h \star_0 k)\big)(x, y) &= \big(\partial f(x) + \partial g(y)\big) + \big(\partial h \circ \varphi(f)(x) + \partial k \circ \varphi(g)(y)\big), \\ \partial\big((f \star_1 h) \star_0 (g \star_1 k)\big)(x, y) &= \big(\partial f(x) + \partial h \circ \varphi(f)(x)\big) + \big(\partial g(y) + \partial k \circ \varphi(g)(y)\big). \end{aligned}$$

One concludes using the associativity and commutativity of $+$ in M. After that, one checks that the differential relations determine the values of a differential interpretation on 2-paths of size $n + 1$ from its values on 2-paths of size $k \leq n$. □

Lemma 2.14 allows one to define a differential interpretation by giving its values on 2-cells.

Example 2.15. The *trivial* functorial interpretation of a 3-polygraph $\mathcal{P}$ sends every 1-cell to some fixed one-element part $*$ of $\mathbb{N}-\{0\}$ and every 2-path from u to v to the only possible map from $\varphi(u) \simeq *$ to $\varphi(v) \simeq *$. Now, let us fix a family X of 2-cells in $\mathcal{P}$. One can check that the map $||\cdot||_X$ is the differential interpretation of $\mathcal{P}$ over the trivial interpretation and into $(\mathbb{N}, +, 0)$, sending a 2-cell $\blacksquare$ to 1 if it is in X and 0 otherwise.

Example 2.16. We consider the differential interpretation of the division polygraphic program $\mathcal{D}$, over the functorial interpretation given in Example 2.4, into $(\mathbb{N}, +, 0)$, sending every constructor and structure 2-cell to zero and:

$$\partial\triangledown(x,y) \;=\; y+1 \qquad \text{and} \qquad \partial\blacktriangledown(x,y) \;=\; xy+x,$$

Example 2.17. For the polygraphic program $\mathcal{F}$ of fusion sort, we consider the differential interpretation, over the functorial interpretation of Example 2.5, into $(\mathbb{N}, +, 0)$, sending every constructor and structure 2-cells to zero and:

$$\partial\blacklozenge(2x{+}1) = 2x^2{+}1, \quad \partial\blacktriangle(2x{+}1) = \lfloor x/2 \rfloor{+}1, \quad \partial\blacktriangledown(2x{+}1, 2y{+}1) = \begin{cases} 1 & \text{if } xy = 0, \\ x+y & \text{otherwise.} \end{cases}$$

Lemma 2.18. *Let $\mathcal{P}$ be a 3-polygraph, with a differential interpretation ∂, over a functorial interpretation φ, into an ordered commutative monoid $(M, +, 0, \preceq)$. Let f, g, h, k be 2-paths such that the inequalities $\varphi(f) \leq \varphi(g)$, $\partial f \preceq \partial g$ and $\partial h \preceq \partial k$ hold. Then, for every $i \in \{0,1\}$ such that $f \star_i h$ is defined, one has $\partial(f \star_i h) \preceq \partial(g \star_i k)$. Moreover, when M is strictly ordered and either $\partial f \prec \partial g$ or $\partial h \prec \partial k$ hold, one has $\partial(f \star_i h) \prec \partial(g \star_i k)$.*

Proof. One computes, for $x \in \varphi(s_1 f)$ and $y \in \varphi(s_1 h)$:

$$\partial(f \star_0 h)(x,y) \;=\; \partial f(x) + \partial h(y) \;\preceq\; \partial g(x) + \partial k(y) \;=\; \partial(g \star_0 k)(x,y).$$

Indeed, the two equalities are given by the differential relations that ∂ satisfies; the inequality uses the hypotheses, the fact that maps are compared pointwise and the monotony of $+$. Moreover, if $+$ is strictly monotone and if one of $\partial f \prec \partial g$ or $\partial h \prec \partial k$ holds, then the middle inequality is strict. Now, one checks:

$$\partial(f \star_1 h) \;=\; \partial f + \partial h \circ \varphi(f) \;\preceq\; \partial g + \partial k \circ \varphi(g) \;=\; \partial(g \star_1 k).$$

The equalities come from the differential relations; the inequality comes from the hypotheses $\partial f \preceq \partial g$, $\partial h \preceq \partial k$ and $\varphi(f) \leq \varphi(g)$, plus the monotony of ∂h and $+$ and the fact that maps are compared pointwise. When $+$ is strictly monotone and when either $\partial f \prec \partial g$ or $\partial h \prec \partial k$ hold, the middle inequality is strict. □

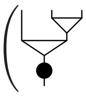
2.4. **Compatible differential interpretations.**

Definition 2.19. Let $\mathcal{P}$ be a 3-polygraph equipped with a functorial interpretation φ and a differential interpretation ∂ of $\mathcal{P}$ over φ and into an ordered commutative monoid M. For every 3-cell α, one says that ∂ is *compatible with* α when $\partial(s_2\alpha) \succeq \partial(t_2\alpha)$ holds. It is said to be *strictly compatible with* α when $\partial(s_2\alpha) \succ \partial(t_2\alpha)$ holds. One says that ∂ is *(strictly) compatible* when it is with every 3-cell of $\mathcal{P}$.

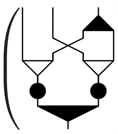
Example 2.20. The differential interpretations given in Examples 2.16 and 2.17 are compatible with every structure 3-cell and strictly compatible with every computation 3-cell of their 3-polygraph.

Indeed, in the source and the target of every structure 3-cell α, only constructor and structure 2-cells appear. The considered differential interpretations sends these to zero, yielding $\partial(s_2\alpha) = \partial(t_2\alpha) = 0$.

For an example of compatibility with a computation 3-cell, we consider the third 3-cell of the fusion sort function 2-cell . On one hand, one gets:

∂ () $(1,1,2x+1) \; = \; \partial$  $(2x+5) \; = \; 2(x+2)^2+1 \; = \; 2x^2+8x+9.$

And, on the other hand, one computes:

∂ () $(1,1,2x+1) \quad = \quad \big\{ \; \partial$  $\big(2\lceil x/2\rceil+3\big) \; + \; \partial$  $\big(2\lfloor x/2\rfloor+3\big)$ $\; + \; \partial$  $(2x+1) \; + \; \partial$  $\big(2\lceil x/2\rceil+3, 2\lfloor x/2\rfloor+3\big)$

$$\begin{aligned} &= 2\cdot\big(\lceil x/2\rceil+1\big)^2+2\cdot\big(\lfloor x/2\rfloor+1\big)^2+x+\lfloor x/2\rfloor+4 \\ &= 2\lceil x/2\rceil^2+2\lfloor x/2\rfloor^2+x+4\lceil x/2\rceil+5\lfloor x/2\rfloor+8 \\ &\leq 2x^2+6x+8. \end{aligned}$$

Proposition 2.21. *Let ∂ be a compatible differential interpretation of a polygraphic program $\mathcal{P}$, over a compatible functorial interpretation φ and into an ordered commutative monoid M. Then, for every 3-path F, the inequality $\partial(s_2F) \succeq \partial(t_2F)$ holds. When M is strictly ordered, ∂ is strictly compatible and F is non-degenerate, then $\partial(s_2F) \succ \partial(t_2F)$ also holds. Moreover, if M is $\mathbb{N}$ equipped with addition, then $|||F||| \leq \partial(s_2F) - \partial(t_2F)$ holds.*

Proof. We proceed by induction on the size of 3-paths. If F is a degenerate 3-path, then one has $s_2F = t_2F$ and, thus, $\partial(s_2F) = \partial(t_2F)$ also.

Let us assume that F is an elementary 3-path. We decompose F using a 3-cell α, 2-paths f, g and 1-paths u, v, yielding:

$$\partial(s_2F) \; = \; \partial\big(f \star_1 (u \star_0 s_2\alpha \star_0 v) \star_1 g\big) \qquad \text{and} \qquad \partial(t_2F) \; = \; \partial\big(f \star_1 (u \star_0 t_2\alpha \star_0 v) \star_1 g\big).$$

By assumption, φ and ∂ are compatible with α, hence $\varphi(s_2\alpha) \geq \varphi(t_2\alpha)$ and $\partial(s_2\alpha) \succeq \partial(t_2\alpha)$ hold. Then one applies Lemmas 2.7 and 2.18 to get $\partial(s_2F) \succeq \partial(t_2F)$ and, when ∂ is strictly compatible with the 3-cell α, $\partial(s_2F) \succ \partial(t_2F)$. If M is $\mathbb{N}$, this means:

$$\partial(s_2F) - \partial(t_2F) \geq 1 = |||F||| \, .$$

Finally, let us fix a non-zero natural number N and assume that the property holds for every 3-path of size N. Let us consider a 3-path F of size $N+1$. Then one decomposes F into $G \star_2 H$ where G is a 3-path of size N and H is an elementary 3-path. Then we apply the induction hypothesis to G and the previous case to H to conclude. □

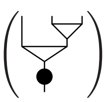
2.5. **Conservative functorial interpretations.** Intuitively, the following definition gives a bound on all the intensities of currents that one can find in the vicinity of any 2-cell inside a 2-path.

Definition 2.22. Let $\mathcal{P}$ be a 3-polygraph equipped with a functorial interpretation φ. One denotes by ∂_φ the differential interpretation of $\mathcal{P}$, over φ and into $(\mathbb{N}, \max, 0)$, sending every 2-cell with valence (m,n), i.e., with arity m and coarity n, to the following map from $\varphi(s_1 \,)$ to $\mathbb{N}$:

$$\partial_\varphi \; = \; \max\left\{\mu_m,\; \mu_n \circ \varphi(\;)\right\},$$

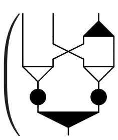
i.e., $\partial_\varphi(x_1,\ldots,x_m) = \max\{x_1,\ldots,x_m,y_1,\ldots,y_n\}$, if $(y_1,\ldots,y_n) = \varphi(\;)(x_1,\ldots,x_m)$. For every 3-cell α of $\mathcal{P}$, one says that φ is *conservative on* α when ∂_φ is compatible with α. One says that φ is *conservative* when it is conservative on every 3-cell of $\mathcal{P}$, i.e., when ∂_φ is compatible.

Example 2.23. The functorial interpretations of Examples 2.4 and 2.5 are conservative. Indeed, we shall see later that their values on structure and constructor 2-cells ensure that they are conservative on structure 3-paths. Let us check conservativeness on, for example, the last computation 3-cell of the sort function 2-cell:

∂_φ () $(1,1,2x+1) = \max\left\{1,\; 2x+1,\; 2x+2,\; 2x+3\right\}$

$$= 2x+3$$

$$= \max\left\{1,\; 2x+1,\; 2\cdot\lfloor x/2\rfloor+1,\; 2\cdot\lceil x/2\rceil+1,\; 2\cdot\lfloor x/2\rfloor+2,\; 2\cdot\lceil x/2\rceil+2,\; 2x+3\right\}$$

$= \partial_\varphi$ () $(1,1,2x+1)$.

When a functorial interpretation is both compatible and conservative, the intensities of currents inside 2-paths do not increase during computations.

Proposition 2.24. *Let φ be a compatible and conservative functorial interpretation of a polygraphic program. Then, for every 3-path F, the inequality $\partial_\varphi(s_2F) \geq \partial_\varphi(t_2F)$ holds.*

Proof. By definition of conservativeness and using Proposition 2.21 on ∂_φ. □

2.6. **Polygraphic interpretations.**

Definition 2.25. A *polygraphic interpretation* of a 3-polygraph $\mathcal{P}$ is a pair (φ, ∂) made of a functorial interpretation φ of $\mathcal{P}$, together with a differential interpretation ∂ of $\mathcal{P}$ over φ and into $(\mathbb{N}, +, 0)$. In that case, φ and ∂ respectively are the *functorial part* and the *differential part* of (φ, ∂).

Let us fix a 3-cell α. A polygraphic interpretation (φ, ∂) is *compatible (with α)* when both φ and ∂ are. It is *strictly compatible (with α)* when φ is compatible with α and ∂ is strictly compatible (with α). It is *conservative (on α)* when φ is.

Example 2.26. The functorial and differential interpretations we have built on the polygraphic programs of division and of fusion sort are two examples of polygraphic interpretations that are conservative, compatible with every structure 3-cell and strictly compatible with every computation 3-cell.

Let us consider the trivial functorial interpretation and the differential interpretation $||\cdot||_X$ over it, for some family X of 2-cells. They form a polygraphic interpretation that is conservative but that has no general reason to be compatible with any 3-cell.

We recall the following theorem:

Theorem 2.27 ([18])**.** *If a* 3*-polygraph has a polygraphic interpretation which is strictly compatible with all of its* 3*-cells, then it terminates.*

Proof. By application of Proposition 2.21, one knows that $\partial(s_2F) > \partial(t_2F)$ holds for every elementary 3-cell F. Furthermore, these are maps with values into $\mathbb{N}$. Since there is no infinite strictly decreasing sequence of such maps for the pointwise order, one concludes that $\mathcal{P}$ must terminate. □

In what follows, we use Theorem 2.27 in several steps, thanks to the following result:

Proposition 2.28. *Let* $\mathcal{P}$ *be a* 3*-polygraph and let* X *be a set of* 3*-cells of* $\mathcal{P}$*. Let us assume that there exists a compatible polygraphic interpretation on* $\mathcal{P}$ *whose restriction to* X *is strictly compatible. Then* $\mathcal{P}$ *terminates if and only if* $\mathcal{P} - X$ *does.*

Proof. If $\mathcal{P}$ terminates, its reduction graph has no infinite path. Since it contains the reduction graph of the 3-polygraph $\mathcal{P} - X$, the latter does not have any infinite path either. Hence $\mathcal{P} - X$ terminates.

Conversely, let us assume that $\mathcal{P}$ does not terminate. Then there exists an infinite sequence $(F_n)_{n\in\mathbb{N}}$ of elementary 3-paths in $\mathcal{P}$ such that, for every $n \in \mathbb{N}$, F_n and F_{n+1} are composable. The polygraphic interpretation is compatible, hence one can apply Proposition 2.21 to get the following infinite sequence of inequalities in $\mathbb{N}$:

$$\partial(s_2F_0) \geq \partial(t_2F_0) = \partial(s_2F_1) \geq (\cdots) = \partial(s_2F_n) \geq \partial(t_2F_n) = \partial(s_2F_{n+1}) \geq (\cdots)$$

Furthermore, for every $n \in \mathbb{N}$ such that $F_n \in \langle X\rangle$, one has a strict inequality $\partial(s_2F_n) > \partial(t_2F_n)$, since the polygraphic interpretation is strictly compatible with every 3-cell of X. Hence, there are only finitely many n in $\mathbb{N}$ such that F_n is in $\langle X\rangle$: otherwise, one could extract, from $(\partial(s_2F_n))_{n\in\mathbb{N}}$, an infinite, strictly decreasing sequence of maps with values in $\mathbb{N}$. Thus, there exists some $n_0 \in \mathbb{N}$ such that $(F_n)_{n\geq n_0}$ is an infinite path in the reduction graph of $\mathcal{P} - X$: this means that $\mathcal{P} - X$ does not terminate. □

Example 2.29. Let us consider the polygraphic programs for division and fusion sort, given in Examples 1.18 and 1.19. We have seen that each one admits a compatible polygraphic interpretation that is strictly compatible with their computation 3-cells. Furthermore, as proved later, the structure 3-cells, alone, terminate. Thus Proposition 2.28 gives the termination of both polygraphic programs.

Actually, in what comes next, we produce a standard differential interpretation that is strictly compatible with structure 3-cells. However, in general, it is not compatible, even in a non-strict way, with computation 3-cells: informally, each application of such a cell can increase the "structure heat". The purpose of the rest of this section is to bound this potential augmentation.

Lemma 2.30. *Let $\mathcal{P}$ be a 3-polygraph equipped with a polygraphic interpretation (φ, ∂). Then, for every 2-path f in $\mathcal{P}$ and every x in $\varphi(s_1 f)$, the following inequality holds in $\mathbb{N}$:*

$$\partial f(x) \;\leq\; \sum_{\blacksquare \in \mathcal{P}_2} ||f||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi f(x),\; \ldots,\; \partial_\varphi f(x) \,\big).$$

Remark 2.31. Let us note that we apply $\partial\blacksquare$ to arguments $\partial_\varphi f(x)$ that are not necessarily in its domain. In that case, one considers an extension of $\partial\blacksquare$ sending x to $\partial\blacksquare(y)$, where y is the maximum element of the set $\varphi(s_1\blacksquare)$ that is below x.

Proof. We proceed by induction on the size of the 2-path f. Let us assume that f is degenerate. Then one has $||f||_{\blacksquare} = 0$ for every 2-cell $\blacksquare$ and, since ∂ is a differential interpretation, $\partial f = 0$. Hence both sides of the sought inequality are equal to 0.

Now, let us consider an elementary 2-path f. One decomposes f into $u \star_0 \blacksquare \star_0 v$, where $\blacksquare$ is a 2-cell and u and v are 1-paths. Then $||f||_{\square}$ is 1 when $\square$ is $\blacksquare$ and 0 otherwise. Let us fix x, y and z respectively in $\varphi(u)$, $\varphi(s_1\blacksquare)$ and $\varphi(v)$. Using the differential relations of ∂ and ∂_φ, one gets $\partial f(x,y,z) = \partial\blacksquare(y)$ and $\partial_\varphi f(x,y,z) = \partial_\varphi\blacksquare(y)$. If $\blacksquare$ has valence (m,n) and $y = (y_1, \ldots, y_m)$, one uses the definition of $\partial_\varphi\blacksquare$ to get, for every $i \in \{1, \ldots, m\}$:

$$\partial_\varphi\blacksquare(y) \;=\; \max\big\{\, \mu_m(y),\; \mu_n \circ \varphi(\blacksquare)(y) \,\big\} \;\geq\; y_i.$$

Then one computes:

$$\begin{aligned}
\sum_{\square \in \mathcal{P}_2} ||f||_{\square} \cdot \partial\square \,\big(\, \partial_\varphi f(x,y,z),\; \ldots,\; \partial_\varphi f(x,y,z) \,\big) \;&=\; \partial\blacksquare \,\big(\, \partial_\varphi\blacksquare(y),\; \ldots,\; \partial_\varphi\blacksquare(y) \,\big) \\
&\geq\; \partial\blacksquare\,(y_1, \ldots, y_m) \\
&=\; \partial f(x,y,z).
\end{aligned}$$

Finally, let us fix a non-zero natural number N and assume that the property holds for every 2-path of size at most N. We consider a 2-path f of size $N+1$: there exists a decomposition $f = g \star_1 h$ where g and h are 2-paths of size at most N. Then, using the differential relations of $||\cdot||_{\blacksquare}$, for any 2-cell $\blacksquare$, and of ∂_φ, one gets:

$$||f||_{\blacksquare} \;=\; ||g||_{\blacksquare} + ||h||_{\blacksquare} \qquad \text{and} \qquad \partial_\varphi(f) \;=\; \max\{\, \partial_\varphi g,\; \partial_\varphi h \circ \varphi(g) \,\}\,.$$

We fix a x in $\varphi(s_1 f)$ and we compute:

$$\begin{aligned}
\partial f(x) \quad &= \quad \partial(g \star_1 h)(x) \\
&= \quad \partial g(x) + \partial h \circ \varphi(g)(x) \\
&\leq \quad \sum_{\blacksquare \in \mathcal{P}_2} ||g||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi g(x),\; \ldots,\; \partial_\varphi g(x) \,\big) \\
&\qquad + \sum_{\blacksquare \in \mathcal{P}_2} ||h||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi h \circ \varphi(g)(x),\; \ldots,\; \partial_\varphi h \circ \varphi(g)(x) \,\big) \\
&\leq \quad \sum_{\blacksquare \in \mathcal{P}_2} ||g||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi f(x),\; \ldots,\; \partial_\varphi f(x) \,\big) \\
&\qquad + \sum_{\blacksquare \in \mathcal{P}_2} ||h||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi f(x),\; \ldots,\; \partial_\varphi f(x) \,\big)
\end{aligned}$$

We factorize the right-hand side to conclude the proof:

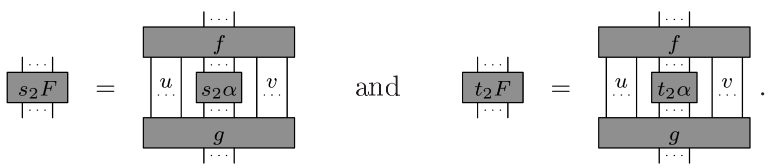
$$\begin{aligned}\partial f(x) \quad &\leq \quad \sum_{\blacksquare\in\mathcal{P}_2} \Big(||g||_{\blacksquare} + ||h||_{\blacksquare} \Big) \cdot \partial\blacksquare \,\big(\, \partial_\varphi f(x),\, \ldots,\, \partial_\varphi f(x) \,\big) \\ &= \quad \sum_{\blacksquare\in\mathcal{P}_2} ||f||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi f(x),\, \ldots,\, \partial_\varphi f(x) \,\big).\end{aligned}$$

□

Proposition 2.32. *Let $\mathcal{P}$ be a 3-polygraph, let α be a 3-cell of $\mathcal{P}$ and let F be an elementary 3-path in $\langle\alpha\rangle$. One assumes that $\mathcal{P}$ is equipped with a polygraphic interpretation (φ, ∂) such that φ is compatible with and conservative on α. Then, for every $x \in \varphi(s_1 F)$, the following inequality holds in $\mathbb{Z}$:*

$$\partial(t_2F)(x) - \partial(s_2F)(x) \;\leq\; \sum_{\blacksquare\in\mathcal{P}_2} ||t_2(\alpha)||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi(s_2F)(x),\, \ldots,\, \partial_\varphi(s_2F)(x) \,\big).$$

Proof. Since F is a 3-path of size 1 in $\langle\alpha\rangle$, one can decompose s_2F and t_2F as follows:



Let us denote by p, q and m the respective sizes of u, v and s_1F. The map $\varphi(f)$ takes its values in a part of $\mathbb{N}^{p+m+q}$: we decompose it into three maps denoted by $\varphi_1(f)$, $\varphi_2(f)$ and $\varphi_3(f)$, with the same domain and respectively taking their values in parts of $\mathbb{N}^p$, $\mathbb{N}^m$ and $\mathbb{N}^q$. Let us fix a $x \in \varphi(s_1F)$. The functorial and differential relations give:

$$\partial(s_2F)(x) \;=\; \partial f(x) \;+\; \partial(s_2\alpha)\circ\varphi_2(f)(x) \;+\; \partial g\big(\, \varphi_1(f)(x),\, \varphi(s_2\alpha)\circ\varphi_2(f)(x),\, \varphi_3(f)(x) \,\big).$$

With the same arguments, one gets the same decomposition for $\partial(t_2F)$, with $s_2\alpha$ replaced by $t_2\alpha$. Thus, the following holds in $\mathbb{Z}$:

$$\begin{aligned}\partial(t_2F)(x) - \partial(s_2F)(x) \;=\; & \partial(t_2\alpha)\circ\varphi_2(f)(x) - \partial(s_2\alpha)\circ\varphi_2(f)(x) \\ & +\; \partial g \,\big(\, \varphi_1(f)(x),\, \varphi(t_2\alpha)\circ\varphi_2(f)(x),\, \varphi_3(f)(x) \,\big) \\ & -\; \partial g \,\big(\, \varphi_1(f)(x),\, \varphi(s_2\alpha)\circ\varphi_2(f)(x),\, \varphi_3(f)(x) \,\big).\end{aligned}$$

Let us prove that $\partial(t_2F)(x) - \partial(s_2F)(x) \leq \partial(t_2\alpha)\circ\varphi_2(f)(x)$ holds. First, one has $\partial(s_2\alpha) \geq 0$. Moreover, φ is compatible with α, which means that $\varphi(s_2\alpha) \geq \varphi(t_2\alpha)$ holds; since the map ∂g is monotone, the following holds in $\mathbb{N}$:

$$\partial g \,\big(\, \varphi_1(f)(x), \varphi(s_2\alpha)\circ\varphi_2(f)(x), \varphi_3(f)(x) \,\big) \;\geq\; \partial g \,\big(\, \varphi_1(f)(x), \varphi(t_2\alpha)\circ\varphi_2(f)(x), \varphi_3(f)(x) \,\big).$$

It remains to bound $\partial(t_2\alpha)\circ\varphi_2(f)(x)$. One applies Lemma 2.30 to $t_2(\alpha)$ to get:

$$\partial(t_2\alpha)\circ\varphi_2(f)(x) \;\leq\; \sum_{\blacksquare\in\mathcal{P}_2} ||t_2(\alpha)||_{\blacksquare} \cdot \partial\blacksquare \,\big(\, \partial_\varphi(t_2\alpha)\circ\varphi_2(f)(x),\, \ldots,\, \partial_\varphi(t_2\alpha)\circ\varphi_2(f)(x) \,\big).$$

By assumption, φ is conservative on α, thus $\partial_\varphi t_2(\alpha)\circ\varphi_2(f)(x) \leq \partial_\varphi s_2(\alpha)\circ\varphi_2(f)(x)$ holds. Moreover, using the differential properties satisfied by ∂_φ, one gets $\partial_\varphi s_2(\alpha)\circ\varphi_2(f)(x) \leq \partial_\varphi(s_2F)$. One concludes by invoking the monotony of $\partial\blacksquare$. □

3. Complexity of polygraphic programs

In this section, we specialize polygraphic interpretations to polygraphic programs to get information on their complexity. In particular, we introduce additive polygraphic interpretations and use them as an estimation of the size of values. This way, we give bounds on the size of computations, with respect to the size of the arguments. We conclude this work with a characterisation of a class of polygraphic programs that compute exactly the FPTIME functions.

3.1. Additive functorial interpretations and the size of values.

Definition 3.1. Let $\mathcal{P}$ be a polygraphic program. One says that a functorial interpretation φ of $\mathcal{P}$ is *additive* when, for every constructor 2-cell $\blacktriangledown$ of arity n, there exists a non-zero natural number $c_{\blacktriangledown}$ such that, for every $(x_1, \dots, x_n)$ in $\varphi(s_1 \blacktriangledown)$, the following equality holds in $\mathbb{N}$:

$$\varphi(\blacktriangledown)(x_1, \dots, x_n) \;=\; x_1 + \cdots + x_n + c_{\blacktriangledown}.$$

In that case, one denotes by γ the greatest of these numbers, i.e., :

$$\gamma \;=\; \max\Big\{ c_{\blacktriangledown},\ \blacktriangledown \in \mathcal{P}_2^C \Big\}.$$

A polygraphic interpretation is *additive* when its functorial part is.

Example 3.2. The functorial interpretations we have built for the polygraphic programs $\mathcal{D}$ and $\mathcal{F}$ are additive. In both cases, γ is 1.

Lemma 3.3. *Let φ be an additive functorial interpretation of a polygraphic program $\mathcal{P}$ and let t be a value with coarity 1. Then the following equality holds in $\mathbb{N}$:*

$$\varphi(t) \;=\; \sum_{\blacktriangledown \in \mathcal{P}_2^C} ||t||_{\blacktriangledown} \cdot c_{\blacktriangledown}.$$

Proof. Let us prove this result by induction on the size of the 2-path t. There is no degenerate value with coarity 1. If t is an elementary value with coarity 1, then t is a constructor 2-cell $\circ$ with arity 0. Since φ is additive, one has $\varphi(\circ) = c_{\circ}$. Moreover, $||t||_{\blacktriangledown}$ is 1 when $\blacktriangledown = \circ$ holds and 0 otherwise, yielding the equality one seeks.

Now, let us fix a non-zero natural number N and assume that the result holds for every value with coarity 1 and size at most N. Let us fix a value t with coarity 1 and size $N+1$. Then t admits a decomposition $t \;=\; \big(t_1 \star_0 \cdots \star_0 t_n\big) \star_1 \blacktriangledown$, where $\blacktriangledown$ is a constructor 2-cell with arity n and each t_i, $i \in \{1, \dots, n\}$, is a value with coarity 1 and size at most N. As a consequence, for every constructor 2-cell $\triangledown$, one has:

$$||t||_{\triangledown} \;=\; \begin{cases} ||t_1||_{\triangledown} + \cdots + ||t_n||_{\triangledown} + 1 & \text{if } \triangledown = \blacktriangledown, \\ ||t_1||_{\triangledown} + \cdots + ||t_n||_{\triangledown} & \text{otherwise.} \end{cases}$$

Finally, one computes:

$$\begin{aligned}
\varphi(t) \;&=\; \varphi(\blacktriangledown) \circ \big(\varphi(t_1) \times \cdots \times \varphi(t_n)\big) && \text{from the functorial relations of } \varphi,\\
&=\; \varphi(t_1) + \cdots + \varphi(t_n) + c_{\blacktriangledown} && \text{since } \varphi \text{ is additive,}\\
&=\; \sum_{\triangledown \in \mathcal{P}_2^C} \Big(||t_1||_{\triangledown} + \cdots + ||t_n||_{\triangledown} \Big) \cdot c_{\triangledown} + c_{\blacktriangledown} && \text{by induction hypothesis}\\
&=\; \sum_{\triangledown \in \mathcal{P}_2^C} ||t||_{\triangledown} \cdot c_{\triangledown} && \text{from previous remark.} \qquad \square
\end{aligned}$$

Proposition 3.4. *Let φ be an additive functorial interpretation of a polygraphic program $\mathcal{P}$. Then, for every value t with coarity 1, the inequalities $||t|| \leq \varphi(t) \leq \gamma\, ||t||$ hold in $\mathbb{N}$. As a consequence, for every value t, one has $\nu(t) \leq \varphi(t) \leq \gamma\nu(t)$, where ν is the functorial interpretation introduced in Example 2.6.*

Proof. Let us assume that t is a value with coarity 1. From Lemma 3.3, one has:

$$\varphi(t) \;=\; \sum_{\blacktriangledown \in \mathcal{P}_2^C} ||t||_{\blacktriangledown} \cdot c_{\blacktriangledown}.$$

By additivity of φ and by definition of γ, one has $1 \leq c_{\blacktriangledown} \leq \gamma$ for every constructor 2-cell $\blacktriangledown$. One concludes by using the following equality, that holds since t is in $\langle \mathcal{P}_2^C \rangle$:

$$||t|| = \sum_{\blacktriangledown \in \mathcal{P}_2^C} ||t||_{\blacktriangledown}\,.$$

When $t_1, \ldots, t_n$ are values with coarity 1 and when $t = t_1 \star_0 \cdots \star_0 t_n$, one concludes thanks to the equalities $\varphi(t) = \big(\varphi(t_1), \ldots, \varphi(t_n)\big)$ and $\nu(t) = \big(\, ||t_1||\,, \ldots, ||t_n||\,\big)$. $\square$

Lemma 3.5. *Let φ be an additive functorial interpretation of a polygraphic program $\mathcal{P}$. For every value t with coarity 1, the equality $\partial_\varphi t = \varphi(t)$ holds. As a consequence, for every value t with coarity n, one has $\partial_\varphi t = \mu_n \circ \varphi(t)$.*

Proof. Let us proceed by induction on the size of t. If ♀ is a constructor 2-cell with arity 0, then the equality holds by definition of ∂_φ♀.

Now, let us fix a non-zero natural number N and assume that the result holds for every value with coarity 1 and size at most N. Let us consider a value t with coarity 1 and size $N+1$. One decomposes t into $t = (t_1 \star_0 \cdots \star_0 t_n) \star_1 \blacktriangledown$, with $\blacktriangledown$ a constructor 2-cell and where t_i is a value with coarity 1 and size at most N, for every $i \in \{1, \ldots, n\}$. Using the differential relations of ∂_φ, one gets:

$$\partial_\varphi t \;=\; \max\big\{\, \partial_\varphi(t_1),\, \ldots,\, \partial_\varphi(t_n),\, \partial_\varphi \blacktriangledown \,\big(\varphi(t_1), \ldots, \varphi(t_n)\big) \,\big\}\,.$$

The definition of $\partial_\varphi \blacktriangledown$ gives:

$$\partial_\varphi \blacktriangledown \,\big(\varphi(t_1), \ldots, \varphi(t_n)\big) \;=\; \max\big\{\, \varphi(t_1),\, \ldots,\, \varphi(t_n),\, \varphi(\blacktriangledown)\big(\varphi(t_1), \ldots, \varphi(t_n)\big) \,\big\}\,.$$

Since φ is additive, $\varphi(\blacktriangledown)\big(\varphi(t_1), \ldots, \varphi(t_n)\big)$ is greater than every $\varphi(t_i)$, which is $\partial_\varphi(t_i)$ by induction hypothesis applied to t_i. Thus one gets the following equality and uses the functorial relations of φ to conclude:

$$\partial_\varphi t \;=\; \varphi(\blacktriangledown)\big(\varphi(t_1), \ldots, \varphi(t_n)\big).$$

Finally, let us consider a value t with coarity n. One denotes by $(t_1, \ldots, t_n)$ the family of values with coarity 1 such that $t = t_1 \star_0 \cdots \star_0 t_n$ holds. One invokes the differential relations of ∂_φ to get the equality $\partial_\varphi t = \max\left\{\partial_\varphi(t_1), \ldots, \partial_\varphi(t_n)\right\}$. One uses the induction hypothesis on each t_i and concludes, thanks to the functorial relations satisfied by φ. $\square$

Proposition 3.6. *Let φ be an additive functorial interpretation on a polygraphic program $\mathcal{P}$. For every function 2-cell $\blacksquare$ and every value t of type $s_1(\blacksquare)$, one has $\partial_\varphi(t \star_1 \blacksquare) = \partial_\varphi \blacksquare \circ \varphi(t)$.*

Proof. Let us assume that $\blacksquare$ has valence (m, n). One uses the differential relations of ∂_φ to produce:

$$\partial_\varphi(t \star_1 \blacksquare) \;=\; \max\left\{\partial_\varphi t, \partial_\varphi \blacksquare \circ \varphi(t)\right\}.$$

But, by definition of ∂_φ, one has $\partial_\varphi \blacksquare \circ \varphi(t) \geq \mu_m \circ \varphi(t)$. There remains to use Lemma 3.5 on t to get $\partial_\varphi t = \mu_n \circ \varphi(t)$. $\square$

Notation 3.7. Let $\blacksquare$ be a function 2-cell with arity m in a polygraphic program $\mathcal{P}$, equipped with an additive functorial interpretation φ. Thereafter, we denote by $M_{\blacksquare}$ the map from $\mathbb{N}^m$ to $\mathbb{N}$ defined by:

$$M_{\blacksquare}(x_1, \ldots, x_m) \;=\; \partial_\varphi \blacksquare\,\left(\,\gamma x_1,\; \ldots,\; \gamma x_m\,\right).$$

The next result uses the map $M_{\blacksquare}$ and the size of the initial arguments to bound the size of intermediate values produced during computations, hence of the arguments of potential recursive calls.

Proposition 3.8. *Let $\mathcal{P}$ be a polygraphic program, equipped with an additive, compatible and conservative functorial interpretation φ. Let $\blacksquare$ be a function 2-cell and let t be a value of type $s_1\blacksquare$. Then, for every 3-path F with source $t \star_1 \blacksquare$, the following inequality holds in $\mathbb{N}$:*

$$\partial_\varphi(t_2 F) \;\leq\; M_{\blacksquare} \circ \nu(t).$$

Proof. The functorial interpretation φ is compatible and conservative: by Proposition 2.24, we know that $\partial_\varphi(t_2 F) \leq \partial_\varphi(t \star_1 \blacksquare)$ holds. Since φ is additive, one may use Proposition 3.6 to produce the equality $\partial_\varphi(t \star_1 \blacksquare) = \partial_\varphi \blacksquare \circ \varphi(t)$. Furthermore, Proposition 3.4 gives $\varphi(t) \leq \gamma\nu(t)$: one argues that ∂_φ is monotone to conclude. $\square$

Example 3.9. Applied to Example 1.19, Proposition 3.8 tells us that, given a list t, any intermediate value produced by the computation of the sorted list $\phi(t)$ has its size bounded by $M_{\phi}(||t||) = ||t||$. This means that recursive calls made during this computation are applied to arguments of size at most $||t||$.

3.2. Cartesian polygraphic interpretations and the size of structure computations.

Here we bound the number of structure 3-cells that can appear in a computation. For that, we consider polygraphic interpretations that take special values on structure 2-cells.

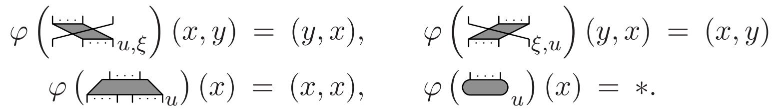
Definition 3.10. Let $\mathcal{P}$ be a polygraphic program. A functorial interpretation φ of $\mathcal{P}$ is said to be *cartesian* when the following conditions hold, for every 1-cells ξ and ζ:

$$\varphi\left(\blacktriangle_\xi\right)(x) \;=\; (x, x) \qquad \text{and} \qquad \varphi\left(\times_{\xi,\zeta}\right)(x, y) \;=\; (y, x).$$

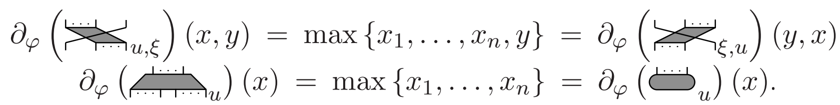
A polygraphic interpretation is *cartesian* when its functorial part is cartesian and when its differential part sends every constructor and structure 2-cell to zero.

Proposition 3.11. *If a functorial interpretation of a polygraphic program $\mathcal{P}$ is cartesian, then it is compatible with and conservative on all the structure 3-cells.*

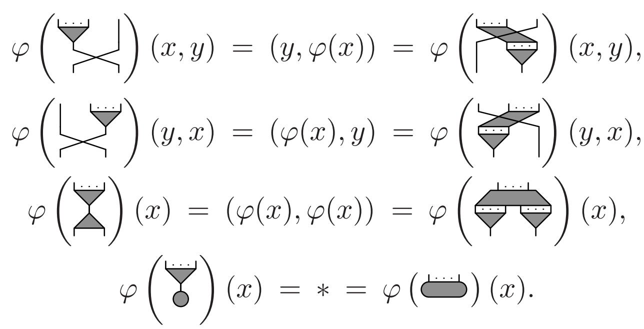
Proof. Let φ be a cartesian functorial interpretation of a polygraphic program $\mathcal{P}$. We start by computing the values of φ and ∂_φ on the structure 2-paths, by induction on their size. This way, one proves that the following equalities hold, for any 1-path u and $x \in \varphi(u)$, any 1-cell ξ and $y \in \varphi(\xi)$:



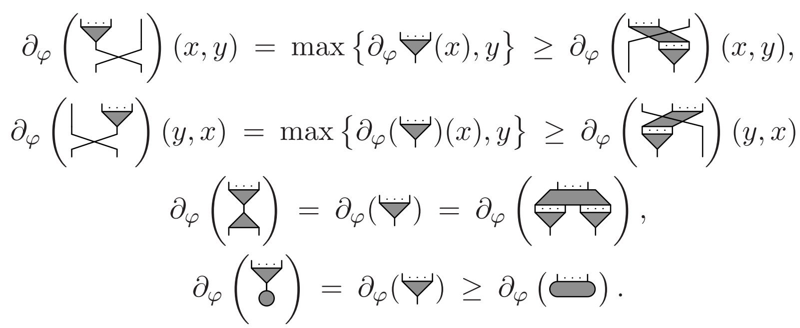
Then, when $u = *$, all these 2-paths are degenerate, so that they are sent on 0 by the differential interpretation ∂_φ. Now, when u is non-degenerate, with $x = (x_1, \ldots, x_n)$, one gets:



Now, we fix a 1-path u, 1-cells ξ, ζ and a constructor 2-cell  : $u \to \xi$ in $\mathcal{P}$. Let us consider $x \in \varphi(u)$ and $y \in \varphi(\zeta)$ and check that the following equalities hold, yielding the compatibility of φ on structure 3-cells:

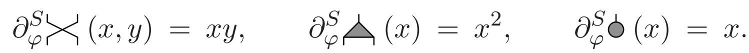


With the same notations, we now check the conservativeness of φ with the structure 3-cells, i.e., the compatibility of ∂_φ with them:



□

Definition 3.12. Let φ be a functorial interpretation of a polygraphic program $\mathcal{P}$. We denote by ∂^S_φ and call *structure differential interpretation generated by* φ the differential interpretation of $\mathcal{P}$, over φ and into $(\mathbb{N}, +, 0)$, that sends every constructor and function 2-cell to zero and such that the following hold:



Lemma 3.13. *Let φ be a functorial interpretation of a polygraphic program $\mathcal{P}$. If φ is both additive and cartesian, then ∂_φ^S is strictly compatible with all the structure 3-cells of $\mathcal{P}$.*

Proof. We start by computing ∂_φ^S on the structure 2-paths, by induction on their size:

$$\partial_\varphi^S\,(\;)\,(x_1,\dots,x_n,y) \;=\; \partial_\varphi^S\,(\;)\,(y,x_1,\dots,x_n) \;=\; y\cdot\textstyle\sum_{1\leq i\leq n} x_i,$$

$$\partial_\varphi^S\,(\;)\,(x_1,\dots,x_n) \;=\; \textstyle\sum_{1\leq i\leq j\leq n} x_i\cdot x_j, \qquad \partial_\varphi^S\,(\;)\,(x_1,\dots,x_n) \;=\; \textstyle\sum_{1\leq i\leq n} x_i.$$

Now, let us fix a constructor 2-cell $(\;)$ with arity n. Let us consider $x=(x_1,\dots,x_n)$ in $\varphi(s_1(\;))$. Since φ is additive, one notes that $\varphi(\;)(x) \;>\; x_1+\dots+x_n$ holds. Then, given a $y\in\mathbb{N}-\{0\}$, one checks that the following strict inequalities hold in $\mathbb{N}-\{0\}$:

$$\partial_\varphi^S\left(\;\right)(x,y) \;=\; y\cdot\varphi(\;)(x) \;>\; y\cdot\sum_{1\leq i\leq n} x_i \;=\; \partial_\varphi^S\left(\;\right)(x,y),$$

$$\partial_\varphi^S\left(\;\right)(x,y) \;=\; y\cdot\varphi(\;)(x) \;>\; y\cdot\sum_{1\leq i\leq n} x_i \;=\; \partial_\varphi^S\left(\;\right)(x,y),$$

$$\partial_\varphi^S\left(\;\right)(x) \;=\; \big(\varphi(\;)(x)\big)^2 \;>\; \sum_{1\leq i\leq j\leq n} x_i\cdot x_j \;=\; \partial_\varphi^S\left(\;\right)(x),$$

$$\partial_\varphi^S\left(\;\right)(x) \;=\; \varphi(\;)(x) \;>\; \sum_{1\leq i\leq n} x_i \;=\; \partial_\varphi^S\,(\;)\,(x).$$

□

The following result gives sufficient conditions on a polygraphic interpretation such that one does not have to bother with the structure 3-cells to prove termination.

Proposition 3.14. *If a polygraphic program admits an additive and cartesian polygraphic interpretation that is strictly compatible with every computation 3-cell, then it terminates.*

Proof. Let (φ,∂) be a polygraphic interpretation with the required properties. One applies Proposition 3.11 to get the compatibility of φ with structure 3-cells. Then Lemma 3.13 tells us that $(\varphi,\partial_\varphi^S)$ is strictly compatible with structure 3-cells: hence Theorem 2.27 yields termination of $\mathcal{P}_3^S$.

Since ∂ sends every constructor and structure 2-cell to zero, one has $\partial(s_2\alpha)=\partial(t_2\alpha)=0$ for every structure 3-cell α: thus (φ,∂) is compatible with every structure 3-cell and, by hypothesis, strictly compatible with every other 3-cell. One applies Proposition 2.28 to conclude. □

Definition 3.15. Let $\mathcal{P}$ be a polygraphic program. One denotes by K the maximum number of structure 2-cells one finds in the targets of computation 3-cells:

$$K = \max\left\{ ||t_2(\alpha)||_{\mathcal{P}_2^S}\,,\alpha\in\mathcal{P}_3^R \right\}.$$

Let φ be an additive functorial interpretation of $\mathcal{P}$. For every function 2-cell $(\;)$ with arity m, one defines $S_{(\;)}$ as the map from $\mathbb{N}^m$ to $\mathbb{N}$ given by:

$$S_{(\;)}(x_1,\dots,x_m) \;=\; K\cdot M_{(\;)}^2(x_1,\dots,x_m).$$

The following lemma proves that, during a computation, if one applies a computation 3-cell, then the structure heat increase is bounded by a polynomial in the size of the arguments.

Lemma 3.16. *Let $\mathcal{P}$ be a polygraphic program, equipped with an additive, cartesian, compatible and conservative functorial interpretation φ. Let $\Box$ be a function 2-cell and t be a value of type $s_1(\Box)$. Let f and g be 2-paths such that $t \star_1 \Box$ reduces into f which, in turn, reduces into g by application of a computation 3-cell α. Then, the following inequality holds in $\mathbb{Z}$:*

$$\partial^S_\varphi g - \partial^S_\varphi f \;\leq\; S_{\Box} \circ \nu(t).$$

Proof. Since φ is compatible and conservative, one can apply Proposition 2.32 on the 3-path from f to g, to get the following inequality:

$$\partial^S_\varphi g - \partial^S_\varphi f \;\leq\; \sum_{\Box \in \mathcal{P}_2} ||t_2(\alpha)||_{\Box} \cdot \partial^S_\varphi \Box \left(\partial_\varphi(f), \, \ldots, \, \partial_\varphi(f) \right).$$

By definition of ∂^S_φ, one has $\partial^S_\varphi \Box = 0$ except when $\Box$ is a structure 2-cell. Thus one gets:

$$\begin{aligned}
&\partial^S_\varphi g - \partial^S_\varphi f \\
&\leq ||t_2(\alpha)||_{\times} \cdot \partial^S_\varphi \times \left(\partial_\varphi(f), \, \partial_\varphi(f) \right) \;+\; ||t_2(\alpha)||_{\blacktriangle} \cdot \partial^S_\varphi \blacktriangle \left(\partial_\varphi(f)\right) \;+\; ||t_2(\alpha)||_{\bullet} \cdot \partial^S_\varphi \bullet \left(\partial_\varphi(f)\right) \\
&= ||t_2(\alpha)||_{\times} \cdot \left(\partial_\varphi(f)\right)^2 \;+\; ||t_2(\alpha)||_{\blacktriangle} \cdot \left(\partial_\varphi(f)\right)^2 \;+\; ||t_2(\alpha)||_{\bullet} \cdot \partial_\varphi(f) \\
&\leq ||t_2(\alpha)||_{\mathcal{P}^S_2} \cdot \left(\partial_\varphi(f)\right)^2 \\
&\leq K \cdot \left(\partial_\varphi(f)\right)^2.
\end{aligned}$$

Finally, we recall that φ is additive, compatible and conservative: an application of Proposition 3.8 to the 3-path with source $t \star_1 \Box$ and target f yields $\partial_\varphi(f) \leq M_{\Box} \circ \nu(t)$ and concludes the proof. □

Example 3.17. For the polygraphic program of Example 1.19, we have $K = 1$. The polynomials bounding the structure interpretation increase after application of one of the computation 3-cells of this polygraphic program are:

$$S_{\bullet}(x) = x^2, \qquad S_{\blacktriangle}(x) = x^2, \qquad S_{\blacktriangledown}(x, y) = (x + y - 1)^2.$$

3.3. **The size of computations.**

Definition 3.18. Let $\mathcal{P}$ be a polygraphic program, with an additive polygraphic interpretation (φ, ∂). For every function 2-cell $\Box$ with arity m, one denotes by $P_{\Box}$ and by $Q_{\Box}$ the maps from $\mathbb{N}^m$ to $\mathbb{N}$ defined by:

$$\begin{aligned}
P_{\Box}(x_1, \ldots, x_m) \;&=\; \partial \Box \left(\gamma x_1, \ldots, \gamma x_m\right), \\
Q_{\Box}(x_1, \ldots, x_m) \;&=\; P_{\Box}(x_1, \ldots, x_m) \cdot \Big(1 + S_{\Box}(x_1, \ldots, x_m)\Big).
\end{aligned}$$

The following result bounds the number of computation 3-cells in a reduction 3-path, with respect to the size of the arguments.

Proposition 3.19. *Let $\mathcal{P}$ be a polygraphic program, equipped with an additive and cartesian polygraphic interpretation (φ, ∂) which is strictly compatible with every computation 3-cell. Let $\Box$ be a function 2-cell and t be a value of type $s_1(\Box)$. Then, for every 3-path F with source $t \star_1 \Box$, the following inequality holds:*

$$|||F|||_{\mathcal{P}^R_3} \;\leq\; P_{\Box} \circ \nu(t).$$

Proof. If F is degenerate, then $|||F|||_{\mathcal{P}_3^R} = 0$ holds. Otherwise, the 3-path F decomposes this way:

$$F \;=\; H_0 \star_2 G_1 \star_2 H_1 \star_2 G_2 \star_2 \cdots \star_2 G_k \star_2 H_k,$$

where each G_i is elementary in $\langle \mathcal{P}_3^R \rangle$ and each H_j lives in $\langle \mathcal{P}_3^S \rangle$. Hence $|||F|||_{\mathcal{P}_3^R} = k$. Since the polygraphic interpretation is cartesian, it is compatible with every structure 3-cell, so that one has $\partial(s_2 H_j) \geq \partial(t_2 H_j)$, for every $j \in \{0, \ldots, k\}$. Since it is also strictly compatible with every computation 3-cell, one applies Proposition 2.21 to get the following chain of (in)equalities, for every $i \in \{0, \ldots, k-1\}$:

$$\partial(s_2 H_i) \geq \partial(t_2 H_i) = \partial(s_2 G_i) > \partial(t_2 G_i) = \partial(s_2 H_{i+1}).$$

By induction on i, one proves the following chain of (in)equalities:

$$\partial(t \star_1 \blacksquare) = \partial(s_2 G_1) > \partial(s_2 G_2) > \cdots > \partial(s_2 G_k) > \partial(t_2 G_k).$$

Furthermore we have $\partial(t_2 G_k) \geq 0$ and, consequently:

$$|||F|||_{\mathcal{P}_3^R} \;\leq\; \partial(t \star_1 \blacksquare).$$

Finally, let us bound $\partial(t \star_1 \blacksquare)$, which is equal to $\partial\blacksquare \circ \varphi(t) + \partial t$, thanks to the differential relations of ∂. But (φ, ∂) is cartesian, yielding $\partial t = 0$, and Proposition 3.4 tells us that $\varphi(t) \leq \gamma \nu(t)$ holds. One uses the definition of $P_{\blacksquare}$ to conclude. □

Proposition 3.20. *Let $\mathcal{P}$ be a polygraphic program, equipped with an additive and cartesian polygraphic interpretation (φ, ∂) which is strictly compatible with and conservative on every computation 3-cells. Let $\blacksquare$ be a function 2-cell and let t be a value of type $s_1\blacksquare$. Then, for every 3-path F with source $t \star_1 \blacksquare$, the following inequality holds:*

$$|||F||| \;\leq\; Q_{\blacksquare} \circ \nu(t).$$

Proof. If $|||F||| = 0$, then the inequality does hold. Otherwise, there exists a 3-cell that we can apply to the starting 2-path $t \star_1 \blacksquare$; moreover, this is a computation 3-cell since no structure 3-cell can be applied to such a 2-path. Hence the 3-path F decomposes this way:

$$F \;=\; G_1 \star_2 H_1 \star_2 G_2 \star_2 \cdots \star_2 G_k \star_2 H_k,$$

where each G_i is elementary in $\langle \mathcal{P}_3^R \rangle$ and each H_j is in $\langle \mathcal{P}_3^S \rangle$. As a consequence, we have:

$$|||F||| \;=\; k + |||H_1||| + \cdots + |||H_k||| \,.$$

Furthermore $k = |||F|||_{\mathcal{P}_3^R}$ holds and, thus, so does $k \leq P_{\blacksquare} \circ \nu(t)$ thanks to Proposition 3.19. We prove that the following inequality holds to conclude:

$$|||H_1||| + \cdots + |||H_k||| \;\leq\; k \cdot \big(S_{\blacksquare} \circ \nu(t)\big).$$

Towards this goal, let us fix an $i \in \{1, \ldots, k\}$. Since ∂_φ^S is strictly compatible with every structure 3-cell, one gets from Proposition 2.21:

$$|||H_i||| + \partial_\varphi^S(t_2 H_i) \;\leq\; \partial_\varphi^S(s_2 H_i).$$

Furthermore, from Lemma 3.16, one knows that the following inequality holds:

$$\partial_\varphi^S(t_2 G_i) \;\leq\; \partial_\varphi^S(s_2 G_i) + S_{\blacksquare} \circ \nu(t).$$

Since $t_2 G_i = s_2 H_i$ holds, one has:

$$|||H_i||| + \partial_\varphi^S(t_2 H_i) \;\leq\; \partial_\varphi^S(s_2 G_i) + S_{\blacksquare} \circ \nu(t).$$

Or, written differently:

$$|||H_i||| \;\leq\; \partial^S_\varphi(s_2G_i) - \partial^S_\varphi(t_2H_i) + S_{\square} \circ \nu(t).$$

One sums this family of k inequalities, one for every i in $\{1,\ldots,k\}$, to produce:

$$|||H_1||| + \cdots + |||H_k||| \;\leq\; \sum_{i=1}^{k} \partial^S_\varphi(s_2G_i) - \sum_{i=1}^{k} \partial^S_\varphi(t_2H_i) + k \cdot S_{\square} \circ \nu(t).$$

By hypothesis, one has $s_2G_1 = t \star_1 \nabla$, $t_2H_k = t_2F$ and, for every $i \in \{1,\ldots,k\}$, $t_2H_i = s_2G_{i+1}$, so that the following inequality holds:

$$|||H_1||| + \cdots + |||H_k||| \;\leq\; \partial^S_\varphi(s_2F) - \partial^S_\varphi(t_2F) + k \cdot S_{\square} \circ \nu(t).$$

Finally, one argues that both $\partial^S_\varphi(t \star_1 \square) = 0$ and $\partial^S_\varphi(t_2F) \geq 0$ hold by definition of ∂^S_φ. □

Example 3.21. Let us compute these bounding maps for the fusion sort function 2-cell $\bullet$ of the polygraphic program $\mathcal{F}$:

$$P_{\bullet}(2x+1) \;=\; 2x^2+1 \qquad \text{and} \qquad Q_{\bullet}(2x+1) \;=\; (2x^2+1)\cdot\big(1+(2x+1)^2\big).$$

Let us fix a list $[i_1;\ldots;i_n]$ of natural numbers. One can check that, in $\mathcal{F}$, this list is represented by a 2-path t such that $\varphi(t) = ||t|| = 2n+1$. The polynomial $P_{\bullet}$ tells us that, during the computation of the sorted list $[\![\bullet]\!]\,(t)$, there will be at most $2n^2+1$ applications of computation 3-cells. The polynomial $Q_{\bullet}$ bounds the total number of 3-cells of any type.

For example, when n is 2, one computes $[\![\bullet]\!]\,(t)$ by building a 3-path of size at most $Q_{\bullet}(5) = 234$, containing no more than $P_{\bullet}(5) = 9$ computation 3-cells. One can check that the 3-path presented in Example 1.28 is (way) below these bounds: it is made of seven 3-cells, six of which are of the computation kind.

3.4. **Polygraphic programs and polynomial-time functions.**

Definition 3.22. Let $\mathcal{P}$ be a polygraphic program. A differential interpretation ∂ of $\mathcal{P}$ is *polynomial* when, for every function 2-cell $\square$, the map $\partial_{\square}$ is bounded by a polynomial. A functorial interpretation φ of $\mathcal{P}$ is *polynomial* when ∂_φ is. A polygraphic interpretation is *polynomial* when both its functorial part and differential part are.

We denote by $\mathbf{P}$ the set of polygraphic programs which are confluent and complete and which admit an additive, cartesian and polynomial polygraphic interpretation that is conservative on and strictly compatible with their computation 3-cells.

Example 3.23. As a consequence of previous results, the two polygraphic programs $\mathcal{D}$, computing euclidean division, and $\mathcal{F}$, computing the fusion sort of lists, are in $\mathbf{P}$.

Definition 3.24. Let us denote by $\mathcal{N}$ the polygraphic program with the following cells:

(1) It has one 1-cell `n`.
(2) Its 2-cells are the three possible structure 2-cells plus:
   (a) Constructor 2-cells: ○ for zero and ○| for the successor.
   (b) Function 2-cells: ∇ for addition and ▼ for multiplication.
(3) Its 3-cells are the eight structure 3-cells plus the following computation 3-cells:

Proposition 3.25. *The polygraphic program $\mathcal{N}$ is in* **P** *and it computes the addition and multiplication of natural numbers.*

Proof. The polygraphic program $\mathcal{N}$ is orthogonal, hence locally confluent, and complete. Furthermore, the following hold:

$$\llbracket \mathtt{n} \rrbracket \simeq \mathbb{N}, \qquad \llbracket \triangledown \rrbracket(\underline{m}, \underline{n}) = \underline{m+n}, \qquad \llbracket \blacktriangledown \rrbracket(\underline{m}, \underline{n}) = \underline{mn}.$$

Then, one checks that the following polygraphic interpretation has all the required properties:

$$\varphi(\mathtt{n}) = \mathbb{N} - \{0\}, \qquad c_{\circ} = c_{\phi} = 1, \qquad \varphi(\triangledown)(x,y) = x+y, \qquad \varphi(\blacktriangledown)(x,y) = xy,$$

$$\partial\triangledown(x,y) = x \qquad \text{and} \qquad \partial\blacktriangledown(x,y) = (x+1)y. \qquad \square$$

Remark 3.26. So $\mathcal{N}$ computes addition and multiplication of natural numbers. As we have seen, it also computes duplication and permutation on them. As a consequence, for every polynomial P in $\mathbb{N}[x]$, one can choose a 2-path P in $\mathcal{N}$ such that $\llbracket P \rrbracket$ is P. Moreover, by induction, one proves that $\varphi(P) = P$ and that ∂P is bounded by a polynomial in $\mathbb{N}[x]$.

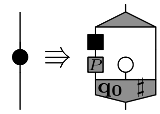
Theorem 3.27. *The polygraphic programs of* **P** *compute exactly the* FPTIME *functions.*

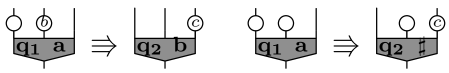
Proof. The fact that a function computed by a polygraphic program in **P** is in FPTIME is a consequence of the results of Proposition 3.20. Indeed, it proves that the size of any computation of $\llbracket \square \rrbracket$ is bounded by $Q_{\square}$ applied to the size of the arguments: from the polynomial assumption and the definition of $Q_{\square}$, this map is itself bounded by a polynomial. Moreover each 3-cell application modifies only finitely many 2-cells: hence the sizes of the 2-paths remain polynomial all along the computation. Furthermore, any step of computation can be done in polynomial time with respect to the size of the current 2-path. Indeed, it corresponds to finding a pattern and, then, replace it by another one: it is just a reordering of some pointers with a finite number of memory allocations. So, the computation involves a polynomial number of steps, each of which can be performed in polynomial time. Thus, the normalization process can be done in polynomial time.

Conversely, let $f : \langle\Sigma\rangle \to \langle\Sigma\rangle$ be a function of class FPTIME. This means that there exists a Turing machine $\mathcal{M} = (\Sigma, Q, q_0, q_f, \delta)$ and a polynomial P in $\mathbb{N}[x]$ such that the machine $\mathcal{M}$ computes f and, for any word w of length n in $\langle\Sigma\rangle$, the number of transition steps required by $\mathcal{M}$ to compute $f(w)$ is bounded by $P(n)$. We extend the polygraphic program $\mathcal{N}$ into $\mathcal{P}(\mathcal{M}, P)$, by adding the following extra cells, adapted from the ones of the polygraphic Turing machine $\mathcal{P}(\mathcal{M})$ used in the proof of Theorem 1.30, in order to use P as a clock:

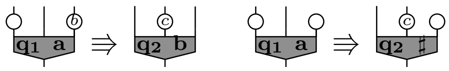
(1) An extra 1-cell $\mathtt{w}$.
(2) Extra 2-cells include the five new structure 2-cells plus:
   (a) Constructor 2-cells: the empty word $\circ : * \Rightarrow \mathtt{w}$ and each letter $\phi : \mathtt{w} \Rightarrow \mathtt{w}$ of Σ.
   (b) Function 2-cells: the main $\bullet : \mathtt{w} \Rightarrow \mathtt{w}$ for f, plus the modified **q a**, $q \in Q$ and $a \in \overline{\Sigma}$, now from $\mathtt{n} \star_0 \mathtt{w} \star_0 \mathtt{w}$ to $\mathtt{w}$, plus an extra size function $\blacksquare : \mathtt{w} \Rightarrow \mathtt{n}$.
(3) Extra 3-cells include the new structure ones plus:
   (a) The computation 3-cells for the auxiliary function $\blacksquare$:

   (b) Timed versions of the computation 3-cells for the Turing machine:

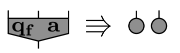




both when $\delta(q_1, a) = (q_2, c, L)$



both when $\delta(q_1, a) = (q_2, c, R)$



One checks that $\mathcal{P}(\mathcal{M}, P)$ is orthogonal and complete. We equip it with the polygraphic interpretation based on the one defined on $\mathcal{N}$ in the proof of Proposition 3.25, extended with the following values:

$$c_{\circ} = c_{\odot} = 1,$$

$$\varphi(\blacksquare)(x) = x, \qquad \varphi(\boxed{\mathbf{q}\ \mathbf{a}})(x, y, z) = x + y + z, \qquad \varphi(\bullet)(x) = P(x) + x + 1,$$

$$\partial\blacksquare(x) = \partial\boxed{\mathbf{q}\ \mathbf{a}}(x, y, z) = x, \qquad \partial\bullet(x) = \partial\boxed{P}(x) + P(x) + x + 1.$$

One checks that this polygraphic interpretation is additive, cartesian, polynomial, compatible with and conservative on all the computation 3-cells. Hence, $\mathcal{P}(\mathcal{M}, P)$ is a polygraphic program in $\mathbf{P}$. Furthermore, one has $[\![\mathtt{n}]\!] \simeq \mathbb{N}$ and $[\![\mathtt{w}]\!] \simeq \langle\Sigma\rangle$. We also note that, among functions computed by $\mathcal{P}(\mathcal{M}, P)$, one proves that $[\![\blacksquare]\!] : [\![\mathtt{w}]\!] \to [\![\mathtt{n}]\!]$ is the length function.

The four middle families of computation 3-cells of $\mathcal{N}$ are once again in bijection with the rules defining the transition relation of the Turing machine $\mathcal{M}$. Hence, the configuration (q, a, w_l, w_r) reduces into (q', a', w'_l, w'_r) in $k \in \mathbb{N}$ steps if and only if, for any $n \geq k$, one has:

$$\big(\underline{n} \star_0 \underline{w_l} \star_0 \underline{w_r}\big) \star_1 \boxed{\mathbf{q}\ \mathbf{a}} \quad \Rrightarrow \quad \Big(\underline{n-k} \star_0 \underline{w'_l} \star_0 \underline{w'_r}\Big) \star_1 \boxed{\mathbf{q'}\ \mathbf{a'}}.$$

Finally, let us fix a word w of length n in $\langle\Sigma\rangle$. The Turing machine computes f, so that $(q_0, \sharp, e, w)$ reduces into a unique configuration $(q_f, a, v, f(w))$, after a finite number k of transition steps. Then we check the following chain of equalities, yielding $[\![\bullet]\!] = f$:

$$[\![\bullet]\!]\,(\underline{w}) \;=\; \Big[\!\!\Big[\boxed{\mathbf{q_0}\ \sharp}\Big]\!\!\Big]\Big(\underline{P(n)} \star_0 \circ \star_0 \underline{w}\Big) \;=\; \Big[\!\!\Big[\boxed{\mathbf{q_f}\ \mathbf{a}}\Big]\!\!\Big]\Big(\underline{P(n)-k} \star_0 \underline{v} \star_0 \underline{f(w)}\Big) \;=\; \underline{f(w)}. \qquad \square$$

Future directions

Polygraphic programs. The definition we have chosen for this study stays close to the one of first-order functional programs. We shall explore generalization along different directions.

We think that an important research trail concerns the understanding of the algebraic properties of the *if-then-else* construction in polygraphic terms. Towards this goal, we want to describe strategies as sets of 4-dimensional cells. The 3-paths will contain all the computational paths one can build when there is no fixed evaluation strategy, while the strategies and conditions will be represented by the 4-paths, seen as normalization processes of 3-paths. In particular, this setting shall allow us to internalize the test used to compute the merge function in the fusion sort algorithm, but also to describe conditional or probabilistic rewriting systems.

On another point, in the polygraphs we consider here, we have fixed a sublayer made of permutations, duplications and erasers, together with natural polygraphic interpretations for them. However, one can see them as a special kind of function 2-cells. Thus, we shall define a notion of hierarchical programs, where one builds functions level after level, giving complexity bounds for them modulo the previously defined functions. However, this does not prevent us to build modules that a programmer can freely use as sublayers, without bothering with the complexity of their functions: for example, a module that describes the evaluation and coevaluation. We think of this module system as a first possibility to integrate polymorphism into the polygraphic setting.

Removing duplication and erasure from the standard definition means that one moves from a cartesian setting to a monoidal one. According to a variant of André Joyal's paradox [29], this is necessary to describe functions such as linear maps on finite-dimensional vector spaces. Thus, one should be able to compute, for example, algebraic cooperations, such as the ones found in Jean-Louis Loday's generalized bialgebras [33], or automorphisms of $\mathbb{C}^n$, such as the universal Deutsch gate [37] of quantum circuits.

Going further, at this step, there will be no reason anymore to consider constructor 2-cells with one output only or values with no output. This way, one could consider algorithms computing, for example, on braids or knots. However, this also suggests to change our notion of function 2-cells to some kind of "polygraphic context", a notion of 2-path with holes whose algebraic structure has yet to be understood. In particular, this is the second solution we think of to describe polymorphic types and functions.

For all this research, we shall consider a more abstract definition of polygraphs: they are special higher-dimensional categories, namely the free ones. This formulation, though leading to a steeper learning curve, shall provide enlightenments about the possibilities one has when one wants to extend the setting. But, more importantly, this will make easier the adaptation of tools from algebra for program analysis.

Analysis tools. In future work, we shall use other possibilities provided by polygraphic interpretations, together with other algebraic tools, to study the computational properties of polygraphs.

We restricted interpretations to be polynomials with integer coefficients. This is close to the tools considered in [8]. Following this last paper, a straightforward characterization of exponential-time (resp. doubly exponential-time) can be done by considering linear (resp. polynomial) interpretations for constructors, instead of additive ones. However, some studies are much more promising. First, to turn to polynomials over reals give some procedures to build interpretations (see [11]) via Alfred Tarski's decidability [44]. Second, we plan to consider differential interpretations with values in multisets (instead of natural numbers), to characterize polynomial-space computations.

For each generalization of the notion of polygraphic program, such as the ones mentioned earlier, we shall adapt polygraphic interpretations in consequence. We think that, if these generalizations are done in an elegant way, this task will be easier. For example, if one considers "symmetric" values, i.e., values with inputs, one can use a third part of polygraphic interpretations we have not used here: ascending currents, described by a contravariant functorial part, such as in the original definition [18].

As pointed earlier, polygraphs are higher dimensional-categories. Philippe Malbos and the second author are currently adapting the finite derivation criterion of Craig Squier [40] to them, as was done before for 1-categories [34]. We think that this will lead us to a computable necessary condition to ensure that a function admits a finite, convergent polygraphic program that computes it.

The same collaboration has more long-term aims: using tools from homological algebra for program analysis. For example, the functorial and differential interpretations are special cases of, respectively, left modules over the 2-category of 2-paths (or bimodules, when there are ascending currents) and derivations of this same 2-category into the given module. Moreover, a well-chosen cohomology theory yields, in particular, information on derivations: thus, one can hope to get new tools such as negative results about the fact that a given algorithm lives in a given complexity class.

Cat. The main concrete objective of this project is to develop a new programming language, codenamed Cat. In this setting, one will build a program as a polygraph, while using the algebraic analysis tools we provide to produce certificates that guarantee several properties of the code, such as grammatical ones, computational ones or semantical ones. As in Caml [13], a Cat program will have two aspects: an implementation and an interface.

In the implementation, one builds the code, describing the cells and assembling them to build paths, i.e., building the data types, the functions, the computation rules and the evaluation strategies. Thanks to the dual nature of polygraphs, one shall be able to perform this using an environment that is either totally graphical, totally syntactical or some hybrid possibility between those.

The interface part contains all the information the programmer can prove on its code, in the form of certificates. These guaranteed properties will range from type information, as in Caml, to polygraphic interpretations proving termination or giving complexity bounds, to proofs of semantical properties in the form of polygraphic three-dimensional proofs [19]. For all these certificates, we shall propose assistants, with tactics that automatize the simpler tasks and leave the programmer concentrate on the harder parts.

Finally, given such a polygraphic program, the question of evaluation arises. One can think of several solutions, whose respective difficulty ranges from "feasible" to "science-fiction": first, a compiler or an interpreter into some existing language, such as Tom [45], a task that has already been started; then, a distributed execution where each 2-cell is translated into a process, whose behaviour is described by the corresponding 3-cells; finally, concrete electronic chips dedicated to polygraphic computation.

References

[1] Colin Adams, *The knot book*, American Mathematical Society, 2004.

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

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

[4] John Baez and Aaron Lauda, *A history of n-categorical physics*, draft version, 2006.

[5] Patrick Baillot and Kazushige Terui, *Light types for polynomial time computation in lambda-calculus*, Proceedings of the 19th Symposium on Logic in Computer Science (LICS 04), 2004, pp. 266–275.

[6] Stephen Bellantoni and Stephen Cook, *A new recursion-theoretic characterization of the poly-time functions*, Computational Complexity **2** (1992), 97–110.

[7] Guillaume Bonfante, Adam Cichon, Jean-Yves Marion, and Hélène Touzet, *Complexity classes and rewrite systems with polynomial interpretation*, Proceedings of the 12th International Workshop on Computer Science Logic (CSL 98), Lecture Notes in Computer Science, vol. 1584, 1999, pp. 372–384.

[8] ———, *Algorithms with polynomial interpretation termination proofs*, Journal of Functional Programming **11** (2001), no. 1, 33–53.

[9] Guillaume Bonfante and Yves Guiraud, *Intensional properties of polygraphs*, Proceedings of the 4th International Workshop on Term Graph Rewriting (TERMGRAPH 07), vol. 203, Electronic Notes in Computer Science, no. 1, 2008, pp. 65–77.

[10] Guillaume Bonfante, Jean-Yves Marion, and Jean-Yves Moyen, *Quasi-interpretations: a way to control resources*, Theoretical Computer Science (2005), to appear.

[11] Guillaume Bonfante, Jean-Yves Marion, Jean-Yves Moyen, and Romain Péchoux, *Synthesis of quasi-interpretations*, Logic and Complexity in Computer Science, 2005.

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

[13] The Caml Language, *caml.inria.fr*.

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

[15] Daniel Dougherty, Pierre Lescanne, and Luigi Liquori, *Addressed term rewriting systems: application to a typed object calculus*, Mathematical Structures in Computer Science **16** (2006), no. 4, 667–709.

[16] Jean-Yves Girard, *Light linear logic*, Information and Computation **143** (1998), no. 2, 175–204.

[17] Yves Guiraud, *Présentations d'opérades et systèmes de réécriture*, Ph.D. thesis, Université Montpellier 2, June 2004.

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

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

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

[21] ______, *Polygraphs for termination of left-linear term rewriting systems*, preprint, 2007.

[22] Dieter Hofbauer and Clemens Lautemann, *Termination proofs and the length of derivations*, Lecture Notes in Computer Science **355** (1988), 167–177.

[23] Martin Hofmann, *A type system for bounded space and functional in-place update*, Nordic Journal of Computing **7** (2000), no. 4, 258–289.

[24] Neil Jones, *Computability and complexity, from a programming perspective*, MIT Press, 1997.

[25] ______, *Logspace and ptime characterized by programming languages*, Theroretical Computer Science **228** (1999), 151–174.

[26] Yves Lafont, *Interaction nets*, Proceedings of the 17th Symposium on Principles of Programming Languages (POPL 90), 1990, pp. 95–108.

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

[28] ______, *Soft linear logic and polynomial time*, Theoretical Computer Science **318** (2004), 163–180.

[29] Joachim Lambek and Philipp Scott, *Introduction to higher-order categorical logic*, Cambridge University Press, 1986.

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

[31] Daniel Leivant, *A foundational delineation of computational feasability*, Proceedings of the 6th Symposium on Logic in Computer Science (LICS 91), 1991, pp. 2–11.

[32] Daniel Leivant and Jean-Yves Marion, *Lambda-calculus characterizations of poly-time*, Fundamenta Informaticae **19** (1993), no. 1-2, 167–184.

[33] Jean-Louis Loday, *Generalized bialgebras and triples of operads*, preprint, 2006.

[34] Philippe Malbos, *For string rewriting systems the homotopical and homological finiteness conditions coincide*, preprint, 2007.

[35] Jean-Yves Marion and Romain Péchoux, *Resource analysis by sup-interpretation*, Lecture Notes in Computer Science **3945** (2006), 163–176.

[36] Yiannis Moschovakis, *What is an algorithm?*, Mathematics Unlimited – 2001 and Beyond (Björn Engquist and Wilfried Schmid, eds.), Springer, 2001, pp. 919–936.

[37] Michael Nielsen and Isaac Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.

[38] Karl-Heinz Niggl and Henning Wunderlich, *Certifying polynomial time and linear/polynomial space for imperative programs*, SIAM Journal of Computing **35** (2006), no. 5, 1122–1147.

[39] Detlef Plump, *Term graph rewriting*, Handbook of Graph Grammars and Computing by Graph Transformation **2** (1999), 3–61.
[40] Craig Squier, *A finiteness condition for rewriting systems*, Theoretical Computer Science **131** (1994), no. 2, 271–294, Revised by Friedrich Otto and Yuji Kobayashi.
[41] Ross Street, *Limits indexed by category-valued* 2*-functors*, J. Pure Appl. Algebra **8** (1976), no. 2, 149–181.
[42] ———, *The algebra of oriented simplexes*, J. Pure Appl. Algebra **49** (1987), no. 3, 283–335.
[43] ———, *Higher categories, strings, cubes and simplex equations*, Appl. Categ. Structures **3** (1995), no. 1, 29–77.
[44] Alfred Tarski, *A decision method for elementary algebra and geometry*, University of California Press, 1951, 2nd edition.
[45] Tom, *tom.loria.fr*.
[46] John von Neumann, *Theory of self-reproducing automata*, University of Illinois Press, 1966.