Programming in the Geometric Machine

Renata H. S. Reiser, Antônio Carlos R. Costa* and Graçaliz P. Dimuro
{reiser, rocha, liz}@atlas.ucpel.tche.br
Universidade Católica de Pelotas, Escola de Informática
Pelotas, CP 402, Brazil

Abstract. This paper presents the programming language induced by the ordered structure of the Geometric Machine Model (GMM). The GMM is an abstract machine model, based on Girard's coherence space, capable of modelling sequential, alternative, parallel (synchronous) and non-deterministic computations on a (possibly infinite) shared memory. The processes of the GMM are inductively constructed in a Coherence Space of Processes. The memory of the GMM, supporting a coherence space of states, is conceived as the set of points of a three dimensional euclidian space. Using the programming language induced by such structure, simple recursive equations and related algorithms are presented, and their domain-theoretic semantics in the machine model is given.

1 Introduction.

Inspired by the work of Scott [7], a constructive and intuitive machine model, with infinite memory, called *Geometric Machine Model* (GMM) is introduced in [5]. Over the GMM ordered structure, we obtain representations of (non-)deterministic processes, labelled by positions of a geometric space, including two special types of parallelism – the temporal and the spacial parallelism. Its most basic notion is that of a coherence relation representing the admissibility of parallelism between two or more basic operations (elementary processes). That relation defines a web over which a coherence space [3] of parallel processes is step-wise and systematically built. In the dual construction, the incoherence relation interprets the condition that models non-determinism. The sequential product and the deterministic sum of parallel and non-deterministic processes are endofunctors in the category of such coherence spaces (with linear functions as morphisms). Following this approach, aspects of Domain Theory [1] and Concurrency Theory [4] are connected to obtain a programming language derived from ordered structure of the GMM, denoted by $\mathcal{L}(\mathbb{D}_\infty)$. In this sense, the aim of this work is the development of examples of the recursive equations defined in the language $\mathcal{L}(\mathbb{D}_\infty)$ and over that it is possible to sample the semantic analysis of concurrent and distributed systems interpreted in such model. Some (possibly recursive) definitions of constructors, the functional composition, the parallel and non-deterministic computations related to synchronization of processes and finally, the spatial definition of a computational process based on a geometric space is presented. This paper is structured as follows. The Section 2 summarizes the ordered structure for the memory states and possible programs in the GMM. A computational state is defined as a linear function from the coherence space of labels to coherence space of values.

*PPPGC/UFRGS, Porto Alegre, CP 15064, Brazil

Labels are intuitively understood as points of a geometric space, so that a computational state is an assignment of values to points of the geometric space and processes are then related to the notion of transformations between computational states. The language derived form the GM model is briefly presented in Section 3. In the following sections recursive equations are illustrated, but only in uni-dimensional memories. In the conclusion, further work and possible applications of the model are mentioned.

2 The Geometric Machine Model

The GM model presented in [5], is an abstract computational machine, with infinite memory and constant memory access time that can be used to analyze the logic and ordered structure of parallel and non-deterministic algorithms. The constructive, intuitive and ordered structure of the Geometric Machine Model is given by the Coherence Space of Processes, denoted by $\mathbb{D}_\infty$, based on the concept of Coherence Spaces, first introduced by Girard [3]. Over $\mathbb{D}_\infty$ it is possible to give interpretation to partial, recursive (possibly) infinite processes, operating on the array structures storable in the GM memory. In addition, the GM memory, supporting a coherence space of states, is conceived as an enumerable set of values labelled by points of a geometric space.

Deterministic machine states are modelled as functions from memory positions to values and non-deterministic machine states are modelled as families of deterministic machine states (with singletons modelling deterministic states). To represent the deterministic sums of processes, we consider the set B of *boolean tests* related to the binary logic and assume the Coherence Space of Boolean Tests as an ordered structure representing the set of all coherent subsets of tests b. Non-determinism enforces a non-traditional treatment of tests. For each boolean test capable of testing a deterministic state s, we consider two forms for tests $\mathbf{b}$: a universal form ($\mathbf{b}_\forall(\mathbf{s}) \equiv \forall s \in \mathbf{s} \,.\, b(s)$) and an existential form ($\mathbf{b}_\exists(\mathbf{s}) \equiv \exists s \in \mathbf{s} \,.\, b(s)$). Both forms coincide with the simple test b when $\mathbf{s}$ is a deterministic singleton set (see Figure 1(a)). This interpretation can be applied to deterministic process constructions, including two special types of parallelism - the temporal parallelism related to the inductive construction (see Figure 1(e)), and the spatial parallelism, with processes defined over array structures, that operate in a synchronized way. The model also provides interpretation to the non-deterministic computations and applies the exponential operator of coherence spaces in the interpretation of the functional space.

Figure 1: The inductive ordered construction of GM model.

The most basic notion of this work is the definition of the coherence relation as the admissibility of parallelism between basic operations called *elementary process*, which modify a single memory position in a single unit of computational time (uct). Its intuitive notion can be described by an elementary transition between deterministic memory states, given by a linear function. Its representation in a one-dimensional machine is in Figure 1(a). The algebraic process constructors are defined by the endofunctors in the $\mathcal{C}osp\mathcal{L}in$ category of the coherence spaces and linear functions. The ordered structure of this model is constructed by levels from the coherence space of the elementary processes, following the Scott's methodology [7].

The basic level of the inductive construction of $\mathbb{D}_\infty$ stars with $\mathbb{D}_0$ denoting the flat *coherence space of elementary processes*. See Figure 1. The next coherence space in the con-

struction, denoted by $\bar{\mathbb{D}}_0$, gives interpretation for *concurrent sets of elementary processes*. The coherence relation between such processes models the admissibility of parallelism between them and essentially says that two elementary processes can be performed in parallel if they do not conflict, i. e., if they do not access the same memory position. That relation defines also the web over which the coherence space of the whole set of processes in the model is step-wise and systematically build. The domain $\bar{\mathbb{D}}_0$ is thus the *domain of parallel products of elementary processes*. See Figure 1(b). In the dual construction, $\bar{\mathbb{D}}_0^{\perp}$, justified by the presence of involutive negation $\perp$ of coherence spaces [3], the incoherence relation models the condition for *non-determinism*, namely, the conflict of memory accesses. See Fig. 1(c). For the construction of the sequential product and the deterministic sum, consider $\mathbb{P}_0 \equiv \mathbb{D}_0 \coprod \bar{\mathbb{D}}_0 \coprod \bar{\mathbb{D}}_0^{\perp}$ as the amalgamated (smash) sum of $\mathbb{D}_0$, $\bar{\mathbb{D}}_0$ and $\bar{\mathbb{D}}_0^{\perp}$. The direct product $\mathbb{P}_0 \prod \mathbb{P}_0$ is the coherence space of sequential products of two (parallel, non-deterministic or elementary) processes, whose execution is performed in $2\,uct$. Figure 1(d) presents a graphic representation of a sequential product between elementary processes. The coherence space $\mathbb{P}_0 \prod_{\mathbb{B}} \mathbb{P}_0$ of the deterministic sums of (parallel, non-deterministic or elementary) processes, performed in $2\,uct$, is defined as the direct product between $\mathbb{B}$ and $\mathbb{P}_0 \prod \mathbb{P}_0$. A graphic representation of a non-deterministic sum between elementary processes is pictured in Figure 1(e). We can put all the above together, in order to obtain the domain

$$\mathbb{D}_1 = \mathbb{P}_0 \coprod (\mathbb{P}_0 \prod \mathbb{P}_0) \coprod (\mathbb{P}_0 \prod_{\mathbb{B}} \mathbb{P}_0) \quad \text{where } \mathbb{P}_0 = \mathbb{D}_0 \coprod \bar{\mathbb{D}}_0 \coprod \bar{\mathbb{D}}_0^{\perp}.$$

The coherence space $\mathbb{D}_1$ encompasses the first step of the construction of the ordered structure of the GMM and provides the representations for all processes performed in at most $2uct$.

Generalizing, each level is identified by a subspace $\mathbb{D}_n$, which reconstructs all the objects from the level before, preserving their properties and relations, and drives the construction of the new objects. Compatible with the algebraic-theoretic approach to computational processes, the relationship between the levels is expressed by linear functions called embedding π_n and projection-functions Π_n, which interpret constructors and destructors of processes, respectively. Induced by the flat domain modelling of the geometric space, the memory position function defined on the domain $\mathbb{D}_0$ of elementary processes can be lifted to the coherent sets of the subsequent constructed domains by a general position-function, formally presented in [5], concerned with the concurrency and conflict relations in such domains. Each element in the web of a domain $\mathbb{D}_{n+1}$ is indexed by two or tree symbols. The leftmost symbol of an index indicates one of the following constructors – (0) (for the simple inclusion of an element of the previous domain $\mathbb{D}_n$ in the new domain $\mathbb{P}_n$ related to the sub-level $\mathbb{D}_n - \mathbb{P}_n$), (1) (indicating the parallel product of elements existing in the previous domain $\mathbb{P}_n$) or (2) (indicating the non-deterministic sum of elements existing in the previous level). The second and third symbols related to the sub-level $\mathbb{P}_n - \mathbb{D}_{n+1}$, if present, mean the following: that the element is the first (02) or the second (12) summand in a deterministic sum represented in $\mathbb{P}_n \prod_{\mathbb{B}} \mathbb{P}_n$, or that it is the first (01) or the second (11) term in a sequential product $\mathbb{P}_n \prod \mathbb{P}_n$. When the index is given by two symbols, the second one indicates the inclusion of the element in the domain $\mathbb{P}_n$ in the new domain $\mathbb{D}_{n+1}$.

The completion procedure guarantees the existence of the least fixed point to the recursive equations defined by infinite composition of morphisms in the coherence space $\mathbb{D}_\infty$. The indexed tokens of coherent subsets in $\mathbb{D}_\infty$ are expressed following the denotation:

1. $: 00 \equiv 00.00.00\ldots$ related to finite processes;
2. $: 001 \equiv 001.001\ldots$ related to infinite sequential processes;

3. $: 002 \equiv 002.002\ldots$ related to infinite deterministic sums.

Figure 1(f) shows a representation for an infinite composition of the morphism representing an infinite sequential product.

3 The textual language $\mathcal{L}(\mathbb{D}_\infty)$

We take use of $\mathbb{D}_\infty$ to obtain a programming language for implementing parallel and non-deterministic algorithms in the GMM. Let $\mathcal{K}$ be the set of constant symbols given by the union $K = I_P \bigcup I_T$, where I_P and I_T denote the set of symbols representing elementary processes (including the `skip` process) and boolean tests of $\mathbb{D}_\infty$, respectively. In addition, let $F_{Op} = \{Id, \|, |, \cdot, +\}$ be the set of symbols representing the constructors of processes:
(i) $Id \in I_P$ is the identity elementary process.
(ii) $\|, |, \cdot : I_P \times I_P \rightarrow I_P$ are binary symbols representing the parallel product, sequential product and non deterministic sum;
(iii) $+ : I_P \times I_P \times I_T \rightarrow I_P$ is a 3-arity symbol representing the deterministic sum, with $\forall b \in I_T$, $+_b : I_P \times I_P \rightarrow I_P$.

The set $\mathcal{L}(\mathbb{D}_\infty)$ of expressions of the language of $\mathbb{D}_\infty$ is defined by:

(i) Variables and the above constant symbols are expressions of $\mathcal{L}(\mathbb{D}_\infty)$.

(ii) If $* \in \{\|, |, \cdot, +_b\}$ and $t_0, t_1, \ldots, t_n, t_{n+1}, \ldots b \in \mathcal{L}(\mathbb{D}_\infty)$ then $*_{i=n+1}^{0}(t_i) = t_{n+1} * t_n * \ldots * t_0$ and $*_{i=0}^{n+1}(t_i) = t_0 * \ldots * t_n * t_{n+1}$ are (finite) expressions of $\mathcal{L}(\mathbb{D}_\infty)$;

(iii) If $t_0, t_1, \ldots, t_n, t_{n+1}, \ldots \in \mathcal{L}(\mathbb{D}_\infty)$ then $*_{n=0}^{\infty} t_n = t_0 * t_1 * \ldots * t_{n+1} * \ldots$ is an (infinite) expression of $\mathcal{L}(\mathbb{D}_\infty)$.

We say that each expression of $\mathcal{L}(\mathbb{D}_\infty)$ is a *representation* of a process of $\mathbb{D}_\infty$, and that the process is the *interpretation* of the expression. The next examples in Section 4 illustrate the resolution of some recursive equations based on the application of the processes constructors. In this examples, the memory and the processes are geometrically conceived as the subset of points corresponding to the natural numbers in the real line.

4 Recursive equations in $\mathcal{L}(\mathbb{D}_\infty)$.

Example 1.

Consider $I \equiv \omega$, $V = \{0, 1\}$ and the computational elements presented in the Table 1. In the following recursive equations,

1. $X_n \equiv$ `zn` $\| X_{n+1}$: X_0 denotes the process that executes, simultaneously, all the elementary processes expressed by `zn` in the Table 1 in $1uct$. After performance, the resulting state has $0 \in V$ in all positions of the GMM memory. Thus, X_0 is represented by the coherent subset $\{\overline{z^{(i)}}_{10:00}\}_I$ in $\mathbb{D}_\infty$.

2. $Y_n \equiv$ `un` $\| Y_{n+1}$: Y_0 describes a constant transformation that executes, simultaneously, all the elementary processes expressed by `un` in the Table 1 in $1uct$. In this case, the resulting computational state has $1 \in V$ in all memory positions and Y_0 is represented by the coherent subset $\{\overline{u^{(i)}}_{10:00}\}_I$ in $\mathbb{D}_\infty$.

Table 1: Elementary processes.

$\mathcal{L}(\mathbb{D}_\infty)$	$\mathbb{D}_\infty$	*computational element*
$\mathsf{t} \equiv \mathsf{k} == \mathsf{n}$	$\{t_{20:00}\}$	computational test
$\mathsf{zn}(s) \equiv (s \mapsto s[i_n := 0])$	$\{z^{(n)}_{:00}\})$	elementary process
$\mathsf{un}(s) \equiv (s \mapsto s[i_n := 1])$	$\{u^{(n)}_{:00}\})$	elementary process
$\mathsf{zn} +_{\mathsf{n==k}} \mathsf{un}$	$\{z^{(n)}_{002:00}, b_{22:00}, u^{(n)}_{012:00}\}$	deterministic sum
$\mathsf{zn} \parallel \mathsf{um}$	$\{\overline{z^{(0)}}_{10:00}, \overline{u^{(1)}}_{10:00}\}$	parallel product
$\mathsf{zn} \mid \mathsf{un}$	$\{\overline{z^{(n)}}_{20:00}, \overline{u^{(n)}}_{20:00}\}$	non-deterministic sum
$\mathsf{zn} \cdot \mathsf{um}$	$\{z^{(n)}_{001:00}, u^{(n)}_{011:00}\}$	sequential product
$\mathsf{copyn}(s) \equiv (s \mapsto s := s[i_{n+1} := i_n])$	$\{copy^{(n+1)}_{:00}\})$	elementary process

3. $Z_n \equiv (\mathsf{zn} \mid \mathsf{un}) \parallel Z_{n+1}$: Z_0 is a description of an arbitrary choice between the values in the set $V = \{0, 1\}$ for each position in the memory, executed in $1\,utc$. Z_0 is represented by the coherent subset $\{\overline{\overline{z^{(i)}}_{20}\overline{u^{(i)}}_{20}}_{10:00}\}_I$ in $\mathbb{D}_\infty$.

4. $\mathsf{kn} \equiv (\mathsf{un} +_{\mathsf{t}} \mathsf{zn})$ and $W_{n,k} \equiv \mathsf{kn} \parallel W_{n+1,k}$: $W_{n,k}$ defines a process that has $1 \in V$ on the k-th position of the resulting computational state, and the value $0 \in V$ in the others memory positions. It is expressed by $\{\overline{u^{(i)}_{002}, z^{(i)}_{012}, t_{22}}_{10:00}\}_I$ in $\mathcal{L}(\mathbb{D})$.

Example 2.

Consider $\{F_i\}_{i\in\omega}$ as a sequence of linear functions $F_i \in \mathcal{F}Lin$ such that $cod(F_i) = X_{i+1} = dom(F_{i+1})$, $X_i \in \{\mathbb{D}_\infty, \mathbb{D}_\infty \prod \mathbb{D}_\infty, \mathbb{D}_\infty \prod_{\mathbb{B}} \mathbb{D}_\infty\}$ where $\mathcal{F}Lin$ is the set of linear functions representing constructors of processes in $\mathbb{D}_\infty$. A finite (infinite) composition of the functions in the sequence $\{F_i\}_{i\in\omega}$ is given by

$$\circledcirc_{i=0}^{n} F_i \;=\; \circledcirc_{i=m+1}^{n} F_i \circ F_m \circ \ldots \circ F_1 \circ F_0 \quad (\circledcirc_{i=0}^{\infty} F_i \;=\; \circledcirc_{i=n+1}^{\infty} F_i \circ \circledcirc_{i=0}^{n} F_i).$$

For constant sequences, $\circledcirc_{i=0}^{n} F_i = F^n$ ($\circledcirc_{i=0}^{\infty} F_i = F^\infty$) and for $n = 0$, $F^0 = Id_{\mathbb{D}_\infty}$.

In order to illustrate the resolution of recursive equations in the domain $\mathbb{D}_\infty$ some examples are presented. They are based on sequential and parallel products represented by the linear functions $\mathbf{F}_{(01)}$ and $\mathbf{F}_{(1)}$, respectively defined in [5].

Example 3.

Consider the sequences of coherent subsets presented in the columns of the Table 2, concerned with the possible solutions of the equation

$$x_{n+1} = \mathbf{F}_{(01)}\,(x_n \sqcap \emptyset). \tag{1}$$

Starting with a unitary coherent subset $x_n^i\mathbb{D}_\infty$, a sequence of solutions is obtained by the iterations of the equation (1). In this case, for each x_n^i, the last row shows $x^i \in \mathbb{D}_\infty^{\rightarrow}$ as the related fixed point of this equation. Thus, $F^\infty_{(01)}(x_n^i \sqcap \emptyset) = x^i$ and, therefore, $x^i = \mathbf{F}_{(01)}(x^i \sqcap \emptyset)$. In addition, the finite union $\bigcup_{i=0}^{n} x^i = \{d^{(k)}_{:001}, d^{(k)}_{011:001}, d^{(k)}_{00.011:001}, \ldots, d^{(k)}_{(00)^{n-1}.011:001}\}$, is also a fixed point for the equation (1) if the iteration starts with

Table 2: Temporal infinite sequential product of spacial infinite parallel processes.

$x'_1 = \{d^{(k)}_{001:00}\}$	$x''_1 = \{d^{(k)}_{011:00}\}$	$\dots$	$x^{(i)}_n = \{d^{(k)}_{(00)^{n-1}.011:00}\}$
$x'_2 = \{d^{(k)}_{(001)^2:00}\}$	$x''_2 = \{d^{(k)}_{011.001:00}\}$	$\dots$	$x^{(i)}_{n+1} = \{d^{(k)}_{(00)^{n-1}.011.001:00}\}$
$\vdots$	$\vdots$		$\vdots$
$x'_n = \{d^{(k)}_{(001)^n:00}\}$	$x''_n = \{d^{(k)}_{011.(001)^n:00}\}$	$\dots$	$x^{(i)}_{2n} = \{d^{(k)}_{(00)^{n-1}.011.(001)^{n-1}:00}\}$
$\vdots$	$\vdots$		$\vdots$
$\bullet\ x' = \mathbf{F}_{(01)}(x' \sqcap \emptyset)$	$\bullet\ x'' = \mathbf{F}_{(01)}(x'' \sqcap \emptyset)$	$\dots$	$\bullet\ x^{(i)} = \mathbf{F}_{(01)}(x^{(i)} \sqcap \emptyset)$
$x' = \{d^{(k)}_{:001}\}$	$x'' = \{d^{(k)}_{011:001}\}$	$\dots$	$x^{(i)} = \{d^{(k)}_{(00)^{n-1}.011:001}\}$

$$\{d^{(k)}_{(001)^n:00}, d^{(k)}_{011.(001)^{n-1}:00}, d^{(k)}_{00.011.(001)^{n-2}:00}, \dots, d^{(k)}_{(00)^{n-1}.011:00}\} \in \pi^{\rightarrow}_{n,\infty}[\mathbb{D}_n].$$

Finally, based on the completion procedure, the process $\mathbf{K}$ (see Figure 2) is represented by the least upper bound (related to the inclusion relation) of the above sequence of fixed points of the equation (1). It is also a fixed point, denoted by $\mathbf{k} \in \mathbb{D}_\infty$ and given by

$$\mathbf{k} = \{d^{(k)}_{:001}, d^{(k)}_{011:001}, d^{(k)}_{00.011:001}, \dots, d^{(k)}_{(00)^{n-1}.011:001}, d^{(k)}_{(00)^{n-1}.011:001}, \dots, d^{(k)}_{:00;011}\}.$$

Observe that $\mathbf{k}$ is a total object representing the sequential product $\mathbf{K}$, step-wise and systematically built, whose factors execute the same action d just in one position of the geometric space, labelled by $k \in I$. It is expressed by $\cdot^{\infty}_{n=0}(\mathsf{dn})^n$ in $\mathcal{L}(\mathbb{D}_\infty)$.

Example 4.

Based on the linear operator $\mathbf{F}_{(10)}$ representing the parallel product in the domain $\mathbb{D}_\infty$, others solutions of the equation (1) are constructed now. For that, consider $k \in \omega = I$ and the objects in the first row of the table 3:

- $x'_1 = \{\overline{d^{(k)}}_{101:00}\}_I = \{\overline{d^0}_{101:00}, \overline{d^1}_{101:00}, \dots, \overline{d^k}_{101:00}, \dots\}$ represents a partial sequential product whose first factor executes the same action d in all positions of a geometric space labelled by the elements of I, simultaneously;
- $x''_1 = \{\overline{d^{(k)}}_{111:00}\}_I = \{\overline{d^0}_{111:00}, \overline{d^1}_{111:00}, \dots, \overline{d^k}_{111:00}, \dots\}$ models the process that displace by one position all factors in x'_1; and
- $x^i_{n+1} = \{\overline{d^{(k)}_{(00)^n.111:00}}\}_I = \{\overline{d^{(0)}_{(00)^n.111:00}}, \overline{d^{(1)}_{(00)^n.111:00}}, \dots, \overline{d^{(k)}_{(00)^n.111:00}}, \dots\}$ represents a partial sequential product. Its last factor executes, simultaneously, the same action d in all positions of a geometric space labelled by the elements of $I = \{0, 1, \dots, k, \dots\}$.

Starting with a coherent subset $x^i_n \in \pi^{\rightarrow}_{n,\infty}[\mathbb{D}_n] \subseteq \mathbb{D}_\infty$, a sequence of solutions is obtained by the iterations of the equation (1). In the same way, for each x^i_n, the last row shows $x^i \in \mathbb{D}^{\rightarrow}_\infty$ as the related fixed point for the equation (1). Thus, $\mathbf{F}^\infty(x^i_n \sqcap \emptyset) = x^i$ and therefore $x^i = \mathbf{F}_{(01)}(x^i \sqcap \emptyset)$. The process $\mathbf{Q}$ pictured in the Figure 3 and represented by the finite union

$$\bigcup x^i = \{\overline{d^{(k)}}_{101:001}\}_I \cup \{\overline{d^{(k)}}_{111:001}\}_I \cup \dots \cup \{\overline{d^{(k)}_{(00)^n.111:001}}\}_I \in \mathbb{D}_\infty$$

Table 3: Temporal infinite sequential product of spacial infinite parallel processes.

$x'_1 = \{\overline{d^{(k)}}_{101:00}\}_I$	$x''_1 = \{\overline{d^{(k)}}_{111:00}\}_I$	$\cdots$	$x^i_{n+1} = \{\overline{d^{(k)}_{(00)^n.111:00}}\}_I$
$x'_2 = \{\overline{d^{(k)}}_{101.001:00}\}_I$	$x''_2 = \{\overline{d^{(k)}}_{111.001:00}\}_I$	$\cdots$	$x^i_{n+1} = \{\overline{d^{(k)}_{(00)^n.111.001:00}}\}_I$
$\vdots$	$\vdots$		$\vdots$
$x'_n = \{\overline{d^{(k)}}_{101.(001)^{n-1}:00}\}_I$	$x''_n = \{\overline{d^{(k)}}_{111.(001)^{n-1}:00}\}_I$	$\cdots$	$x^{(i)}_{2n+1} = \{\overline{d^{(k)}_{(00)^n.111.(001)^n:00}}\}_I$
$\vdots$	$\vdots$		$\vdots$
• $x' = \mathbf{F}_{(01)(11)}(x' \sqcap \emptyset)$	• $x'' = \mathbf{F}_{(01)(11)}(x'' \sqcap \emptyset)$	$\cdots$	• $x^{(i)} = \mathbf{F}_{(01)(11)}(x^{(i)} \sqcap \emptyset)$
$x' = \{\overline{d^{(k)}}_{101:001}\}_I$	$x'' = \{\overline{d^{(k)}}_{111:001}\}_I$	$\cdots$	$x^{(i)} = \{\overline{d^{(k)}_{(00)^n.111:001}}\}_I$

is another fixed point to equation (1), whenever if the iteration start with

$$\{\overline{d^{(k)}}_{101.(001)^n:00}\}_I \cup \{\overline{d^{(k)}}_{111.(001)^n:00}\}_I \cup \ldots \cup \{\overline{d^{(k)}_{(00)^n.111:00}}\}_I \in \pi^{\rightarrow}_{n,\infty}[\mathbb{D}_{n+1}].$$

Observe that this coherent subset is a temporal finite object since it is performed in $1\ utc$, in a synchronized way.

Figure 2: The computational process $\mathbf{K}$.

Figure 3: The computational process $\mathbf{Q}$.

5 Concluding remarks.

We take use of the advantages of coherence spaces to obtain the domain-theoretic structure of the GM model. The completion procedure of the ordered structure of the GM model provides solutions for temporal recursive equations defined over the representations of the algebraic process constructors – sequential or parallel products and (non-)deterministic sums. In addition, based on the inductive structure of the set of labels ($I \equiv \omega$) it is also possible to represent the solutions for spacial recursive equations. In particular, a semantic modelling of algorithms related to the representations in $\mathbb{D}_\infty$ can be obtained based on algebraic process constructors. As an application of such model, an interval version of the GMM, called *Interval Geometric Machine Model* (IGMM) is defined in [6]. The possibly infinite set of memory positions in IGMM are labelled by points of the three-dimensional euclidian geometric space and the coherence space $\mathbb{IQ}$ of rational intervals [2] is taken into account to define its memory values. Finally, this model can be applied, to various kinds of computations involving array structures, such as matrix computations and cellular automata. Following this approach, it is possible to introduce a more generic version of GM model with a transfinite global memory shared by synchronized processes distributed over an enumerable set of geometric machines. That model is formalized by the coherence space of transfinite computational processes, defined over a web of tokens indexed by transfinite ordinal numbers.

Acknowledgements. This work is partially supported by CNPq/CTINFO and FAPERGS.

References

[1] S. Abramsky and A. Jung, Domain Theory, in: Handbook of Logic in Comp. Sc., Clarendon Press (1994).

[2] G. Dimuro, A. C. Costa and D. Claudio, A Coherence Space of Rational Intervals for a Construction of IR, Reliable Computing **6**(2) (2000) 139–178.

[3] J. -Y. Girard, Linear logic, Theoretical Computer Science **1** (1987) 187–212.

[4] R. Milner, Communication and Concurrency, Prentice Hall, Engl. Cliffs (1990).

[5] R. Reiser, A. C. Costa and G. Dimuro, First steps in the construction of the Geometric Machine, in: Seleta XXIV CNMAC eds. E.X.L. de Andrade et al., TEMA **3**(1) (2002) 183–192.

[6] R. Reiser, A. C. Costa and G. Dimuro, A programming language for the Inteval Geometric Machine Model, in: Eletronic Notes in Theoretical Computer Science **84** (2003). 12p.

[7] D. Scott, The lattice of flow diagrams, Lect. Notes in Math. **188** (1971) 311–372.