Electronic Notes in Theoretical Computer Science 84 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume84.html 12 pages

A Programming Language for the Interval Geometric Machine

Renata Hax Sander Reiser [1] Antônio Carlos da Rocha Costa [2,4]
Graçaliz Pereira Dimuro [3]

Escola de Informática
Universidade Católica de Pelotas
Pelotas, 96010-000, Brazil

Abstract

This paper presents an interval version of the Geometric Machine Model (GMM) and the programming language induced by its structure. 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. The version of the GMM presented here operates with real intervals, and is defined to model the semantics of algorithms of Interval Mathematics. Using the programming language induced by such structure, simple interval algorithms are presented, and their domain-theoretic semantics in the machine model is given.

1 Introduction

In this work, aspects of Domain Theory [1] and Concurrency Theory [4] are connected in order to obtain a semantics for interval algorithms [5] involving non-deterministic and synchronous parallel computations performed over matrix structures, based on Girard's Coherence Spaces [3]. The work presents an interval version of the *Geometric Machine Model* (GMM) where the possibly infinite set of memory positions are labelled by points of the three-dimensional euclidian geometric space. This model is able to describe partial computations and to formalize non-determinism and concurrency in the accesses to memory

[1] Email: reiser@ucpel.tche.br
[2] Email: rocha@ucpel.tche.br
[3] Email: liz@ucpel.tche.br
[4] PPGC, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil

positions. To define this model, the following ordered structures are considered: the *domain of states* $\mathbb{S}$, the *domain of boolean tests* $\mathbb{B}$ and the *domain of processes* $\mathbb{D}_\infty^\rightarrow$. Also, the coherence space $\mathbb{IQ}$ of rational intervals [2] is taken into account, to model the data operated by the processes. The input and of output data of the GMM are defined on the the extended set of real intervals $\mathrm{IR} = IR \bigcup \{(-\infty, +\infty)\}$. The interval version of the GMM is called *Interval Geometric Machine* (IGM).

This paper is organized as follows. In Section 2 basic concepts of coherence spaces and linear functions are considered together with the ordered structure of the IGM model. Section 3 introduces the language $\mathcal{L}(\mathbb{D}_\infty^\rightarrow)$ derived from $\mathbb{D}_\infty^\rightarrow$. The formal definition of the IGM model is presented in Section 5 and the last section shows some intervals algorithms expressed in $\mathcal{L}(\mathbb{D}_\infty^\rightarrow)$.

2 The Ordered Structure of the IGM Model

2.1 Coherence Spaces and Linear Functions

A web $\mathbf{W} = (W, \approx_W)$ is a pair consisting of a set W with a symmetric and reflexive relation $\approx_W$, called coherence relation. A subset of this web with pairwise coherent elements is called a coherent subset. The collection of coherent subsets of the web $\mathbf{W}$, ordered by the inclusion relation, is called a *coherence space*, denoted by $\mathbb{W} \equiv (Coh(\mathbf{W}), \subseteq)$ [10].

Linear functions are continuous functions in the sense of Scott [1] also satisfying the stability and linearity properties that assure the existence of least approximations in the image set [10]. Considering the coherence spaces $\mathbb{A}$ and $\mathbb{B}$, a linear function $f : \mathbb{A} \to \mathbb{B}$ is identified by its *linear trace*, a subset of $A \times B$ given by $ltr(f) = \{(\alpha, \beta) \,|\, \beta \in f(\alpha)\}$. Let $\mathbf{A} \multimap \mathbf{B} = (A \times B, \approx_{\multimap})$ be the web of linear traces with the coherence relation given by $(\alpha, \beta) \approx_{\multimap} (\alpha', \beta') \leftrightarrow ((\alpha \approx_{\mathbf{A}} \alpha' \rightarrow \beta \approx_{\mathbf{B}} \beta')$ and $(\beta = \beta' \rightarrow \alpha = \alpha'))$.

The collection of coherent subsets of the web $\mathbf{A} \multimap \mathbf{B}$, ordered by inclusion relation, defines the domain $\mathbb{A} \multimap \mathbb{B} \equiv (Coh(\mathbf{A} \multimap \mathbf{B}), \subseteq)$ of the linear traces of functions from $\mathbb{A}$ to $\mathbb{B}$.

In the following, we summarize the construction of the ordered structure of the IGM model, based on coherence spaces and traces of linear functions. For more details, see [6].

2.2 The Coherence Space of Machine States

The notion of *memory state* in the IGM model is formalized as follows. Consider the flat domain $\mathbb{R}^3 \equiv (Coh(\mathrm{R}^3, =), \subseteq)$ of *memory positions*, representing names of variables. Let $\mathbb{IQ} \equiv (Coh(\mathbf{IQ}), \subseteq)$ be the *Bi-structured Coherence Space of Rational Intervals* [2], representing the values that can be assigned to the variables. This domain is defined on the web $\mathbf{IQ} = (IQ, \approx_{IQ})$, given by the set of rational intervals with the coherence relation $p \approx_{IQ} q \leftrightarrow p \cap q \neq \emptyset$. This representation of the computable real numbers and the real intervals was

introduced in [2].

As suggested in [8], deterministic machine states are modelled as functions from memory positions to values. Non-deterministic machine states are modelled as families of deterministic machine states (with singletons modelling deterministic states).

Definition 2.1 Let $\mathbb{R}^3 \multimap \mathbb{IIQ}$ be the coherence space modelling deterministic machine states, and $\mathbf{S} = (Coh(\mathbb{R}^3 \multimap \mathbb{IIQ}), \approx_S)$ be the web given by the set of all coherent subsets of $\mathbb{R}^3 \multimap \mathbb{IIQ}$, together with the trivial (i.e., universal) coherence relation $\approx_S$. The collection of all coherent subsets of $\mathbf{S}$, ordered by inclusion, defines the coherence space that models the non-deterministic (and the singleton deterministic) machine states, denoted by $\mathbb{S} \equiv (Coh(\mathbf{S}), \subseteq)$.

A machine state is conceived then as a coherent set of linear traces of strict, continuous, stable and linear functions from $\mathbb{R}^3$ to $\mathbb{IIQ}$, one trace for each deterministic component of the non-deterministic state.

2.3 The Set $\mathcal{D}_0$ of Elementary Processes

The intuitive notion of an *elementary process*, which modifies a single memory position in a single unit of computational time (*uct*), can be described by an elementary transition between deterministic memory states, given by a linear function. Its intuitive representation in a one-dimensional machine is in Fig. 1(a). It is a function $\mathrm{d}^{(k)}$ satisfying:

Proposition 2.2 *Let $\mathbb{A} \equiv [\mathbb{R}^3 \multimap \mathbb{IIQ}] \multimap \mathbb{IIQ}$ be the coherence space of the so-called computational actions. If $\mathrm{d}, \mathrm{pr}^{(k)} \in \mathbb{A}$, with $\mathrm{pr}^{(k)}(s) = s(k)$ then the function $\mathrm{d}^{(k)} \in [\mathbb{R}^3 \to \mathbb{IIQ}] \to [\mathbb{R}^3 \to \mathbb{IIQ}]$ given by*

$$\mathrm{d}^{(k)}(s)(i) = \begin{cases} \mathrm{pr}^{(i)}(s) = s(i) & \text{if } i \neq k, \\ \mathrm{d}(s) & \text{if } i = k \end{cases}$$

is a linear function.

Proof. Let $s', z' \in [\mathbb{R} \to \mathbb{IIQ}]$, $i, k \in \mathbb{IIQ}$ and $s' = \mathrm{d}^{(k)}(s), z' = \mathrm{d}^{(k)}(z)$. We show that, for all $(s, s'), (z, z')$ in the graph of the function $\mathrm{d}^{(k)}$ the following properties hold. (1) If $s \approx_{\multimap} z$ then $s' \approx_{\multimap} z'$: Suppose $s \approx_{\multimap} z$. If $i \neq k$ and $\mathrm{d} \in [\mathbb{R} \multimap \mathbb{IIQ}] \multimap \mathbb{IIQ}$ then $\mathrm{d}^{(k)}(s)(i) = s(i) \approx_{\mathbf{IQ}} z(i) = \mathrm{d}^{(k)}(z)(i)$. In the other case, if $i = k$ then $\mathrm{d}^{(k)}(s)(k) = \mathrm{d}(s) \approx_{\multimap} \mathrm{d}(z) = \mathrm{d}^{(k)}(z)(k)$. Thus, $s' \approx_{\multimap} z'$. (2) If $s' = z'$ then $s = z$: Suppose $s' = z'$. When $i \neq k$ and $\mathrm{d} \in \mathbb{A}$ is a linear function, $s(i) = \mathrm{d}^{(k)}(s)(i) = s'(i) = z'(i) = \mathrm{d}^{(k)}(z)(i) = z(i)$ then $s = z$. Otherwise, if $i = k$, $\mathrm{d}^{(k)}(s)(i) = \mathrm{d}(s)(k) = \mathrm{d}(z)(k) = \mathrm{d}^{(k)}(z)(i)$ and $s = z$ based on the linearity of function d. Thus, in any case, $s = z$, and $\mathrm{d}^{(k)}$ is linear. □

Fig. 1(b) schematically represents the elementary processes in $\mathrm{d}^{(k)}$. The identity function interprets the undefined process **skip** (Fig. 1(c)).

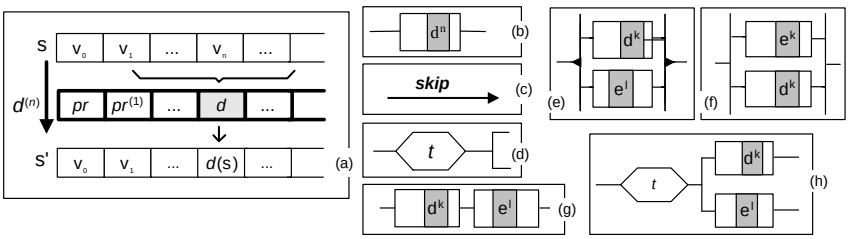

Fig. 1. Diagrammatic representation of one-dimensional computational elements.

2.4 The Coherence Space of Boolean Tests

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 $\mathbb{B}$ of Boolean Tests as the set of all coherent subsets of tests of the discrete web $\mathbf{B} \equiv (B, =)$, ordered by the inclusion relation. The coherence space $\mathbb{S} \multimap \mathbb{B}$ models the boolean tests t (see Figure 1(d)). Non-determinism enforces a non-traditional treatment of tests. For each boolean test b capable of testing a deterministic state s, we consider two forms for tests $\mathbf{b}$ on non-deterministic states $\mathbf{s}$:

(i) a universal form ($\mathbf{b}_{\forall}(\mathbf{s}) \equiv \forall s \in \mathbf{s} \,.\, b(s)$) and

(ii) 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.

2.5 The Coherence Space of Processes $\mathbb{D}_{\infty}^{\rightarrow}$

In this section, we summarize the main properties of the Coherence Space $\mathbb{D}_{\infty}^{\rightarrow}$ of processes in the IGM model, inductively constructed [5] from the coherence space $\mathbb{D}_0$ of elementary processes.

Following the methodology proposed in [9], each level in the inductive construction is identified by a subspace $\mathbb{D}_n$, which reconstructs all the objects from the level below it, preserving their properties and relations, and constructs the new objects specific of this level, see Fig. 2. The constructive relationship between the levels is expressed by linear functions called embedding and projection functions, interpreting constructors and destructors of processes, respectively.

In the next subsection, we summarized important aspects of each level of such inductive construction.

2.5.1 The basic level of the inductive construction of $\mathbb{D}_{\infty}^{\rightarrow}$

Let $\mathbf{D}_0 \equiv (\mathcal{D}_0, =)$ be a discrete web defined by the set $\mathcal{D}_0$ of elementary processes (2.3) with the equality relation. $\mathbb{D}_0 = (Coh(\mathbf{D}_0), \subseteq)$ denotes the

[5] The Coherence Spaces constructors used in the definition of $\mathbb{D}_{\infty}^{\rightarrow}$ can be found in [10].

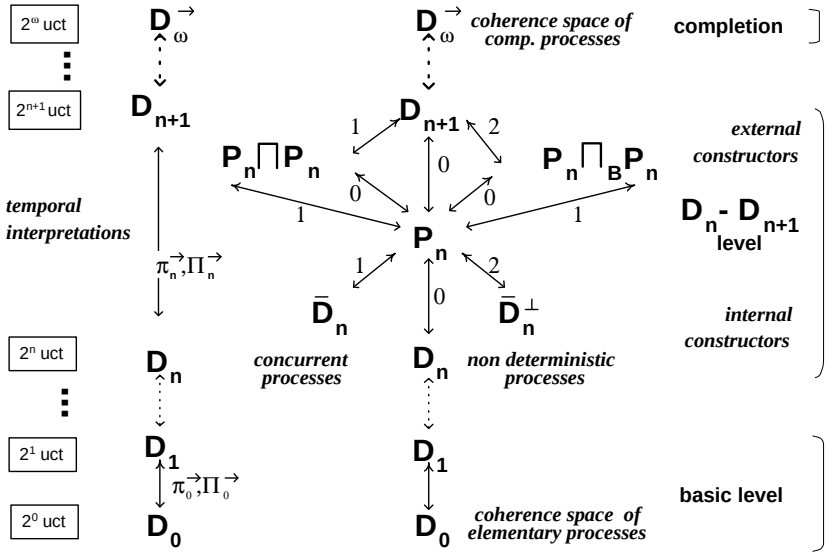

Fig. 2. The inductive construction of the coherence space $\mathbb{D}_\infty^{\rightarrow}$.

flat *coherence space of elementary processes*, with $Coh(\mathbf{D}_0) = \{\emptyset\} \cup \{\overline{d^{(k)}}\}$ where $\overline{d^{(k)}} \equiv \{d^{(k)}\} \in \mathbb{D}_0$.

The next coherence space in the construction, related with the family $\bar{\mathcal{D}}_0 \equiv Coh(\mathbf{D}_0)$, 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 $\bar{\mathbb{D}}_0 \equiv (Coh(\bar{\mathbf{D}}_0), \subseteq)$ is thus the *domain of parallel products of elementary processes*, where the web $\bar{\mathbf{D}}_0 \equiv (\bar{\mathcal{D}}_0, \approx)$ is given by the coherence relation $\overline{d^{(k)}} \approx \overline{e^{(l)}} \Leftrightarrow d^{(k)} = e^{(l)}$ or $k \neq l$. See Fig. 1(e).

In the dual construction, $\bar{\mathbb{D}}_0^{\perp} \equiv (Coh\,(\bar{\mathbf{D}}_0^{\perp}), \subseteq)$, where $\bar{\mathbf{D}}_0^{\perp} \equiv (\bar{\mathcal{D}}_0, \approx^{\perp})$, justified by the presence of involutive negation $\perp$ of coherence spaces [10], the incoherence relation $x \approx^{\perp} y \Leftrightarrow x \not\approx y$ models the condition for *non-determinism*, namely, the conflict of memory accesses. See Fig. 1(f).

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 $2\,uct$. See Fig. 1(g).

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 $\mathbb{P}_0 \prod \mathbb{P}_0$. See Fig. 1(h).

We can put all the above together, in order to obtain the domain

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

The coherence space $\mathbb{D}_1$ encompasses the first step of the construction of the ordered structure of the IGM model and provides the representations for all IGM computational processes performed in at most $2uct$. Each element in the web of such domain 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 level in the new one), (1) (indicating the parallel product of elements existing in the previous level) or (2) (indicating the non-deterministic sum of elements existing in the previous level). The second and third symbols, if present, mean the following: that the element is the first (02) or the second (12) summand in a deterministic sum, or that it is the first (01) or the second (11) term in a sequential product.

2.5.2 The $\mathbb{D}_n - \mathbb{D}_{n+1}$ level of the construction of $\mathbb{D}_{\infty}^{\rightarrow}$.
The ideas presented until above be generalized to the equation

$$\mathbb{D}_{n+1} = \mathbb{P}_n \coprod (\mathbb{P}_n \prod \mathbb{P}_n) \coprod (\mathbb{P}_n \prod_{\mathbb{B}} \mathbb{P}_n), \mathbb{P}_n = \mathbb{D}_n \coprod \bar{\mathbb{D}}_n \coprod \bar{\mathbb{D}}_n^{\perp} \tag{1}$$

The memory position information induced by R^3 on the domain of elementary processes $\Upsilon_{\mathbb{D}_0} : \mathbb{D}_0 \rightarrow \wp(\mathrm{R}^3)$, $\Upsilon_{\mathbb{D}_0}(x) = \{k \,|\, d^{(k)} \in x\}$ can be lifted to the coherent sets of the constructed domains by the position-function defined below, which defines the concurrency and conflict relations in them.

Definition 2.3 Let $\alpha \in \{0, 1, 2\}$ and

$$\begin{aligned}
&\upsilon_{\mathbf{P}_0} : \mathbf{P}_0 \rightarrow \wp(\mathrm{R}^3), \upsilon_{\mathbf{P}_0}(\mathbf{a}_0) = \Upsilon_{\mathbb{D}_0}(\mathbf{a}), \text{ and } \upsilon_{\mathbf{P}_0}(\mathbf{a}_{\alpha(\alpha\neq 0)}) = \{k | \overline{d^{(k)}} = \mathbf{a}\} \\
&\Upsilon_{\mathbb{P}_0} : \mathbb{P}_0 \rightarrow \wp(\mathrm{R}^3), \Upsilon_{\mathbb{P}_0}(x) = \cup \upsilon_{\mathbf{P}_0}(\mathbf{a}), \forall \mathbf{a}_{\alpha} \in x \\
&\Upsilon_{\sqcap} : \mathbb{P}_0 \prod \mathbb{P}_0 \rightarrow \wp(\mathrm{R}^3), \Upsilon_{\sqcap}(x) = \cup \Upsilon_{\mathbf{P}_0}(\mathbf{a}), \forall \mathbf{a}_{\alpha} \in x \\
&\Upsilon_{\sqcap_{\mathbb{B}}} : \mathbb{P}_0 \prod_{\mathbb{B}} \mathbb{P}_0 \rightarrow \wp(\mathrm{R}^3), \Upsilon_{\sqcap_{\mathbb{B}}}(x) = \cup \Upsilon_{\mathbf{P}_0}(\mathbf{a}), \forall \mathbf{a}_{\alpha(\alpha\neq 2)} \in x.
\end{aligned}$$

Then, $\Upsilon_{\mathbb{D}_{n+1}} : \mathbb{D}_{n+1} \rightarrow \wp(\mathrm{R}^3)$ is given by: $\Upsilon_{\mathbf{D}_{n+1}}(x) = \cup \Upsilon_{\mathbf{P}_n}(\mathbf{a})$, $\forall \mathbf{a}_0 \in x$ or $\Upsilon_{\mathbf{D}_{n+1}}(x) = \cup \Upsilon_{\sqcap}(\mathbf{a})$, $\forall \mathbf{a}_1 \in x$ or $\Upsilon_{\mathbf{D}_{n+1}}(x) = \cup \Upsilon_{\sqcap_{\mathbb{B}}}(\mathbf{a})$, $\forall \mathbf{a}_2 \in x$.

Let $Coh(\mathbf{D}_n)$ be the collection of coherent subsets of $\mathbb{D}_n$ and $x, y \in Coh(\mathbf{D}_n)$. In the web $\bar{\mathbf{D}}_n \equiv (Coh(\mathbf{D}_n), \approx)$, the coherence relation defined by $x \approx y \equiv \Upsilon_{\mathbb{D}_n}(x) \bigcap \Upsilon_{\mathbb{D}_n}(y) = \emptyset$ models the concurrence relation. That means, the pair (x, y) verifying such coherence relation represents concurrent computational processes whose execution is performed in 2^n uct.

By the other side, over the complementary web defined by the incoherence relation in $\bar{\mathbf{D}}_n$ is constructed the coherence space of non-deterministic computational processes performed at most 2^n uct, denoted $\bar{\mathbb{D}}_n^{\perp}$.

2.5.3 The Completion of $\mathbb{D}_{\infty}^{\rightarrow}$

Definition 2.4 The coherence space $\mathbb{D}_{\infty}^{\rightarrow}$ of processes in the IGM model is defined as the least fixed point for the equation 1 [6] . That is,

$$\mathbb{D}_{\infty}^{\rightarrow} = \mathbb{P}_{\infty}^{\rightarrow} \coprod (\mathbb{P}_{\infty}^{\rightarrow} \prod \mathbb{P}_{\infty}^{\rightarrow}) \coprod (\mathbb{P}_{\infty}^{\rightarrow} \prod_{\mathbb{B}} \mathbb{P}_{\infty}^{\rightarrow}), \mathbb{P}_{\infty}^{\rightarrow} = \mathbb{D}_{\infty}^{\rightarrow} \coprod \bar{\mathbb{D}}_{\infty}^{\rightarrow} \coprod \bar{\mathbb{D}}_{\infty}^{\rightarrow \perp}.$$

The inductive construction of the coherence space $\mathbb{D}_{\infty}^{\rightarrow}$ and the corresponding completion procedure provides a domain-theoretical structure for the representation of the set of all computational processes involving concurrency (and non determinism) and assures the existence of the least fixed point of the recursive equations defined by (possibly infinite) compositions of the above process constructors.

In order to express the indexed tokens of the coherent subsets in $\mathbb{D}_{\infty}^{\rightarrow}$, the follow denotation is considered:

(i) $: 00 \equiv 00.00.00 \ldots$ related to finite processes;

(ii) $: 001 \equiv 001.001.001 \ldots$ related to infinite sequential processes; and

(iii) $: 002 \equiv 002.002.002 \ldots$ related to infinite deterministic sums.

Based on the extension of the position function presented in Definition 2.3, indicated by $\Upsilon_{\mathbf{D}_{\infty}^{\rightarrow}}$, the concurrence relation between (infinity) computational processes is formalized in [7]. For instance, let $\mathrm{d}^{(k)}, \mathrm{e}^{(l)} \in [\mathbb{R}^3 \rightarrow \mathbb{IIQ}] \rightarrow [\mathbb{R}^3 \rightarrow \mathbb{IIQ}]$ be linear functions as presented in Proposition 2.2 and $k \neq l$. That means, $\Upsilon_{\mathbf{D}_{\infty}^{\rightarrow}}(\{\mathrm{d}_{:00}^{(k)}\}) = \{k\} \bigcap \{l\} = \Upsilon_{\mathbf{D}_{\infty}^{\rightarrow}}(\{\mathrm{e}_{:00}^{(l)}\}) = \emptyset$. The related parallel product between these elementary processes is represented as a partial objet given by the coherent subset

$$\{\overline{\mathrm{d}^{(k)}}_{01:00}, \overline{\mathrm{e}^{(l)}}_{01:00}\} \in \mathbb{D}_{\infty}^{\rightarrow}.$$

3 The Language Derived from $\mathbb{D}_{\infty}^{\rightarrow}$

We take use of $\mathbb{D}_{\infty}^{\rightarrow}$ to obtain a programming language for implementing parallel and non-deterministic interval algorithms.

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 $\mathbb{D}_{\infty}^{\rightarrow}$, respectively. In addition, let $F_{Op} = \{Id, \| , |, \cdot, + \}$ denote the set of symbols representing the constructors of processes of $\mathbb{D}_{\infty}^{\rightarrow}$, where

(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 $\forall b \in I_T$, $+_b : I_P \times I_P \rightarrow I_P$.

[6] See [6] for a proof that such fixpoint exists.

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

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

(ii) If $* \in \{\|\,, \mid, \cdot, +_b\}$ and $t_0, t_1, \ldots, t_n, t_{n+1}, \ldots b \in \mathcal{L}(\mathbb{D}_\infty^{\rightarrow})$ 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 $\mathcal{L}(\mathbb{D}_\infty^{\rightarrow})$;

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

We say that each expression of $\mathcal{L}(\mathbb{D}_\infty^{\rightarrow})$ is a *representation* of a process of $\mathbb{D}_\infty^{\rightarrow}$, and that the process is the *interpretation* of the expression.

4 The Machine Model and its Computations

We define the IGM model and its computations following [8].

Definition 4.1 The IGM model is defined as a tuple of functions $\mathcal{M} = (\mathcal{M}_{\mathbf{I}}, \mathcal{M}_{\mathbb{D}_\infty^{\rightarrow}}, \mathcal{M}_{\mathbb{B}}, \mathcal{M}_{\mathbf{O}})$ such that, for input and output values taken in the set IR of real intervals, represented in the center-radius form $[a_c, a_r]$, and using $s[i,j,k]$ for $s((i,j,k))$:

(i) $\mathcal{M}_{\mathbf{I}} : \mathrm{IR} \to \mathbb{S}$ is the *input function*. When $\mathbf{I} = \{\mathbf{i}_{00}\}$,

$$\mathcal{M}_{\{\mathbf{i}_{00}\}}([a_c, a_r]) = \{s\}, s(i) = \begin{cases} x_{a_c} \in \mathbb{IQ}, \text{ if } i = (0,0,0), \\ x_{a_r} \in \mathbb{IQ}, \text{ if } i = (0,0,1), \\ \emptyset, \qquad\quad \text{otherwise.} \end{cases}$$

where x_{a_c} and x_{a_r} are the coherent sets of $\mathbb{IQ}$ that best approximate a_c and a_r, see *[2]*.

(ii) $\mathcal{M}_{\mathbb{D}_\infty^{\rightarrow}} : \mathcal{L}(\mathbb{D}_\infty^{\rightarrow}) \multimap (\mathbb{S} \multimap \mathbb{S})$ is the *program interpretation function*, such that $\mathcal{M}_{\mathbb{D}_\infty^{\rightarrow}}(p) = x$ interprets the program p as the process x, as such process is defined in section 2.5 (see examples below).

(iii) $\mathcal{M}_{\mathbb{B}} : I_T \multimap (\mathbb{S} \multimap \mathbb{B})$ is the *test interpretation function*, such that $\mathcal{M}_{\mathbb{B}}(\mathbf{b}) = t$ interprets the symbol $\mathbf{b}$ as the test t, as it is defined in 2.4.

(iv) $\mathcal{M}_{\mathbf{O}} : \mathbb{S} \to \mathrm{IR}$ is the *output function*. When $\mathbf{O} = \{\mathbf{o}_{ij}\}$, then

$$\mathcal{M}_{\{\mathbf{o}_{ij}\}}(\mathbf{s}) = \{[a_c, a_r] \mid s[i,j,0] = x_{a_c}, s[i,j,1] = x_{a_r} \in \mathbb{IQ}, s \in \mathbf{s}\}.$$

The computation of a program p with an input data *in* results in the production of the output data $out = \mathcal{M}_{\mathbf{O}} \circ \mathcal{M}_{\mathbb{D}_\infty^{\rightarrow}}(p) \circ \mathcal{M}_{\mathbf{I}}(in)$.

5 Sample Interval Algorithms Expressed in $\mathcal{L}(\mathbb{D}_\infty^{\rightarrow})$

The next result follows from the definition of interval arithmetic operations found in [5], using the center-radius form of intervals.

Proposition 5.1 *Consider* $a, b \in \mathrm{IR}$ *and* $M = \{A, B, C, D\}$ *with*

$A = a_c \cdot b_c + a_r \cdot b_c + a_c \cdot b_r + a_r \cdot b_r$, $B = a_c \cdot b_c - a_r \cdot b_c + a_c \cdot b_r - a_r \cdot b_r$,
$C = a_c \cdot b_c + a_r \cdot b_c - a_c \cdot b_r - a_r \cdot b_r$, $D = a_c \cdot b_c - a_r \cdot b_c - a_c \cdot b_r + a_r \cdot b_r$.

The arithmetic operations $+, -, \cdot, / : \mathrm{IR}^2 \to \mathrm{IR}$ *are given by*

$$a + b = [a_c + b_c, a_r + b_r], a \cdot b = [\frac{(max(M) + min(M))}{2}, \frac{(max(M) - min(M))}{2}],$$

$$a - b = [a_c - b_c, a_r + b_r], \qquad a/b = \begin{cases} a \cdot [\frac{b_c}{b_c^2 - b_r^2}, \frac{b_r}{b_c^2 - b_r^2}] & if |b_c| > b_r, \\ undefined & if |b_c| \leq b_r. \end{cases}$$

An interval stored in the memory of the IGM machine is labelled by a *reference position* $(l, i) \in \mathrm{R}^2$ and a third index indicating its center (0) and radius (1), so $s[l, i, 0], s[l, i, 1]$ respectively represent the center and the radius of an interval stored in reference position (l, i).

Some expressions of elementary processes (assignment statements) and their corresponding interpretation in the domain $\mathbb{D}_\infty^\rightarrow$ are given in Table 1, where $\alpha = s[m, j, 0] \cdot s[n, k, 0]$, $\beta = s[m, j, 1] \cdot s[n, k, 0]$, $\gamma = s[m, j, 1] \cdot s[n, k, 0]$ and $\theta = s[m, j, 1] \cdot s[n, k, 1]$.

Table 1
Expressions of elementary processes and their domain interpretations

$\mathcal{L}(\mathbb{D}_\infty^\rightarrow)$	$\mathbb{D}_\infty^\rightarrow$
$\mathtt{sum\_c}(l, m, n, i, j, k) \equiv (s[l, i, 0] := s[m, j, 0] + s[n, k, 0])$	$\{sum\_c_{:00}^{(li0)}\}$
$\mathtt{sum\_r}(l, m, n, i, j, k) \equiv (s[l, i, 1] := s[m, j, 1] + s[n, k, 1])$	$\{sum\_r_{:00}^{(li1)}\}$
$\mathtt{sub\_c}(l, m, n, i, j, k) \equiv (s[l, i, 0] := s[m, j, 0] - s[n, k, 0])$	$\{sub\_c_{:00}^{(li0)}\}$
$\mathtt{sub\_r}(l, m, n, i, j, k) \equiv (s[l, i, 1] := s[m, j, 1] - s[n, k, 1])$	$\{sub\_r_{:00}^{(li1)}\}$
$\mathtt{p\_2}(l, m, n, i, j, k, u) \equiv (s[l, i, u + 2] := \alpha + \beta + \gamma + \theta)$	$\{p\_2_{:00}^{(l,i,u+2)}\}$
$\mathtt{p\_3}(l, m, n, i, j, k, u) \equiv (s[l, i, u + 3] := \alpha - \beta + \gamma + \theta)$	$\{p\_3_{:00}^{(l,i,u+3)}\}$
$\mathtt{p\_4}(l, m, n, i, j, k, u) \equiv (s[l, i, u + 4] := \alpha + \beta - \gamma + \theta)$	$\{p\_4_{:00}^{(l,i,u+4)}\}$
$\mathtt{p\_5}(l, m, n, i, j, k, u) \equiv (s[l, i, u + 5] := \alpha - \beta - \gamma + \theta)$	$\{p\_5_{:00}^{(l,i,u+2)}\}$

In the following, we illustrate the representation of some compound (non elementary) processes. Let $F_{(10)} : \mathbb{D}_\infty^\rightarrow \sqcap \mathbb{D}_\infty^\rightarrow \to \mathbb{D}_\infty^\rightarrow$ be the parallel product operator of $\mathbb{D}_\infty^\rightarrow$, represented by the symbol $\|$ in $\mathcal{L}(\mathbb{D}_\infty^\rightarrow)$.

(i) $\mathtt{sum}(l, m, n, i, j, k) \equiv (\mathtt{sum\_c}(l, m, n, i, j, k) \| \mathtt{sum\_r}(l, m, n, i, j, k))$ is the expression of the sum of two intervals, labelled by positions (m, j) and (n, k), with the result placed in the (l, i) position, and with the sum of the centers and the radiuses performed in parallel, 1*uct*. Its interpre-

tation

$$x_i = \{\overline{sum\_c^{(li0)}}_{10:00}, \overline{sum\_r^{(li1)}}_{10:00}\} \in \mathbb{D}_\infty^{\rightarrow}$$

can be computed from the interpretation of each element in the expression, by

$$F_{(10)}(\{\overline{sum\_c^{(li0)}}_{10:00}\} \sqcap \{\overline{sum\_r^{(li1)}}_{10:00}\}).$$

Analogous considerations can be done for the subtraction: $\texttt{sub}(l,m,n,i,j,k) \equiv (\texttt{sub\_c}(l,m,n,i,j,k) \parallel \texttt{sum\_r}(l,m,n,i,j,k))$.

(ii) If $\texttt{sum\_row}(l,m,n,i) \equiv (\texttt{sum}(l,m,n,i,i,i) \parallel \texttt{sum\_row}(l,m,n,i+1)$, the expression $\texttt{sum\_row}(l,m,n,0)$ represents the process that executes in parallel the addition of the sequence of intervals stored in the m-th and n-th rows, and assigns the result to the corresponding positions in the l-th row, in $1uct$. The coherent subset

$$x = \{\overline{sum\_c^{(li0)}}_{10:00}, \overline{sum\_r^{(li1)}}_{10:00}\}_{i\in\omega}$$

represents this process in $\mathbb{D}_\infty^{\rightarrow}$. This means that x is the least fixed point of the (spatial) recursive equation $x_{n+1} = F_{(10)}(x_i \sqcap x_0)$.
In the same way, the corresponding subtraction is defined by: $\texttt{sub\_row}(l,m,n,i) \equiv (\texttt{sub}(l,m,n,i,i,i) \parallel \texttt{sub\_row}(l,m,n,i{+}1,i{+}1,i{+}1)$.

(iii) Let $\mathbf{Q}_{li}$ be the interval algorithm performed in $2^{i-1}uct$, defined by

$$\begin{cases} \texttt{sum\_row}(l,0) \equiv \textbf{skip} \\ \texttt{sum\_row}(l,i) \equiv (\texttt{sum\_row}(l,i-1); \texttt{sum}(l,l,l,i-1,i,i-1)) \end{cases}$$

In this case, $\mathbf{Q}_{02}$ is represented in $\mathbb{D}_\infty^{\rightarrow}$ by the subset

$$\{\overline{sum\_c^{(010)}}_{101:00}, \overline{sum\_r^{(011)}}_{101:00}, \overline{sum\_c^{(000)}}_{111:00}, \overline{sum\_r^{(001)}}_{111:00}\}$$

The same analysis can be applied to the next process:

$$\begin{cases} \texttt{sum\_col}(0,i) := \textbf{skip} \\ \texttt{sum\_col}(l,i) := \texttt{sum\_col}(l-1,i); \texttt{sum}(l-1,l,l-1,i,i,i). \end{cases}$$

(iv) The process $\mathbf{P}$ that executes, simultaneously, the process $\mathbf{Q}$ in the first l-th rows can be given by

$$\begin{cases} \texttt{sum\_row}(0,i) \equiv \textbf{skip} \\ \texttt{sum\_row}(l,i) \equiv (\texttt{sum\_row}(l,i) \parallel \texttt{sum\_row}(l-1,i)) \end{cases}$$

(v) Consider the processes:
(a) the parallel product:

$$\begin{aligned} \texttt{p\_par}(l,m,n,i,j,k) \equiv\ & \texttt{p\_2}(l,m,n,i,j,k,2) \parallel \\ & \texttt{p\_3}(l,m,n,i,j,k,2) \parallel \\ & \texttt{p\_4}(l,m,n,i,j,k,2) \parallel \\ & \texttt{p\_5}(l,m,n,i,j,k,2) \end{aligned}$$

(b) the recursive process related to a minimal element:

$$\begin{cases} \texttt{Min}(l,i,0) := \texttt{skip} \\ \texttt{Min}(l,i,u) := \texttt{mini}(l,i,u-1); \texttt{Min}(l,i,u-1) \end{cases}$$

when
$\texttt{mini}(l,i,0) = \texttt{mini}(l,i,1) = \texttt{mini}(l,i,2) \equiv \texttt{skip}$ and
$\texttt{mini}(l,i,u+3) \equiv (s[l,i,u+2] := \texttt{min}(s[l,i,u+2], s[l,i,u+3]))$.

(c) the recursive process related to a maximal element:

$$\begin{cases} \texttt{Max}(l,i,0) := \texttt{skip} \\ \texttt{Max}(l,i,u) := \texttt{maxi}(l,i,u-1); \texttt{Max}(l,i,u-1). \end{cases}$$

when
$\texttt{maxi}(l,i,0) = \texttt{maxi}(l,i,1) = \texttt{maxi}(l,i,2) \equiv \texttt{skip}$ and
$\texttt{maxi}(l,i,u+3) \equiv (s[l,i,u+2] := \texttt{max}(s[l,i,u+2], s[l,i,u+3]))$,

(d) the recursive process:

$$\begin{aligned} \texttt{p}(l,m,n,i,j,k) \equiv & (\texttt{p\_par}(l,m,n,i,j,k); \\ & (\texttt{p\_c}(l,m,n,i,j,k) \parallel \texttt{p\_r}(l,m,n,i,j,k))), \end{aligned}$$

when

$$\begin{aligned} \texttt{p\_c}(l,m,n,i,j,k) &\equiv (s[l,i,0] := (Max(l,i,5) + Min(l,i,5))/2) \\ \texttt{p\_r}(l,m,n,i,j,k) &\equiv (s[l,i,1] := (Max(l,i,5) - Min(l,i,5))/2) \end{aligned}$$

in order to obtain the recursive expression

$$\texttt{prod\_row}(l,m,n,i) \equiv (\texttt{p}(l,m,n,i,i,i) \parallel \texttt{prod\_row}(l,m,n,i+1))$$

interpreting the parallel product of corresponding intervals labelled by positions (m,i) and (n,i) with the results placed in the (l,i) position, $i \in \omega$.

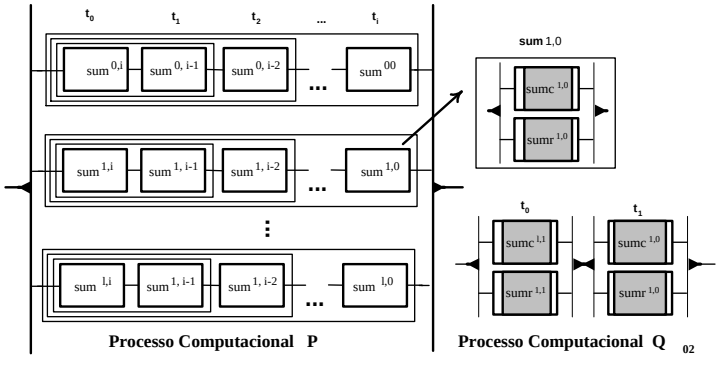

Fig. 3. Diagrammatics representations of processes in the IGM model.

6 Conclusions

In this work we described the main characteristics of the IGM model. Also, we defined a programming language extracted from $\mathbb{D}_{\infty}^{\rightarrow}$, and we presented some examples of programs, as well as their representation in $\mathbb{D}_{\infty}^{\rightarrow}$. The IGM model is an abstract machine, with infinite memory and constant memory access time that can be use to analyze the logic and ordered structure of interval algorithms, including partial, concurrent and non-deterministic ones. In this sense, the development of algorithms in the IGM model may happen to improve the analysis of algorithms for interval applications. The application of that analysis to concrete problems is work in progress.

Acknowledgement

This work is partially supported by the Brazilian funding agencies CNPq, CAPES and FAPERGS. The authors are very grateful to them.

References

[1] Abramsky, S., and A. Jung, *Domain Theory*, in: Handbook of Logic in Computer Science, Clarendon Press, 1994.

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

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

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

[5] Moore, R. E., *Methods and Applications of Interval Analysis*, SIAM, 1979.

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

[7] Reiser, R. H. S., *The Geometric Machine - a computational model for concurrence and non-determinism based on the coherence spaces*, Ph.D. thesis, CPGCC/UFRGS, Porto Alegre, 2002.

[8] Scott, D., *Some Definitional Suggestions for Automata Theory*, Journal of Computer and System Sciences **188** (1967), 311-372.

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

[10] Troelstra, A. S., *Lectures on Linear Logic*, CSLI Lecture Notes **29** (1992).