The geometry of conservative programs

Emmanuel Haucourt

Under consideration for publication in Math. Struct. in Comp. Science

The geometry of conservative programs

EMMANUEL HAUCOURT[1]

emmanuel.haucourt@lix.polytechnique.fr

[1] *LIX - UMR 7161, 1 rue Honoré d'Estienne d'Orves, Bât. Alan Turing*
Campus de l'École Polytechnique, 91120, Palaiseau, France

Received 8 September 2017

The programs we consider are written in a restricted form of the language introduced by (Dijkstra (1968)). A program is said to be *conservative* when each of its loops restores all the resources it consumes. We define the geometric model of such a program and prove that the collection of directed paths on it is a reasonable overapproximation of its set of execution traces. In particular, two directed paths that are close enough with respect to the *uniform distance* result in the same action on the memory states of the system. The same holds for *weakly dihomotopic* directed paths. As a by-product, we obtain a notion of independence which is favourably compared to more common ones. The geometric models actually belong to a handy class of *local pospaces* whose elements are called *isothetic regions*. The local pospaces we use differ from the original ones, we carefully explain why the alternative notion should be preferred. The title intentionally echoes the article by (Carson and Reynolds Jr. (1987)).

1. Introduction

The importance of the ideas introduced by (Dijkstra (1968)) cannot be overestimated, neither from the theoretical point of view (Hansen, 2002, p.7-12) nor from the practical one, as evidenced by their influence on the POSIX Threads Programming norm. Its underlying philosophy consists of having things done in parallel *unless* otherwise specified by the programmer. Strictly speaking, Dijkstra's PV language refers to an extension of the language ALGOL60 with the synchronization mechanism `parbegin ... parend` (also called 'parallel composition') and the the primitives $P(\_)$ and $V(\_)$ with which one takes and releases general semaphores. It is not hard to see that the same would have been possible with any reasonable sequential language (*e.g.* Parallel Pascal in (Cridlig (1995, 1997))). The PV language allows parallel composition to occur anywhere in a program so they can be nested. In this paper, we only allow it in outermost position so the programs we consider are parallel compositions of sequential processes.

```
parbegin
  process_1 ;
  ...
  process_N ;
parend
```

A program is thus made of a pool of sequential processes executing their instructions in parallel and sharing a pool of resources. The limited amount of resources is thus a constraint on the simultaneous execution of processes. In our framework it is also constrained by the 'wait' instruction $W(\_)$ which synchronizes a given group of *existing* processes. Indeed it stops any process which meets it until a certain number of processes get blocked by it. When the threshold is reached the executions of all the processes stalled by the instruction resume. The 'Wait' instruction can be seen as a weakened form of parallel composition. Indeed the latter *creates* the processes that it synchronizes while the former does not. These restrictions are commonly met in the design of asynchronous control command systems, to which the methods that we will describe were dedicated in the first place.

The idea that higher dimensional geometric structures offer a natural framework for modelling such a kind of program was implicitly suggested by (Dijkstra (1968)) but the concept was later made explicit by (Coffman et al. (1971)) who introduced the *progress spaces*[†]. Coffman and its co-authors emphasized that the instruction pointers of a set of tasks $\{T_1, \ldots, T_n\}$ can be gathered to become the coordinates of a point on a shape of dimension n. They also pointed out that the evolution of this point (with respect to time) induces a 'trajectory' (on the shape) which can never result in a decrease in either coordinate, the progress being irreversible. Then (Carson and Reynolds Jr. (1987)) definitely made things clear stating that a *progress graph* is a 'multidimensional, Cartesian graph in which the progress of each of a set of concurrent processes is measured along an independent time axis'. They also used the term 'solid geometry' and mentioned 'areas of relative progress disallowed by synchronization primitives invoked by the processes', specifying that these *forbidden regions* are represented as finite unions of rectangles. Their terminology strongly suggests that the model of a program is obtained by extruding the forbidden region from some Cartesian product.

However the only models considered in those articles are *cubical regions*, that is finite unions of n-dimensional hyperrectangles (*i.e.* n-fold products of intervals of $\mathbb{R}$). In the abstract of (Carson and Reynolds Jr. (1987)) one can actually read that 'the model is shown to be exact for systems composed of [...] concurrent processes, each consisting of a straight line sequence of [...] ordered semaphore operations.' In particular the trajectories that (Coffman et al. (1971)) talked about are monotonic continuous paths on $\mathbb{R}^n$. On one hand it suggests that the standard topology and order on $\mathbb{R}$ can be used to model certain concurrent programs. It also highlights the importance of the mathematical structure carried by the collection of cubical regions, which will be proven to be a restriction of the one carried by the collection of *isothetic regions*. On the other hand it brings out the limitations on the class of programs one can model that way: neither branching nor loop is allowed.

Contribution. In line with the theoretical pioneering works on geometric models of concurrency, this article aims at going beyond the limitation mentioned above yet pay-

[†] For further details on the rise of topological methods in concurrency theory see (Goubault (2000)).

ing more attention to practice. Our first contribution is to emphasize the role of the metric structure of geometric models by proving that two execution traces that are close enough (with respect to the uniform distance) have the same effect on the states of the abstract machine associated with a given toy language (Theorem 6.1). As an immediate consequence, the same holds for weakly dihomotopic execution traces (Corollary 6.2). Another interesting by-product of that approach is its relation to program independence (Theorem 6.2). We insist that our geometric models are obtained from discrete ones by elementary mathematical methods. In particular, we completely avoid the traditional use of precubical sets (Pratt (1991); van Glabbeek (1991)) and realization functors (Fajstrup et al. (2006)). Discrete models faithfully reflect the runtime behaviour of programs (Theorem 4.1); the geometric models do so because of their relation to discrete ones (Section 6.2). The geometric model construction that we provide only applies to conservative programs (Definition 4.1), a class that was independently studied by (Fahrenberg (2002)) and that comes with a breadth-first traversal algorithm deciding whether a program belong to it (end of Section 4.1). In return, it can be automated and the models of such programs are isothetic regions (Corollary 6.1). The collection of isothetic regions enjoys a rich and tractable mathematical structure (Proposition 7.3, Theorems 7.1 and 7.2). The crucial fact is that all the relevant operations on isothetic regions can be automated from basic operations on intervals of $\mathbb{R}$. By the way, we return to the notion of local pospaces and propose a seemingly slight change in the original definition (Fajstrup et al. (2006)). We hope that Remark 5.5 will convince the reader of the advantage of that change.

Organization of the paper. In Section 2 we introduce the program representations from which the models are defined. Such representations are obtained by standard methods of compilation. We also introduce the parallel composition of such representations and the notion of syntactically independent programs (Definitions 2.3 and 2.4). The abstract machine of the language is described in Section 3 thus providing it with a semantics from which we introduce the notion of observationally independent programs (Definition 3.6). Conservative programs and their discrete models are introduced in Section 4. In Section 5 we make a foray into the realm of directed topology adopting the formalism of local pospaces. In Section 6 we switch to geometric models and provide results justifying that change. Building on metric graphs we introduce and study isothetic regions in Section 7. Section 8 is dedicated to related works and open problems.

2. Middle-end representations of programs

Instead of specifying a toy language we describe the abstract representations of the programs we are interested in. Those representations are based on *control flow graphs*[‡] (Allen (1970)) which are built the same way as *transition graphs* (Fajstrup et al., 2016, p.7-14, 27) except that the instructions are carried by vertices instead of arrows. Echoing the remark from (Pratt (1991)) control flow graphs are only defined for sequential processes,

[‡] Control flow graphs are very similar to *flowcharts* (Floyd (1967)). The former were introduced in the context of compiler optimization, the latter in that of program verification.

one of the purpose of the present article is precisely to provide a control flow structure to programs made of several cooperating sequential processes.

The *variables*, *semaphores*, and *barriers* occuring in a program are respectively collected in the finite sets $\mathcal{X}$, $\mathcal{S}$, and $\mathcal{B}$. Each semaphore or barrier z comes with its *arity* $\alpha(z) \in \mathbb{N} \cup \{\infty\}$. A *valuation* is a mapping $\nu : \mathcal{X} \to \mathbb{R}$. An *expression* is a mapping $\varepsilon : \{\text{valuations}\} \to \mathbb{R}$ together with a subset $\mathcal{F}(\varepsilon) \subseteq \mathcal{X}$ such that if the valuations ν and ν' match on $\mathcal{F}(\varepsilon)$ then $\varepsilon(\nu) = \varepsilon(\nu')$. The *free variables* of ε are the elements of $\mathcal{F}(\varepsilon)$. The set of expressions occuring in the program is denoted by $\mathcal{E}$. The *assignments* are the elements of $\mathcal{X} \times \mathcal{E}$ yet we write $x := \varepsilon$ instead of (x, ε). By extension the set of free variables of such an assignment is $\mathcal{F}(\varepsilon)$. Given a graph

$$G : A \underset{\partial^+}{\overset{\partial^-}{\rightrightarrows}} V$$

a *conditional branching* at vertex $v \in V$ is a mapping

$$\beta : \{\text{valuations}\} \to \{a \in A \mid \partial^- a = v\}$$

together with a subset $\mathcal{F}(\beta) \subseteq \mathcal{X}$ such that if the valuations ν and ν' match on $\mathcal{F}(\beta)$ then $\beta(\nu) = \beta(\nu')$. No such mapping exists when the vertex v has no outgoing arrows. An *instruction* of the language is either an assignment, a branching, a *request* $P(s)$ (resp. a *release* $V(s)$) for some semaphore $s \in \mathcal{S}$, or a *synchronization* $W(b)$ for some barrier $b \in \mathcal{B}$. An instruction which 'does nothing', namely *Skip*, is also provided. Except for assignments and conditional branchings, the instructions have no free variables.

Definition 2.1. A *process* is a graph together with a distinguished vertex, its *origin*, and a labelling $\lambda : V \to \{\text{instructions}\}$ such that if $\lambda(v)$ is not a *Skip* instruction, then v has at least one outgoing arrow, and if $\lambda(v)$ is not a branching, then v has at most one outgoing arrow. We also impose that the origin carries a *Skip* instruction and that all the vertices and all the arrows of the graph are met by some path starting at the origin of the graph. In reference to (Allen (1970)) we also write 'control flow graph' as a synonym for 'process'. The arrows are interpreted as intermediate positions of the instruction pointer so a *point* on a control flow graph is either a vertex or an arrow.

Definition 2.2. A *program* consists of:

— the finite sets $\mathcal{X}$, $\mathcal{S}$, and $\mathcal{B}$ which collect the variables, the semaphores, and the barriers appearing in the program,
— the arity map $\alpha : \mathcal{S} \sqcup \mathcal{B} \to \mathbb{N} \cup \{\infty\}$,
— the initial valuation $\nu : \mathcal{X} \to \mathbb{R}$ which provides the values of the variables at the beginning of each execution of the program, and
— the tuple $(G_1, \ldots, G_n)$ of processes which are launched simultaneously at the beginning of each execution of the program.

Following (Coffman et al. (1971)) the instruction pointer of a program is defined as the n-tuple which gathers the instruction pointers of its running processes. It is thus natural to define a *point* p of $(G_1, \ldots, G_n)$ as an n-tuple whose i^{th} component, which is refered to as p_i, is a point of G_i in the sense of Definition 2.1. The *origin* of the program

is the tuple of the origins of its running processes. By analogy with the multisets of actions from (Cattani and Sassone (1996)), we define a *multi-instruction* as a partial map $\mu : \{1, \ldots, n\} \to \{\text{instructions}\}$. A multi-instruction is said to be *trivial* when, as a mapping, it always returns *Skip*. By a slight abuse of notation, we define $\lambda(p)$ as the multi-instruction defined by $\lambda(p)(i) = \lambda_i(p_i)$ over the set below.

$$\big\{ i \in \{1, \ldots, n\} \;\big|\; p_i \text{ is a vertex of } G_i \big\}$$

Our language neither allows dynamic creation of processes nor runtime alteration of semaphore and barrier arities. Under obvious assumptions we can form the parallel composition of programs.

Definition 2.3. Two programs P and Q are said to be *compatible* when their initial valuations and their arity maps coincide on the intersection of their domains of definition. In that case we can form the *parallel composition* $P|Q$. The sets of variables, semaphores, and barriers appearing in $P|Q$ as well as the initial valuation and the arity map are defined in the obvious way. The tuple of running processes of $P|Q$ is the appending of the tuples of the running processes of P and Q. By extension we define the parallel composition of $P_1, \ldots, P_N$ when the programs are pairwise compatible.

It is then natural to ask when a program can be split into a parallel composition of simpler programs. We propose a straightforward approach.

Definition 2.4. Two programs are said to be *syntactically independent* when $\mathcal{X} \cap \mathcal{X}' = \emptyset$, $\mathcal{S} \cap \mathcal{S}' = \emptyset$, and $\mathcal{B} \cap \mathcal{B}' = \emptyset$. The programs $P_1, \ldots, P_N$ are *syntactically independent* when they are pairwise syntactically independent.

Syntactically independent programs are compatible so they can be composed. Conversely if one can split the running processes of a program into groups of processes without common resources, then one can 'parallelize' this program in the obvious way up to the order of the terms of the tuple of running processes.

3. Abstract machine

The abstract machine 'executes' programs. Its functioning explains on the way the use of barriers and semaphores as well as the behaviour of the related instructions, namely $P(s)$, $V(s)$ for $s \in \mathcal{S}$, and $W(b)$ for $b \in \mathcal{B}$. First we define its internal states denoting by n the number of running processes of the program under consideration.

Definition 3.1. A *state* of the abstract machine is a mapping σ defined over the disjoint union $\mathcal{X} \sqcup \mathcal{S}$ such that:

— for all variables x, $\sigma(x) \in \mathbb{R}$ (*i.e.* the restriction of σ to $\mathcal{X}$ is a valuation), and
— for all semaphores s, $\sigma(s)$ is a mapping from $\{1, \ldots, n\}$ to $\mathbb{Z}$.

A state σ is said to be *initial* when $\sigma(s)$ is the zero map for all semaphores $s \in \mathcal{S}$. The valuation part of σ is the *memory state* of the machine. The elements of the set $\{1, \ldots, n\}$ should be thought of as *process identifiers*. The number of occurrences of the semaphore

s held by the i^{th} process is therefore given by $\sigma(s)(i)$, namely the coefficient of i in the multiset $\sigma(s)$. Yet, it is convenient to allow negative values in order to make subsequent definitions more concise.

Definition 3.2. Two instructions *conflict* when:

— both modify the same variable (*i.e.* the instructions are $x := \varepsilon$ and $x := \varepsilon'$ regardless of ε and ε' being identical), one says that it is a *write-write* conflict, or

— one of them alters a free variable of the other (*i.e.* one of the instructions is $x := \varepsilon$ and the other is a conditional branching or an assignment in which x occurs as a free variable), one says that it is a *read-write* conflict.

A multi-instruction μ is *conflicting* when there are $i, j \in M$ with $i \neq j$ such that $\mu(i)$ and $\mu(j)$ conflict. As an immediate byproduct we soundly define the action of a *non-conflicting* multi-instruction μ on the right side of a state σ:

— for all variables $x \in \mathcal{X}$, there is at most one $i \in \{1, \ldots, n\}$ such that the instruction $\mu(i)$ modifies x. In that case the instruction is an assignment $x := \varepsilon$ and one has

$$(\sigma \cdot \mu)(x) \quad = \quad \varepsilon(\sigma|_{\mathcal{X}})$$

the latter being well-defined because for all $j \in \{1, \ldots, n\} \setminus \{i\}$ the instruction $\mu(j)$ does not alter any free variable of ε. In the other case one has $(\sigma \cdot \mu)(x) = \sigma(x)$.

— for all semaphores $s \in \mathcal{S}$, the mapping $(\sigma \cdot \mu)(s) : \{1, \ldots, n\} \to \mathbb{Z}$ is given below:

$$i \quad \mapsto \quad \begin{cases} \sigma(s)(i) + 1 & \text{if } i \in \operatorname{dom}\mu \text{ and } \mu(i) = P(s) \\ \sigma(s)(i) - 1 & \text{if } i \in \operatorname{dom}\mu \text{ and } \mu(i) = V(s) \\ \sigma(s)(i) & \text{in all other cases.} \end{cases}$$

The *abstract machine* is the mapping which sends each (σ, μ) to $\sigma \cdot \mu$ when μ is non-conflicting. It provides our language with an operational semantics allowing several processes to execute their current instruction simultaneously.

The arity of a semaphore s is the number of available occurrences of s while the arity of a barrier is the maximal number of processes that it can stop. In particular any process trying to acquire an occurrence of a semaphore of null arity is definitely blocked. The same happens when a process comes across a synchronization barrier whose arity is infinite. On the contrary, semaphores of infinite arity and barriers of null arity are harmless: the latter are roughly speaking ignored and the former cannot bring about the process stalling.

Definition 3.3. A non-conflicting multi-instruction μ is said to be *admissible* at state σ when for all semaphores $s \in \mathcal{S}$, we have the following inequalities

$$0 \quad \leqslant \quad |\sigma(s)| + \operatorname{card}\{i \in M \mid \mu(i) = P(s)\} - \operatorname{card}\{i \in M \mid \mu(i) = V(s)\} \quad \leqslant \quad \alpha(s)$$

and for all synchronizing barriers $b \in \mathcal{B}$, $\operatorname{card}\{i \in M \mid \mu(i) = W(b)\}$ is either null or strictly greater than $\alpha(b)$. It is worth noticing that our semaphore semantics slightly differs from the one found in (Dijkstra (1968)) which does not take the concept of 'owner' of a semaphore occurrence into account. By extension, a sequence $\mu_0 \cdots \mu_q$ of multi-instructions is said to be *non-conflicting* when so are all its elements. It is said to be

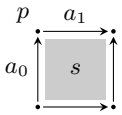


Fig. 1. Discrete directed paths are 'continuous'.

admissible at state σ when for all $k \in \{0, \dots, q\}$, the multi-instruction μ_k is admissible at state $\sigma \cdot \mu_0 \cdots \mu_{k-1}$.

Definition 3.4. A *directed path* γ on a given tuple of graphs $(G_1, \dots, G_n)$, typically the running processes of a program, is a sequence $(\gamma(k))_{k \in \{0,\dots,q\}}$ of points (in the sense of Definition 2.2) such that for all $k \in \{0, \dots, q\}$ we have

— $\gamma_i(k) = \gamma_i(k+1)$ or $\partial^- \gamma_i(k+1) = \gamma_i(k)$ for all $i \in \{1, \dots, n\}$, or
— $\gamma_i(k) = \gamma_i(k+1)$ or $\partial^+ \gamma_i(k) = \gamma_i(k+1)$ for all $i \in \{1, \dots, n\}$.

The constraints imposed in Definition 3.4 might seem surprising at the first sight. They actually force the directed paths to be 'continuous' in the following sense: assume that the picture in Figure 1 is the unit square $[0,1]^2$ split into points (its corners), open segments (the interior of its edges), and open unit square (its interior). A naive definition would accept sequences of points such that $\gamma_i(k) = \gamma_i(k+1)$, $\partial^- \gamma_i(k+1) = \gamma_i(k)$, or $\partial^+ \gamma_i(k) = \gamma_i(k+1)$ for all $i \in \{1, \dots, n\}$. In doing so, we would make the sequence (a_0, a_1) in Figure 1 a directed path although no continuous increasing path on the unit square can go from a_0 to a_1 without meeting p or s.

Each directed path γ is associated with a sequence of multi-instructions $(\mu_k)_{k \in \{0,\dots,q-1\}}$. The domains of definition of the maps μ_k, for $k \in \{0, \dots, q-1\}$, are given below.

$$\operatorname{dom} \mu_k \quad = \quad \big\{ i \in \{1, \dots, n\} \;\big|\; \gamma_i(k+1) = \partial^+ \gamma_i(k) \text{ or } \lambda_i(\gamma_i(k+1)) = W(\_) \big\}$$

Then we have $\mu_k(i) = \lambda_i(\gamma_i(k+1))$ for all $k \in \{0, \dots, q-1\}$ and all $i \in \operatorname{dom} \mu_k$. This convention implements a specific behaviour: any instruction is triggered at the very moment it is reached by the instruction pointer. It also reveals that the instruction $W(\_)$ is persistent in the sense that it remains active until the instruction pointer leaves the vertex carrying it. On the contrary, the instructions $P(\_)$ and $V(\_)$ instantly alter the internal state of the machine whereafter their effect is over. With the abuse of notation introduced in Definition 2.2 and for $k \in \{0, \dots, q-1\}$, the multi-instruction μ_k seen as a partial map is a restriction of $\lambda(\gamma(k+1))$. The point $\gamma(0)$ is intentionally ignored so that the sequence of multi-instructions associated with a concatenation $\gamma \cdot \gamma'$ be the concatenation of the sequences of multi-instructions associated with γ and γ'. A directed path is said to be *non-conflicting* when so is its associated sequence of multi-instructions. It is said to be *admissible* when its associated sequence of multi-instructions is admissible at the initial state of the program. In both cases, we define the *action* of γ on the right of σ as follows.

$$\sigma \cdot \gamma \quad = \quad \sigma \cdot \mu_0 \cdots \mu_{q-1}$$

An admissible path is an *execution trace* when all the conditional branchings met along the way are respected, in other words the following equality holds for all $k \in \{0, \dots, q-2\}$

and all $i \in \{1,\dots,n\}$ such that $\mu_k(i)$ is a branching.

$$\big(\mu_k(i)\big)(\sigma \cdot \mu_0 \cdots \mu_{k-1}) \quad = \quad \gamma_i(k+2)$$

Let n and N be non-zero natural numbers. Thinking of the elements of the set $\{1,\dots,n\}$ as the process identifiers of a program, we consider a mapping g from $\{1,\dots,n\}$ onto $\{1,\dots,N\}$ as a way of gathering processes in view of a parallelization of the program. In that regard, we say that two multi-instructions μ and μ' (whose domains of definition are contained in $\{1,\dots,n\}$) can be *swapped* when the sets of 'groups' of processes they trigger are disjoint. Formally, if J (resp. J') is the set of indices $j \in \{1,\dots,N\}$ such that $\operatorname{dom}\mu$ (resp. $\operatorname{dom}\mu'$) meets $g^{-1}(\{j\})$, then $J \cap J' = \emptyset$.

The permutations π of the set $\{0,\dots,q-1\}$ act on the left of sequences of length q in the usual way: for all $k \in \{0,\dots,q-1\}$ the element at position k in the sequence s is moved to position $\pi(k)$ in the sequence $\pi \cdot s$. Dually, the element at position k in $\pi \cdot s$ is the one that was at position $\pi^{-1}(k)$ in s. Interpreting sequences as mappings defined over $\{0,\dots,q-1\}$, the sequence $\pi \cdot s$ is the mapping $s \circ \pi^{-1}$.

Definition 3.5. Given a sequence $\mu_0,\dots,\mu_{q-1}$ of multi-instructions such that $\operatorname{dom}\mu_k \subseteq \{1,\dots,n\}$ for all $k \in \{0,\dots,q-1\}$, a permutation π of the set $\{0,\dots,q-1\}$ is said to be *compatible* with that sequence when it is order preserving on all pairs $\{k,k'\}$ such that μ_k and $\mu_{k'}$ cannot be swapped. The permutation π is said to be *compatible* with the directed path γ when it is compatible with its associated sequence of multi-instructions.

Suppose that the programs $P_1,\dots,P_N$ are compatible in the sense of Definition 2.3 and that P_j has n_j running processes. For $j \in \{0,\dots,N\}$ define $s_j = n_1 + \cdots + n_j$ with the convention that $s_0 = 0$. In particular the set $\{1,\dots,n_1+\cdots+n_N\}$ of indexes of the running processes is divided into subsets $S_1,\dots,S_N$ where $S_j = \{s_{j-1}+1,\dots,s_j\}$. If $i \in S_j$ then the i^{th} process of the parallel composition comes from the program P_j. With the notation introduced before, we have $g^{-1}(\{j\}) = S_j$ for all $j \in \{1,\dots,N\}$. With the notation of Definition 3.4, we introduce another notion of independence.

Definition 3.6. The programs $P_1,\dots,P_N$ are said to be *observationally independent* when for all execution traces γ and all permutations π compatible with the sequence of multi-instructions $(\mu_0 \cdots \mu_{q-1})$ associated with γ, the sequence $\pi \cdot (\mu_0 \cdots \mu_{q-1})$ is associated with an execution trace γ' having the same action on the system state than γ, that is to say $\sigma \cdot (\mu_0 \cdots \mu_{q-1}) = \sigma \cdot (\mu_{\pi^{-1}(0)} \cdots \mu_{\pi^{-1}(q-1)})$.

Remark 3.1. Syntactically independent programs are observationally independent but the converse is false. For example, consider the program whose unique process has a unique instruction, namely the assignment $x := cst$ where the expression cst has no free variable (hence cst always returns the same value when it is evaluated). This program is observationally independent from itself, but not syntactically.

Syntactical independence is decidable yet too restrictive. On the contrary observational independence is purely theoretic because it cannot be stated until all the execution trace prefixes have been tested or a mathematical proof has been given. We complete this section with a technical result that will be used in the proof of Theorem 6.2.

Definition 3.7. Let us define a *rolling* as a cyclic permutation ρ of the following form

$$(\; x \quad x+1 \quad \cdots \quad x+y \;)$$

for some $x, y \in \mathbb{N}$, that is to say defined by $\rho(x+t) = x + (t+1 \mod y+1)$.

Lemma 3.1. Given a permutation π which is not an identity, the rolling

$$\rho \quad = \quad (\; x \quad x+1 \quad \cdots \quad \pi^{-1}(x) \;)$$

where $x = \min\{x \in \operatorname{dom} \pi \mid \pi(x) \neq x\}$ is the unique one such that $\pi \circ \rho^{-1}(x) = x$.

Proof. By definition of x we have $\pi^{-1}(x) > x$. From the relation $\pi \circ \rho^{-1}(x) = x$ we deduce that $\rho(\pi^{-1}(x)) = x$. Hence x and $\pi^{-1}(x)$ are the least and the greatest elements of $\{x \in \operatorname{dom} \rho \mid \rho(x) \neq x\}$ which characterizes the rolling ρ. □

The next lemma, which is stated with the notation introduced in Lemma 3.1, justify our interest in rollings.

Lemma 3.2. If the permutation π is compatible with a sequence of multi-instructions $\mu_0 \cdots \mu_{q-1}$, then so is the rolling ρ. Moreover the permutation $\pi \circ \rho^{-1}$ is compatible with sequence of multi-instructions $\rho \cdot (\mu_0 \cdots \mu_{q-1})$.

Proof. Let x be the least element of $\operatorname{dom}(\pi)$ such that $\pi(x) \neq x$. We have $\pi^{-1}x > x$ and $\pi(k) > x$ for all $k \in \{x, \ldots, \pi^{-1}x - 1\}$. Since π is compatible with the sequence $(\mu_0 \cdots \mu_{q-1})$, the multi-instruction $\mu_{\pi^{-1}x}$ can be swapped with all the multi-instructions μ_k for $k \in \{x, \ldots, \pi^{-1}x - 1\}$. The rolling ρ is thus compatible with the sequence $(\mu_0 \cdots \mu_{q-1})$.

Let $(\mu'_0 \cdots \mu'_{q-1})$ be the sequence of multi-instructions $\rho \cdot (\mu_0 \cdots \mu_{q-1})$. In other words the equality $\mu'_k = \mu_{\rho^{-1}(k)}$ holds for all $k \in \{0, \ldots, q-1\}$. Let $k, k' \in \{0, \ldots, q-1\}$ be such that the multi-instructions μ'_k and $\mu'_{k'}$ cannot be swapped and assume that $k < k'$. One has to prove that $\pi \circ \rho^{-1}(k) < \pi \circ \rho^{-1}(k')$. To do so, it suffices to check that $\rho^{-1}(k) < \rho^{-1}(k')$ because the permutation π is compatible with the sequence $(\mu_0 \cdots \mu_{q-1})$. That inequality is readily satisfied when $k < x$ or $\pi^{-1}x < k'$ so we can suppose that both k and k' belong to the set $\{x, \ldots, \pi^{-1}x\}$. In particular $\rho^{-1}(k)$ and $\rho^{-1}(k')$ also belong to that set. Then we have $\rho^{-1}(k) \neq \pi^{-1}x$ otherwise we would have $\rho^{-1}(k) > \rho^{-1}(k')$ and $\pi \circ \rho^{-1}(k) < \pi \circ \rho^{-1}(k')$, the latter inequality being deduced from the definition of x. But then μ'_k and $\mu'_{k'}$ would be able to swap because the permutation π is compatible with the sequence $(\mu_0 \cdots \mu_{q-1})$. It follows that k differs from x. We conclude by noting that ρ^{-1} is order-preserving on the set $\{x+1, \ldots, \pi^{-1}x\}$. □

Definition 3.8. Let π_0 be a permutation which is not an identity. Inductively applying Lemma 3.1 we define a finite sequence of permutations $\pi_0, \ldots, \pi_A$ and a finite sequence of rollings $\rho_0, \ldots, \rho_A$ by the relation $\pi_{a+1} = \pi_a \circ \rho_a^{-1}$ for $a \in \{0, \ldots, A-1\}$. The natural number A is the least one such that π_A is a rolling. It is well-defined by Lemma 3.1 which ensures that the sequence $x_a = \min\{x \in \operatorname{dom} \pi_a \mid \pi_a(x) \neq x\}$, for $a \in \{0, \ldots, A\}$, is strictly increasing. We have $\pi_0 = \rho_A \circ \cdots \circ \rho_0$ and the sequence $\rho_0, \ldots, \rho_A$ is called the *rolling decomposition* of the permutation π_0. By convention, the rolling decomposition of the identity is the empty sequence.

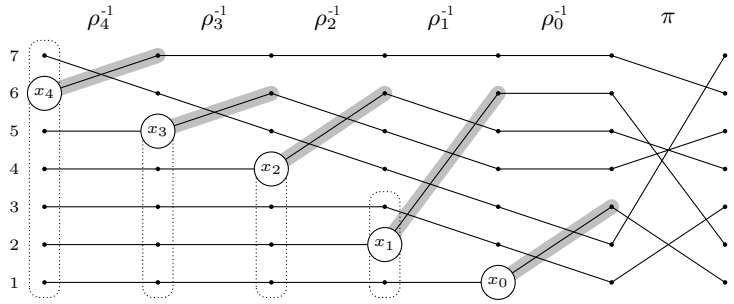


Fig. 2. The rolling decomposition of the permutation $\pi = (1\ 3)(2\ 7\ 6)(4\ 5)$.

Example 3.1. Let π_0 be the permutation $(1\ 3)(2\ 7\ 6)(4\ 5)$.

a	π_a	x_a	$\pi_a^{-1}(x_a)$	ρ_a^{-1}
0	$(1\ 3)(2\ 7\ 6)(4\ 5)$	1	3	$(3\ 2\ 1)$
1	$(2\ 3\ 7\ 6)(4\ 5)$	2	6	$(6\ 5\ 4\ 3\ 2)$
2	$(4\ 7\ 6)$	4	6	$(6\ 5\ 4)$
3	$(5\ 7\ 6)$	5	6	$(6\ 5)$
4	$(6\ 7)$	6	7	$(7\ 6)$
5	id	–	–	–

Corollary 3.1. Let $\rho_0, \ldots, \rho_A$ be the rolling decomposition of a permutation π_0 that is compatible with a sequence of multi-instructions $(\mu_0 \cdots \mu_q)$. For all $a \in \{0, \ldots, A\}$

— the rolling ρ_a is compatible with the sequence $(\rho_{a-1} \circ \cdots \circ \rho_0) \cdot (\mu_0 \cdots \mu_q)$, and
— the permutation π_{a+1} is compatible with the sequence $(\rho_a \circ \cdots \circ \rho_0) \cdot (\mu_0 \cdots \mu_q)$.

Proof. By iterated applications of Lemma 3.2. □

4. Discrete models of conservative programs

A directed path γ that is not admissible 'crashes' the abstract machine in the sense that $\sigma \cdot \gamma$ is not well-defined (Definition 3.4). Under mild hypotheses on a given program, we define a collection of 'forbidden' points such that the admissible directed paths are 'almost' the ones that do not meet any of them, the subtlety being explained in Theorem 4.1. We will be done if we can prove that admissibility no longer depends on the state of the abstract machine (Definition 3.3). The crucial point is that only the constraints on semaphores depend on the state σ. From a physicist point of view, the situation suggests to compare the action of a directed path on the semaphores sharing with the work of a force along a curve. As a physicist saying that a force is conservative when this work only depends on the starting and the ending of the curve, we say that a program is

conservative when the action (on the abstract machine states) of a directed path (over the control flow graphs of the running processes) only depends on its endpoints.

4.1. *Conservative programs*

Conservative programs were independently introduced by (Fahrenberg (2002)) as parallel composition of *well behaved processes*. The notion of a conservative program given below is slightly more general. The idea is that the amount of semaphores held by the running processes of a program can be encoded in its model.

Definition 4.1. A *PV-test* is a program written only with instructions $P(\_)$, $V(\_)$, and *Skip*, and such that the arities of its semaphores are infinite. Denoting by $G_1, \ldots, G_n$ for the running processes of a PV-test and following Definition 3.2, every directed path on $(G_1, \ldots, G_n)$ is non-conflicting. A PV-test is said to be *conservative* when for all directed paths γ on $(G_1, \ldots, G_n)$ the restriction of $\sigma \cdot \gamma$ to $\mathcal{S}$ (*i.e.* the sharing of semaphores between the running processes) is nonnegative[§] and only depends on the endpoint of γ, being understood that σ is an initial state (Definition 3.1) and that γ starts at the origin of the program (Definition 2.2). Every program is associated with a PV-test by changing the arities of all its semaphores to ∞ and replacing all the instructions occurring in it but $P(\_)$ and $V(\_)$ by *Skip*. A program is said to be conservative when so is its associated PV-test. In that case, its *sharing function* is the canonical map

$$F : \{\text{points of } (G_1, \ldots, G_n)\} \times \mathcal{S} \to \{\text{multisets over } \{1, \ldots, n\}\}$$

which associates (p, s) with $\sigma \cdot \gamma(s)$ where γ is any directed path from the origin of the program (Definition 2.2) to p (such a directed path always exists by Definition 2.1). The *mass* of the multiset $F(p, s)$ (seen as a mapping from $\{1, \ldots, n\}$ to $\mathbb{N}$) is the following sum

$$|F(p,s)| \quad = \quad \sum_{i=1}^{n} \big(F(p,s)\big)(i) \ .$$

The *potential function* of a conservative program is the mapping $|F|$.

If the program under consideration has a single running process, viz $n = 1$, then the sharing and the potential functions from Definition 4.1 can be identified. For each $i \in \{1, \ldots, n\}$ the i^{th} sequential projection of a program is obtained by removing all its running processes but G_i. The next result is the reason why it is so important to keep track of the owners of the semaphore occurrences.

Lemma 4.1. A program is conservative iff so are all its sequential projections. Moreover for all points p and all semaphores $s \in \mathcal{S}$, the multiset $F(p, s)$ is the following map

$$F(p,s) : i \in \{1, \ldots, n\} \mapsto F_i(p_i, s) \in \mathbb{N}$$

where F_i is the sharing function of the i^{th} process. In particular the potential function

[§] For all semaphores s, the range of the mapping $\sigma(s)$ is contained in $\mathbb{N}$, so it can be seen as a multiset.

of the program is the sum of the potential functions of its sequential projections.

$$|F(p,s)| \quad = \quad \sum_{i=1}^{n} F_i(p_i,s)$$

Proof. Suppose that the program is conservative. To check that the i^{th} sequential projection is conservative, it suffices to consider the directed paths whose points have their j^{th} coordinate on the origin of the graph G_j for all $j \in \{1,\dots,n\} \setminus \{i\}$.

Conversely, let σ be an initial state and γ be a directed path on $(G_1,\dots,G_n)$ starting at the origin of the program. Following Definitions 3.1 and 3.4, the amount of occurrences of the semaphore s held by the i^{th} process at the end of the directed path γ is $\big(\sigma\cdot\gamma(s)\big)(i)$. Given $i \in \{1,\dots,n\}$, let γ_i be the directed path on G_i obtained by replacing the j^{th} coordinate of each point of γ, for $j \neq i$, by the origin o_j of G_j. In other words one applies the following mappings to all points of γ.

$$\begin{array}{ccccc} (G_1,\dots,G_n) & \to & G_i & \hookrightarrow & (G_1,\dots,G_{i-1},G_i,G_{i+1},\dots,G_n) \\ (p_1,\dots,p_n) & \mapsto & p_i & \mapsto & (o_1,\dots,o_{i-1},p_i,o_{i+1},\dots,o_n) \end{array}$$

From Definition 3.2, no running process but the i^{th} one can alter the amount of occurrences of the semaphore s held by the i^{th} running process of the program. Therefore the next equality is satisfied and the conclusion follows.

$$\big(\sigma\cdot\gamma(s)\big)(i) \quad = \quad \big(\sigma\cdot\gamma_i(s)\big)(i) \quad = \quad F_i(p_i,s)$$

□

Lemma 4.1 would not be satisfied if we had allowed processes to release occurrences of tokens they do not hold. Consider indeed the concurrent execution of the instructions $P(s)$ and $V(s)$ for some semaphore s with only one occurrence left.

Remark 4.1. From a given control flow graph G (Definition 2.1) one can write a program whose unique running process is G. We can then say that G is conservative when so is that program. Indeed, two such programs only differ over their declarations, which are anyway 'forgotten' by the corresponding PV-tests. Hence G being conservative does not depend on the chosen program.

Examples of conservative and nonconservative processes are given in Figures 3 and 4. The latter suggests that certain nonconservative control flow graphs can be turned into conservative ones by a mild transformation, the overall idea being that we try to encode the amount of semaphore occurrences held by a process into its finite control flow graph. As shown by Figure 5 there are situations where no such transformation is possible.

For a given control flow graph, the property of being conservative can be checked by a breadth-first traversal algorithm that was independently discribed by (Fahrenberg (2002)). Let us denote the commutative group of mappings from $\mathcal{S}$ to $\mathbb{Z}$ by $\mathbb{Z}^{\mathcal{S}}$. We inductively define a sequence of partial functions of type $\{\text{points}\} \to \mathbb{Z}^{\mathcal{S}}$. The first term π_0 is only defined at the origin of the graph and $\pi_0(\text{origin})$ is the empty multiset (*i.e.* the map sending all $s \in \mathcal{S}$ to 0). Suppose that π_n is defined for some $n \in \mathbb{N}$. If there exists an ordered pair of points (p,p') such that:

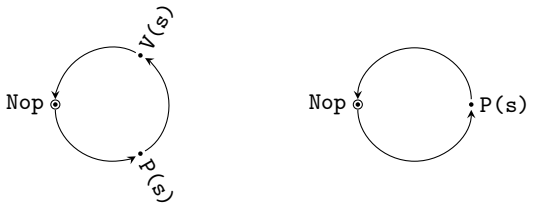


Fig. 3. Conservative loop *vs* nonconservative loop.

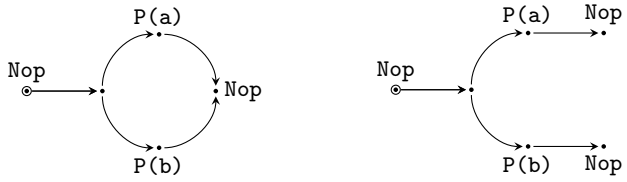


Fig. 4. Conservative process may be obtained by duplicating a vertex.

— $\pi_n(p)$ is defined but not $\pi_n(p')$, and
— $\partial^- p' = p$ or $p' = \partial^+ p$,
we define a strict extension of π_n by setting:

$$p' \quad \mapsto \quad \begin{cases} \pi_n(p) & \text{if } \partial^- p' = p \\ \pi_n(p) \cdot \lambda(p') & \text{if } p' = \partial^+ p \end{cases}$$

If all those extensions are compatible, then π_{n+1} is their union; otherwise the induction stops and the graph is not conservative. If the mapping π_n cannot be extended (*i.e.* no ordered pair as above exists), then it is defined everywhere because every point of a control flow graph can be reached from its origin (Definition 2.1). Moreover G is conservative if and only if π_n is nonnegative and the following property holds for all ordered pairs of points (p, p') such that $\partial^- p' = p$ or $p' = \partial^+ p$.

$$\pi_n(p') \quad = \quad \begin{cases} \pi_n(p) & \text{if } \partial^- p' = p \\ \pi_n(p) \cdot \lambda(p') & \text{if } p' = \partial^+ p \end{cases}$$

In that case, the sharing function of the control flow graph G is π_n. Applying the

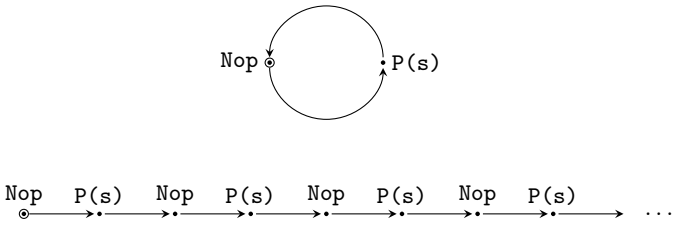


Fig. 5. A nonconservative loop and its unfolding.

algorithm to the left hand control flow graph in Figure 4 results in a sequence of length 5, the map π_6 being undefined. On the contrary, if the algorithm is applied to the right hand control flow graph from Figure 3, one obtains a mapping π_3 which is defined everywhere but does not satisfy the latter property.

4.2. *Discrete models*

We introduce the discrete models of conservative programs and explain the purpose for which they are studied.

Definition 4.2. A point $p = (p_1, \dots, p_n)$ of the conservative program is said to be:
— *conflicting* when $\lambda_i(p_i)$ and $\lambda_j(p_j)$ conflict for some $i \neq j$ (Definition 3.2),
— *exhausting* when there is some semaphore $s \in \mathcal{S}$ such that

$$|F(p_1, \dots, p_n, s)| \quad > \quad \text{arity}(s) \ ,$$

— *desynchronizing* when there is some synchronization barrier $b \in \mathcal{B}$ such that

$$0 \quad < \quad \text{card}\big\{i \in \{1, \dots, n\} \;\big|\; \lambda_i(p_i) = W(b)\big\} \quad \leqslant \quad \alpha(b) \ ,$$

— *terminal* when the sequence of multi-instructions associated with any directed path starting at p only contains trivial multi-instructions (Definitions 2.2 and 3.4).

The *forbidden* set of the program P gathers all the conflicting, exhausting, and desynchronizing points.

$$\{\text{fobidden}\} \quad = \quad \{\text{conflicting}\} \cup \{\text{exhausting}\} \cup \{\text{desynchronizing}\}$$

The *discrete model* of the program is the complement of its forbidden set.

$$\{\text{points of the program}\} \setminus \{\text{forbidden}\}$$

A *deadlock* is a non-terminal point p such that any directed path *on the model* starting at p is associated with a sequence of trivial multi-instructions. Equivalently, a deadlock is a point p such that any directed path which starts at p and triggers some non trivial multi-instruction actually meets a forbidden point. The latter formulation exactly states that the forbidden region hinders the expected functioning of the program.

The following result, which is illustrated by Figure 6, is one of the main motivations for introducing conservative programs and discrete models.

Theorem 4.1. Any directed path on the discrete model (*i.e.* which does not meet any forbidden point) is admissible. Conversely, for each admissible path γ there exists a directed path on the discrete model whose sequence of multi-instructions is the same as the one of γ.

Proof. Let γ be a directed path that is not admissible and let k be such that the multi-instruction μ_k from Definition 3.4 is not admissible at state $\sigma \cdot \mu_0 \cdots \mu_{k-1}$. If μ_k contains a conflict then so does $\lambda(\gamma(k+1))$, the latter being an extension of the former as explained in Definition 3.4. By definition of μ_k, the equality

$$\{i \mid \mu_k(i) = W(b)\} \quad = \quad \{i \mid \lambda_i(\gamma_i(k+1)) = W(b)\}$$

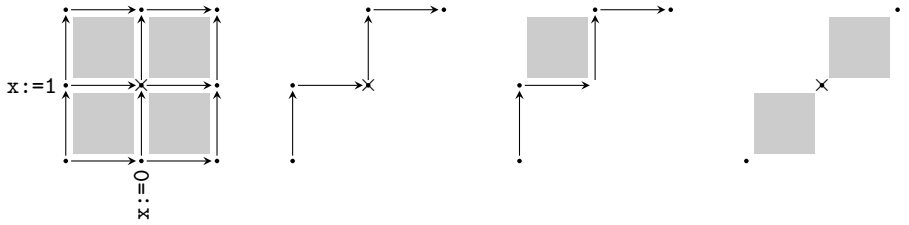


Fig. 6. A discrete model, an admissible path on it that meets a forbidden point, a possible replacement, and a nonadmissible path.

holds for all barriers $b \in \mathcal{B}$; hence if μ_k forces a barrier (*i.e.* the left hand set above is not empty and its cardinal is less or equal than $\alpha(b)$) then the point $\gamma(k)$ is forbidden in the sense of Definition 4.2. Since the program is conservative, the equality

$$\sigma \cdot \mu_0 \cdots \mu_k(s) \quad = \quad F(\gamma(k), s)$$

holds for all semaphores $s \in \mathcal{S}$; hence if the mass $|F(\gamma(k), s)|$ is strictly greater than the arity $\alpha(s)$, then the point $\gamma(k)$ is forbidden.

Conversely, let γ be an admissible path such that $\gamma(k)$ is forbidden, and suppose that k is minimum. By similar arguments to those exposed above, we conclude that there exist $i, i' \in \{1, \ldots, n\}$ such that $\lambda_i(\gamma_i(k))$ and $\lambda_{i'}(\gamma_{i'}(k))$ conflict in the sense of Definition 4.2. For γ is admissible, there must be some $k' < k$ such that the following holds for all $k''' \in \{k', \ldots, k\}$.

$$\lambda_{i'}(\gamma_{i'}(k''')) \quad = \quad \lambda_{i'}(\gamma_{i'}(k))$$

In less formal words, the i'^{th} process is stalled on an assignment in conflict with the instruction $\lambda_i(\gamma_i(k))$. The interpretation of arrows as interlude between instructions plays a role here. Denote by k'' the first index such that the i'^{th} coordinate of $\gamma(k'')$ is the unique arrow α outgoing from $\gamma_{i'}(k')$. If no such index exists, then $k'' = \infty$. To obtain the expected directed path, it suffices to change the i'^{th} coordinate of points $\gamma(k''')$ into α for all $k' < k''' < k''$, which amounts to set the instruction pointer of the i'^{th} process in an intermediate position. One readily deduces from Definition 3.4 that the altered directed path induces the same sequence of multi-instructions than the original one. Of course there might be another $i'' \in \{1, \ldots, n\}$ such that $\lambda_i(\gamma_i(k))$ and $\lambda_{i''}(\gamma_{i''}(k))$ conflict, but then it suffices to iterate the preceding construction until all such indices have been treated. Finally we obtain a directed path γ' whose initial segment $\gamma'(0), \ldots, \gamma'(k)$ does not meet any forbidden point and induces the same sequence of multi-instructions than $\gamma(0), \ldots, \gamma(k)$. We conclude by a straightforward induction over k. □

Remark 4.2. Theorem 4.1 guarantees that restricting the class of admissible directed paths to those which do not meet any forbidden point does not result in a significant loss. Moreover, it allows to treat the instructions $P(\_)$, $V(\_)$, and $W(\_)$ *statically* rather than *dynamically*. Indeed, since directed paths on the model are admissible in the sense of Definition 3.3, it is no longer needed to check at runtime whether resources are exhausted

or barriers are violated. Moreover, our language does not offer any feature to access the current distribution of resources. Consequently, it is no longer needed to keep it up-to-date during an execution. In other words, once the discrete model is built, the relevant part of an abstract machine state is reduced to its memory (Definition 3.1), and the instructions $P(\_)$, $V(\_)$, and $W(\_)$ can be ignored, that is to say replaced by *Skip*.

Theorem 4.1 demands a notion of conflicting instructions that can be decided statically. For example, according to Definition 3.2, every assignment conflicts with itself. Our credo is, as far as possible, to statically treat concurrency by encoding in the model of a program all the constraints imposed on its executions.

5. Locally ordered spaces

Locally ordered spaces are mathematical objects combining topology and order. They were introduced by (Fajstrup et al. (2006)) as a generalization of partially ordered spaces (Nachbin (1965)) in order to model the directed circle. Since then, local pospaces have often been cited and used but never studied for themselves. Moreover, some local pospace definitions found in the literature (Bubenik and Worytkiewicz (2006); Bubenik (2009); Kahl (2009)) do not match the original one (Fajstrup et al. (2006)). As a discriminating example, the original approach allows an ordered atlas on the circle containing only two ordered charts while some other ones do not. It therefore seemed appropriate to explicitly define the local pospaces we are working with, which are yet different, and gather some facts giving a rather good account of their features. The characteristic of our approach is that all the ordered atlases are explicitly required to induce a basis of topology of the underlying space. This condition is in accordance with the idea that local pospaces are obtained from pospaces as n-dimensional manifolds from $\mathbb{R}^n$, viz by methods inspired from *sheaves*. The resulting definition is strictly weaker than the original one.

There is an obvious price to pay: unlike d-spaces (Grandis (2003)) and streams (Krishnan (2009)), which behave much like topological spaces, local pospaces are closer to manifolds. A colimit of local pospaces, as long as it exists, may not respect the underlying topology. In comparison, the category of d-spaces and that of streams, which are actually close to each other (Haucourt (2012)), are complete and cocomplete, and their colimits behave well with respect to the underlying topology. They even admit Cartesian closed subcategories enjoying the same convenient features (Goubault-Larrecq (2014)). Nevertheless we prefer local pospaces because they are much more constrained than streams and d-spaces. For example, a local pospace has no vortex (Remark 5.4).

Definition 5.1. (Nachbin (1965)) A *partially ordered space* (or *pospace*) is a topological space X together with a partial order $\sqsubseteq$ on (the underlying set of) X whose graph

$$\{(a,b) \in X \times X \mid a \sqsubseteq b\}$$

is a closed subset of $X \times X$. A pospace morphism is an order-preserving continuous map. Pospaces and their morphisms form the category **PoTop**. The underlying space of a pospace is Hausdorff (Nachbin, 1965, p.27).

Example 5.1.
The ordered real line (*i.e.* $\mathbb{R}$ with its standard topology and order).
The sub-pospaces of a pospace (*i.e.* its subsets with the induced topology and order).
The category of pospaces is cocomplete (Haucourt (2012)). However due to antisymmetry of partial orders, the following coequalizer in **PoTop** is reduced to a point.

$$\{*\} \underset{*\mapsto 1}{\overset{*\mapsto 0}{\rightrightarrows}} [0,1] \longrightarrow \{*\}$$

To overcome this difficulty we consider patchworks of pospaces.

Definition 5.2. (Fajstrup et al. (2006)) Let X be a Hausdorff space. An *(ordered) chart* on X is a pospace U whose underlying space S_U is an open subset of X. An *(ordered) atlas* is a collection $\mathcal{U}$ of ordered charts whose underlying spaces form a basis of the topology of X. Moreover, it is required that for all $U, V \in \mathcal{U}$, for all $x \in U \cap V$, there exists $W \in \mathcal{U}$ such that $x \in W \subseteq U \cap V$ and denoting by $\sqsubseteq_{U|_W}$ the relation induced by $\sqsubseteq_U$ on the underlying set of W, the restrictions of $\sqsubseteq_U$ and $\sqsubseteq_V$ to W match $\sqsubseteq_W$.

$$\sqsubseteq_{U|_W} \;=\; \sqsubseteq_W \;=\; \sqsubseteq_{V|_W}$$

Remark 5.1. Let $\mathcal{U}$ be a collection of charts whose underlying spaces form an open cover of X and such that for all $x \in X$, all $U_0, U_1 \in \mathcal{U}$ that both contain x, there exists an open neighborhood W of x in $U_0 \cap U_1$ such that both U_0 and U_1 induce the same pospace structure on W. Then the collection of all open sub-pospaces of all the charts in $\mathcal{U}$ forms an atlas in the sense of Definition 5.2.

Since there is no possible confusion with the atlases from differential manifolds, we will just write 'atlas' instead of 'ordered atlas'. As we shall see, an atlas might contain two globally incompatible ordered charts upon the same open subset. In particular, writing $U \subseteq U'$ for $U, U' \in \mathcal{U}$ we mean that the underlying space of U is included in the one of U' without taking the compatibility of their partial orders into account. Therefore $U \cap U'$ refers to the intersection of the underlying subspaces while $U \wedge U'$ refers to $U \cap U'$ equipped with the partial order $\sqsubseteq_U \cap \sqsubseteq_{U'}$ and thus forms a pospace.

Definition 5.3. Two atlases on the same space are said to be *compatible* when their union is still an atlas.

The following results are easily checked and provide a good insight of the behaviour of atlases. Moreover they lead to the notion of local pospaces.

Lemma 5.1. The atlases $\mathcal{U}$ and $\mathcal{V}$ are compatible iff for all $U \in \mathcal{U}$, for all $x \in U$, there exists $V \in \mathcal{V}$ such that $x \in V \subseteq U$ and $\sqsubseteq_V = \sqsubseteq_{U|_V}$.

Lemma 5.2. The notion of compatibility induces an equivalence relation over the collection of atlases sharing the same underlying space. In particular the union of all the atlases of a compatibility class is an atlas.

Definition 5.4. An *atlas morphism* from $\mathcal{U}$ to $\mathcal{V}$ is a map f from the underlying set of $\mathcal{U}$ to that of $\mathcal{V}$ such that for all $x \in \operatorname{dom} f$, there exists an ordered chart $U \in \mathcal{U}$ and an ordered chart $V \in \mathcal{V}$ such that $x \in U$ and f induces a pospace morphism from U to V.

Lemma 5.3. A mapping is an atlas morphism iff for all $x \in X$ and all $V \in \mathcal{V}$ containing $f(x)$, there is $U \in \mathcal{U}$ containing x such that f induces a pospace morphism from U to V.

By Lemma 5.3, an atlas morphism is necessarily continuous because any atlas induces a basis of its underlying topology.

Lemma 5.4. Let f be a map. Let $\mathcal{U}$ and $\mathcal{U}'$ (resp. $\mathcal{V}$ and $\mathcal{V}'$) be equivalent atlases on $\operatorname{dom} f$ (resp. $\operatorname{codom} f$). Then $f : \mathcal{U} \to \mathcal{V}$ is an atlas morphism iff $f : \mathcal{U}' \to \mathcal{V}'$ is so.

Definition 5.5. A *locally ordered space* (or *local pospace*) is a Hausdorff space together with an equivalence class of ordered atlases. A *local pospace morphism* is an atlas morphism (Lemma 5.4). The category of local pospaces is denoted by **LpoTop**.

Remark 5.2. Given an atlas $\mathcal{U}$ on a space X and a subspace Y of X, the collection of sub-pospaces of the form $U \cap Y$ with $U \in \mathcal{U}$ is an atlas on Y. The inclusion map $Y \hookrightarrow X$ thus becomes a local pospace morphism, and Y is said to be a sub-local pospace of X.

Remark 5.3. The collection of ordered charts induced by a pospace on its open subsets is an atlas. However, the 'obvious' functor $A : \mathbf{PoTop} \to \mathbf{LpoTop}$ is neither full nor injective on objects. Indeed, let X be the sub-pospace of $\mathbb{R}$ over $[0,1] \cup [2,3]$. The homeomorphism from $[0,1] \cup [2,3]$ to itself that swaps the connected components induces an endomorphism of X in **LpoTop**, but not in **PoTop**. Moreover if we let X' be the coproduct in **PoTop** of $[0,1]$ and $[2,3]$, then X and X' have the same image under A though they are not isomorphic in **PoTop** (Sokołowski (2002)).

Definition 5.6. The local pospace induced by a pospace X is its image under A.

The following proposition is an immediate consequence of Lemma 5.1.

Proposition 5.1. Two pospaces on the same underlying space X induce the same local pospace iff every point of X admits an open neighborhood upon which both partial orders coincide. Moreover, a local pospace lies in the image of functor A iff its greatest atlas contains an ordered chart supported by its whole underlying space.

Example 5.2. The *locally ordered real line* is the image under A of the ordered real line in **PoTop** (Example 5.1 and Definition 5.6). The following atlases over $\mathbb{R}$ are equivalent.

1 $\{(I, \leqslant) \mid I \text{ open interval of } \mathbb{R}\}$,
2 $\{(U, \leqslant) \mid U \text{ open subset of } \mathbb{R}\}$,
3 $\{(U, \sqsubseteq_U) \mid U \text{ open subset of } \mathbb{R}\}$ where $x \sqsubseteq_U y$ stands for $x \leqslant y$ and $[x,y] \subseteq U$, and
4 $\{(U, \sqsubseteq'_U) \mid U \text{ open subset of } \mathbb{R}\}$ where $x \sqsubseteq'_U y$ is any extension of $\sqsubseteq_U$.

In particular any subinterval of $\mathbb{R}$ inherits a local pospace structure. A *directed path* on a local pospace X is a morphism from a locally ordered compact interval to X.

Proposition 5.2. If X is a pospace and δ is a directed path on AX starting at x and ending at x', then $x \sqsubseteq_X x'$.

Proof. Let the domain of δ be $[0,r]$. The collection $\mathcal{V}$ of open sub-pospaces of X is an atlas of AX. In particular we have $\sqsubseteq_{X|_V} = \sqsubseteq_V$ for all $V \in \mathcal{V}$. By Definition 5.4, we have

an atlas $\mathcal{U}$ of $[0,r]$ such that for all $U \in \mathcal{U}$ the mapping δ induces a pospace morphism from U to some $V \in \mathcal{V}$. From Example 5.2, we can suppose that all the elements of $\mathcal{U}$ are open intervals of $[0,r]$. Given a finite sequence

$$0 = t_0 < \cdots < t_n = r$$

such that $t_k - t_{k-1}$ is strictly less than the Lebesgue number of the covering $\mathcal{U}$, we have a chart $V_k \in \mathcal{V}$ such that $\delta(t_{k-1}) \sqsubseteq_{V_k} \delta(t_k)$ for all $k \in \{1,\dots,n\}$. Since we have $\sqsubseteq_{X|_{V_k}} = \sqsubseteq_{V_k}$ for all $k \in \{1,\dots,n\}$, we actually have $x = \delta(0) \sqsubseteq_X \delta(r) = x'$. □

Corollary 5.1. If X is a pospace such that $x \sqsubseteq_X x'$ implies the existence of a directed path on AX from x to x' (Example 5.2), then the next equality holds for all pospaces Y.

$$\mathbf{LpoTop}(AX, AY) = \mathbf{PoTop}(X, Y)$$

Proof. Let f be a local pospace morphism from AX to AY and suppose that we have $x \sqsubseteq_X x'$. If γ is a directed path on AX from x to x', then $f \circ \gamma$ is a directed path on AY from $f(x)$ to $f(x')$. From Proposition 5.2 it comes that $f(x) \sqsubseteq_Y f(x')$; and thus f is actually a pospace morphism. □

Corollary 5.2. A directed path δ on a local pospace X is constant iff its extremities are equal and its image is contained in some ordered chart of an atlas of X.

Proof. Let U be a chart as in the statement of the corollary. The restriction of δ to a non-empty interval $[t,t']$ is a directed path on the local pospace AU. From Proposition 5.2 we have $\delta(t) \sqsubseteq_U \delta(t')$. Hence δ is a constant directed path on the pospace U. □

Remark 5.4. From Corollary 5.2 we deduce that every non-constant directed loop on a local pospace meets at least two charts. Put another way, a local pospace has no vortex (*i.e.* no point every neighbourhood of which contains a non-constant directed loop). In particular the complex plane cannot be provided with a local pospace structure whose set of directed paths would be as below, since the origin would then be a vortex.

$$\{\rho(t) \cdot \mathsf{e}^{i\theta(t)} \mid r \geqslant 0 \ ; \ \theta : [0,r] \to \mathbb{R} \ ; \ \rho : [0,r] \to \mathbb{R}_+ \ ; \ \rho, \theta \text{ nondecreasing}\}$$

Example 5.3. The *locally ordered circle* is the local pospace obtained by exporting the ordered real line (Example 5.2) to $\mathbb{S}^1$ through the exponential map.

$$t \in \mathbb{R} \mapsto e^{it} \in \mathbb{S}^1$$

In particular the directed paths δ on the directed circle can be written as $\delta = \mathsf{e}^{i\gamma}$ where γ is a directed path on $\mathbb{R}$. An arc is a connected proper subspace of $\mathbb{S}^1$. Any arc is the image of some interval of $\mathbb{R}$ under the exponential map. The following atlases on $\mathbb{S}^1$ are compatible, the last of them being the greatest one.

1 $\{(A, \leqslant) \mid A \text{ open arc}\}$ where $\leqslant$ is the order induced by $\mathbb{R}$ and the restriction of the exponential map to an open subinterval of $\{t \in \mathbb{R} \mid \mathsf{e}^{\mathsf{i}t} \in A\}$ of length at most 2π,
2 $\{(U, \sqsubseteq_U) \mid U \text{ proper open subset of } \mathbb{S}^1\}$ where $x \sqsubseteq_U y$ means that the anticlockwise compact arc from x to y is included in U, and
3 $\{(U, \sqsubseteq'_U) \mid U \text{ proper open subset of } \mathbb{S}^1\}$ where $\sqsubseteq'_U$ is any extension of the partial order $\sqsubseteq_U$.

In fact, the locally ordered circle is the following coequalizer in **LpoTop**.

$$\{*\} \underset{*\mapsto 1}{\overset{*\mapsto 0}{\rightrightarrows}} [0,1] \longrightarrow \mathbb{S}^1$$

No ordered chart is supported by $\mathbb{S}^1$ hence the locally ordered circle does not arise from a pospace (Proposition 5.1). However, the *unordered circle* (*i.e.* the discrete pospace over $\mathbb{S}^1$) induces a local pospace over the circle. We insist on this example to emphasize the difference between loops in algebraic topology and directed loops.

Remark 5.5. Example 5.3 also offers an opportunity to compare our local pospaces to the original ones. An *original* atlas is a Hausdorff space X together with a collection of pospaces $\mathcal{U}$ whose underlying spaces form an open cover of X and such that every point $x \in X$ comes with a pospace W_x such that

— the underlying space of W_x is a neighborhood of x, and
— given any $U \in \mathcal{U}$ containing x, the pospaces induced on $U \cap W_x$ by U and W_x coincide.

An *original* local pospace is an equivalence class of orginal atlases, both of them being equivalent when their union is again an original atlas. Given a point $x \in X$, the pospace W_x only depends on x. We think that it should not be so. For example, the second atlas described in Example 5.3 is not an atlas in the original sense. Indeed, it contains open dense subsets of arbitrary low non-zero measure (*i.e.* the cumulated length of its connected components). Let x be any point of $\mathbb{S}^1$, its associated pospace W_x contains a neighborhood A of x which is a closed arc. Then there exists an open dense subset D of $\mathbb{S}^1$ which contains x and whose measure is strictly less than the length of A. Let a_0 and a_1 be the extremities of A with a_0 coming before a_1 in the partial order on W_x. Then a_0 and a_1 are not comparable in D because $A \not\subseteq D$ for measure consideration.

From there, one may ask how annoying is that situation. We might as well consider such an atlas as irrelevant. But then, observe that the identity map on $\mathbb{S}^1$ induces an isomorphism between the first and the second atlases on the list of Example 5.3. In fact, given an atlas in the original sense, we obtain an atlas in the sense of Definition 5.2 by gathering, for all $x \in X$, all the open sub-pospaces of W_x containing x (Remark 5.1). In categorical terms the category of original atlases is a full subcategory of the category of atlases in the sense of Definition 5.2. The question of whether the full inclusion is an equivalence, which amounts to have an original atlas in each equivalence classes (Lemma 5.2), is open.

We advocate for shifting to the new formalism for, at least, the following three reasons. First, the original notion of an atlas requires a mapping, namely the one that associates each point with a pospace, besides the collection of charts, whereas the latter should be, in the author's point of view, intrinsic. Second, the new definition is better suited to the analogy with manifolds. Indeed, a collection of charts forms a manifold when the charts are compatible two-by-two. Finally, it is also better suited to local orders arising from a given class of paths of a space, that is to say when for all charts U of the atlas $\mathcal{U}$, we write $x \sqsubseteq_U y$ when there exists a path of the distinguished class from x to y and whose image is wholly contained in U. Such local pospaces may naturally arise when the class of distinguished paths consists of the smooth curves γ on a manifold $\mathcal{M}$ with vector fields

$f_1, \ldots, f_n$ such that for all $t \in \operatorname{dom}\gamma$ the vector $\dot{\gamma}(t)$ is a linear combination of vectors $f_1(t), \ldots, f_n(t)$ with non-negative coefficients. Note that the directed circle falls into this range of examples.

A routine verification proves that the category **LpoTop** is finitely complete and has all coproducts. As we shall see in Section 6.2, local pospaces are sufficiently supple to model all conservative programs.

6. Geometric models of conservative programs

This section aims at building the geometric models of conservative programs from their discrete ones in an elementary way. Let $G_1, \ldots, G_n$ be the running processes of a program P, we denote the set of arrows and vertices of G_i by A_i and V_i. In terms of instruction pointer dynamics, this approach imposes a paradigm shift about the meaning of arrows of control flow graphs. In Definition 2.1, they are interpreted as intermediate positions between instructions. In this context, being inbetween instructions is a qualitative statement. After each arrow of a control flow graph has been replaced by a copy of the open segment $]0,1[$ in the associated metric graph, the abovementioned statement becomes quantitative. Indeed, an intermediate point can be 'close to' the preceding instruction or 'almost on' the next one (which are respectively carried by the source and the target of the arrow along with which the instruction pointer moves).

6.1. *Locally ordered metric graphs*

A *metric graph* is a metric space obtained by assigning a strictly positive length to each arrow of a graph $G : A \rightrightarrows V$. The description we provide is inspired from that of (Bridson and Haefliger (1999)) but in our case, the length of every arrow is supposed to be 1. The underlying set of the metric graph $|G|$ is the disjoint union of V and $A\times]0,1[$. Two elements of that set are said to be *neighbours* when they both belong to $\{\alpha\}\times]0,1[\sqcup \{\partial^-\alpha, \partial^+\alpha\}$ for some arrow α. If $\partial^-\alpha \neq \partial^+\alpha$, there is an obvious bijection ϕ from $\{\alpha\}\times]0,1[\sqcup \{\partial^-\alpha, \partial^+\alpha\}$ to $[0,1]$. If $\partial^-\alpha = \partial^+\alpha$, its codomain is $[0,1[$. The distance between two neighbours p and p' is defined as follows

$$d(p,p') \quad = \quad \begin{cases} |t-t'| & \text{if } \partial^-\alpha \neq \partial^+\alpha \\ \min\left\{|t-t'|\ ,\ 1-t+t'\ ,\ 1-t'+t\right\} & \text{if } \partial^-\alpha = \partial^+\alpha \end{cases}$$

where $t = \phi(p)$ and $t' = \phi(p')$. An *itinerary* is a finite sequence $p_0, \ldots, p_q$ of points of $|G|$ such that p_{k-1} and p_k are neighbours for all $k \in \{1, \ldots, q\}$; its length is defined as the following sum.

$$\ell(p_0, \ldots, p_q) \quad = \quad \sum_{k=1}^{q} d(p_{k-1}, p_k)$$

The distance between two points p and p' of $|G|$ is the greatest lower bound of the set of lengths of itineraries from p to p'. It extends the distance between neighbours.

$$d(p,p') \quad = \quad \inf\left\{\ell(p_0, \ldots, p_q) \;\middle|\; p_0, \ldots, p_q \text{ is an itinerary from } p \text{ to } p'\right\}$$

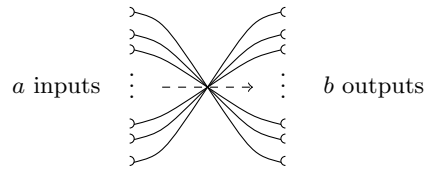


Fig. 7. Ordered open stars.

Let $\mathcal{B}$ be the collection of open balls of radius $\leqslant \frac{1}{3}$ that are centred at a vertex together with the open subintervals of length $\leqslant \frac{1}{3}$ contained in $\{\alpha\}\times]0,1[$ for some arrow α. The collection $\mathcal{B}$ is a basis of the topology induced by the metric graph $|G|$. The open ball of radius $0 < \varepsilon \leqslant \frac{1}{3}$ centred at a vertex v is the disjoint union $\{v\} \cup v_\varepsilon^+ \cup v_\varepsilon^-$ with v_ε^+ (resp. v_ε^-) being the union of $\{\alpha\}\times]0,\varepsilon[$ (resp. $\{\alpha\}\times]1-\varepsilon,1[$) for all $\alpha \in A$ such that $\partial^-\alpha = v$ (resp. $\partial^+\alpha = v$). The other elements of $\mathcal{B}$ are the subsets $\{\alpha\} \times U$ with $\alpha \in A$ and U being an open interval of length $0 < \varepsilon \leqslant \frac{1}{3}$ contained in $]0,1[$. The latter inherit their pospace structures from that of U. The former are equipped with the partial order $\sqsubseteq$ characterized by the following statements (see also Figure 7):

— $v_\varepsilon^- \sqsubset \{v\} \sqsubset v_\varepsilon^+$, and
— each branch inherits from the total order of $\mathbb{R}$.

Each element of the basis $\mathcal{B}$ is thus a chart of $|G|$. Because of the numerical constraint imposed on the radii and the lengths of the elements of the collection $\mathcal{B}$, it is stable under intersection. The valuable property is that for all $B, B' \in \mathcal{B}$, the partial order on $B \cap B'$ is the restriction of the partial order on B, which is much stronger than stating that the collection $\mathcal{B}$ is an atlas (Definition 5.2). The *locally ordered metric graph* associated with G, denoted by $\restriction G \restriction$, is the resulting local pospace together with the distance described above.

Remark 6.1. The locally ordered metric graph associated with G is related to a basic graph transformation. Note that each chart of the atlas $\mathcal{B}$ is isomorphic with the colimit over 0 of a finite family of copies of $\mathbb{R}_+$ and $\mathbb{R}_-$ (*i.e.* the extremities of the half-lines are identified). Such a pospace is called an *ordered open star* and it is entirely characterized by (a,b) with a (resp. b) being the number of copies of $\mathbb{R}_-$ (resp. $\mathbb{R}_+$). In particular, the ordered open star associated with $(0,0)$ is the point $\{0\}$ while the one associated with $(1,1)$ is the ordered real line $\mathbb{R}$. Any point p of the underlying space $|G|$ has a basis of neighborhoods all the elements of which are isomorphic with the same open star, in other words an element of $\mathbb{N}\times\mathbb{N}$ which we call the *type* of p. It is not difficult to see that the collection B of points of $\restriction G \restriction$ whose type differs from $(1,1)$ is finite. No more difficult it is to find a (necessarily finite) collection C that meets every connected component of $\restriction G \restriction$ that is isomorphic with the locally ordered circle exactly once. Then denote the union $B \cup C$ by V' and note that $\restriction G \restriction \setminus V'$ is a finite disjoint union of connected components that are all isomorphic with the ordered open segment $]0,1[$. These components are the arrows of the *reduced* graph of G, denoted by $\mathrm{red}(G)$, while its set of vertices is V'. The source and the target maps are defined with respect to the local pospace structure of $\restriction G \restriction$

in the obvious way. For all finite graphs G and G' we have the following equivalence.

$$\mathrm{red}(G) \quad \cong \quad \mathrm{red}(G') \qquad \Leftrightarrow \qquad \uparrow G\downarrow \quad \cong \quad \uparrow G'\downarrow$$

Given a finite sequence of graphs $G_1, \ldots, G_n$, we would like to equip the local pospace product $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ with a distance. Since the duration of a parallel execution is the duration of the longest execution, the maximum metric is certainly better suited to our context than the Euclidean one. For all $i \in \{1, \ldots, n\}$, let d_i be the distance from the metric graph $|G_i|$. Then every sub-local pospace of $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ is equipped with the distance

$$d_X(p, q) \quad = \quad \max\big\{d_i(p_i, q_i) \;\big|\; i \in \{1, \ldots, n\}\big\}$$

where p_i and q_i are the i^{th} coordinates of p and q.

6.2. *Switching to the geometric framework*

The next proposition is a consequence of (Haucourt, 2012, Corollary 6.7) which plays an important role in the sequel, especially in the proof of Theorem 6.1.

Proposition 6.1. For all directed paths γ on $\uparrow G\downarrow$ and all finite unions X of connected subsets of $\uparrow G\downarrow$, the inverse image of X by γ has finitely many connected components.

Proposition 6.1 is specific to directed topology in the sense that it does not hold for undirected paths on the metric graph $|G|$. For example, consider a path oscillating infinitely many times around to origin of $\mathbb{R}$, the magnitude of the oscillations getting close to zero.

Definition 6.1. For each point p of $(G_1, \ldots, G_n)$ in the sense of Definition 2.2, the *canonical block* B_p is the subset of $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ whose i^{th} component is $\{p_i\}$ when p_i is a vertex, and $\{p_i\}\times]0, 1[$ when it is an arrow. The canonical blocks form the *canonical partition* of $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$. It is finite because the graphs G_i are so. The dimension of B_p is the number of arrows appearing in the n-tuple p. Any union of canonical blocks is said to be a *canonical* subset of $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$. The notion of a directed path on $(G_1, \ldots, G_n)$ (Definition 3.4) was motivated by the following lemma.

Lemma 6.1. If there exists a directed path starting in B_p, ending in $B_{p'}$, and whose image is contained in $B_p \cup B_{p'}$ then one of the following facts is satisfied:
— for all $i \in \{1, \ldots, n\}$, $p_i = p'_i$ or p_i is the source of the arrow p'_i, or
— for all $i \in \{1, \ldots, n\}$, $p_i = p'_i$ or p'_i is the target of the arrow p_i.

Proof. The conclusion is obviously satisfied when $p = p'$ so we assume that p differs from p'. The inverse images $\gamma^{-1}(B_p)$ and $\gamma^{-1}(B_{p'})$ have finitely many connected components, it is a consequence of Proposition 6.1 and the standard equality below.

$$\gamma^{-1}(B_p) \quad = \quad \bigcap_{i=1}^{n} \gamma_i^{-1}(\mathrm{proj}_i(B_p))$$

Hence we have a partition of the domain of γ into intervals which are alternatively contained in $\gamma^{-1}(B_p)$ and $\gamma^{-1}(B_{p'})$, the first interval I being contained in the former while

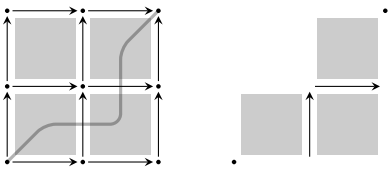


Fig. 8. A directed path and its discretization.

the second one J is contained in the latter. In particular we have $\sup I = \inf J$ which either belongs to I or J. The conclusion follows. □

Definition 6.2. Given a directed path γ on the local pospace $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ we have, by the same arguments as in the proof of Lemma 6.1, a finite partition $I_0 < \cdots < I_N$ of $\operatorname{dom}\gamma$ such that for all $k \in \{0,\ldots,N\}$, there exists a (necessarily unique) point p^k such that $\gamma(I_k) \subseteq B_{p^k}$. By Lemma 6.1, the sequence $p^0,\ldots,p^N$ is a directed path in the sense of Definition 3.4. It is called the *discretization* of γ and denoted by $D(\gamma)$. Conversely, given a directed path δ on $(G_1,\ldots,G_n)$ (Definition 3.4), it is not difficult to find a directed path on $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ whose discretization is δ. Such a directed path is said to be a *lifting* of δ. Discretization and lifting are illustrated in Figure 8.

Lemma 6.2. For all directed paths γ on $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ and all points p' in the canonical block containing $\partial^+\gamma$ (resp. $\partial^-\gamma$), there exists a directed path γ' such that $\partial^+\gamma' = p'$ (resp. $\partial^-\gamma' = p'$) and both γ and γ' have the same discretization.

Proof. Assume that $D(\gamma)$ is the sequence of canonical blocks $B_0,\ldots,B_q$ with $q \in \mathbb{N}$. If $q = 0$, then γ is contained in a single block so the constant path standing on p' fulfils the requirements. Suppose that $q > 0$ and let $p' \in B_q$ be the point that γ' should reach. Then choose a point $p'' \in B_{q-1}$ such that there exists a directed path γ'' from p'' to p' whose discretization is the two elements sequence (B_{q-1}, B_q). By induction, we have a directed path γ''' arriving at p'' whose discretization is $B_0,\ldots,B_{q-1}$. The concatenation of γ'' followed by γ''' is the expected directed path γ'. □

Definition 6.3. The sequence of multi-instructions associated with a directed path γ on $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ is the one associated with its discretization. Following Definition 3.4, the directed path γ is *admissible* (resp. is an *execution trace*) when so is $D(\gamma)$. We define the *sharing function*

$$F' : \uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow \times \mathcal{S} \to \{\text{multisets over } \{1,\ldots,n\}\}$$

by setting $F'(p',s) = (\sigma\cdot D(\gamma))(s)$ where γ is an admissible directed path from the origin of the program to p', and σ is an initial state. As a consequence of Lemma 6.2, for all semaphores $s \in \mathcal{S}$, the mapping $F'(\_,s)$ is constant on every reachable canonical block. Moreover, Definition 4.2 still makes sense for points of $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$ instead of $(G_1,\ldots,G_n)$, and the potential function $|F'|$ instead of $|F|$. We thus obtain the *geometric forbidden region* and the *geometric model* $[\![P]\!]$ of the program, the latter being the complement of the former in $\uparrow G_1\downarrow \times \cdots \times \uparrow G_n\downarrow$. Due to the correspondence between discretization and lifting (Definition 6.2), Theorem 4.1 is still valid with the words 'directed

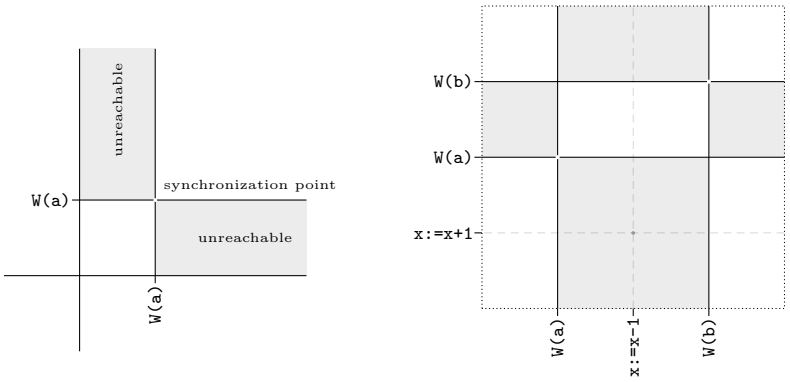


Fig. 9. Binary synchronization: producer *vs* consumer on a flat torus (the opposite edges of the dotted frame are identified).

path', 'forbidden point', and 'admissible' being understood in the geometric context. The notion of a *terminal* point and that of a *deadlock* point from Definition 4.2 readily adapts to the geometric framework $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$, we collect the latter to form the *deadlock space*. The *deadlock attractor* is the set of points x such that for every directed path γ starting at x, there exists a directed path γ' starting at $\partial^+\gamma$ such that $\partial^+\gamma'$ is a deadlock point. The following result is an immediate consequence of Lemma 6.2.

Corollary 6.1. The geometric model, the deadlock space, and the deadlock attractor of a program are canonical (Definition 6.1).

Example 6.1. The right hand model in Figure 9 represents a 'producer *vs* consumer' situation. Using synchronization barriers we ensure that items are made and delivered just-in-time. This example lie on the directed torus, each dotted edge being identified with its opposite. The grayed out subspace is *unreachable* from the origin of the model.

Example 6.2. The models shown in Figure 10 are standard examples of deadlocking programs. The 'Swiss Cross' model and the 'Dining Philosophers' problem appear in (Coffman et al. (1971)) and (Dijkstra (1971)) respectively.

Example 6.3. The models shown in Figure 11 illustrate how drastically sensitive the model of a program is to the arities of the semaphores it uses. The left hand model is obtained with a semaphore of arity 1 while the right hand one is obtained by setting the arity to 2.

Example 6.4. The model depicted in Figure 12 was introduced by (Lipski and Papadimitriou (1981)) as an example of program without deadlock though its 'request graph' has cycles. A careful examination reveals that there is indeed a 'tunnel' going through the geometric forbidden space.

The 'replacement' shown in Figure 6 might seem cryptic in the discrete setting but as shown in Figure 13, it becomes obvious in the light of the continuous one.

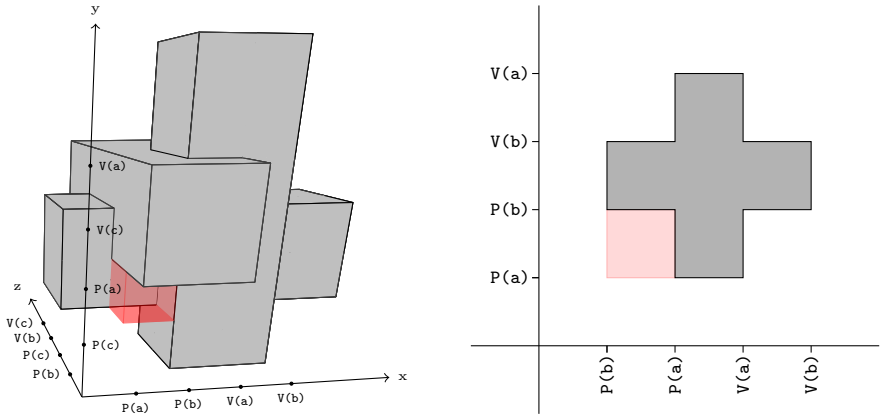


Fig. 10. The three dining philosophers and the Swiss Cross with their deadlock attractors (in red).

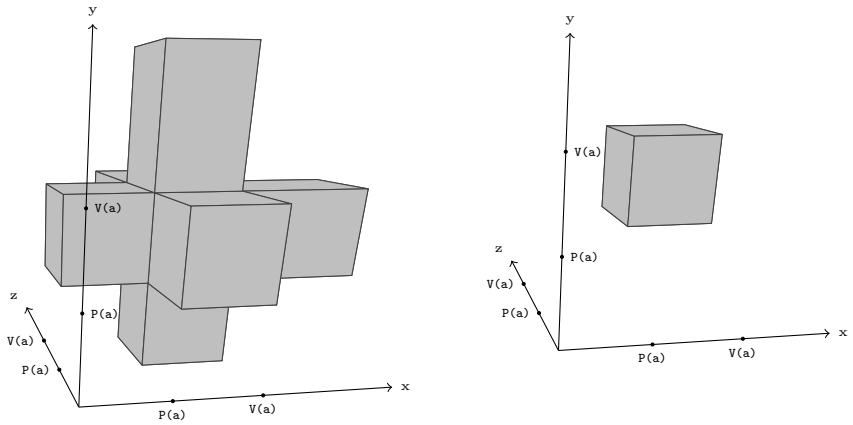
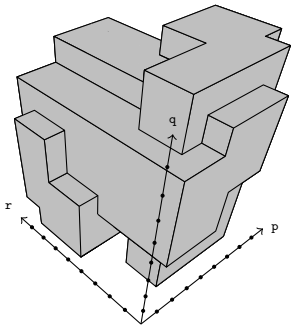


Fig. 11. The tetrahemihexacron a.k.a. 3D Swiss Cross, and the 'floating' cube.




```
sem 1: u v w x y z
proc:
  p = P(a);P(y);P(z);V(a);P(w);V(z);V(y);V(w)
  q = P(u);P(v);P(a);V(u);P(z);V(v);V(a);V(z)
  r = P(y);P(w);V(y);P(u);V(w);P(v);V(u);V(v)
init: p q r
```

Fig. 12. The geometric model of the Lipski algorithm.

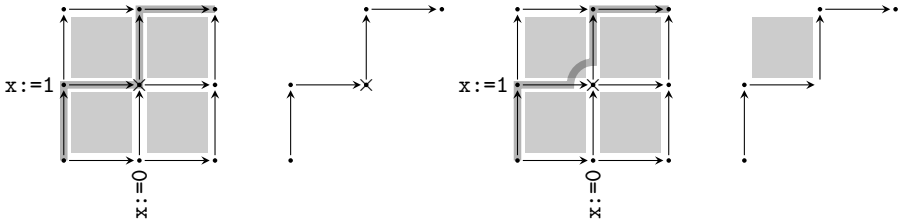


Fig. 13. Comparing the discrete and the continuous approaches: an admissible directed path that meets a forbidden point and a possible replacement for it.

6.3. *Uniform distance and weakly directed homotopy*

Given a compact Hausdorff space K and a metric space (X, d_X), the set of continuous maps from K to X can be equipped with the *uniform distance* (Manetti, 2015, p.122)

$$d(f,g) \quad = \quad \max\{d_X(f(k),g(k)) \mid k \in K\} \ .$$

Any subset $X \subseteq \uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ is equipped with the distance d_X defined in Section 6.1. The projection of a directed path γ on $\uparrow G_i \downarrow$ is denoted by γ_i.

From the computer science point of view, the next theorem states that the uniform distance on directed paths is compatible with the action of admissible directed paths and execution traces, which justifies the use of geometric models in the study of concurrency. Let X be the geometric model of the program P. Given two subsets B and B' of X, we denote by $dX^{[0,r]}(B,B')$ for the set of directed paths $\gamma : [0,r] \to X$ such that $\partial^- \gamma \in B$ and $\partial^+ \gamma \in B'$. This set inherits from the uniform distance.

Theorem 6.1. For all γ in $dX^{[0,r]}(B_p, B_{p'})$, there exists an open ball Ω centred at γ such that all the elements of Ω induce the same action on valuations as γ, and if γ is an execution trace, then so are all the elements of Ω.

Proof. By Section 6.2, the locally ordered space X is a subobject of $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ with each G_i being the control flow graph of a process. The set of vertices of G_i is denoted by V_i. For each coordinate $i \in \{1, \ldots, n\}$, the subspace ${\gamma_i}^{-1}(V_i)$ is the union of *finitely* many pairwise disconnected compact intervals because γ is a directed path (Proposition 6.1). As a consequence there exists a real number $\delta > 0$ such that for all $i, j \in \{1, \ldots, n\}$, for all connected components I and J of ${\gamma_i}^{-1}(V_i)$ and ${\gamma_j}^{-1}(V_j)$, if I and J are disjoint, then so are their δ-neighbourhoods[¶].

In particular, for each $i \in \{1, \ldots, n\}$ and each connected component I of ${\gamma_i}^{-1}(V_i)$, the points $\gamma_i(\min I - \delta)$ and $\gamma_i(\max I + \delta)$ do not belong to V_i. Because each set V_i is finite, the numbers $d_X(\gamma_i(\min I - \delta), V_i)$ and $d_X(\gamma_i(\max I + \delta), V_i)$ are non-zero. The collection of numbers $d_X(\gamma_i(\min I - \delta), V_i)$, $d_X(\gamma_i(\max I + \delta), V_i)$, for i and I ranging through $\{1, \ldots, n\}$ and the connected components of ${\gamma_i}^{-1}(V_i)$, is thus finite and does not contain

[¶] The δ-neighbourhood of a subset of a metric space is the union of all the open balls of radius δ centred at some point of that set (*e.g.* the δ-neighbourhood of $[a,b]$ is $[a-\delta, b+\delta]$).

zero. Note that we implicitly ignore $\min I - \delta$ if it is strictly less than zero, and $\max I + \delta$ if it is strictly greater than r. We let $\varepsilon > 0$ be the least element of that collection, and define $\Omega \subseteq dX^{[0,r]}(B_p, B_{p'})$ as the open ball of radius ε centred at γ.

Given $\gamma' \in \Omega$, the points $\gamma'_i(\min I - \delta)$ and $\gamma'_i(\max I + \delta)$ belong to the same connected component of $|G_i| \setminus V_i$ as the points $\gamma_i(\min I - \delta)$ and $\gamma_i(\max I + \delta)$. Since γ'_i is a local pospace morphism, the set ${\gamma'_i}^{-1}(V_i) \cap [\min I - \delta, \max I + \delta]$ consists of a compact interval. Indeed, the image of the restriction of γ'_i to $[\min I - \delta, \max I + \delta]$ is entirely contained in some ordered open star centred at $\gamma_i(\min I)$, and $\gamma'_i(\min I - \delta)$ is *before* the centre while $\gamma'_i(\max I + \delta)$ is *after*.

The case where I is the first (*i.e.* the leftmost) connected component of ${\gamma_i}^{-1}(V_i)$ deserves a special treatment. If $0 \in I$ (*i.e.* $\gamma_i(0) \in V_i$) then we also have $\gamma'_i(0) \in V_i$ because both γ and γ' *start in the same canonical block*. Otherwise, for the same reason, neither $\gamma_i(0)$ nor $\gamma'_i(0)$ belong to V_i. The same observation applies to the last (*i.e.* the rightmost) connected component of ${\gamma_i}^{-1}(V_i)$, yet with r instead of 0. Moreover, if I_0 and I_1 are two *consecutive* connected components of ${\gamma_i}^{-1}(V_i)$, then the image of the interval $[\max(I_0) + \delta, \min(I_1) - \delta]$ by γ_i does not meet V_i.

It follows that ${\gamma_i}^{-1}(V_i)$ and ${\gamma'_i}^{-1}(V_i)$ have the same number of connected components. We denote by ϕ_i the poset isomorphism from the totally ordered collection of connected components of ${\gamma_i}^{-1}(V_i)$ to that of ${\gamma'_i}^{-1}(V_i)$. Then for any connected component I of ${\gamma_i}^{-1}(V_i)$, the instructions $\lambda_i(\gamma_i(\min I))$ and $\lambda_i(\gamma'_i(\min \phi_i(I)))$ are the same and $\phi_i(I)$ is contained in the δ-neighbourhood of I:

$$\phi_i(I) \quad \subseteq \quad [\min I - \delta; \max I + \delta] \ .$$

In particular for all $i \in \{1, \ldots, n\}$ the dipaths γ_i and γ'_i *share the same discretization*. Following Remark 4.2, we can suppose that the only instructions met are assignments, branchings, and *Skip*. In order to avoid useless distinctions between assignments and branching, the *output* of an instruction will indistinguishably denote the content of the altered variable (in case of an assignment) or the chosen branch (otherwise).

For all $i \in \{1, \ldots, n\}$ and all connected components I of ${\gamma_i}^{-1}(V_i)$, define $\tau_i(I)$ as the greatest element of the set $\{\min I, \min \phi_i(I)\}$. If I is the k^{th} connected component of ${\gamma_i}^{-1}(V_i)$, starting the set of indices at 1, then $\tau_i(I)$ is the first instant at which both γ_i and γ'_i have executed exactly k instructions. In particular one always has the following inequality:

$$\min I - \delta \quad \leqslant \quad \tau_i(I) \quad \leqslant \quad \max I + \delta$$

We prove that for all $i \in \{1, \ldots, n\}$ and all connected components I of ${\gamma_i}^{-1}(V_i)$, the instructions $\lambda_i(\gamma_i(\min I))$ and $\lambda_i(\gamma'_i(\min \phi_i I))$ (which are the same) return the same output after the executions of the sequences of multi-instructions associated with $\gamma|_{[0,\tau_i(I)]}$ and $\gamma'|_{[0,\tau_i(I)]}$. The result is obtained by induction on the finite totally ordered set

$$\mathcal{I} \quad = \quad \big\{ I \subseteq [0,r] \ \big| \ I \text{ connected component of } {\gamma_i}^{-1}(V_i) \text{ for some } i \in \{1, \ldots, n\} \big\}$$

equipped with the lexicographic order (*i.e.* $[a,b] \leqslant [c,d]$ when $a < c$, or $a = c$ and $b \leqslant d$).

The output of the instruction $\lambda_i(\gamma_i(\min I))$ only depends on the contents of finitely many variables that are gathered in the set $\mathcal{F}$. The idea is to apply the induction hy-

pothesis to every ordered pair (j, J) where $j \in \{1, \ldots, n\}$ and J is the greatest connected component of ${\gamma_j}^{-1}(V_j)$ that is strictly less than I. That hypothesis provides us with information about the behaviour of γ and γ' up to $\tau_j(J)$, but the latter might differ from $\tau_i(I)$. So we have to analyse what can happen during the interval of time bounded by the instants $\tau_i(I)$ and $\tau_j(J)$.

If the interval J meets the interval I, then neither the instruction $\lambda_j(\gamma_j(\min J))$ nor the instruction $\lambda_j(\gamma'_j(\min \phi_j(J)))$ (which are the same), alter the content of a variable of $\mathcal{F}$. The reason is that both γ and γ' are *dipath on the geometric model* so they *do not meet any conflicting point*. In that case, we may have $\tau_i(I) \leqslant \tau_j(J)$ but it is harmless with regards to the preceding remark. If the interval J does not meet the interval I then, by definition of δ, we also have the inequality

$$\max J + \delta \quad < \quad \min I - \delta$$

from which we deduce that $\tau_j(J) < \tau_i(I)$. Whether the intervals I and J meet or not, we still have to treat the case of the connected components J' of ${\gamma_j}^{-1}(V_j)$ that are strictly greater than J. As above, either J' meets I and therefore $\lambda_j(\gamma_j(\min J))$ does not alter any variable appearing in $\mathcal{F}$, or $\max I < \min J'$. But in the later case, by definition of δ, we also have $\max I + \delta < \min J' - \delta$, from which we deduce that $\tau_i(I) < \{\min J', \min \phi_j(J')\}$.

If γ is an *execution trace* and if the instruction $\lambda_i(\gamma_i(\min I))$ is a branching, then its output is the arrow α of the graph G_i such that γ_i visits the segment $\{\alpha\}\times]0, 1[$ just after it leaves the point $\gamma_i(\min I)$. According to what we have proven, the branchings $\lambda_i(\gamma_i(\min I))$ and $\lambda_i(\gamma'_i(\min \phi_i I))$, which are the same, return the same output at the end of the directed paths $\gamma|_{[0,\tau_i(I)]}$ and $\gamma'|_{[0,\tau_i(I)]}$. Hence γ' is an execution trace too. □

Continuous deformations of paths are basic objects in algebraic topology which have a natural counterpart in directed topology. Weakly directed homotopies have been introduced by (Fajstrup et al. (2006)), yet Definition 6.4 slightly differs from the original concept: instead of requiring that the intermediate paths be inextendible, we require that they all have the same endpoints, as usual in algebraic topology (Brown, 2006, p.207).

Definition 6.4. Let $\gamma, \delta : [0, r] \to X$ be two paths on a topological space with $r \geqslant 0$, $\gamma(0) = \delta(0)$, and $\gamma(r) = \delta(r)$. A homotopy h from γ to δ is a continuous map defined over $[0, r] \times [0, q]$ for some $q \geqslant 0$ such that:

— for all $s \in [0, q]$, $h(0, s) = \gamma(0)$ and $h(r, s) = \gamma(r)$, and

— for all $t \in [0, r]$, $h(t, 0) = \gamma(t)$ and $h(t, q) = \delta(t)$.

Assuming that X is a local pospace, h is said to be *weakly directed* when all the intermediate paths $h(\_, s)$ are directed. Two directed paths are said to be *weakly dihomotopic* when there exists a weakly directed homotopy between them. Given a conservative program, we thus obtain an equivalence relation upon the collection of directed paths on its geometric model. The sequence of weakly directed homotopies on the "flat" torus which are depicted in Figure 14 shows that, as in classical algebraic topology, the "vertical" directed circle is weakly dihomotopic to the "horizontal" one. The effect of weakly directed homotopies on the corresponding sequences of multi-instructions is illustrated in Figure 15 and explained below.

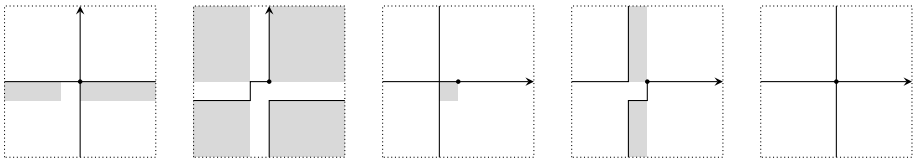


Fig. 14. Dihomotopies on the flat torus.

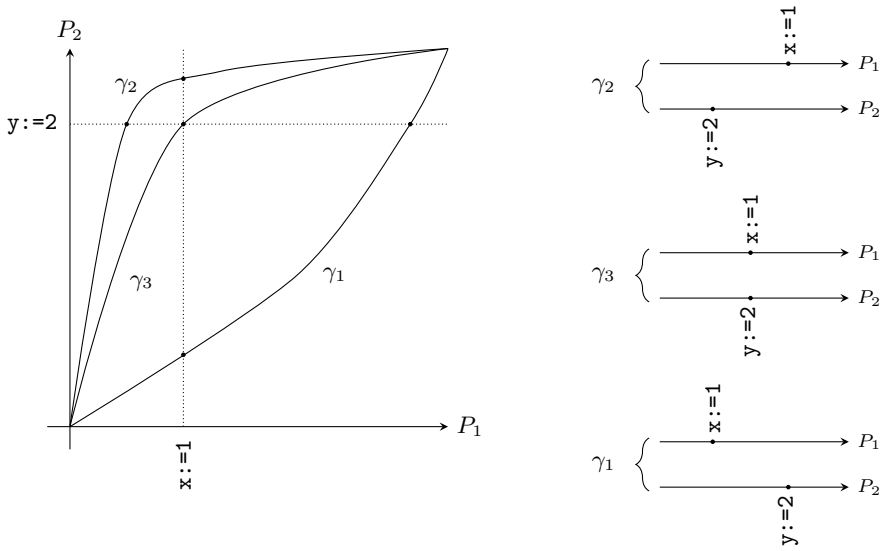


Fig. 15. Timelines and sequences of multi-instructions interpreting weakly dihomotopic directed paths.

Corollary 6.2. Two weakly dihomotopic directed paths on the geometric model of a conservative program induce the same action on the valuations. Moreover, one is an execution trace if and only if the other is so.

Proof. Let h be a weakly directed homotopy. By a standard result from general topology (Manetti, 2015, p.152) and with the notation of Definition 6.4, the mapping

$$\hat{h} : s \in [0,q] \mapsto (t \in [0,r] \mapsto h(t,s) \in X)$$

is a path on $dX^{[0,r]}(B_p, B_{p'})$. Its image is thus compact, we cover it with open balls given by Theorem 6.1. By the Lebesgue number theorem (Manetti, 2015, p.192) there exists a real number $\varepsilon > 0$ such that $|s - s'| < \varepsilon$ implies that $\hat{h}(s)$ and $\hat{h}(s')$ belong to the same open ball from the covering. The conclusion follows. □

Building on the geometric models and following the intuition that Cartesian product of models should represent parallel composition, we introduce another notion of independence that only applies to conservative programs.

Definition 6.5. The conservative programs $P_1, \dots, P_N$ are said to be model independent when they are compatible (Definition 2.3) and the following equality holds.

$$[\![P_1|\cdots|P_N]\!] \quad = \quad [\![P_1]\!] \times \cdots \times [\![P_N]\!]$$

Remark 6.2. It readily comes from Definition 2.4 that syntactically independent programs are model independent. However, model independent programs might not be syntactically independent. It suffices to consider the program P made of two copies of the process $P(a); P(c); V(c); V(a)$ and the program Q made of two copies of the process $P(b); P(c); V(c); V(b)$ assuming that a and b are semaphores of arity 1 while c is a semaphore of arity 2. Then P and Q are compatible in the sense of Definition 2.3 therefore we can consider their parallel composition $P|Q$. A direct yet rather tedious calculation proves that $[\![P|Q]\!] = [\![P]\!] \times [\![Q]\!]$. But one can also get convinced of this fact by a simple reasoning. Due to the semaphore a, the two processes of P cannot hold more that one occurrence of c. As the same applies to program Q, the parallel composition $P|Q$ never uses more than two occurrences of c. Consequently, the forbidden region generated by c is included in the forbidden region generated by a and b. From the computer science point of view, it means that c has no influence on the executions of the program $P|Q$ so it is harmless to drop it. Denote by P' and Q' the programs obtained by removing the instructions $P(c)$ and $V(c)$ from P and Q, then P' and Q' are separated and we have $[\![P|Q]\!] \cong [\![P'|Q']\!]$. This example is due to (Balabonski and Haucourt (2010)).

Theorem 6.2. Model independent programs are observationally independent.

Proof. Let $S_1 \sqcup \cdots \sqcup S_N$ be the associated partition of the set of running processes of the parallel composition $P_1|\cdots|P_N$ (Definition 3.6) with n being the total number of running processes. Let δ be an execution trace (Definition 3.4) and denote by $(\mu_0, \dots, \mu_{q-1})$ its associated sequence of multi-instructions. First, we treat the case where the permutation is a rolling $(0 \cdots q-1)$ compatible with δ (Definitions 3.6 and 3.7). The idea is to consider a lifting of δ and to deform it (by a weakly directed homotopy) so it becomes a directed path whose discretization is associated with the sequence of multi-instructions $(\mu'_0, \dots, \mu'_{q-1})$ with $\mu'_k = \mu_{\rho^{-1}(k)}$. If we succeed, then we have, by Corollary 6.2, that both sequences of multi-instructions $(\mu_0, \dots, \mu_{q-1})$ and $(\mu'_0, \dots, \mu'_{q-1})$ come from execution traces and induce the same action on valuations. The general case will follow from the rolling decomposition of the compatible permutation to treat (Corollary 3.1). As ρ is compatible with δ we have $J \cap J' = \emptyset$ with J and J' defined below.

$$J \quad = \quad \big\{ j \in \{1, \dots, N\} \;\big|\; \operatorname{dom} \mu_{q-1} \cap S_j \neq \emptyset \big\}$$

$$J' \quad = \quad \big\{ j \in \{1, \dots, N\} \;\big|\; \operatorname{dom} \mu_k \cap S_j \neq \emptyset \text{ for some } k \in \{0, \dots, q-2\} \big\}$$

The projections proj_J and $\mathrm{proj}_{J'}$ send a point of $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ to the extracted tuples of components whose indices respectively belong to the following sets.

$$\bigcup_{j \in J} S_j \qquad\qquad \bigcup_{j' \in J'} S_{j'}$$

Therefore we have a lifting γ of δ (Definition 6.2) that can be written as a concatenation $\gamma = \gamma_{J'} \cdot \gamma_J$ such that $\mathrm{proj}_J \circ \gamma_{J'}$ and $\mathrm{proj}_{J'} \circ \gamma_J$ are constant. As an instance of Godement

exchange law between composition and concatenation, we have the directed paths γ_1 and γ_2 defined below.

$$\gamma_1 = \mathrm{proj}_{J'} \circ \gamma = \gamma_{J'} \cdot \mathrm{cst} \qquad \gamma_2 = \mathrm{proj}_J \circ \gamma = \mathrm{cst} \cdot \gamma_J$$

So we have $\gamma = (\gamma_1, \gamma_2)$ with the convention that we omit all the 'harmless' components, that is to say the ones in following set.

$$\{1, \ldots, n\} \quad \setminus \bigcup_{j \in J \cup J'} S_j \ .$$

Since $P_1, \ldots, P_N$ are model independent, the geometric model $[\![P_1 | \cdots | P_N]\!]$ is the local pospace product $[\![P_1]\!] \times \cdots \times [\![P_N]\!]$. Therefore, assuming that both γ_1 and γ_2 are defined over $[0,1]$, the mapping below induces a local pospace morphism from $[0,1]^2$ to the geometric model of the parallel composition.

$$(x, y) \mapsto (\gamma_1(x), \gamma_2(y))$$

By precomposing with a weakly directed homotopy from $[0,1] \times \{0\} \ \cup \ \{1\} \times [0,1]$ to $\{0\} \times [0,1] \ \cup \ [0,1] \times \{1\}$ we obtain a weakly directed homotopy from $\gamma = \gamma_{J'} \cdot \gamma_J$ to $\gamma' = \gamma'_J \cdot \gamma'_{J'}$ with $\gamma_{J'}$, γ_J, γ'_J, and $\gamma'_{J'}$ being characterized by the relations below:

— $\mathrm{proj}_J \circ \gamma'_J = \mathrm{proj}_J \circ \gamma_J \quad$ and $\quad \mathrm{proj}_{J'} \circ \gamma'_{J'} = \mathrm{proj}_{J'} \circ \gamma_{J'}$, and

— $\mathrm{proj}_J \circ \gamma'_{J'} = \mathrm{cst} \quad$ and $\quad \mathrm{proj}_{J'} \circ \gamma'_J = \mathrm{cst}$.

The sequence of multi-instructions associated with γ' is $(\mu'_0, \ldots, \mu'_{q-1})$ and we are done. □

Remark 6.3. From Remark 3.1 we deduce that observationally independent programs might not be model independent. Gathering the results of the article, we have proven the following chain of strict implications.

$$\text{syntactic independence} \Rightarrow \text{model independence} \Rightarrow \text{observational independence}$$

Relevance of model independence goes beyond the theoretical aspect since there exists a unique decomposition theorem for cubical regions together with factorization algorithms (Balabonski and Haucourt (2010); Ninin (2017)). In addition, the Boolean algebra of n-dimensional cubical regions is the n-fold tensor product of the Boolean algebra of 1-dimensional cubical regions (Haucourt and Ninin (2014)). These results should be extendible to all isothetic regions and gathered in a more general formulation.

7. Isothetic regions over compact metric graphs

The canonical subsets of the local pospace $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ obviously form a finite Boolean subalgebra of its powerset. Moreover, one easily checks that the topological interior and closure of a canonical subset of $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ is again canonical. In fact we even prove that the set of points of $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ that are visited by directed paths starting in a canonical subset of $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ is canonical. The purpose of this section is to extend the family of canonical subsets to a collection enjoying the same properties but that depends on the local pospace structure of $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$ instead

of the graphs $G_1, \ldots, G_n$. As a Boolean algebra, it is generated by the connected subsets of the compact[‖] metric graphs $\uparrow G_i \downarrow$ for $i \in \{1, \ldots, n\}$ and comes with basic operators which are extensively described in the sequel. Our intention is to convince the reader that all of them derive from basic operations over the intervals of $\mathbb{R}$ and therefore can be automated.

A finite product of nonempty intervals is called a *cube*. One of the remarkable properties of the collection of cubes is that it is stable under monotonic union. This feature is entirely due to the fact that a monotonic union of connected sets is connected. Another remarkable property is that the collection of finite unions of cubes of a given dimension n forms a Boolean subalgebra of $\mathrm{Pow}(\mathbb{R}^n)$. In this section, we adapt the preceding result to products of compact metric graphs instead of $\mathbb{R}^n$. In order to facilitate the presentation, the terms of that product are supposed to be equal to $\uparrow G \downarrow$ for a given finite graph G.

Proposition 7.1. The collection $\mathcal{R}_1 G$ of all finite unions of connected subsets of $|G|$ forms a Boolean subalgebra of $\mathrm{Pow}(|G|)$ which is isomorphic to the following Cartesian product of Boolean algebras

$$\mathcal{R}_1 G \quad \cong \quad \mathrm{Pow}(V) \times (\mathcal{R}_1]0,1[)^{\mathrm{card}\, A}$$

with A (resp. V) being the set of arrows (resp. vertices) of G, and $\mathcal{R}_1]0,1[$ being the Boolean algebra of finite unions of subintervals of $]0,1[$.

Proof. First note that $\mathcal{R}_1 G$ is always stable under binary unions. Moreover, there is a canonical bijection between the connected components of $|G|$ and that of G. Since the latter is finite, $|G|$ belongs to $\mathcal{R}_1 G$. Let C be a connected subset of $|G|$. For all arrows α, the set $\{\alpha\}\times]0,1[\cap C^c$ is the union of at most two separated intervals (*i.e.* contained in disjoint open subsets). From the last remark and the finiteness of G, we deduce that $\mathcal{R}_1 G$ is stable under complement and therefore that it is indeed a Boolean subalgebra of $\mathrm{Pow}(|G|)$. The isomorphism readily follows. □

Definition 7.1. A *block* of dimension $n \in \mathbb{N}$, or *n-block*, is the product of n connected nonempty subsets of the metric graph $|G|$. A collection of blocks is called a *block covering* of $X \subseteq |G|^n$ when the union of its elements is X. A block contained in X is said to be a block of X. Such a block is said to be *maximal* when no block of X strictly contains it. The *maximal block covering* of $X \subseteq |G|^n$ is the set of all its maximal blocks, it is denoted by $\alpha_n(X)$. In particular $\alpha_1(X)$ is the collection of connected components of X. Also note that $\alpha_n(X) = \emptyset$ if and only if $X = \emptyset$. An *isothetic region* of dimension n is a subset of $|G|^n$ that admits a finite block covering. The collection of isothetic regions of dimension n is denoted by $\mathcal{R}_n G$ and the collection of sets of n-blocks is denoted by $\mathrm{Cov}_n G$. The latter is preordered by the relation $\preccurlyeq$ defined below.

$$C \preccurlyeq C' \quad \equiv \quad \forall w \in C \; \exists w' \in C', \; w \subseteq w'$$

If G is isomorphic to $0 \to 1 \to \cdots \to x \to x+1$ for some $x \in \mathbb{N}$, then the corresponding regions are said to be *cubical*.

[‖] It is well-known that the metric graph $|G|$ is compact iff the graph G is finite.

The term 'isothetic region' is borrowed from (Preparata and Shamos, 1985, p.329). By Corollary 6.1, the geometric model of any conservative program is an isothetic region.

Remark 7.1. The maximal block covering of X may contain strict sub-coverings of X. For example, consider the maximal block $[1,2]\times[0,3]$ of $[0,2]^2\cup[1,3]^2$.

Remark 7.2. Given two subsets X and X' of $|G|^n$, any maximal block of the intersection $X\cap X'$ is a maximal block of $w\cap w'$ for some maximal blocks w of X and w' of X'.

Lemma 7.1. Any block of $X\subseteq|G|^n$ is contained in a maximal block of X.

Proof. Let w be a block of $X\subseteq|G|^n$. By the Hausdorff maximal principle there exists some maximal $\subseteq$-chain of blocks of X containing w. The union of this chain is calculated component by component, each of them being a monotonic union of connected subsets of the metric graph $|G|$. The result is a maximal block of X containing w. □

Remark 7.3. Given two maximal block coverings D and D', the following are equivalent:

— $D=D'$

— $\bigcup D=\bigcup D'$

— any block of D is included in a block of D', and vice versa.

Proposition 7.2. We obtain a Galois connection (γ_n,α_n) between Cov_nG and $\mathrm{Pow}(|G|^n)$ defining $\gamma_n(D)$ as $\bigcup D$ for all $D\in\mathrm{Cov}_nG$.

$$\mathrm{Cov}_nG\underset{\alpha_n}{\overset{\gamma_n}{\rightleftarrows}}\mathrm{Pow}(|G|^n)$$

In particular $\gamma_n\circ\alpha_n=\mathrm{id}$. That Galois connection induces an isomorphism of Boolean algebras between $\mathrm{Pow}(|G|^n)$ and the image of α_n (*i.e.* the collection of maximal block coverings).

Proof. The collection Cov_nG is equipped with the preorder $\preccurlyeq$ from Definition 7.1. One readily checks that both α_n and γ_n are monotonic. A point of X is connected, so it can be seen as a block. The equality $\gamma_n\circ\alpha_n=\mathrm{id}$ thus follows from Lemma 7.1. Remark 7.3 actually states that the preorder $\preccurlyeq$ induces a partial order on the image of α_n and that the restriction of $\alpha_n\circ\gamma_n$ to the image of α_n is the identity. Hence (γ_n,α_n) induces a poset isomorphism between $\mathrm{Pow}(|G|^n)$ and the image of α_n. Boolean algebras can be defined as partially ordered sets satisfying some properties expressed in terms of finite least upper bounds and finite greatest lower bounds. Their morphisms are poset morphisms that preserve these bounds. Thus any poset that is isomorphic to a Boolean algebra is a Boolean algebra itself. Hence the image of α_n is a Boolean algebra. □

We now prove that $X\subseteq|G|^n$ is an isothetic region iff its maximal block covering is finite, thus giving the higher dimensional version of Proposition 7.1 and an analog of Proposition 7.2 for isothetic regions. To achieve this aim, we examine how the maximal blocks of a binary intersection and of a complement are obtained from the maximal blocks of their operands. The next statement is a standard result from general topology.

Lemma 7.2. The connected components of a product of spaces are the products of connected components of those spaces.

Lemma 7.3. If $w = w_1 \times \cdots \times w_n$ is a block, then $\alpha_n(w^c)$ is the collection of blocks w' such that there exists $i \in \{1, \ldots, n\}$ such that w'_i is a connected component of $|G| \setminus w_i$ and for $j \neq i$, w'_j is a connected component of $|G|$. Moreover, if the graph G is finite, then w^c has finitely many maximal blocks.

Proof. The complement of a block w can be written as the finite union of the sets

$$X_i \quad = \quad |G| \times \cdots \times |G| \times \underbrace{w_i^c}_{i^{th} \text{ position}} \times |G| \times \cdots \times |G|$$

for i ranging through $\{1, \ldots, n\}$. Given a block w' of w^c there exists $i \in \{1, \ldots, n\}$ such that $w'_i \not\subseteq w_i$. Since w' is connected so is w'_i which is therefore contained in some connected component of w_i^c (*i.e.* $|G| \setminus w_i$). In other words w' is contained in X_i and thus in one of its connected components, which has the expected form by Lemma 7.2. The finiteness derives from Proposition 7.1. □

Remark 7.4. Computing the maximal block covering of $X \cap X'$ amounts to gather all the maximal block coverings of $w \cap w'$ for $w \in \alpha_n(X)$ and $w' \in \alpha_n(X)$, and then to drop all the elements that are strictly contained in another one. Fortunately, the maximal block covering of a given intersection $w \cap w'$ is easy to compute. It is indeed the collection of products $c_1 \times \cdots \times c_n$ where for all $i \in \{1, \ldots, n\}$, c_i is a connected component of $w_i \cap w'_i$. That description is obtained from the set theoretical relation below and Lemma 7.2.

$$w \cap w' = (w_1 \cap w'_1) \times \cdots \times (w_n \cap w'_n)$$

The collection of finite sets of n-blocks is denoted by $\mathrm{Cov}_{nf}\, G$.

Theorem 7.1. Suppose that the graph G is finite. The collection of n-dimensional isothetic regions $\mathcal{R}_n G$ forms a Boolean subalgebra of $\mathrm{Pow}(|G|^n)$ and the Galois connection from Proposition 7.2 restricts to a Galois connection between $\mathrm{Cov}_{nf}\, G$ and $\mathcal{R}_n G$, which induces an isomorphism of Boolean algebras between $\mathcal{R}_n G$ and the image of α_n (*i.e.* the collection of finite maximal block coverings).

$$\mathrm{Cov}_{nf}\, G \underset{\alpha_n}{\overset{\gamma_n}{\rightleftarrows}} \mathcal{R}_n G$$

Proof. From Remark 7.2, Lemma 7.3 (which holds because G is finite), and De Morgan's laws, we deduce that the complement of an isothetic region has finitely many maximal blocks. It follows that any subset of $|G|^n$ is an isothetic region iff it has finitely many maximal blocks. The restriction of α_n is therefore well-defined. □

The collection $\mathcal{R}_n G$ has further interesting stability properties which we now examine. The *interior* and the *closure* of X are respectively the greatest open subset of $|G|$ contained in X and the the least closed subset of $|G|$ containing X. They are denoted by $\mathrm{int}(X)$ and $\mathrm{clo}(X)$. The closure of X is also denoted by $\overline{X}$. Also denote the *boundary* of X (*i.e.* the set difference $\mathrm{clo}(X) \setminus \mathrm{int}(X)$) as $\mathrm{bnd}(X)$.

Proposition 7.3. The Boolean algebra $\mathcal{R}_nG$ is stable under closure, boundary, and interior operators.

Proof. The closure operator preserves finite products, therefore it preserves blocks. Moreover it preserves finite unions, hence it preserves isothetic regions. The boundary of a subset is the intersection of its closure and the closure of its complement (Manetti, 2015, p.44), hence it also preserves isothetic regions. The interior of a set is difference between its closure and its boundary. It follows that the interior operator also preserves isothetic regions. □

From a practical point of view, the crucial property of isothetic regions is that the Boolean algebra $\mathcal{R}_nG$ also inherits from the local pospace structure of $\uparrow G \downarrow^n$.

Definition 7.2. The *forward* operator is defined for all subsets A and B of a local pospace X as the union of the images of the directed paths on $A \cup B$ starting in A.

$$\mathrm{frw}(A,B) = \bigcup \big\{ \mathrm{img}(\delta) \;\big|\; \delta \text{ directed path of } X;\ \partial^- \delta \in A;\ \mathrm{img}(\delta) \subseteq A \cup B \big\}$$

Dually, the *backward* operator is defined as the union of the images of the directed paths on $A \cup B$ ending in A.

$$\mathrm{bck}(A,B) = \bigcup \big\{ \mathrm{img}(\delta) \;\big|\; \delta \text{ directed path of } X;\ \partial^+ \delta \in A;\ \mathrm{img}(\delta) \subseteq A \cup B \big\}$$

Until the end of this section, capital letters A, B and C, with or without index, are subsets of a local pospace X. Yet, Definition 7.2 obviously makes sense for d-spaces (Grandis (2003)) instead of local pospaces.

Remark 7.5. Both operators are $\subseteq$-increasing in both variables. Also note that for all subsets A and B, we have $A \subseteq \mathrm{frw}(A,B) \subseteq A \cup B$, and that if A and B are disconnected (*i.e.* neither A nor B meets the closure of the other) then $\mathrm{frw}(A,B) = A$. Note however that the converse is false (*e.g.* $A = [1,2]$ and $B = [0,1]$). The same obviously holds for the backward operator. Moreover, if A is (path) connected then so are $\mathrm{frw}(A,B)$ and $\mathrm{bck}(A,B)$.

Lemma 7.4. If $A_k \subseteq B_k$ for all $k \in \{1,\dots,n\}$, then the forward and the backward operators preserve products.

$$\mathrm{frw}(A_1 \times \cdots \times A_n, B_1 \times \cdots \times B_n) = \mathrm{frw}(A_1,B_1) \times \cdots \times \mathrm{frw}(A_n,B_n)$$

$$\mathrm{bck}(A_1 \times \cdots \times A_n, B_1 \times \cdots \times B_n) = \mathrm{bck}(A_1,B_1) \times \cdots \times \mathrm{bck}(A_n,B_n)$$

Proof. If δ is a directed path on $B_1 \times \cdots \times B_n$ with $\partial^- \delta \in A_1 \times \cdots \times A_n$, then for all $k \in \{1,\dots,n\}$, $\mathrm{pr}_k \circ \delta$ is a directed path on B_k whose source belongs to A_k. Conversely, given some n-tuple of directed paths $(\delta_1,\dots,\delta_n)$ with δ_k directed path on B_k with its source in A_k, the directed path defined by $t \mapsto (\delta_1(t),\dots,\delta_n(t))$ has its source in $A_1 \times \cdots \times A_n$ and its image contained in $B_1 \times \cdots \times B_n$. □

The inclusion assumption cannot be dropped from the statement of Lemma 7.4, indeed taking the disconnected sets $A = [0,1[\times [0,1]$ and $B = [1,2] \times]1,2]$ we have $\mathrm{frw}(A,B) = A$ though $\mathrm{frw}([0,1[,[1,2]) = \mathrm{frw}([0,1],]1,2]) = [0,2]$.

Definition 7.3. For all $A \subseteq X$ we define

— the *future closure* $\overline{A}^{\mathrm{f}}$ as $\mathrm{frw}(A, \overline{A})$ and the *past closure* $\overline{A}^{\mathrm{p}}$ as $\mathrm{bck}(A, \overline{A})$.

— the *future cone* $\mathrm{cone}^{\mathrm{f}}(A)$ as $\mathrm{frw}(A, X)$ and the *past cone* $\mathrm{cone}^{\mathrm{p}}(A)$ as $\mathrm{bck}(A, X)$.

— The subset A is said to be *future stable* when $\mathrm{cone}^{\mathrm{f}} A = A$. The *past stable* subsets are defined dually.

Remark 7.6. The future (resp. past) cone is future (resp. past) stable. A subset is future stable iff its complement is past stable. The collection of future (resp. past) stable subsets of X forms a sub-complete lattice of $\mathrm{Pow}(X)$.

Remark 7.7. As a consequence of Lemma 7.4, the future closure and the future cone operators preserve products.

$$\overline{A_1 \times \cdots \times A_n}^{\mathrm{f}} = \overline{A_1}^{\mathrm{f}} \times \cdots \times \overline{A_n}^{\mathrm{f}}$$

$$\mathrm{cone}^{\mathrm{f}}(A_1 \times \cdots \times A_n) = \mathrm{cone}^{\mathrm{f}}(A_1) \times \cdots \times \mathrm{cone}^{\mathrm{f}}(A_n)$$

Dually, the same holds for the past closure and the past cone operators.

From the operators of Definition 7.3 one defines some meaningful new ones.

Definition 7.4. The *future escape* of A is the set of points of X whose future cones avoid A. It is denoted by $\mathrm{escape}^{\mathrm{f}} A$. the *past attractor* of A is the set of points $x \in X$ such that any directed path starting at x can be extended to a directed path arriving in A. It is denoted by $\mathrm{att}^{\mathrm{p}} A$. The *past escape* and the *future attractor* of A, denoted by $\mathrm{escape}^{\mathrm{p}} A$ and $\mathrm{att}^{\mathrm{f}} A$, are defined dually.

From a mere reformulation of Definition 7.4 we obtain the following fact.

Proposition 7.4. The past attractor is future stable and can be written in terms of past cone, future escape, and Boolean operators.

$$\mathrm{att}^{\mathrm{p}} A \quad = \quad \mathrm{cone}^{\mathrm{p}} A \setminus \mathrm{cone}^{\mathrm{p}}\Big((\mathrm{cone}^{\mathrm{p}} A)^c\Big) \quad = \quad \mathrm{escape}^{\mathrm{f}}(\mathrm{escape}^{\mathrm{f}} A)$$

In general, the past attractor is not past stable.

Lemma 7.5. For all $A \subseteq B \subseteq C$ we have

$$\mathrm{frw}(A, C) = \mathrm{frw}(\mathrm{frw}(A, B), C) \quad \text{and} \quad \mathrm{bck}(A, C) = \mathrm{bck}(\mathrm{bck}(A, B), C)$$

Proof. The left member is contained in the right one by Remark 7.5 and because $A \subseteq \mathrm{frw}(A, B)$ (no assumption upon A, B, nor C is required here). Conversely we have $\mathrm{frw}(A, B) \subseteq B \subseteq C$, then let γ be a directed path from $\mathrm{frw}(A, B)$ to C whose image is contained in C. In particular there exists a directed path δ from A to $\partial^- \gamma$ whose image is contained in B, hence in C. Their concatenation $\gamma \cdot \delta$ therefore starts in A and has its image contained in C. □

Once again the inclusion assumption cannot be dropped. Taking $A = [0, 1]$, $B = [1, 2]$, and $C = [2, 3]$ provides an obvious counter-example. The next proposition is a key ingredient when it comes to practical computations.

Proposition 7.5. Let A be a subset of a local pospace X. Under the assumption that $\delta^{-1}(A)$ has finitely many connected components for all directed paths δ on X, the forward and the backward operators can be written as follows.

$$\begin{aligned} \mathrm{frw}(A,B) &= A \cup \mathrm{frw}(\overline{A}^{\mathrm{f}} \cap B, B) \cup \mathrm{frw}(A \cap \overline{B}^{\mathrm{p}}, B) \\ \mathrm{bck}(A,B) &= A \cup \mathrm{bck}(\overline{A}^{\mathrm{p}} \cap B, B) \cup \mathrm{bck}(A \cap \overline{B}^{\mathrm{f}}, B) \end{aligned}$$

Proof. Let δ be a directed path on $A \cup B$ starting in A and such that $\partial^+\delta \notin A$ (hence $\partial^+\delta \in B$). Let C be the connected component of $\delta^{-1}(B)$ that contains 1 and denote by t_0 its greatest lower bound. Then any neighborhood of t_0 contains some $t < t_0$ such that $\delta(t) \notin B$, and therefore $\delta(t) \in A$. By hypothesis on A the last connected component of $\delta^{-1}(A)$ makes sense. Its least upper bound is t_0. Then $\delta(t_0) \in \overline{A}^{\mathrm{f}}$. If $\delta(t_0) \in B$ then $\delta|_{[t_0,1]}$ is a directed path whose image is contained in B. It follows that $\partial^+\delta \in \mathrm{frw}(\overline{A}^{\mathrm{f}} \cap B, B)$. If $\delta(t_0) \notin B$ the image of $\delta|_{[t_0,1]}$ is then included in $\overline{B}^{\mathrm{p}}$ and $\delta(t_0) \in A$ (since $\mathrm{img}(\delta) \subseteq A \cup B$). It follows the image of $\delta|_{[t_0,1]}$ in included in $A \cap \overline{B}^{\mathrm{p}} \cup B$ and therefore $\partial^+\delta \in \mathrm{frw}(A \cap \overline{B}^{\mathrm{p}}, B)$.

Conversely, suppose there exists a directed path δ starting in $\overline{A}^{\mathrm{f}} \cap B$ and whose image is contained in B. Then consider a directed path γ starting in A and such that $\partial^+\gamma = \partial^-\delta$. The image of the conctenation $\delta \cdot \gamma$ is then contained in $A \cup B$. Therefore $\partial^+\delta \in \mathrm{frw}(A,B)$. The inclusion $\mathrm{frw}(A \cap \overline{B}^{\mathrm{p}}, B) \subseteq \mathrm{frw}(A,B)$ follows from Remark 7.5. The result for the backward operator is obtained by duality. □

Note that the extra hypothesis is only required for the first parameter. Also note that in the case where X is the d-space $\mathbb{R}$ equipped with the chaotic direction (*i.e.* all continuous maps from $[0,1]$ to $\mathbb{R}$ are directed) this property fails for all A but $\emptyset$ and $\mathbb{R}$. Given any $t \in \mathbb{R}$ there is indeed a path converging to t that oscillates infinitely many times around t. As we shall see the isothetic regions behave much better. From now on we strengthen the hypotheses upon local pospaces represented by capital letters A, B, and C assuming they are actually isothetic regions contained in $X = \uparrow G \downarrow^n$. The next result is an immediate consequence of Proposition 6.1.

Proposition 7.6. For all isothetic regions A and all directed paths γ on $\uparrow G \downarrow^n$, the inverse image $\gamma^{-1}(A)$ has finitely many connected components.

Corollary 7.1. Under the hypotheses of Lemma 6.1 (*i.e.* there exists a directed path starting in B_p, ending in $B_{p'}$, and whose image is contained in $B_p \cup B_{p'}$) we have

$$\mathrm{frw}(B_p, B_{p'}) = \mathrm{bck}(B_{p'}, B_p) = B_p \cup B_{p'} \ .$$

Theorem 7.2.
The Boolean algebra $\mathcal{R}_n G$ is stable under forward and backward operators.

Proof. First we prove that for all isothetic regions A and B compatible with the canonical partition (Definition 6.1) and for all directed paths γ on $A \cup B$ starting in A, every canonical block met by γ is contained in $\mathrm{frw}(A,B)$. Let $B_{p_1}, \ldots, B_{p_N}$ be the discretization of γ (Definition 6.2). By definition B_{p_1} meets A which contains it by compatibility with

the canonical partition. Suppose that $N \geqslant 2$. By Corollary 7.1 and because the forward operator is monotonic in both arguments, $\mathrm{frw}(A, B)$ also contains B_{p_2}. In particular we have $\mathrm{frw}(A, B) = \mathrm{frw}(A \cup B_{p_2}, B)$. Let γ' be a restriction of γ whose discretization is $B_{p_2}, \ldots, B_{p_N}$. Then γ' is a directed path on $A \cup B$ starting in $A \cup B_{p_2}$. The result immediately follows from an induction on the parameter $N \geqslant 1$. Formally we have proven the following inclusion, its converse being an immediate consequence of Definition 7.2.

$$\bigcup \{ B_p \mid \exists \delta \text{ directed path on } A \cup B \text{ that starts in } A \text{ and meets } B_p \} \subseteq \mathrm{frw}(A, B)$$

The general case is obtained by replacing each graph G_i by a graph G'_i with the same reduced graph (Remark 6.1), the graphs G'_i being chosen so that A and B are compatible with the canonical partition of $\uparrow G'_1 \downarrow \times \cdots \times \uparrow G'_n \downarrow$. For $i \in \{1, \ldots, n\}$, let G'_i be the graph whose vertices are all the vertices of G_i plus all the points of $\uparrow G_i \downarrow$ that belongs to the boundary of $\mathrm{proj}_i(M)$ for some maximal block M of A or B. The set V'_i of vertices of G'_i is thus a discrete subspace of $\uparrow G_i \downarrow$, the arrows of G'_i are the connected components of $\uparrow G_i \downarrow \setminus V'_i$. Each of them is isomorphic, as a pospace, to $]0, 1[$, so the source and the target maps of G'_i are defined accordingly. In particular one has a local pospace isomorphism Φ_i from $\uparrow G_i \downarrow$ to $\uparrow G'_i \downarrow$ whose restriction to V'_i is the identity. The tuple $(\Phi_1, \ldots, \Phi_n)$ is thus a local pospace isomorphism Φ from $\uparrow G'_1 \downarrow \times \cdots \times \uparrow G'_n \downarrow$ to $\uparrow G_1 \downarrow \times \cdots \times \uparrow G_n \downarrow$. Due to its specific form Φ also induces an isomorphism between the corresponding Boolean algebras of isothetic regions. In addition $\Phi(A)$ and $\Phi(B)$ are compatible with the canonical partition of $\uparrow G'_1 \downarrow \times \cdots \times \uparrow G'_n \downarrow$ and therefore $\mathrm{frw}(A, B) = \Phi^{-1}(\mathrm{frw}(\Phi(A), \Phi(B)))$ is an isothetic region. Note that G_i and G'_i have isomorphic reduced graphs. □

Corollary 7.2. Given isothetic regions A and B, the sets $\overline{A}^{\mathrm{f}}$, $\overline{A}^{\mathrm{p}}$, $\mathrm{frw}(\overline{A}^{\mathrm{f}} \cap B, B)$, $\mathrm{bck}(\overline{A}^{\mathrm{p}} \cap B, B)$, $\mathrm{frw}(A \cap \overline{B}^{\mathrm{p}}, B)$, and $\mathrm{bck}(A \cap \overline{B}^{\mathrm{f}}, B)$ are isothetic regions.

Corollary 7.2 combined with Propositions 7.5 and 7.6 provides the algorithms that compute the forward and backward operators on isothetic regions. The next corollary immediately derives from Proposition 7.4.

Corollary 7.3. If $A, X \in \mathcal{R}_n G$ with $A \subseteq X$, then the future and the past attractors of A (in X) belong to $\mathcal{R}_n G$.

8. Perspectives and related works

We discuss some related works, further developments, and open problems related to computer science and mathematics.

8.1. *Precubical sets*

The category of *precubical sets* can be defined as the presheaf category $\mathbf{Set}^{\square^{+\mathrm{op}}}$ where $\square^+$ is the small category generated by the face inclusions with $\varepsilon \in \{0, 1\}$, $n \in \mathbb{N}$, and $i \in \{0, \ldots, n\}$.

$$\delta^n_{i,\varepsilon} : (x_0, \ldots, x_{n-1}) \in [0,1]^n \mapsto (x_0, \ldots, x_{i-1}, \varepsilon, x_i, \ldots, x_{n-1}) \in [0,1]^{n+1}$$

Fig. 16. Cartesian product *vs* tensor product of precubical sets.

Its connection with concurrency theory is brought out by labelling precubical set elements to obtain the notion of a higher dimensional automaton (Pratt (1991); van Glabbeek (1991))[††]. Given a precubical set K, the elements of $K(n)$ for $n \in \mathbb{N}$ are its n-dimensional elements. The dimension of K is the least natural number d such that $K(n)$ is empty for all $n > d$. The category of graphs thus appears as the full subcategory of 1-dimensional precubical sets. Following the way sequences of multi-instructions are built from directed paths on tuple of graphs (Definition 3.4) an element of dimension n is an intermediate position from which n processes can execute their next instruction simultaneously. This strongly suggests that the running processes $G_1, \ldots, G_n$ of a program (Definition 2.2) should be combined to form a precubical set of dimension n from which the forbidden elements would be removed. Cartesian product naturally comes to mind but actually does not fit at all because, for example, a Cartesian product of graphs is still a graph. In fact, the right notion is that of *tensor product* of precubical sets (see Figure 16) which is a slight and even simpler variation on the tensor product of cubical sets (Brown et al., 2011, p.373). In particular the elements of $G_1 \otimes \cdots \otimes G_n$ are precisely the points of $G_1, \ldots, G_n$ (Definition 2.2). Tensor products of graphs thus seem to be a reasonable alternative to Cartesian products of metric graphs. However, removing an element e from a precubical set is not an obvious operation: in doing so, one also has to drop all the elements e' whose border contains e, and so on so forth. From the mathematical point of view, the problem is that the subobjects of a precubical set do not form a Boolean algebra. Therefore, modelling parallel programs by means of higher dimensional automata, we implicitly make a strong assumption which actually derives from the mere presence of the face operators. The upper corner of an n-dimensional cube c is labelled with a tuple of instructions among which only a subset of n components, namely $\{i \in \{1, \ldots, n\} \mid c_i \text{ is an arrow}\}$, can be executed in parallel. The upper corner of any back face of that cube has the same upper corner, but the subset of available instructions now contains n' elements, with n' being the dimension of that face. According to the foregoing interpretation, it means that if a multi-instruction is admissible, all its sub-multi-instructions are so. In other words, the expressiveness of higher dimensional automata easily allows one to prevent processes from synchronizing, but not to force them to synchronize. In this context, we cannot express that all the instructions of a given multi-instruction *must* be executed simultaneously. Specifically, the 'wait' instruction $W(\_)$ has no obvious semantics in terms

[††] A detailed account of the origin of higher dimensional automata as well as their relation to other models of concurrency can be found in (Pratt (2000)). The similar notion of a *higher dimensional transition system* induces a category that is proven to be equivalent to that of non-degenerate higher dimensional automata (Cattani and Sassone (1996)). The relation between both notions has been thoroughly studied by (Gaucher (2010)). Many traditional models of concurrency are actually subsumed by higher dimensional automata (van Glabbeek (2006); Goubault and Mimram (2012)).

of higher dimensional automata. For example, if b is a synchronization barrier of arity 2 and s is a square in some higher dimensional automata, then the upper corner of s cannot be labelled with $(W(b), W(b))$. As suggested by (Fahrenberg and Legay (2015)) one could relax the definition of precubical sets allowing the face maps to be partial, but this approach would come with many technical issues we do not want to deal with here. One of them being that the face maps, which send n-dimensional elements to $(n-1)$-dimensional ones, do not 'generate' partial precubical sets. To conclude this section, we draw reader's attention to the fact that synchronizing instruction like 'wait' are by no means artificial, they are in fact an essential feature of modern parallel programming (POSIX Thread).

8.2. *A closely related work*

The present paper was written in parallel with a book dealing with the same subject (Fajstrup et al. (2016)). However, their approaches differ on several points. The book is based on the toy language PIMP (p.8), viz a parallel extension of the language IMP (Winskel (1993)) together with Dijkstra's instructions $P(\_)$ and $V(\_)$ (p.27), allowing the parallel composition operator to occur anywhere in a program. On the contrary we only consider programs with parallel composition in outermost position. For example we reject the program `x:=0;(x:=1||y:=1);y:=0`. That restriction allows simple middle-end representations of programs (Section 2) and actually preserves us from handling oversized models. In the book, deciding whether a program is conservative or not actually requires to compute (at least) its *transition graph* (p.13), that is to say the one-dimensional skeleton of its precubical model. By opposition, we only have to deal with each sequential process independently from the others (Lemma 4.1). In fact the geometric models described in the book are built inductively, and involve tensor products of precubical sets which are responsible for combinatorial explosions (p.62). On the contrary, a careful examination of the framework offered by isothetic regions reveals that in practice, it can be used without performing any tensor product of precubical sets. From the semantic point of view, the geometric models defined in the book do not take conflicts (nor synchronization barriers) into account. Instead, a program is said to be *coherent* precisely when it satisfies the conclusion of Corollary 6.2. Moreover, the presence of synchronisation instructions $W(\_)$ in a given program is not without consequences for the topology of its geometric model. If such an instruction actually contributes to the forbidden region of a program, then its geometric model is not locally compact. In particular, the results from (Fajstrup (2005)) and (Krishnan (2013)) cannot be applied to such isothetic regions. The cubical region $\{(x, y) \in \mathbb{R}^2 \mid x = 0 \Leftrightarrow y = 0\}$, which is depicted in Figure 9, is the simplest illustration of that phenomenon.

8.3. *Hemi-metric spaces*

A *hemi-metric*[‡‡] on a set X is a mapping d from $X \times X$ to $\mathbb{R}_+ \cup \{\infty\}$ satisfying the triangle inequality and $d(x,x) = 0$ for all $x \in X$. The ordered pair (X,d) is called a *hemi-metric space* (Goubault-Larrecq, 2013, p.203). For example, a natural hemi-metric on the unit circle consists of defining the distance $d(p,q)$ from p to q as the minimal length of clockwise arcs from p to q. In that case $d(q,p) = 2\pi - d(p,q)$. In general, the length $\ell(\gamma)$ of a path γ on a metric space (X,d) is the supremum of the sums

$$\sum_{k=0}^{N} d\big(\gamma(t_k), \gamma(t_{k-1})\big)$$

taken over all $N \in \mathbb{N}$ and all the finite sequences $t_0 < \ldots < t_N$ with $\operatorname{dom}\gamma = [t_0, t_N]$ (Bridson and Haefliger, 1999, p.12). If X is an isothetic region together with the distance d_X introduced in Section 6.1, we obtain a hemi-metric on X defining the distance from p to q as the infimum of $\ell(\gamma)$ over all the *directed* paths γ from p to q, with the convention that the infimum of the empty set is ∞ (Bridson and Haefliger, 1999, p.32).

Together with 1-Lipschitz maps, viz $f : X \to Y$ such that $d_Y(f(x), f(x')) \leqslant d_X(x,x')$ for all $x, x' \in X$, hemi-metric spaces form a cocomplete category (Goubault-Larrecq, 2013, p.236). Since the latter category naturally contains cubes in all dimensions we can define the hemi-metric realization of precubical sets, which is studied and compared to the usual geometric realization by (Goubault and Mimram (2016)).

8.4. *Continuous Kripke structures*

Drawing from (Clarke et al., 2000, p.14) we define a Kripke structure over a set AP of *atomic propositions* as a triple $M = (S, R, L)$ where:

— S is the finite set of states.
— $R \subseteq S \times S$ is a binary relation called the *accessibility* relation.
— $L : S \to \mathrm{Pow}(\mathrm{AP})$ labels each state with the set of atomic propositions that are true in that state.

Following (Blackburn et al., 2008, p.9), the syntax of the modal language is given below.

$$\phi ::= p \mid \bot \mid (\neg\phi) \mid (\phi \ \vee \ \phi) \mid \Diamond\phi$$

It is interpreted, for any state s, as follows:

— $M, s \Vdash p$ iff $p \in L(s)$
— $M, s \Vdash \bot$ is false
— $M, s \Vdash \phi_1 \vee \phi_2$ iff $M, s \Vdash \phi_1$ or $M, s \Vdash \phi_2$
— $M, s \Vdash \Diamond\phi$ iff there exists a sequence of states $s_0 s_1 s_2 \ldots s_N$ such that $M, s_N \Vdash \ \phi$, $s_0 = s$, and $s_{i-1} R s_i$ or $s_{i-1} = s_i$ for all $i \in \{1, \ldots, N\}$.

In doing so, we drift from the orginal approach (Blackburn et al., 2008, p.9) by implicitly assuming that (S, R) is a preorder. Therefore it is natural to focus on the corresponding

[‡‡] See (Lawvere (1973)) for a historical reference.

modal logic, to wit **S4**. In particular (S, R) can be understood as a graph G whose vertices are the states and whose arrows are the ordered pairs (s, s') such that sRs'. We then ask whether (S, R), which is called a *frame* in (Blackburn et al., 2008, p.16), can be replaced by the locally ordered metric graph $\uparrow G\downarrow$. In that case the sequences of states are replaced by directed paths on $\uparrow G\downarrow$ and the language is interpreted as above except for the operator $\Diamond$ whose semantics is adapted accordingly:

— $M, s \Vdash \Diamond\phi$ iff there exists a directed path on $\uparrow G\downarrow$ from s to s' such that $M, s' \Vdash \phi$.

According to the description of the metric graph given at the beginning of Section 6.1, the set S is actually a finite subspace of $\uparrow G\downarrow$. Hence we have to equip $\uparrow G\downarrow$ with a labelling L' that extends L in order to obtain some kind of Kripke structure over $\uparrow G\downarrow$. In this regard we observe that, as a consequence of Definition 6.2 in the one-dimensional case, the following equivalence holds for all $s \in S$, all extensions L' of L, and all modal logic formula ϕ.

$$(S, R, L), s \Vdash \phi \qquad \Leftrightarrow \qquad (\uparrow G\downarrow, L'), s \Vdash \phi$$

Conversely, given a local pospace Σ together with a labelling $L : U\Sigma \to \mathrm{Pow}(\mathrm{AP})$ over the underlying set of Σ we have what we call a *continuous Kripke structure*. One easily deduce a usual Kripke model of **S4** taking the elements of $U\Sigma$ as states together with the following accessibility relation.

$$R \quad = \quad \{(x, y) \mid \text{there exists a directed path on } X \text{ from } x \text{ to } y\}$$

As before we observe that, as a consequence of Definition 6.2 in the one-dimensional case, the following equivalence holds for all $s \in U\Sigma$, and all modal logic formula ϕ.

$$(\Sigma, L), s \Vdash \phi \qquad \Leftrightarrow \qquad (U\Sigma, R, L), s \Vdash \phi$$

Modelling **S4** by standard Kripke structures or continuous ones is therefore a matter of taste. Similar arguments would have led to the conclusion that one can indifferently model **S4** with small categories. Indeed their corresponding accessibility relation turn them into preordered sets. This is because the satisfaction of an **S4** formula only depends on the existence of a (directed) path between two given points.

One is tempted to go further and consider temporal logics. For this purpose we introduce $\mathrm{CTL}^*_\varepsilon$, a modified version of CTL^* (Clarke et al., 2000, p.42) as a case study. The syntax of $\mathrm{CTL}^*_\varepsilon$ offers two types of formulas, the state ones and the path ones. They are respectively gathered in the collections SF and PF which are mutually recursively defined using the atomic propositions, the path quantifiers E and A, the temporal operators X_ε for $\varepsilon \in \mathbb{R}_+$, F, G, U, and all the Boolean connectives (Clarke et al., 2000, p.29). In other words SF and PF are characterized by the following constraints:

— $\mathrm{AP} \subseteq \mathrm{SF} \subseteq \mathrm{PF}$,
— both SF and PF are stable under Boolean connectives,
— $\{Ef, Af \mid f \in \mathrm{PF}\} \subseteq \mathrm{SF}$, and
— $\{X_\varepsilon f, Ff, fGg, fUg \mid f, g \in \mathrm{PF},\ \varepsilon \in \mathbb{R}_+\} \subseteq \mathrm{PF}$.

In particular all the formulas of $\mathrm{CTL}^*_\varepsilon$ are path formulas but each one is assigned a type according to the smallest set of the filtration $\mathrm{AP} \subseteq \mathrm{SF} \subseteq \mathrm{PF}$ it belongs to. The type of a $\mathrm{CTL}^*_\varepsilon$ formula only matters when it comes to the semantics. Let $M = (\Sigma, L)$ be

a continuous Kripke structure. As for standard CTL* semantics, we have to consider infinite directed paths on Σ to define CTL$^*_\varepsilon$ semantics, in other words local pospace morphisms $\pi : \mathbb{R}_+ \to \Sigma$. Given a state (resp. path) formula f the notation $M, s \vDash f$ (resp. $M, \pi \vDash f$) means that f holds at state s (resp. along path π) in M. The inductive definition of $\vDash$ is a carbon copy of the one given in (Clarke et al., 2000, p.29) except that i, j and k belongs to $\mathbb{R}_+$ instead of $\mathbb{N}$. In particular the operators X_ε and U are interpreted as below with π^ε denoting the ε-suffixe of π (*i.e.* $t \in \mathbb{R}_+ \mapsto \pi(t+\varepsilon) \in \Sigma$):

— $M, \pi \vDash X_\varepsilon f$ iff $M, \pi^\varepsilon \vDash f$
— $M, \pi \vDash fUg$ iff $\exists r \in \mathbb{R}_+ \ (\forall \varepsilon \in [0, r[\ \ M, \pi^\varepsilon \vDash f$ and $M, \pi^r \vDash g)$

On that occasion we remark that the equivalences thereinafter hold.

$$X_0 f \ \equiv \ f \qquad\qquad X_{\varepsilon_1 + \cdots + \varepsilon_n} f \ \equiv \ X_{\varepsilon_1} \cdots X_{\varepsilon_n} f$$

In particular the operator X_0 is useless and for all $n \in \mathbb{N}$ the operator X_n is obtained from X_1 setting $\varepsilon_k = 1$ for all $k \in \{1, \ldots, n\}$ in the preceding formula. The preceding observation no longer holds in the context of continuous Kripke structure so we indeed need to introduce the operators X_ε for all $\varepsilon \in \mathbb{R}_+$, though $\mathbb{R}_+$ could have been replaced by any of its nontrivial initial segments.

Let us go back to our initial motivation and let Σ be the geometric model of a conservative program P. Following Definition 2.2, we let $\mathcal{X}$ be the finite set of all the variables appearing in the program P, all of them ranging through $\mathbb{R}$. Assuming that $\mathcal{X}$ is totally ordered and contains N elements, the set $\mathcal{X}$ can be seen as an N-tuple of variables without repetitions. In this context any valuation δ can be seen as a point of $\mathbb{R}^N$, hence $\mathrm{AP} = \mathbb{R}^N$ and $L(s) \subseteq \mathbb{R}^N$. As a by-product of Definitions 3.4 and 6.2, each infinite directed path π on Σ is associated with a sequence of valuations (δ_*) and a strictly increasing sequence (a_*) of elements of $\mathbb{R}_+$ such that for all $t \in [a_k, a_{k+1}[$, the valuation δ_k is $\delta_0 \cdot \pi|_{[0,t]}$. Note that both sequences may be finite (*e.g.* when π is constant beyond a given value). However if one is infinite, then so is the other and the sequence (a_*) goes to infinity.

$$\lim_{n \to +\infty} a_n = +\infty$$

Mimicking the notion of collecting semantics, one defines the interpretation L as below.

$$L(s) \quad = \quad \{\delta_0 \cdot \pi|_{[0,t]} \mid \pi(t) = s\}$$

Everything has been settled to obtain the next statement for all $k \in \mathbb{N}$ that is less than the length of the sequence (δ_*) .

$$M, \pi \quad \vDash \quad (\mathcal{X} = \delta_0) \ U \ (\mathcal{X} = \delta_1) \ U \ (\mathcal{X} = \delta_2) \ U \ \cdots \ U \ (\mathcal{X} = \delta_k) \ U \ \top$$

Moreover, given $a_i \leqslant b < a_{i+1}$ and $a_j \leqslant c < a_{j+1}$, if $\pi|_{[b,c]}$ and γ, both defined on $[b, c]$, are weakly dihomotopic then we have the following fact by Corollary 6.2.

$$M, \pi|_{[0,b]} \cdot \gamma \cdot \pi^c \quad \vDash \quad (\mathcal{X} = \delta_0) \ U \ \cdots \ U \ (\mathcal{X} = \delta_i) \ U \ (\mathcal{X} = \delta_j) \ U \ \cdots \ U \ (\mathcal{X} = \delta_k) \ U \ \top$$

In doing so we have formalized, in CTL$^*_\varepsilon$ logic, the intuitive fact that one can locally replace a segment of an execution trace by a weakly dihomotopic one without altering its long time behaviour. However, a global reparametrization like $\theta : t \in \mathbb{R}_+ \mapsto e^t \in \mathbb{R}_+$

may drastically change the validity of a formula along a path in a given model since the operators X_ε allows, for example, to express the fact that some property is satisfied at a given instant.

Acknowledgements

The author would like to express his warmest thanks to the first reviewer for the outstanding quality of his work.

References

Allen, F. E. Control Flow Analysis. In *Proc. of a Symp. on Compiler Optimization*, pp. 1–19, New York, NY, USA, 1970. ACM.

Balabonski, T. and Haucourt, E. A Geometric Approach to the problem of Unique Decomposition of Processes. In *Concurrency Theory 21th Internat. Conf.*, vol. 6269 of *Lect. Notes in Computer Sci.*, pp. 132–146. Springer, 2010.

Blackburn, P., de Rijke, M., and Venema, Y. *Modal Logic.* Tracts in Theoret. Computer Sci. (53). Cambridge University Press, 3rd print. ed., 2008.

Bridson, M. R. and Haefliger, A. *Metric Spaces of Non-Positive Curvature*, vol. 319 of *Grundlehren der mathematischen Wissenschaften.* Springer, 1999.

Brown, R. *Topology and Groupoids.* BookSurge Publishing, 2006.

Brown, R., Higgins, P. J., and Sivera, R. *Nonabelian Algebraic Topology.* Tracts in Mathematics. European Math. Society, 2011.

Bubenik, P. Context for Models of Concurrency. In *Proc. of the Workshops on Geom. and Topological Methods in Concurrency Theory (GETCO 2004+2005+2006)*, vol. 230 of *Electronic Notes in Theoret. Computer Sci.*, pp. 3–21. Elsevier, 2009.

Bubenik, P. and Worytkiewicz, K. A model category for local pospaces. *Homology, Homotopy and Applications*, 8(1):263–292, 2006.

Carson, S. D. and Reynolds Jr., P. F. The Geometry of Semaphore Programs. *ACM Trans. on Programming Lang. and Syst.*, 9(1):25–53, 1987.

Cattani, G. L. and Sassone, V. Higher Dimensional Transition Systems. In *11th Symp. of Logics in Computer Sci., LICS '96*, pp. 55–62. IEEE, 1996.

Clarke, E. E., Grumberg, O., and Peled, D. A. *Model Checking.* MIT Press, 2nd print. ed., 2000.

Coffman, E. G., Elphick, M., and Shoshani, A. System Deadlocks. *ACM Computing Surv.*, 3(2):67–78, 1971.

Cridlig, R. Semantic Analysis of Shared-memory Concurrent Languages Using Abstract Model-checking. In *Proc. of the 1995 ACM SIGPLAN Symp. on Partial Evaluation and Semantics-based Program Manipulation*, pp. 214–225, New York, NY, USA, 1995.

Cridlig, R. Implementing a Static Analyzer of Concurrent Programs: Problems and Perspectives. In Dam, M., editor, *Analysis and Verification of Multiple-Agent Lang., selected papers from the 5th LOMAPS, June 24-26, 1996*, vol. 1192, pp. 244–259, London, UK, 1997. Springer.

Dijkstra, E. W. Cooperating sequential processes. In Genuys, F., editor, *Programming Lang.: NATO Advanced Study Institute*, proc. of the summer school held at Villars-de-Lans, 1966, pp. 43–112. Academic Press, 1968. Reprint of the eponymous technical report published by the Technological Univ. of Eindhoven, The Netherlands, 1965. Also in Hansen, P. B., editor, *The Origin of Concurrent Programming*, Springer 2002.

Dijkstra, E. W. Hierarchical Ordering of Sequential Processes. *Acta Informatica*, 1: 115–138, 1971.

Fahrenberg, U. and Legay, A. Partial Higher-Dimensional Automata. In Moss, L. S. and Sobociński, P., editors, *6th Conf. on Algebra and Coalgebra in Computer Sci.*, vol. 35 of *Leibniz Internat. Proc. in Informatics*, pp. 101–115, Dagstuhl, Germany, 2015.

Fahrenberg, U. Towards an Efficient Algorithm for Detecting Unsafe States in Timed Concurrent Sytems. Master's thesis, The Faculty of Engineering and Sci., 2002.

Fajstrup, L. Dipaths and dihomotopies in a cubical complex. *Advances in Appl. Math.*, 35(2):188–206, 2005.

Fajstrup, L., Goubault, É., and Raußen, M. Algebraic Topology and Concurrency. *Theoret. Computer Sci.*, 357(1):241–278, 2006. Also technical report R-99-2008, Department of Math. Sci.s, Aalborg university, DK-9220 Aalborg Øst. 1999.

Fajstrup, L., Goubault, É., Haucourt, E., Mimram, S., and Raußen, M. *Directed Algebraic Topology and Concurrency.* Springer Briefs in Appl. Sci. and Technology - Math. Methods. Springer, 2016.

Floyd, R. W. Assigning meanings to programs. In Schwartz, J. T., editor, *Math. Aspects of Computer Sci.*, vol. 19 of *Proc. of Symposia in Appl. Mathematics*, pp. 19–32. American Math. Society, 1967.

Gaucher, P. Directed algebraic topology and higher dimensional transition systems. *New York Jrnl. of Math.*, 16:409–461 (electronic), 2010.

Goubault, É. Geometry and Concurrency: A user's guide. *Math. Struct. in Comp. Science*, 10(4):411–425, 2000.

Goubault, É. and Mimram, S. Formal Relationships Between Geometrical and Classical Models for Concurrency. *Electronic Notes in Theoret. Comp. Sci.*, 283:77–109, 2012.

Goubault, É. and Mimram, S. Directed Homotopy in Non-Positively Curved Spaces. Submitted to *Logical Methods in Computer Science.*

Goubault-Larrecq, J. *Non-Hausdorff Topology and Domain Theory*, vol. 22 of *New Math. Monographs.* Cambridge University Press, 2013.

Goubault-Larrecq, J. Exponentiable Streams and Prestreams. *Appl. Categorical Structures*, 22:514–549, 2014.

Grandis, M. Directed Homotopy Theory, I. The Fundamental Category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 44(4):281–316, 2003.

Hansen, P. B. *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls.* Springer, 2002.

Haucourt, E. Streams, d-Spaces and their Fundamental Categories. *Electronic Notes in Theoret. Computer Sci.*, 283:111–151, 2012.

Haucourt, E. and Ninin, N. The Boolean Algebra of Cubical Areas as a Tensor Product in the Category of Semilattices with Zero. In *7th Interaction and Concurrency Experience (ICE 2014)*, Electronic Proc. in Theoret. Computer Sci., 2014.

Kahl, T. A fibration category of local pospaces. *Electronic Notes in Theoret. Computer Sci.*, 230:129–140, 2009.

Krishnan, S. A Convenient Category of Locally Preordered Spaces. *Appl. Categorical Structures*, 17(5):445–466, 2009.

Krishnan, S. Cubical Approximation for Directed Topology I. *Appl. Categorical Structures*, 23(2):177–214, 2013.

Lawvere, F. W. Metric Spaces, Generalized Logic, and Closed Categories. In *Rendiconti del seminario matématico e fisico di Milano*, vol. XLIII, pp. 135–166, 1973. Reprints in Theory and Applic. of Categories, No. 1 (2002) pp. 1-37.

Lipski, W. and Papadimitriou, C. H. A Fast Algorithm for Testing for Safety and Detecting Deadlocks in Locked Transaction Systems. *Jrnl. of Algorithms* (2), pp. 211–226, Academic Press, 1981.

Manetti, M. *Topology*, vol. 91 of *La Matematica per il 3+2*. Springer, 2015.

Nachbin, L. *Topology and Order*, vol. 4 of *Math. Studies*. Van Nostrand, 1965.

Ninin, N. *Factorisation des régions cubiques et applications à la concurrence*. PhD thesis, Université Paris 11 Orsay, 2017.

Pratt, V. Modeling Concurrency with Geometry. In Wise, D. S., editor, *Proc. 18th Ann. ACM Symp. on Princ. of Programming Lang.*, pp. 311–322, 1991.

Pratt, V. Higher dimensional automata revisited. *Math. Struct. in Comp. Science*, 10 (4):525–548, 2000.

Preparata, F. P. and Shamos, M. I. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Sci.. Springer, 2nd print. ed., 1985.

Sokołowski, S. A case for po-manifolds. In *Preliminary Proc. of the Workshop on Geom. and Topology in Concurrency Theory GETCO'02*, BRICS Notes Series. Basic Reasearch in Computer Sci., 2002.

van Glabbeek, R. J. Bisimulations for Higher Dimensional Automata. Manuscript available electronically at `http://theory.stanford.edu/~rvg/hda`, 1991.

van Glabbeek, R. J. Erratum to 'On the Expressiveness of Higher Dimensional Automata'. *Theoret. Computer Sci.*, 368(1-2):168–194, 2006.

Winskel, G. *The Formal Semantics of Programming Languages: An Introduction*. MIT Press, Cambridge, 1993.