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

Geometry and Concurrency: A User's Guide

ERIC GOUBAULT

LETI (CEA - Technologies Avancées)†
DEIN-SLA-CEA F 91191 Gif-sur-Yvette Cedex, France
Email: Eric.Goubault@cea.fr

Received 31 January 2000

Geometrical methods in concurrency theory (and in distributed systems theory) have appeared recently for modelling and analyzing systems' behaviours and also for solving computability and complexity issues. We identify some of the main directions of research and survey some of the major ideas on which all this is based on (some of which are more than thirty years old).

1. Introduction

"Geometry and Concurrency" is not yet a well-established domain of research, but is rather made of a collection of seemingly related techniques, algorithms and formalizations, coming from different application areas, accumulated over a long period of time. There is currently a certain amount of effort made for unifying these (in particular see the article (Gunawardena, 1994)), following the workshop "New Connections between Computer Science and Mathematics" held at the Newton Institute in Cambridge, England in November 1995 (and sponsored by HP/BRIMS). More recently, the first workshop on the very same subject has been held in Aalborg, Denmark (see `http://www.math.auc.dk/~raussen/admin/workshop/workshop.html` where the articles of this issue, among others, have been first sketched.

But what is "Geometry and Concurrency" composed of then? It is an area of research made of techniques which use geometrical reasoning for describing and solving problems appearing in concurrent systems. Here geometrical reasoning means mostly using techniques from algebraic topology†, therefore involving an idea of "topological invariance" under some form of homotopy, as will be explained in next section. Graphs and partial orders do include some form of geometric reasoning indeed, but these are now standard techniques in computer science, to which quite a few conferences and journals are devoted, whereas we will be more interested in "higher-dimensional" phenomena here.

These techniques have cropped up in different areas of computer science for over thirty

† Work partly done when the author was at Ecole Normale Supérieure, Paris.
† More or less abstract, from simple topological notions like connectedness (Spanier, 1966), simplicial complexes (May, 1967; Gabriel and Zisman, 1967) etc. to abstract homotopy categories (Baues, 1989).

years now. We have somewhat artificially subdivided them into three main strands. One is concerned with giving semantics to concurrent machines and languages, for formalizing and analyzing their properties. Another one has been mostly concerned with distributed databases and the scheduling of transactions (with a view to their correctness). Finally, the most recent one is in the field of fault-tolerant protocols for distributed systems, where the main application is the determination of the computability (and then complexity) of some "functions" on given distributed architectures. After a brief history in Section 2, we review the main contributions to date in all three directions. In Sections 3, 4 and 5, we present the articles of this issue and how they naturally fall into one of these pre-existing strands. Finally in Section 6 we give a few directions for future work, in particular some areas of research for which these techniques seem also to have appeared or might be useful.

2. A brief history

The first "algebraic topological" model I am aware of is called *progress graph* and has appeared in operating systems theory, in particular for describing the problem of "deadly embrace"[‡] in "multiprogramming systems". *Progress graphs* are introduced in (Coffman et al., 1971), but attributed there to E. W. Dijkstra. In fact they also appeared slightly earlier (for editorial reasons it seems) in (Shoshani and Coffman, 1970).

The basic idea is to give a description of what can happen when several processes are modifying shared ressources. Given a shared resource a, we see it as its associated semaphore that rules its behaviour with respect to processes. For instance, if a is an ordinary shared variable, it is customary to use its semaphore to ensure that only one process at a time can write on it (this is mutual exclusion). Then, given n deterministic sequential processes $Q_1, \ldots, Q_n$, abstracted as a sequence of locks and unlocks on shared objects, $Q_i = R^1 a_i^1.R^2 a_i^2 \cdots R^{n_i} a_i^{n_i}$ (R^k being P or V[§]), there is a natural way to understand the possible behaviours of their concurrent execution, by associating to each process a coordinate line in $\mathbb{R}^n$. The state of the system corresponds to a point in $\mathbb{R}^n$, whose ith coordinate describes the state (or "local time") of the ith processor.

Consider a system with finitely many processes running altogether. We assume that each process starts at (local time) 0 and finishes at (local time) 1; the P and V actions correspond to sequences of real numbers between 0 and 1, which reflect the order of the P's and V's. The initial state is $(0, \ldots, 0)$ and the final state is $(1, \ldots, 1)$. An example consisting of the two processes $T_1 = Pa.Pb.Vb.Va$ and $T_2 = Pb.Pa.Va.Vb$ gives rise to the two dimensional *progress graph* of Figure 1.

The shaded area represents states which are not allowed in any execution path, since they correspond to mutual exclusion. Such states constitute the *forbidden area*. An *execution path* is a path from the initial state $(0, \ldots, 0)$ to the final state $(1, \ldots, 1)$ avoiding the forbidden area and increasing in each coordinate - time cannot run backwards. We

[‡] as E. W. Dijkstra originally put it in (Dijkstra, 1968), now more usually called deadlock.

[§] Using E. W. Dijkstra's notation P and V (Dijkstra, 1968) for respectively acquiring and releasing a lock on a semaphore.

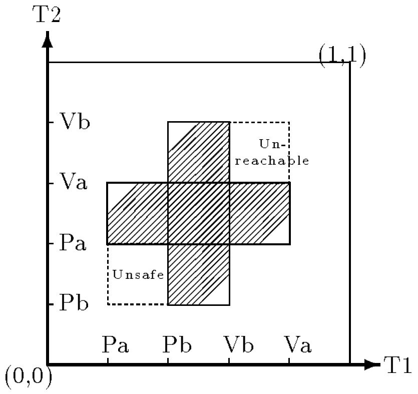

Fig. 1. Example of a progress graph

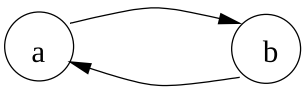

Fig. 2. The corresponding request graph

call these paths *directed paths* or dipaths. This entails that paths reaching the states in the dashed square underneath the forbidden region, marked "unsafe" are deemed to deadlock, i.e. they cannot possibly reach the allowed terminal state which is $(1,1)$ here. Similarly, by reversing the direction of time, the states in the square above the forbidden region, marked "unreachable", cannot be reached from the initial state, which is $(0,0)$ here. Also notice that all terminating paths above the forbidden region are "equivalent" in some sense, given that they are all characterized by the fact that T_2 gets a and b before T_1 (as far as resources are concerned, we call this a *schedule*). Similarly, all paths below the forbidden region are characterized by the fact that T_1 gets a and b before T_2 does.

On this picture, one can already recognize many ingredients that are at the center of the main problem of algebraic topology, namely the classification of shapes modulo "elastic deformation". As a matter of fact, the actual coordinates that are chosen for representing the times at which Ps and Vs occur are unimportant, and these can be "stretched" in any manner, so the properties (deadlocks, schedules etc.) are invariant under some notion of deformation, or *homotopy*. This is a particular kind of homotopy though, and this will be at the center of many difficulties in later work. We call it (in subsequent work) *directed homotopy* or *dihomotopy* in the sense that it should preserve the direction of time. For instance, the two homotopic shapes, all of which have two holes, of Figure 3 and Figure 4 have a different number of dihomotopy classes of dipaths. In Figure 3 there are essentially four dipaths up to dihomotopy (i.e. four schedules corresponding to all possibilities of accesses of resources a and b) whereas in Figure 4, there are essentially three dipaths up to dihomotopy.

There is another method to determine deadlocks in such situations, which was of course known long ago and was entirely graph-theoretic, known as the *request graph*. Figure 2 depicts the request graph corresponding to the progress graph of Figure 1. Nodes of this graph are resources of the concurrent system, i.e. here, semaphores. There is an oriented edge from a resource x to a resource y if there is a process which has locked x and needs to lock y at a given time. A sufficient condition for such systems to be deadlock-

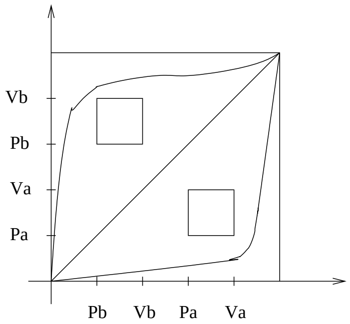

Fig. 4. The progress graph corresponding to $Pb.Vb.Pa.Va \mid Pa.Va.Pb.Vb$

free is that their corresponding request graphs be acyclic¶. Unfortunately, this is not a necessary condition in general. For instance a request graph cannot capture the notion of n-semaphores, i.e. resources that can be shared by up to n processes but not $n+1$ (for instance, asynchronous buffers of communication of size n which can be "written" on by at most n processes). This in fact really calls for some higher-dimensional versions of graphs.

Starting from progress graphs, the article (Coffman et al., 1971) developed an algorithm in $O(n^2)$ (n is the number of tasks) to determine freedom from deadlocks. The notion of unsafe state was also introduced with the hope to determine automatically the right schedulers that would prevent the whole system from running into a deadlock situation. This was limited to binary semaphores only though‖. A fully worked out deadlock detection algorithm on progress graphs (including the determination of the unsafe region) is described in (Carson and Reynolds Jr, 1987) which takes care of this limitation. This was unfortunately not an optimal algorithm.

3. Semantics and Analysis of Concurrency

The semantics community came back to these geometric considerations with the development of "truly-concurrent" semantics, as opposed to "interleaving" semantics. The base of the argument was that interleaving semantics, i.e. the representation of parallelism by non-determinism ignores real asynchronous behaviours that actually exist††: $a \mid b$ where a and b are atomic is represented by the same transition system as the non-deterministic choice a then b or b then a (see Figure 5).

This fact creates problems in particular in static analysis of (asynchronous) concurrent systems in that interleaving builds a lot of uninteresting states in the modelisation,

¶ Note that this is a very geometric condition indeed.

‖ There is a way to translate general semaphores into binary semaphores, see (Dijkstra, 1968), but this uses an encoding with integers which cannot be represented in progress graphs.

†† For instance a distributed system which does not have a global clock is such a system.

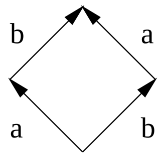

Fig. 5. Interleaving semantics of $a \mid b$

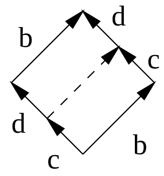

Fig. 6. A refinement of $a \mid b$

hence inducing a high cost in verification. This is called the *state-space explosion problem*. Some techniques exist now to circumvent part of the problem, actually mostly derived from truly-concurrent considerations (originally Petri nets in (Valmari, 1990) and Mazurkiewicz trace theory in (Godefroid and Wolper, 1991)). Another problem is that *refinement*, which is a very convenient technique in program analysis, is very difficult to apply in interleaving semantics, see (van Glabbeek and Goltz, 1989) for instance: suppose that action a is in fact non-atomic and is composed of two subactions c then d, then $a \mid b$ in interleaving semantics is no longer equivalent to a then b or b then a. The path c then b then d is missing there (see Figure 6, the missing part is represented by the dashed line). This implies that interleaving semantics is bound to describe all atomic actions for the derived analyses to be correct, hence incurring a great complexity.

Quite a few models for true-concurrency have appeared (see in particular the account of (Winskel and Nielsen, 1994)) but it is only in 1991 that geometry is proposed to solve the problem, in (Pratt, 1991). The diagnosis is that interleaving is only the boundary of the real picture. $a \mid b$ is really the filled-in square whose boundary is the non-deterministic choice a then b or b then a (the hollow square). The natural combinatorial notion, extension of transition systems, is that of a *cubical set*, which is a collection of points (states), edges (transitions), squares, cubes and hypercubes (higher-dimensional transitions representing the truly-concurrent execution of some number of actions). This is introduced in (Pratt, 1991) as well as possible formalizations using n-categories, and a notion of homotopy[‡‡]. This is actually a combinatorial view of some kind of progress graph. Look back to Figure 1. Consider all interleavings of actions Pa, Pb, Va and Vb: they form a subgrid of the progress graph. Take as 2-transitions (i.e. squares in the cubical set we are building) the filled-in squares. Only the forbidden region is really interleaved. Cubical sets generalize progress graphs, in that they allow any amount of non-deterministic choices as well as dynamic creation of processes. These cubical sets are called *Higher-Dimensional Automata* (HDA) following (Pratt, 1991) because it really makes sense to consider a hypercube as some form of transition. Actually at about the same time, a bisimulation semantics was given in (van Glabbeek, 1991). Notice that 2-transitions or squares are nothing but a *local commutation relation* as in Mazurkiewicz trace theory (), *independence relation* as in asynchronous transition systems, see (Bednarczyk, 1988) and (Shields, 1985), as in trace automata (as used in e.g. (Kahn, 1974; Kahn and MacQueen, 1977)), as in transition systems with independence (Sassone et al., 1994), or (indirectly) as with the "confluence" relation of concurrent transition systems (Stark, 1989). There

‡‡ Which appeared later, see (Fajstrup et al., 1999), not to be completely adequate.

are two more ingredients with HDA: the elegance and the power of the tools of geometric formalisations, and the natural generalisation to higher-dimensions (i.e. "higher-order independence relation" or n-ary independence relations).

The later semantic papers on the subject were very much influenced by (Pratt, 1991). In 1992, homological methods (see (Mac Lane, 1963) for a start) for studying the properties of Higher-Dimensional Automata were advocated in (Goubault and Jensen, 1992) given that they provide computable invariants of homotopy (at least in the usual case). A semantics of CCS was also discussed in a suitable category of HDA (also studied in (Lanzmann, 1993)) together with a notion of bisimulation[§§]. This homology theory is not the definitive one, in particular it cannot distinguish some situations which are quite simple. Some further work was needed and began in (Gaucher, 1997a) and (Gaucher, 1997b). In "Homotopy invariants of higher-dimensional categories and concurrency in computer science (I) and (II)" in this issue, P. Gaucher extends his work in using strict globular ω-categories (generalizing (Pratt, 1991)) to formalize the execution paths of a parallel automaton. He also defines three new homology theories which are more adequate than the one proposed in (Goubault, 1995a). The first properties of these theories are proven, in particular Hurewicz morphisms, are constructed, relating homotopy and homology. It is to be noted that at the very time this was written up, R. Brown from Bangor University and R. Steiner from Glasgow University made an important contribution very much related to this, in showing that the category of ω-categories is equivalent to the category of cubical sets with connections and compositions (see (Brown and Higgins, 1981a) and (Brown and Higgins, 1981b) for an introduction).

More general languages were modeled in (Goubault, 1993) with HDAs. Then a few analyses of programs by abstract interpretation (see (Cousot and Cousot, 1977) for a start) were designed: some earlier steps in (Cridlig and Goubault, 1993), then the automatic determination of a superset of possible schedules from the geometry of executions (i.e. histories of resource usage) in (Goubault, 1995a) and (Goubault, 1995b), some applications to model-checking in 1995 in (Cridlig, 1995) (for shared-memory programming languages), and in (Cridlig, 1996) (on CML, i.e. message-passing languages). A prototype Parallel Pascal Analyser has been implemented along these lines by R. Cridlig, see `http://www.dmi.ens.fr/~cridlig`.

In "On the classification of dipaths in geometric models of concurrency" in this issue, M. Raussen focusses on the determination of the dihomotopy classes in cubical complexes, deadlock detection and serializability (see next section) being only special cases. An algorithm is given for the 2-dimensional case and some ideas are given for solving the general case. In "Loops, Ditopology and Deadlocks" in this issue, L. Fajstrup gives a geometric model of a truly-concurrent PV system of n processes with finitely many (nested) loops and proves that it suffices to study a finite number of deloopings of each loop to determine the unsafe region and the deadlocks. Hence the previous algorithm on deadlock detection of (Fajstrup et al., 1998) can be applied to these finitely many deloopings and give the precise unsafe area in concurrent systems with loops.

[§§] Also there is a way to fully compute the branchings, mergings and deadlocks inductively on this language.

Some proposals have been made to use the scheduling information obtained from the HDA semantics to derive automatic parallelization algorithms. This has been fully treated for CCS in (Takayama, 1995), (Takayama, 1996).

Most of the models used since V. Pratt's article were based on some form of cubical set. Some more recent work, in (Fajstrup and Raussen, 1996) and (Fajstrup et al., 1998) in particular, introduced topological models, the *local po-space* models, generalizing the previous models, for being able to reason in a similar way as in ordinary algebraic topology, i.e. by reasoning directly on "continuous" shapes and not through their combinatorial representations (simplicial sets in general). Of course, as in the standard case, there is, as shown in (Fajstrup et al., 1999) again, a natural relationship between combinatorial and topological representations through a pair of adjoint functors, geometric realization and singular cube functors. The expected applications of this modelization, as well as some of the basic notions about "directed homotopy" and schedules are described in (Fajstrup et al., 1999). More recently, V. Pratt proposed *Chu Spaces* (see for instance (Pratt, 1999)) as a model for concurrency, starting back to the failure identified in (Pratt, 1991) of the natural duality event/schedule, in interleaving semantics. V. Pratt in "Higher-Dimensional Automata Revisited" in this issue, develops this idea and expresses the HDA geometric model in terms of Chu spaces.

Some potentially related semantic models are the n-categorical formulations of (Buckland and Johnson, 1996) and some other recent combinatorial or categorical formulations in (Fiore et al., 1997), (Sassone and Cattani, 1996), (Sokolowski, 1998a), (Sokolowski, 1998b), (Sokolowski, 1998c).

4. Correctness of Distributed Databases

Another strand of research is concerned with distributed databases and in fact, this is very much related to the "multiprogramming" and "semaphore" strand described before. But the kind of properties which have been studied are slightly different, and the focus has been put on devising optimal algorithms for the analysis of simple transaction models. As a matter of fact, a distributed database can be seen as a shared-memory machine (containing items) on which processes (called transactions) act by reading and writing, getting permissions to do so by using the appropriate functions on attached semaphores. One of the main purposes of this area is to ensure coherence of the distributed database while ensuring good performance, through a definition of suitable policies (protocols) for transactions to perform their own actions (with P and V). This entails that deadlock-freedom of transactions is of importance. Correctness of a distributed database is itself very often expressed by some form of a *serializability* condition. Look for instance at Figure 4. This could describe a database with two transactions $T_1 = Pb.Vb.Pa.Va$ and $T_2 = Pa.Va.Pb.Vb$ trying to modify two items a and b. All paths of execution above the left hole are equivalent to a serial execution of transaction T_2 then transaction T_1. All paths of execution below the right hole are equivalent to the serial execution of transaction T_1 then T_2. The third type of dipath is not a serial dipath: it describes several equivalent cases, for instance: T_1 acquires b, T_2 acquires a, then T_1 acquires a and T_2 acquires b. Think of the database to represent airplane tickets (for instance b is the return ticket corresponding to the one-way ticket a), and the two transactions to represent remote booking booths, the action between a P and its corresponding V is writing a name on the ticket. The situation here is that T_1 will have reserved its one-way ticket and T_2 will have reserved its return ticket only. This is not an allowed behaviour. It is not equivalent to a purely serial schedule which are the only ones that are specified as correct (only one of T_1 or T_2 gets the whole lot of tickets).

Testing serializability is unfortunately known to be a NP-complete problem (in (Papadimitriou, 1979)), even when the model is only based on simple binary semaphores.

The progress graph approach to the study of distributed databases was really initiated in (Yannakakis et al., 1979) and then in (Papadimitriou, 1983). In (Lipski and Papadimitriou, 1981) an algorithm for proving the safety through serializability of distributed databases with only two transactions expressed as progress graphs was described. The underlying algorithmics is relying on proving the connectedness of the closure of the set of forbidden rectangles[¶¶]. Of course the real problem in our previous example was that the forbidden region was disconnected, allowing dipaths to interleave some of the requests of different transactions. Another algorithm, for proving freedom from deadlock for two transactions synchronizing with binary semaphores only was also described. It was shown to have $O(n \log n \log\log n)$ (where n is the number of forbidden rectangles) time complexity. A notion of "directed homotopy" was also defined. The generalization of safety conditions to higher-dimensions through the method of (Yannakakis et al., 1979) reducing to 2-dimensional problems, was shown to be in $O(nd2^d + d^2 \log n \log\log n)$ time complexity (d is the dimension, i.e. the number of transactions). Much work has been done in algorithmics of these geometric problems and the algorithm above for safety is improved (actually it is optimal then) in (Soisalon-Soininen and Wood, 1985) achieving $O(n \log n)$ time and $O(n)$ space complexity for 2 transactions, then relying on M. Yannakakis result for the extension to any dimension. This is not the best that can be done in higher dimensions: the next step is achieved in (Fajstrup et al., 1998) where a direct method for unsafe regions is described and where it is shown that the closure of the forbidden region in higher-dimension is not a strong enough condition for serializability in general. An application to proving the 2-phase locked protocol is given in (Gunawardena, 1994), and, using dihomotopy, in (Fajstrup et al., 1999).

5. Computability and Complexity of Fault-Tolerant Distributed Protocols

The strand of work here is concerned with the robust or fault-tolerant implementation of distributed programs. More precisely, the interest here is in *wait-free* or *t-resilient* ($0 \leq t \leq n-1$ where n is the number of processors involved, wait-free being $n-1$-resilient) implementations on a distributed machine composed of several units communicating through a shared memory via atomic read/write or FIFO queues etc. or through synchronous/asynchronous message passing. This means that the processes executed on the n processors must be as loosely coupled as possible so that even if t processors fail

[¶¶] Also cited in chapter "The geometry of rectangles" in (Preparata and Shamos, 1993).

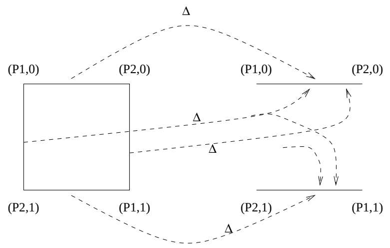

Fig. 7. The binary consensus decision map from initial global states to final global states

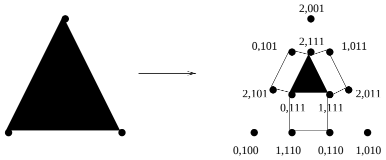

Fig. 8. The synchronous protocol complex (for 3 processes) after one round

to terminate, the others will carry on their computation and find a correct partial result (as observed on the non-faulty ones).

Consider as an example a machine with two processors, P_1 and P_2, communicating by atomic reads and writes on a shared memory. Each of these processes has a local binary variable x_1 (for P_1) and x_2 (for P_2) respectively. The binary *consensus* problem is to design an algorithm for having P_1 and P_2 agree on one value, which, moreover, has to be a value that one of them started with. Therefore if $x_1 = 0$ and $x_2 = 1$ at the beginning, we want them to be $x_1 = 0$ and $x_2 = 0$ or $x_1 = 1$ and $x_2 = 1$ at the end. Of course this is complicated by the fact that we ask the solution to be wait-free. If one of the two processes fails (the other process being unable to know whether this process is dead or just very slow) then the non-faulty one must terminate with one of the values P_1 or P_2 started with originally.

It is unfortunately proved in (Fisher et al., 1985) that this is not possible – in fact it was proven for the equivalent case of an asynchronous message-passing system, in the general 1-resilient case. The article used an argument from graph theory (but already quite geometric in nature). It has in fact originated the geometric point of view on this field of research as we are going to explain[‖‖].

The main argument can be understood in broad terms ("similarity chain") as a connectedness result. If we represent the local states of each processor P_i $(i = 1, 2)$, at some point of the execution of a program, by a vertex (P_i, x_i) and the global states of the system by an (un-oriented) edge linking the two local states which compose them, then it is easy to see that the semantics of atomic reads and writes imposes that the reachable global states starting from one initial global state have to form a connected graph. There is no way a program on such a machine can "break" connectivity. This implies that the binary consensus cannot be implemented here since we must be able to reach from the global state $((P_1, 0), (P_2, 1))$ two disconnected global states $((P_1, 0), (P_2, 0))$ and $((P_1, 1), (P_2, 1))$ as shown on Figure 7.

Further work generalizes this simple geometric argument to the needed higher-dimensional cases. In (Biran et al., 1988) a characterization of a class of problems solvable in

‖‖ We refer the reader to the book (Lynch, 1996) to have a flavour of the vast amount of results that have been proven in the field of fault-tolerant distributed protocols.

asynchronous message-passing systems in the presence of a single failure was given. No generalization to more failures has since been solved using the same kinds of graph techniques. It was then a rather shared belief that one would have to use more powerful techniques in that case. The conjecture (Chaudhuri, 1990) that the k-set agreement problem (a kind of weak consensus; all is required is that the non-faulty processes eventually agree on a subset of at most k values taken from the input values) cannot be solved in certain asynchronous systems was finally proven in three different papers independently, (Borowsky and Gafni, 1993), (Saks and Zaharoglou, 1993) and (Herlihy and Shavit, 1993).

The basic idea of (Herlihy and Shavit, 1993) is to generalize such pictures as the one of Figure 7 using *simplicial sets* instead of graphs (which are special cases of the former). Simplicial sets are made up of vertices, edges but also triangles, simplexes in general glued altogether. A simplex of dimension n represents the global state, at some point of the execution, of n processes. Vertices are still pairs composed of the name of the process, together with its local state. Again, given an initial global state, the semantics of the operations of the distributed machine we want to study is defined by the reachable global states, at any time. For instance, Figure 8 shows the simplicial set, called protocol complex, after one round of communication on a synchronous message-passing machine, which broadcasts the local states to all processes at each step. If there is a simplicial map from it to some suitable set of global states, respecting the specification of the problem, then there exists a corresponding wait-free protocol. This enables us also to compute how many rounds of communication might be necessary to solve a given problem. Notice that the simplexes of the protocol complex are really the schedules, and this should be related to the directed homotopy approach of Section 3. This has been hinted in (Goubault, 1996a), (Goubault, 1996b), (Goubault, 1997) for simple cases only.

It was also proved in (Herlihy and Shavit, 1993) that in a shared-memory model with single reader/single writer registers providing atomic read and write operations, k-set agreement requires at least $\lfloor f/k \rfloor + 1$ rounds where f is the number of processes that can fail.

The renaming task (processes must try to agree on a smaller set of names between each other than the original set of names), first proposed in (Attiya et al., 1990) was also finally solved in (Herlihy and Shavit, 1993). There is a wait-free protocol for the renaming task in certain asynchronous systems if the output name space is sufficiently large. It was already known that there is a wait-free solution for the renaming task for $2n+1$ or more output names on a system of $n+1$ asynchronous processors and none for $n+2$ or fewer output names. M. Herlihy and N. Shavit refined this result and showed that there was no solution for strictly less that $2n+1$ output names.

Not only impossibility results can be given but also constructive means for finding algorithms follow from this work (see for instance (Herlihy and Shavit, 1994)).

More generally, datatypes do matter for computability results. It is known for quite a long time that consensus for two processes can be solved with shared memory with atomic reads and writes plus a FIFO queue, or plus an atomic test&set operation. If we define the consensus number of a data type as the maximal number of asynchronous processors (having atomic read and write) on which it can implement wait-free consensus, then,

— atomic read/write registers have consensus number 1,
— test&set and fetch&add registers, queues, and stacks have consensus number 2,
— n-register assignment has consensus number $2n-2$,
— load-locked, store-conditional and compare-and-swap registers have consensus number ∞.

These facts motivated the introduction of the following general problem, dealing with the power of the architecture of distributed machines. We say that a datatype, or object, is an (m, j)-consensus object if it allows any set of m processes to solve j-set agreement tasks. Herlihy and Rajsbaum in (Herlihy and Rajsbaum, 1994) (see also (Borowsky and Gafni, 1993)) proved that is is impossible to implement $(n+1, k)$-consensus using (m, j)-consensus objects if $n/k > m/j$.

Further work on this can be found in (Jayanti, 1993), (Jayanti, 1997) and (Schenk, 1997). In particular, the problem identified in (Jayanti, 1997) is that the hierarchy of data objects as briefly sketched above is not robust, in the sense that it is possible to implement some datatypes with consensus number k using several datatypes with consensus numbers strictly less than k.

This of course is due to some subtle interactions between the use of these data objects and we could hope that the more general study of the schedules with such datatypes could lead to some better classifications.

In "Algebraic Spans" in this issue, M. Herlihy and S. Rajsbaum introduce a new tool (related to the one described in (Herlihy and Rajsbaum, 1995) for proving impossibility results, based on a core theorem of algebraic topology, the acyclic carrier theorem, which unifies, generalizes and extends earlier results.

6. Some perspectives

There are numerous perspectives in static analysis of concurrent programs, computability and complexity issues in fault-tolerant distributed systems as well as in concurrent database theory, as I have been trying to explain in the previous sections. The aim here is not to list the possible research that could be carried on in these directions (good references for this are (Fajstrup et al., 1999) and (Herlihy and Rajsbaum, 1999)), but to look at other possible use of these techniques. For instance, Squier's theorem in rewriting systems theory, which gives a necessary condition for the existence of a presentation of a given monoid by a finite canonical rewriting system in terms of its homology (must be of finite dimension), seems very much related to the techniques presented above. It is definitely a computability result, as we have in fault-tolerant distributed systems theory, but for something which looks sequential (rewriting). As hinted in (Goubault, 1995a), this can be understood as a problem of concurrency theory in that the study of the confluence of rewriting systems is related to parallel reduction techniques (as in (Lévy, 1978) for instance). The resolutions used in most of the proofs of this theorem, like in (Kobayashi, 1990), (Groves, 1991), (Anick, 1986), (Farkas, 1992) and (Lafont and Prouté, 1990) are very much like a Knuth-Bendix completion procedure, where higher-dimensional objects are filling in possible defects of local confluence. This looks like building higher-dimensional transitions implementing the parallel (confluent) reductions (see in particular (Groves, 1991) where the resolution is a cubical complex and in dimension one it is generated by the transition system coming from the reduction relation). Some other proof techniques use something which is very much like some kind of directed homotopy, as in (Squier et al., 1994) for instance. Other interesting relations should be studied concerning "higher-dimensional" word problems, as in (Burroni, 1991). A hope is that geometry can also give some insight in logics, especially modal logics as in (Goubault-Larrecq and Goubault, 1999). Finally, there is some intuition from theoretical physics that seems relevant to semantics, in particular concerning time and dynamical systems. Some of the concepts of M. Raussen's and L. Fasjtrup's articles in this issue are based on similar notions as in (Penrose, 1972): some areas of physics (not classical mechanics though) have to consider time as non-reversible, hence have to construct some kind of directed topology.

Acknowledgments

I feel very much indebted to Giuseppe Longo who gave me the opportunity to act as a guest editor of this special issue, and who very kindly helped me through the many different steps that editing necessitates. My very warm thanks to the authors who did a very nice job, in a relatively short time. Finally, I wish to thank the referees of this issue, who sometimes found "bugs" that I would not have found, and who suggested very useful improvements to the authors. The referees were, Nir Shavit, Stefan Sokolowski, Christophe Tabacznyj, Jean Goubault-Larrecq, Arnaud Venet, Ronnie Brown and Tim Porter as well as some of the authors who cross-checked other authors' articles, Philippe Gaucher, Martin Raussen and Lisbeth Fajstrup.

References

Anick, D. J. (1986). On the homology of associative algebras. *Transactions of the American Mathematical Society*, 296:641–659.

Attiya, H., Bar-Noy, A., Dolev, D., Peleg, D., and Reischuk, R. (1990). Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524–548.

Baues, H. J. (1989). Algebraic homotopy. In *Cambridge Studies in Advanced Mathematics*, volume 15. Cambridge University Press.

Bednarczyk, M. A. (1988). *Categories of asynchronous systems.* PhD thesis, University of Sussex.

Biran, O., Moran, S., and Zaks, S. (1988). A combinatorial characterization of the distributed tasks which are solvable in the presence of one faulty processor. In *Proc. 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 263–275. ACM Press.

Borowsky, E. and Gafni, E. (1993). Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proc. of the 25th STOC*. ACM Press.

Brown, R. and Higgins, P. J. (1981a). Colimit theorems for relative homotopy groups. *Journal of Pure and Applied Algebra*, (22):11–41.

Brown, R. and Higgins, P. J. (1981b). On the algebra of cubes. *Journal of Pure and Applied Algebra*, (21):233–260.

Buckland, R. and Johnson, M. (1996). ECHIDNA: A system for manipulating explicit choice higher dimensional automata. In *AMAST'96: Fifth Int. Conf. on Algebraic Methodology and Software Technology*, Munich.

Burroni, A. (1991). Higher dimensional word problem. *Lecture notes in computer science*, (530).

Carson, S. D. and Reynolds Jr, P. F. (1987). The geometry of semaphore programs. *ACM Transactions on Programming Languages and Systems*, 9(1):25–53.

Chaudhuri, S. (1990). Agreement is harder than consensus: set consensus problems in totally asynchronous systems. In *Proc. of the 9th Annual ACM Symposium on Principles of Distributed Computing*, pages 311–334. ACM Press.

Coffman, E. G., Elphick, M. J., and Shoshani, A. (1971). System deadlocks. *Computing Surveys*, 3(2):67–78.

Cousot, P. and Cousot, R. (1977). Abstract interpretation: A unified lattice model for static analysis of programs by construction of approximations of fixed points. *Principles of Programming Languages 4*, pages 238–252.

Cridlig, R. (1995). Semantic analysis of shared-memory concurrent languages using abstract model-checking. In *Proc. of PEPM'95*, La Jolla. ACM Press.

Cridlig, R. (1996). Semantic analysis of Concurrent ML by abstract model-checking. In *Proceedings of the LOMAPS Workshop.*

Cridlig, R. and Goubault, E. (1993). Semantics and analyses of Linda-based languages. In *Proc. of WSA'93*, number 724 in LNCS. Springer-Verlag.

Dijkstra, E. (1968). *Cooperating Sequential Processes.* Academic Press.

Fajstrup, L., Goubault, E., and Raussen, M. (1998). Detecting deadlocks in concurrent systems. In *Proceedings of the 9th International Conference on Concurrency Theory, also available at http://www.dmi.ens.fr/˜goubault.* Springer-Verlag.

Fajstrup, L., Goubault, E., and Raussen, M. (1999). Algebraic topology and concurrency. *submitted to Theoretical Computer Science, also technical report, Aalborg University.*

Fajstrup, L. and Raussen, M. (1996). Detecting deadlocks in concurrent systems. Technical report, BRICS Research Report, Aalborg University.

Farkas, D. R. (1992). The Anick resolution. *Journal of Pure and Applied Algebra*, 79.

Fiore, M., Plotkin, G., and Power, J. (1997). Complete cuboidal sets in axiomatic domain theory (extended abstract). In *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science*, pages 268–279, Warsaw, Poland. IEEE Computer Society Press.

Fisher, M., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed commit with one faulty process. *Journal of the ACM*, 32(2):374–382.

Gabriel, P. and Zisman, M. (1967). Calculus of fractions and homotopy theory. In *Ergebnisse der Mathematik und ihrer Grenzgebiete*, volume 35. Springer Verlag.

Gaucher, P. (1997a). Connexion de flux d'information en algèbre homologique. Technical report, IRMA, Strasbourg, available at http://irmasrv1.u-strasbg.fr/˜gaucher/activite.html.

Gaucher, P. (1997b). Etude homologique des chemins de dimension 1 d'un automate. Technical report, IRMA, Strasbourg, available at http://irmasrv1.u-strasbg.fr/˜gaucher/activite. html.

Godefroid, P. and Wolper, P. (1991). Using partial orders for the efficient verification of deadlock freedom and safety properties. In *Proc. of the Third Workshop on Computer Aided Verification*, volume 575, pages 417–428. Springer-Verlag, Lecture Notes in Computer Science.

Goubault, E. (1993). Domains of higher-dimensional automata. In *Proc. of CONCUR'93*, Hildesheim. Springer-Verlag.

Goubault, E. (1995a). *The Geometry of Concurrency.* PhD thesis, Ecole Normale Supérieure. also available at http://www.dmi.ens.fr/˜goubault.

Goubault, E. (1995b). Schedulers as abstract interpretations of HDA. In *Proc. of PEPM'95*, La Jolla. ACM Press, also available at http://www.dmi.ens.fr/~goubault.

Goubault, E. (1996a). The dynamics of wait-free distributed computations. Technical report, Research Report LIENS-96-26.

Goubault, E. (1996b). A semantic view on distributed computability and complexity. In *Proceedings of the 3rd Theory and Formal Methods Section Workshop*. Imperial College Press, also available at http://www.dmi.ens.fr/~goubault.

Goubault, E. (1997). Optimal implementation of wait-free binary relations. In *Proceedings of the 22nd CAAP*. Springer Verlag.

Goubault, E. and Jensen, T. P. (1992). Homology of higher-dimensional automata. In *Proc. of CONCUR'92*, Stonybrook, New York. Springer-Verlag.

Goubault-Larrecq, J. and Goubault, E. (1999). Order-theoretic, geometric and combinatorial models of intuitionistic s4 proofs. In *proceedings of IMLA'99*.

Groves, J. R. J. (1991). Rewriting systems and homology of groups. In Kovacs, L. G., editor, *Groups – Canberra 1989*, number 1456, pages 114–141. Lecture notes in Mathematics, Springer-Verlag.

Gunawardena, J. (1994). Homotopy and concurrency. In *Bulletin of the EATCS*, number 54, pages 184–193.

Herlihy, M. and Rajsbaum, S. (1994). Set consensus using arbitrary objects. In *Proc. of the 13th Annual ACM Symposium on Principles of Distributed Computing*. ACM Press.

Herlihy, M. and Rajsbaum, S. (1995). Algebraic spans (preliminary version). In *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 90–99, Ottawa, Ontario, Canada.

Herlihy, M. and Rajsbaum, S. (1999). New perspectives in distributed computing. In Kutylowski, M., Pacholski, L., and Wierzbicki, T., editors, *24th International Symposium on Mathematical Foundations of Computer Science*, volume LNCS 1672, pages 170–186. Springer-Verlag.

Herlihy, M. and Shavit, N. (1993). The asynchronous computability theorem for t-resilient tasks. In *Proc. of the 25th STOC*. ACM Press.

Herlihy, M. and Shavit, N. (1994). A simple constructive computability theorem for wait-free computation. In *Proceedings of STOC'94*. ACM Press.

Jayanti, P. (1993). On the robustness of Herlihy's hierarchy. In *Proceedings of the Twelth Annual ACM Symposium on Principles of Distributed Computing*, pages 145–157, Ithaca, New York, USA.

Jayanti, P. (1997). Robust wait-free hierarchies. *Journal of the ACM*, 44(4):592–614.

Kahn, G. (1974). The semantics of a simple language for parallel programming. *Information Processing*, (74).

Kahn, G. and MacQueen, D. B. (1977). Coroutines and networks of parallel processes. *Information Processing*, (77).

Kobayashi, Y. (1990). Complete rewriting systems and homology of monoid algebras. *Journal of Pure and Applied Algebra*, 65:263–275.

Lafont, Y. and Prouté, A. (1990). Church-Rosser property and homology of monoids. Technical report, Ecole Normale Supérieure.

Lanzmann, E. (1993). Automates d'ordre supérieur. Master's thesis, Université d'Orsay.

Lévy, J.-J. (1978). *Réductions Correctes et Optimales dans le Lambda-Calcul*. PhD thesis, Université Paris VII.

Lipski and Papadimitriou (1981). A fast algorithm for testing for safety and detecting deadlocks in locked transaction. *ALGORITHMS: Journal of Algorithms*.

Lynch, N. (1996). *Distributed Algorithms*. Morgan-Kaufmann.

Mac Lane, S. (1963). Homology. In *Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen*, volume 114. Springer Verlag.

May, J. P. (1967). *Simplicial objects in algebraic topology.* D. van Nostrand Company, inc.

Papadimitriou, C. H. (1979). The serializability of concurrent database updates. *Journal of the ACM*, 26(4):631–653.

Papadimitriou, C. H. (1983). Concurrency control by locking. *SIAM Journal on Computing*, 12(2):215–226.

Penrose, R. (1972). *Techniques of Differential Topology in Relativity*, volume 7 of *Conference Board of the Mathematical Sciences, Regional Conference Series in Applied Ma thematics.* SIAM, Philadelphia, USA.

Pratt, V. (1991). Modeling concurrency with geometry. In *Proc. of the 18th ACM Symposium on Principles of Programming Languages.* ACM Press.

Pratt, V. (1999). Chu spaces. In *Course notes for the School in Category Theory and Applications.* Coimbra, Portugal.

Preparata, F. P. and Shamos, M. I. (1993). *Computational Geometry, an Introduction.* Springer-Verlag.

Saks, M. and Zaharoglou, F. (1993). Wait-free k-set agreement is impossible: The topology of public knowledge. In *Proc. of the 25th STOC.* ACM Press.

Sassone, V. and Cattani, G. L. (1996). Higher-dimensional transition systems. In *Proceedings of LICS'96.*

Sassone, V., Nielsen, M., and Winskel, G. (1994). Relationships between models of concurrency. In *Proceedings of the Rex'93 school and symposium.*

Schenk, E. (1997). The consensus hierarchy is not robust. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, page 279, Santa Barbara, California.

Shields, M. (1985). Concurrent machines. *Computer Journal*, 28.

Shoshani, A. and Coffman, E. G. (1970). Sequencing tasks in multiprocess systems to avoid deadlocks. In *Conference Record of 1970 Eleventh Annual Symposium on Switching and Automata Theory*, pages 225–235, Santa Monica, California. IEEE.

Soisalon-Soininen, E. and Wood, D. (1985). An optimal algorithm for testing for safety and detecting deadlocks in locked transaction systems. In *Symposium on Principles of Database Systems (PODS '82)*, pages 108–116.

Sokolowski, S. (1998a). Homotopy in concurrent processes. Technical report, Institute of Computer Science, Gdansk Division.

Sokolowski, S. (1998b). Investigation of concurrent processes by means of homotopy functors. Technical report, Institute of Computer Science, Gdansk Division.

Sokolowski, S. (1998c). Point glueing in cpo-s. Technical report, Institute of Computer Science, Gdansk Division.

Spanier, E. J. (1966). *Algebraic Topology.* McGraw Hill.

Squier, C. C., Otto, F., and Kobayashi, Y. (1994). A finiteness condition for rewriting systems. *Theoretical Computer Science*, 131:271–294.

Stark, A. (1989). Concurrent transition systems. *Theoretical Computer Science*, 64:221–269.

Takayama, Y. (1995). Cycle filling as parallelization with expansion law. In *submitted to publication.*

Takayama, Y. (1996). Extraction of concurrent processes from higher-dimensional automata. In *Proceedings of CAAP'96*, pages 72–85.

Valmari, A. (1990). A stubborn attack on state explosion. In *Proc. of CAV'90.* Springer Verlag, LNCS.

van Glabbeek, R. (1991). Bisimulation semantics for higher dimensional automata. Technical report, Stanford University, Manuscript available on the web as http://theory.stanford.edu/~rvg/hda.

van Glabbeek, R. and Goltz, U. (1989). Partial order semantics for refinement of actions. *Bulletin of the EATCS*, (34).

Winskel, G. and Nielsen, M. (1994). *Models for concurrency*, volume 3 of Handbook of Logic in Computer Science, pages 100–200. Oxford University Press.

Yannakakis, M., Papadimitriou, C. H., and Kung, H. T. (1979). Locking policies: Safety and freedom from deadlock. In *20th Annual Symposium on Foundations of Computer Science*, pages 286–297, San Juan, Puerto Rico. IEEE.