Homology of Higher Dimensional Automata*

Eric Goubault[1] and Thomas P. Jensen[2]

[1] LIENS, Ecole Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, FRANCE, email:goubault@dmi.ens.fr

[2] Dept. of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, U.K., email:tpj@doc.ic.ac.uk

Abstract. Higher dimensional automata can model concurrent computations. The topological structure of the higher dimensional automata determines certain properties of the concurrent computation. We introduce bicomplexes as an algebraic tool for describing these automata and develop a simple homology theory for higher dimensional automata. We then show how the homology of automata has applications in the study of branching-time equivalences of processes such as bisimulation.

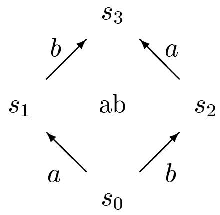
1 Introduction

Geometry has been suggested as a tool for modeling concurrency using higher dimensional objects to describe the concurrent execution of processes. This contrasts with earlier models based on the interleaving of computation steps to capture all possible behaviours of a concurrent system. Interleaving models must necessarily commit themselves to a specific choice of *atomic action* which makes them unable to distinguish between the execution of two truly concurrent actions and two mutually exclusive actions as these are both modeled by their interleaving. In [9] and [1] Pratt and Glabbeek advocate a model of concurrency based on geometry and in particular on the notion of a higher–dimensional automaton (HDA). Higher–dimensional automata are generalisations of the usual non–deterministic finite automata as described in *e.g.* [2]. The basic idea is to use the higher dimensions to represent the concurrent execution of processes. Thus for two processes, a and b, we model the mutually exclusive execution of a and b by the automaton

$$\begin{tikzcd} & s_3 & \\ s_1 \arrow[ur, "b"] & & s_2 \arrow[ul, "a"'] \\ & s_0 \arrow[ul, "a"] \arrow[ur, "b"'] & \end{tikzcd}$$

whereas their concurrent execution is modeled by including the two–dimensional surface delineated by the (one–dimensional) a– and b–transitions as a transition in the automaton. This is pictured as

* This work was partially supported by ESPRIT BRA 3074 SEMAGRAPH



A computation is modeled by a path in this higher–dimensional automaton. Now several properties of computational relevance are determined by the topology of the HDA. *E.g.* a HDA is deterministic if for any two paths in the automaton one can be transformed into the other in a continuous fashion, *i.e.* non–determinism arises from *holes* in the automaton that prevent the transformation of one path into another. Furthermore certain differences in the topologies of two HDA imply that a computation is possible in one HDA but not the other, *i.e.* information about the topology of HDA can be used to answer questions about *bisimulation* between the HDA.

The field of algebraic topology offers several techniques for giving an algebraic description of topological properties of geometric objects. In this paper we develop a theory of *homology* of HDA. To each HDA we associate a sequence of groups that characterises the essential branchings and mergings in the HDA. These *homology groups* seem to be more amenable to automated computation than the fundamental groups associated with homotopy theory.

We introduce HDA in section 2. Section 3 defines the notion of bicomplex and show how HDA can be described by bicomplexes. In section 4 we give a translation of a CCS–like process language into bicomplexes. Section 5 develops the theory of homology of bicomplexes and show how our process language can be translated into homology groups. Finally section 6 shows how differences in the homology groups of two bicomplexes imply that the associated bicomplexes are not bisimular. Section 7 concludes.

Notation. We denote by $\mathbf{Z}_2$ the group $(\mathbf{Z}/2\mathbf{Z}, +)$ (which is also a field with multiplication being multiplication modulo 2). For Q a set, we write $\overline{Q}$ for the free $\mathbf{Z}_2$-vector space generated by Q (or Vect(Q)). For f a function from a set A to a set B, we define $\overline{f}$ from $\overline{A}$ to $\overline{B}$ as being the linear extension of f. We write $\otimes$ for the tensor product between two $\mathbf{Z}_2$-vector spaces, $\oplus$ for the direct sum of two $\mathbf{Z}_2$-vector spaces. The vector space generated by the cartesian product of two sets is the tensor product[3] of the vector spaces generated by each of these sets. We write $\{*\}$ for the trivial structure (e.g. group, or vector space), (x) for the $\mathbf{Z}_2$-vector space generated by x, Ker f for the kernel of the function f, and Im f, for the image of the function f. Id is the identity function. Given V a $\mathbf{Z}_2$-vector space, whose basis is $\{e_i$ / i=1...$\alpha\}$, we define the scalar product of two vectors x and y as being $\langle x, y\rangle = \Sigma_{i=1\ldots\alpha} x_i.y_i$ (with value in $\mathbf{Z}$), where $x = \Sigma_{i=1\ldots\alpha} x_i.e_i$ and $y = \Sigma_{i=1\ldots\alpha} y_i.e_i$.

[3] for a full definition see [4]

2 Higher–dimensional automata

We have already given one example of a higher–dimensional automaton, *viz.* the automaton in the introduction with the interior filled. In this section we define the concept of a higher–dimensional automaton (HDA) precisely and explain how this definition extends the usual definition of a finite automaton. Furthermore we define the notion of a *path* through a HDA, used to describe a concurrent computation.

The description of a finite automaton over an alphabet Σ consists of a set of states S together with a transition function $t : S \times \Sigma \to S$ such that $t(s, m)$ is the state reached when reading symbol m in state s. In addition, there is an initial state $s_0 \in S$ and a set of final states $F \subseteq S$. In this framework there is a clear distinction between states, where the automaton "rests" and transitions where the automaton is "in action". We call such an automaton a one–dimensional automaton, for reasons to become clear shortly.

This way of viewing an automaton is inadequate when the automaton is capable of performing several actions simultaneously. Such an automaton can be more or less active according to how many actions are being performed. We can picture such an automaton as a network of one–dimensional automata in which some automata rest and some are in action. A state of such a network is then a mixture of resting states and transitions/actions and the automaton changes from one state to another by initiating or terminating one or more actions. The number of actions is called the *dimension* of the state. We shall call such an automaton a higher–dimensional automaton.

The classical finite automaton can be described in this fashion. The new set of states consists of the old set of states, all elements of which have dimension zero, together with the states $(s, m, t(s, m))$ of dimension one, which represent the transitions. Note that the dimension of a state agrees with the standard way of drawing finite automata: A state is represented by a point, *i.e.* a zero–dimensional object, and a transition by a line, *i.e.* an object of dimension one.

The following is Glabbeek's definition of a HDA from [1]:

Definition 1. A higher–dimensional automaton is a tuple $(S, d, \sigma, \tau, s_0, F, \ell)$ where

- S is a set of states
- $d : S \to \mathbf{N}$ is the dimension of a state
- $\sigma, \tau : S \times \mathbf{N} \rightharpoonup S$ are partial functions. For $s \in S$ and $k < d(s)$, $\sigma(s, k)$ and $\tau(s, k)$ are the start state and the end state of the action in the k'th dimension. The functions σ, τ must satisfy the *cubical* laws (cf. [1]): For $i \leq j$

$$i) \quad d(\sigma(s,k)) = d(\tau(s,k)) = d(s) - 1$$

$$ii) \quad \sigma(\sigma(s,i),j) = \sigma(\sigma(s,j+1),i) \qquad iii) \quad \sigma(\tau(s,i),j) = \tau(\sigma(s,j+1),i)$$

$$iv) \quad \tau(\sigma(s,i),j) = \sigma(\tau(s,j+1),i) \qquad v) \quad \tau(\tau(s,i),j) = \tau(\tau(s,j+1),i)$$

- $s_0 \in S$ is the initial state and $F \subseteq S$ is the set of final states. They must satisfy $d(s_0) = d(f_i) = 0$ for all $f_i \in F$

– $\ell : S \rightharpoonup \Sigma$ is the labeling function, that assigns a label to every state of dimension one, *i.e.* $\ell(s)$ is defined if and only if $d(s) = 1$. Furthermore we require that

$$\ell(\sigma(s,i)) = \ell(\tau(s,i)) \qquad \text{for } i = 0, 1$$

Note Instead of specifying the dimension function $d : S \to \mathbf{N}$ we shall sometimes present S as a family of sets $\{S_i\}_{i \in \mathbf{N}}$ where S_i is the set of states of dimension i.

The first cubical law states that the dimension of state increases by one when an action is initiated and decreases by one when an action is terminated. Representing a state as a list of the actions that is being performed we can explain the other cubical laws. Let us as example take the law $iv)$ and assume that $i < j$. The $j+1$'th element of the list representing s will be the j'th element in the list representing the state just before s where the i'th action has not been initiated, *i.e.* the state $\sigma(s,i)$. Hence the adjustment in the index. The rules $ii), iii), v)$ can be explained in a similar way.

Example As an example we have the automaton from the introduction that performs the two actions a and b concurrently. It can be drawn as follows:

$$\begin{tikzcd}
 & s_3 & \\
s_1 \arrow[ur, "b''"] & ab & s_2 \arrow[ul, "a''"'] \\
 & s_0 \arrow[ul, "a'"] \arrow[ur, "b'"'] &
\end{tikzcd}$$

We name one–dimensional states (*i.e.* transitions) by their label and distinguish between different transitions with the same label using primes. In our example there are two transitions with the label a which are denoted by a' and a'' respectively[4]. Furthermore we shall write $\sigma(s,i)$ and $\tau(s,i)$ as $\sigma_i(s)$ and $\tau_i(s)$.

The labeling function ℓ extends to a function ℓ^* on higher–dimensional states by stipulating that $\ell^*(s)$ is the string of labels of the one–dimensional states constituting s (. is the string concatenation). Finally, ℓ^* of a sum of states is the set of ℓ^* of each of the summands.

Definition 2. A path in a higher-dimensional automaton $(S, d, \sigma, \tau, s_0, F, \ell)$ is a sequence p=$(p_i)_{0 \leq i \leq n}$ such that:

– $\forall i, p_i \in \cup_j S_j$
– $p_0 = s_0$
– $\forall i : 0 \leq i \leq n-1 : \sigma_j(p_{i+1}) = p_i$ or $\tau_k(p_i) = p_{i+1}$ for some $j, k < d(p_i)$

n+1 is the length of the path p. A word is w=$(\ell^*(p_i))_{0 \leq i \leq n}$. An acceptable path is a path such that $p_n \in F$. An acceptable word is a word whose associated path is acceptable. The language accepted by $(S, d, \sigma, \tau, s_0, F, \ell)$ is the set of all acceptable words.

[4] Thus the primes should not appear in the drawing. We have added them only for the sake of clarity

3 Complexes

In this section we introduce some concepts from algebraic topology that are useful for studying higher-dimensional automata. One of the fundamental problems studied in algebraic topology is how to characterise geometric objects that are equal modulo a continuous deformation. That is, two objects are considered equal if one can be deformed into the other by stretching, bending *etc* but not tearing nor piercing it. For connected objects, the essential problem here is to define rigorously the notion of a *hole* in an object and algebraic topology offers several possibilities. It was first done via the notion of fundamental group (a group of paths on a manifold[5]) in [8], and more generally (and later) by the homotopy groups. This notion is very close to the geometric intuition: A hole is something that prevents a path on a manifold passing on one side of the hole to be continuously deformed into another path passing on the other side of the hole. But it is difficult to compute these homotopy groups in general. To overcome this problem Poincaré introduced the notion of homology [6]. It defines holes in an even more algebraic and constructive way. First of all, a given manifold is represented in a discrete way by triangulating it. Thus the manifold is represented as a union of points, edges, triangles, tetrahedrons *etc*. These discrete sets are called simplicial sets. Then these objects are oriented geometrically; here it suffices to give an order on points. Afterwards, the notion of a *boundary* of an object of dimension n+1 is defined as a formal sum (keeping in mind the orientation problems) of objects of dimension n. We can call boundary everything which is the result of taking the boundary of an (or a sum of) object(s). The boundary of a line segment is its endpoints and the boundary of a triangle is the edge of the triangle. A *cycle* is a formal sum of objects whose boundary is null. All boundaries are cycles, but the converse is not true. It is false when there is a hole inside a cycle. A characterization of holes is therefore, objects which are cycles but not boundaries.

More precisely an n–dimensional manifold can be represented as a collection of k–dimensional objects ($k = 0, \ldots, n$) and is fully described by listing for each object of dimension k its $(k-1)$–dimensional boundaries. Thus a manifold is given by the diagram

$$\ldots \quad Q_i \xrightarrow{\partial^i} Q_{i-1} \xrightarrow{\partial^{i-1}} \ldots \xrightarrow{\partial^1} Q_0$$

where Q_i is the collection of i–dimensional objects and ∂^i is the *boundary* map. Since the definition of boundaries and cycles involve formal sums of objects, it is more convenient to study this diagram where Q_i is replaced by $\overline{Q_i}$ and ∂^i is replaced by its linear extension. A boundary of an (i+1)–dimensional object can then be described as an object lying in the image of ∂^{i+1}. Similarly, as a cycle is an object with zero boundary, the cycles of dimension i are precisely the objects in the kernel of ∂^i. As all boundaries are cycles we have that $Im\partial^{i+1} \subseteq Ker\partial^i$, so $\overline{\partial}^i \circ \overline{\partial}^{i+1}$=0: this equation makes the diagram a *complex*. As both $Im\,\partial^{i+1}$ and $Ker\,\partial^i$ are subgroups

[5] We shall not define an n dimensional manifold precisely, but just use it to denote a topological space that locally "looks like" a Euclidean space $\mathbf{R}^n$

[6] Here, we just speak of simplicial homology

of $\overline{Q_i}$ we can form the quotient

$$Ker\,\partial^i / Im\,\partial^{i+1}.$$

This is called the i–th *homology group* of the complex. As a first indication of the information present in the homology groups we note that non–zero homology in dimension i indicates an i–cycle that is not the boundary of an $i+1$–dimensional object, *i.e.* an i–dimensional hole in the manifold.

3.1 Transition systems and bicomplexes

In computer science, transition systems, which can also be seen as discrete formalizations of continuous processes, are central in the semantic definitions of programming languages. Paths are traces of execution, and one can imagine that a deformation of a path into another one is not possible because of branchings or mergings, equivalent of holes in the preceding case. In the sequential case, what we have just said is trivial. But if we generalize transition systems to represent concurrent processes, it is no more the case. A very nice geometrical way of doing it can be found in [1] and [9].Then, a semantic equivalence on these objects (in the manner of bisimulation) can be expressed in terms of deformation of paths, that is homotopy. It is then natural to consider an algebraic equivalent, much more computable, in the form of some kind of homology. A difficulty is to be able to take into account the (irreversible) flow of time *i.e.* to talk about the beginning and the end of a path. One can think of several ways to do it. One is to consider homology of monoids as in [3]. Another, as proposed in [9] is to use the formalism of n-categories. We suggest bicomplexes because they capture the difference between start–and end–states and fit well with the cubical laws while still being easy to understand and compute.

Definition 3. A bicomplex $(A_i, \delta_0^i, \delta_1^i)$ is a sequence of groups, A_i, together with two sequences of group homomorphisms δ_0^i, δ_1^i:

$$\begin{tikzcd} \dots & A_3 \arrow[r, shift left, "\delta_1^3"] \arrow[r, shift right, "\delta_0^3"'] & A_2 \arrow[r, shift left, "\delta_1^2"] \arrow[r, shift right, "\delta_0^2"'] & A_1 \arrow[r, shift left, "\delta_1^1"] \arrow[r, shift right, "\delta_0^1"'] & A_0 \end{tikzcd}$$

such that

$$\delta_0^i \circ \delta_0^{i+1} = \delta_1^i \circ \delta_1^{i+1} = 0$$
$$\delta_0^i \circ \delta_1^{i+1} = \delta_1^i \circ \delta_0^{i+1}$$

Let S_i denote the states of dimension i. For each S_i we can construct the free $\mathbf{Z}_2$–vector space generated by S_i, denoted by $\overline{S}_i$. The elements of $\overline{S}_i$ are formal sums

$$s_1 + \ldots + s_n \qquad s_i \in S_i, n \in \mathbf{N}$$

where each $s_i \in S_i$ appears at most once. Addition of two elements is defined as usual by

$$\sum_{s\in S_i} n_j s + \sum_{s\in S_i} n_k s = \sum_{s\in S_i} (n_j + n_k)s$$

where all coefficients are in $\mathbf{Z}_2$, *i.e.* an s appearing in both sums disappears. It is straightforward that each $\overline{S}_i$ forms an abelian group with the empty sum (all coefficients $= 0$) as neutral element.

The mappings $\sigma, \tau : S \times \mathbf{N} \rightharpoonup S$ from a HDA induce a sequence of mappings

$$\sigma^i, \tau^i : S_{i+1} \to S_i$$

where σ^i and τ^i are the restrictions of σ and τ to states of dimension $i+1$. By linear extension we obtain two group homomorphisms $\delta^i_\sigma, \delta^i_\tau : \overline{S}_{i+1} \to \overline{S_i}$ by:

$\delta^i_\sigma(s_1 + \ldots + s_n) = \delta^i_\sigma(s_1) + \ldots + \delta^i_\sigma(s_n)$ and $\delta^i_\tau(s_1 + \ldots + s_n) = \delta^i_\tau(s_1) + \ldots + \delta^i_\tau(s_n)$.

The cubical laws can then be used to show the following theorem

Theorem 4. *Let $(S, d, \sigma, \tau, s_0, F, \ell)$ be a higher–dimensional automaton. Then*

$$\cdots \qquad \overline{S_3} \underset{\delta^3_\tau}{\overset{\delta^3_\sigma}{\rightrightarrows}} \overline{S_2} \underset{\delta^2_\tau}{\overset{\delta^2_\sigma}{\rightrightarrows}} \overline{S_1} \underset{\delta^1_\tau}{\overset{\delta^1_\sigma}{\rightrightarrows}} \overline{S_0}$$

is a bicomplex.

Proof: Show $\delta_\sigma\delta_\sigma = 0$. Cubical law $ii)$ implies that

$$\sum_{i=1}^{n}(\sum_{j=1}^{i-1}\sigma(\sigma(q,i),j) + \sum_{j=i}^{n-1}\sigma(\sigma(q,i),j)) = \sum_{i=1}^{n}\sum_{j=1}^{i-1}\sigma(\sigma(q,i),j) + \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sigma(\sigma(q,j),i) =$$
$$\sum_{i=2}^{n}\sum_{j=1}^{i-1}\sigma(\sigma(q,i),j) + \sum_{j=1}^{n}\sum_{i=j+1}^{n}\sigma(\sigma(q,i),j) = \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sigma(\sigma(q,i),j) + \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sigma(\sigma(q,i),j) = 0.$$

The rest of the proof is similar. From now on, all HDA will be presented by bicomplexes (Q,∂_0,∂_1) and an initial state i, a set of final states F, and a labeling operator ℓ, written as a tuple (Q,∂_0,∂_1,i,F,ℓ).

4 Translation of process algebra into bicomplexes

We now develop a truly concurrent semantics of a process algebra by translating the terms of the algebra into bicomplexes. First a quick review of the CCS–like syntax: We assume given a set of actions $\Sigma \cup \{\tau\}$ such that when Σ contain an action ω it also contains its complementary action $\overline{\omega}$. The action τ is called the internal action. Furthermore we have the idle process **nil**. Terms are then formed according to the following grammar:

$$t ::= \omega \mid \mathbf{nil} \mid (t_1 + t_2) \mid (t_1 \mid t_2) \mid (t_1\ ;\ t_2)$$

where $+$ is choice, $|$ parallel composition and ; is sequential composition. For expository reasons we have divided the parallel operator in two operators: $t_1 \| t_2$, which is parallel composition without communication, and the general one $|$.

The translation is defined by structural induction over the terms. For each construct we specify the resulting bicomplex $(\overline{\{T_i\}_{i\in\mathbf{N}}}, \partial_0, \partial_1, j, G, L)$. T_i are states of dimension i, *i.e.* generators of the i–th vector space and j and G are the initial state and the set of final states respectively. We introduce a special 0–dimensional state 1 (neutral for $\otimes$) to represent the idle action **nil**.

(case 1) $t = \omega \in \Sigma$. Then T_0=$\{1, \beta\}$, T_1=$\{\gamma\}$, j=$\{1\}$, G=$\{\beta\}$, $\partial_0(\gamma) = 1$, $\partial_1(\gamma) = \beta$, and L($\gamma$)=$\omega$, where β,γ are fresh state names.

(case 2) t=**nil**. Then T_0=$\{1\}$, J=$\{1\}$, G=$\{1\}$.

(case 3) t=q+q'. We assume that the translation of q is $(Q, \partial_0, \partial_1, l, i, F)$, and of q' is $(Q', \partial_0, \partial_1, l', i', F')$. Then:
- $T_0 = (Q_0 \cup Q'_0)/\{i = i'\}$
- $\forall i \geq 1, T_i = Q_i \cup Q'_i$, *i.e.* $\overline{T_i} = \overline{Q_i} \oplus \overline{Q'_i}$
- j=i
- G=F∪F'
- $\partial_j[t] = \partial_j[q] \oplus \partial_j[q']$
- and for the labeling function:

$$\forall x \in T_1 : L(x) = \begin{cases} l(x) & \text{if x is in } Q_1 \\ l'(x) & \text{if x is in } Q'_1 \end{cases}$$

(case 4) t=q ; q'. We have the translation of q : $(Q, \partial_0, \partial_1, l, i, F)$, and of q' : $(Q', \partial_0, \partial_1, l', i', F')$. Then, if F=$\{f_1, ..., f_m\}$

$$\forall i \geq 0, T_i = Q_i \ \cup \ f_1 \times Q'_i \ \cup \ ... \ \ f_m \times Q'_i$$

We have also,
- j=i
- $G = f_1 \times F' \ \cup \ ... \ \cup \ f_m \times F'$
- $\partial_j[T] = \partial_j[Q] \oplus Id \otimes \partial_j[Q']$
- for x in T_1, L(x)=l(x) if x is in Q_1, otherwise x=(f_i,y) with y element of Q'_1 and L(x)=l'(y).

(case 5) t=(q $\|$ q'). By considering the valid transitions for t, we see that we need that the paths of t be isomorphic to the cartesian product of the paths of q and q'. Thus it is natural to form the product of the cubical sets underlying the corresponding automata:

$$\forall k \geq 0, T_k = \bigcup_{i=0\ldots k} Q_i \times Q'_{k-i}.$$

And for the boundary operators:

$$\partial^k_j[T] = \bigoplus_{i=0\ldots k} (\partial^i_j[Q] \otimes Id \ \oplus \ Id \otimes \partial^{k-i}_j[Q'])$$

Moreover, j=i⊗i', G=F×F'. Finally, the labeling operator is given by:

$$L^*_{|\oplus_{i,j}\overline{Q}_i\otimes\overline{Q}_j} = \Sigma_{i,j} l^*_{|\overline{Q}_i}.l^*_{|\overline{Q}_j}$$

(case 6) t=(q | q'). There is here a possibility of communication between q and q', represented by equations on tensor products between states which can interact. We define the new bilinear product, quotient of the tensor product by these equations, as follows:

- x $\in Q_1$, y $\in Q'_1$ and l(x)=$\overline{l}(y)$, then x$\otimes_c$y=s, where s is in T_1 such that $L(s)=\tau$, $\partial_0(s)=\partial_0(x)\otimes\partial_0(y)$, $\partial_1(s)=\partial_1(x)\otimes\partial_1(y)$.
- otherwise x$\otimes_c$y=x⊗y (no interaction).

The symbol τ is a distinguished element in Σ, used (see for instance [7]) to represent the internal synchronisation action.

Now we define T as being given by almost the same equations as in (case 5).

- $\overline{T}_0 = \overline{Q}_0 \otimes \overline{Q}'_0$
- $\overline{T}_1 = \overline{Q}_0 \otimes \overline{Q}'_1 \oplus \overline{Q}_1 \otimes \overline{Q}'_0 \oplus$ Vect($\{x \otimes_c y / x \in Q_1, y \in Q'_1$ and $l(x) = \overline{l(y)}\}$)
- $\overline{T}_2 = \overline{Q}_0 \otimes \overline{Q}'_2 \oplus \overline{Q}_2 \otimes \overline{Q}'_0 \oplus$ Vect($\{x \otimes y / x \in Q_1, y \in Q'_1$ and $l(x) \neq \overline{l(y)}\}$)
- $\forall k \geq 3, T_k = \bigcup_{i=0\ldots k} Q_i \times Q'_{k-i}$.

And for the boundary operators, for $q \in Q_i$ and $q' \in Q'_{k-i}$:

$$\partial^k_j[T](q\otimes q') = \partial^i_j[Q](q)\otimes q' \ \oplus \ q\otimes\partial^{k-i}_j[Q'](q')$$

and

$$\partial^{k-1}_j[T](q\otimes_c q') = \partial^i_j[Q](q)\otimes\partial^{k-i}_j[Q'](q'), \text{ if } k=2, i=1, l(q)=\overline{l'(q')}$$

Moreover, j=i⊗i', G=F×F'. Finally, the labeling operator is given (on 1-states) by:

$$L(x\otimes_c y) = \tau \quad \text{if } x\in Q_1, y\in Q'_1 \text{ and } l(x)=\overline{l'(y')}$$

otherwise

$$L(x\otimes y) = \begin{cases} l(x) & \text{if } x\in Q_1, y\in Q'_0 \\ l'(y) & \text{if } x\in Q_0, y\in Q'_1 \end{cases}$$

For a term t, we denote by $[\![t]\!]$ the result of its translation into a higher-dimensional automaton, as described above. We then have a correctness result about the translation, which states that the paths of $[\![t]\!]$ describe the fully concurrent execution of t (as can be inferred from a presentation of CCS via rules of transition, in [7]). We prefer in this article to give a few examples instead of a fully abstract treatment of that property.

Examples

(1) We use the inductive construction above to translate the CCS-term (a|b) (which is equal to (a||b)). We have:

$$1 \xrightarrow{a} \alpha \qquad\qquad 1 \xrightarrow{b} \beta$$

to represent $[\![a]\!]$ and $[\![b]\!]$ respectively. We now form the tensor product:

$$\begin{tikzcd}
& \alpha\otimes\beta & \\
\alpha \arrow[ur, "\alpha\otimes b"] & a\otimes b & \beta \arrow[ul, "a\otimes\beta"'] \\
& 1 \arrow[ul, "a"] \arrow[ur, "b"'] &
\end{tikzcd}$$

The paths of ($[\![a\|b]\!]$) are:

$$\begin{array}{ll}
1.(1, a, \alpha, \alpha\otimes b, \alpha\otimes\beta) & 4.(1, b, \beta, a\otimes\beta, \alpha\otimes\beta) \\
2.(1, a, a\otimes b, \alpha\otimes b, \alpha\otimes\beta) & 5.(1, b, a\otimes b, \alpha\otimes b, \alpha\otimes\beta) \\
3.(1, a, a\otimes b, a\otimes\beta, \alpha\otimes\beta) & 6.(1, b, a\otimes b, a\otimes\beta, \alpha\otimes\beta)
\end{array}$$

Paths 3 and 5 are not considered in [1]. We can nevertheless give an intuitive meaning to each of them. Associate action a to processor p_1 and action b to processor p_2. Then for path 3 we have:
1: p_1 and p_2 are idle
a: p_1 is computing a while p_2 is idle
a⊗b: p_1 continues to compute a while p_2 is computing b
a$\otimes\beta$**:** p_1 still computes a while p_2 is idle (it has finished its computation)
$\alpha\otimes\beta$**:** p_1 and p_2 are idle (they have both finished their computations)

(2) We now compute $[\![$(a+b)$\|$c$]\!]$.

$$\begin{tikzcd}
& \alpha \arrow[rr, "\alpha\otimes c"] \arrow[d, phantom, "a\otimes c"] & & \alpha\otimes\gamma \\
1 \arrow[ur, "a"] \arrow[rr, "c"] \arrow[dr, "b"'] & {} & \gamma \arrow[ur, "a\otimes\gamma"'] \arrow[dr, "b\otimes\gamma"] & \\
& \beta \arrow[rr, "\beta\otimes c"'] \arrow[u, phantom, "b\otimes c"] & & \beta\otimes\gamma
\end{tikzcd}$$

5 Homology of bicomplexes

Let (Q,∂_0,∂_1) be a bicomplex arising from a HDA. Then if two transitions of dimension one a, b have a common start point, *i.e.* $\partial_0(a) = \partial_0(b)$, then the sum a+b will belong to the kernel of ∂_0 since we work modulo 2:

$$\partial_0(a + b) = \partial_0(a) + \partial_0(b) = 0$$

i.e. $a + b \in Ker\partial_0$. So a+b is a potential branching. However it is not a non-deterministic choice if a and b are boundaries of a higher-dimensional transition, *i.e.* if there exists a 2-dimensional transition A such that $\partial_0(A) = a+b$. These branchings could have been defined in a more standard way, but this is particularly amenable to generalisation for higher-dimensional transitions. Intuitively, a non-deterministic choice of dimension i is an element of the kernel of ∂_0^i modulo the boundaries of i+1-dimensional objects, *i.e.* is an element of the i-th homology group:

$$Ker\partial_0^i/Im\partial_0^{i+1}$$

This is for branching. A similar relationship holds for mergings and the maps ∂_1^i.

Definition 5. For (Q,∂_0,∂_1) a $\mathbf{Z}_2$-bicomplex, we define two sequences of homology (see for instance [5]) groups (or homology vector spaces):

$$H_i(Q,\partial_0) = Ker\partial_0^i/Im\partial_0^{i+1} \qquad H_i(Q,\partial_1) = Ker\partial_1^i/Im\partial_1^{i+1}$$

An element of $Ker\partial_j^i$ is an i-cycle, and an element of $Im\partial_j^{i+1}$ is an i-boundary. An element of $H_i(Q,\partial_0)$ is called a branching of dimension i. An element of $H_i(Q,\partial_1)$ is called a merging of dimension i. An element x satisfying $\langle x, H_i(Q,\partial_0)\rangle \neq 0$ is called a branching choice of dimension i. We write $H_*(T,\partial_j)$ for $\bigoplus_{k\geq 0} H_k(T,\partial_j)$.

We begin by giving some intuition about these homology groups. We consider first of all the homology groups of dimension 0.

Lemma 6. *Consider a finite automaton given by (Q,∂_0,∂_1,i,F,ℓ), with no cycle*

- *i=$H_0(Q,\partial_1)$ implies that all states of Q are reachable*
- *F=$H_0(Q,\partial_0)$ implies that all states of Q are co-reachable*

Now we give a few examples of the groups $H_i(Q,\partial_j)$.

(1) We consider the first examples of the last section. Then, $H_0([\![a\|b]\!],\partial_0) = (\alpha\otimes\beta)$ (it is the end state), $H_0([\![a\|b]\!],\partial_1) = (1)$ (it is the initial state), and the other H_i are null (there is no branching nor merging, it is deterministic).

(2) We study now Q=$[\![a \mid \overline{a}]\!]$. We now have, $H_0(Q,\partial_0) = (\alpha\otimes\beta)$, $H_0(Q,\partial_1) = (1)$, $H_1(Q,\partial_0) = (a+\tau)\oplus(\tau+\overline{a})$ (we have two branchings of dimension one, that is a choice between a, τ, and $\overline{a}$), $H_1(Q,\partial_1) = (a\otimes\beta+\tau)\oplus(\tau+\alpha\otimes\overline{a})$ (we have two mergings of dimension 1), and the other H_i are null.

5.1 Some results from homology theory

We now list some results concerning the calculation of homology groups of direct sums and tensor products of complexes. These results are needed when we later are to interpret operators of our process algebra as operators on homology groups.

We first look at the sum of two manifolds. If $(\overline{Q}_*, \partial[Q])$ and $(\overline{Q}'_*, \partial[Q'])$ are the complexes corresponding to two manifolds, then the one arising from their disjoint union $(\overline{T}_i, \partial[T])$, is given by:

$$\forall i, \overline{T}_i = \overline{Q}_i \oplus \overline{Q'}_i$$

and,

$$\forall i, \partial^i[T] = \partial^i[Q] \oplus \partial^i[Q']$$

Now for bicomplexes, the analogous is fairly obvious: let $(\overline{Q}_*, \partial_0[Q], \partial_1[Q])$ and $(\overline{Q'}_*, \partial_0[Q'], \partial_1[Q'])$ be two bicomplexes. Then we form $(\overline{T}_*, \partial_0[T], \partial_1[T])$ with

$$\forall i, \overline{T}_i = \overline{Q}_i \oplus \overline{Q'}_i,\ \partial_0^i[T] = \partial_0^i[Q] \oplus \partial_0^i[Q'],\ \partial_1^i[T] = \partial_1^i[Q] \oplus \partial_1^i[Q']$$

We can compute the homology groups of T given those of Q and Q' as follows:

Lemma 7. *The homology groups of T, disjoint union of Q and Q' are:*

$$\forall i, j, H_i(T, \partial_j) = H_i(Q, \partial_j) \oplus H_i(Q', \partial_j).$$

More generally, we have a relation between the homologies of bicomplexes Q, Q' and $Q \cap Q'$, using the so-called Mayer-Vietoris sequence (see for instance [6]). In particular, it permits us to adapt the preceding lemma to a case where we sum two bicomplexes, and identify their initial states (used in next section).
Now for products: One can compute the homology of the cartesian product of two manifolds, given the homology of both of them. If the cartesian product of two (concrete) manifolds M_1 and M_2 is based on the set of pairs of points of M_1 and M_2 respectively, we have to describe the discrete equivalent of such an operation on the simplicial sets and complexes arising from a triangulation of them. Let Q_i and Q'_i be the simplicial sets of objects of dimension i associated with the manifolds M_1 and M_2. Then the simplicial sets T_i associated with the product of M_1 and M_2 are:

$$\forall k \geq 0, T_k = \bigcup_{i=0\ldots k} Q_i \times Q'_{k-i}.$$

Therefore, the generated simplicial complex is:

$$\forall k \geq 0, \overline{T}_k = \bigoplus_{i=0\ldots k} \overline{Q}_i \otimes \overline{Q}'_{k-i}.$$

And the boundary operator is :

$$\partial^k[T] = \bigoplus_{i=0\ldots k} (\partial^i[Q] \otimes Id \ \oplus \ Id \otimes \partial^{k-i}[Q']).$$

This construction is the tensor product of the two complexes associated with M_1 and M_2 (see [6]). Now the homology groups are given by the Künneth formula:

Lemma 8.

$$H_k(T, \partial) = \bigoplus_{i=0\ldots k} (H_{k-i}(Q, \partial) \otimes H_i(Q', \partial))$$

The reader can verify that the definition of tensor product we have given is also correct for bicomplexes.

5.2 Translation of process algebra into homology groups

In this section we demonstrate that the operators of our process algebra can be interpreted as operators on homology groups such that the interpretation of a term gives the homology groups of the bicomplex corresponding to the term.

(case 1) $t = \omega \in \Sigma$. We have $H_i = \{*\}$ for all i≥1. $H_0([\![t]\!], \partial_0) = (\alpha)$, and $H_0([\![t]\!], \partial_1) = (1)$.
(case 2) t=**nil**. Here we also have $H_i = \{*\}$ for all i≥1. $H_0([\![t]\!], \partial_j) = (1)$.
(case 3) t=q+q'. We interpret choice as a connected sum of the complexes of q and q'. The homology groups are:
 - $H_1(T, \partial_j) = H_1(Q, \partial_j) \oplus H_1(Q', \partial_j) \oplus N$ where,
 N=({0,X+X'},+) with i=$\partial_0(X)$, i'=$\partial_0(X')$, $X \in Q$, $X' \in Q'$
 - $\forall i \neq 1, H_i(T, \partial_j) = H_i(Q, \partial_j) \oplus H_i(Q', \partial_j)$.

(case 4) t=q ; q'. The homology groups are then:

$$H_0(T, \partial_0) = \overline{F} \otimes H_0(Q', \partial_0) \quad H_0(T, \partial_1) = (1)$$

$$\forall i \geq 1, H_i(T, \partial_j) \oplus \overline{F} \otimes H_i(Q', \partial_j).$$

(case 5) t=(q ∥ q'). The parallel composition without communication is interpreted as the tensor product of complexes. For the homology groups, we then have (Künneth formula):

$$H_0(T, \partial_j) = H_0(Q, \partial_j) \otimes H_0(Q', \partial_j)$$

$$H_k(T, \partial_j) = \bigoplus_{i=0\ldots k} (H_{k-i}(Q, \partial_j) \otimes H_i(Q', \partial_j))$$

(case 6) t=(q | q'). There is here a possibility of communication between p and q, represented by equations on tensor products between states which can interact. For the homology groups, we have:

$$H_1(T, \partial_j) = H_1(Q, \partial_j) \otimes H_0(Q', \partial_j) \oplus H_0(T, \partial_j) \otimes H_1(Q', \partial_j) \oplus I(Q, Q'),$$

where I(Q,Q') is the interaction term:

$$I(Q, Q') = Vect\{x \otimes_c (y + \partial_j(y)), (x + \partial_j(x)) \otimes_c y \ / \ x \in Q_1, y \in Q'_1, l(x) = \overline{l(y)}\}$$

and for kŒ1, $H_k(T, \partial_j)$ is given by the Künneth formula from above.

Remark In what we have presented here, we have always $F = H_0(T, \partial_0)$. One can show (using the Mayer-Vietoris sequence) that as soon as we have the restriction operator \, we can have elements of $H_0(T, \partial_0)$ not in F. They are called deadlocks. One can use such a definition, and tools from homological algebra, to study deadlocks of processes, or also failure pairs if one wishes to study failure equivalence. In the next section we concentrate on bisimulation.

6 Bisimulation equivalence

Glabbeek defined the notion of bisimulation of HDA. In this section we demonstrate how the homology groups of HDA can be used to show that two HDA are not bisimulation equivalent.

Definition 9. S is a bisimulation between (Q,∂_0, ∂_1,l,I,F) and (Q',∂_0, ∂_1,l',I',F') if:

- S is a relation between $\cup_i Q_i$ and $\cup_j Q'_j$
- all $s \in I$ are related to an $s' \in I'$ and vice-versa
- (s,s')∈S ⇒ (s∈F ⇔ s'∈F')
- (s,s')∈S ⇒ (∀ q a path for Q such that ∃i, q_i=s, ∃q' a path for Q' such that ∃j, q'_j=s' and (q_{i+1}, q'_{j+1}) ∈S, $l^*(q_{i+1}) = l^*(q'_{j+1})$)
- (s,s')∈S ⇒ (∀ q' a path for Q' such that ∃j, q'_j=s', ∃q a path for Q such that ∃i, q_i=s and (q_{i+1}, q'_{j+1}) ∈S, $l^*(q_{i+1}) = l^*(q'_{j+1})$)

(Q,∂_0, ∂_1,l,I,F) and (Q',∂_0, ∂_1,l',I',F') are bisimulation equivalent if and only if there exists a bisimulation between them.

This notion of bisimulation equivalence "naturally" generalizes the usual notion (as found in [7]) of observational equivalence, or bisimulation equivalence on one dimensional automata. In our setting, the description of bisimulation equivalence is more complex than in the sequential case. Nevertheless, we can show that it is still a branching (in our sense) time equivalence, that is, it locally preserves some geometric shapes.

We introduce now local invariants of bisimulation equivalence. Let $\equiv$ be the smallest congruence on Λ containing the relation G, defined by: $uGv \Leftrightarrow u = l^*(x)$, $v = l^*(y)$ and $\exists z \in Im\partial_0$ such that $\langle x, z\rangle$=1, $\langle y, z\rangle$=1 Then l^* induces a map $[l^*] : H_i(Q, \partial_0) \rightarrow \Lambda/\equiv$. Let S(x), for $x \in \oplus_i H_i(Q, \partial_0)$ be $[l^*](x)$ if card $l^*(x) \geq 2$, 0 otherwise. Under the hypothesis that the number of states at a finite distance (equal to the length of the minimal path to reach them) of a given state is finite, we have the following result,

Proposition 10. *If (Q,∂_0, ∂_1,l,I,F) and (Q',∂_0, ∂_1,l',I',F') are bisimulation equivalent then (∀i, $S(H_i(Q, \partial_0)) = S(H_i(Q', \partial_0))$ (as a subset of $\Lambda/\equiv = \Lambda/\equiv'$) provided that all (i-1)-states (for i≥2) of elements of H_i are in the ∂_0- boundary of i-states which have all different labels in $\Lambda/\equiv$).*

This states that the branchings whose branches have distinct labels are preserved (and possibly duplicated) by bisimulation equivalence.

Examples:

(1) Let q=a.(b+c) and q'=a.b+a.c. We compute their homology groups: $H_1([\![q]\!], \partial_0) = H_0([\![a]\!], \partial_0) \otimes H_1([\![b+c]\!], \partial_0) = H_0([\![a]\!], \partial_0) \otimes (H_1([\![b]\!], \partial_0) \oplus H_1([\![c]\!], \partial_0) \oplus (b+c)) = (\alpha \otimes b + \alpha \otimes c)$, and, $H_1([\![q']\!], \partial_0) = (a + a')$ (where a' is a distinct copy of a). Thus S($H_1([\![q]\!], \partial_0)$)={b+c}, and S($H_1([\![q']\!], \partial_0)$)={0}. Thus, by proposition 10 $[\![q]\!]$ and $[\![q']\!]$ are not bisimulation equivalent.

(2) Let q=(a+b)‖(c+d) and q'=a‖c+b‖c+a‖d+b‖d. We compute their homology groups: $H_1([\![q]\!],\partial_0) = H_1([\![a+b]\!],\partial_0) \otimes H_0([\![c+d]\!],\partial_0) \oplus H_0([\![a+b]\!],\partial_0) \otimes H_1([\![c+d]\!],\partial_0) = (a \otimes \gamma + b \otimes \gamma) \oplus (a \otimes \delta + b \otimes \delta) \oplus (\alpha \otimes c + \alpha \otimes d) \oplus (\beta \otimes c + \beta \otimes d)$. And $H_2([\![q]\!],\partial_0) = H_1([\![a+b]\!],\partial_0) \otimes H_1([\![c+d]\!],\partial_0) = (a \otimes c + b \otimes c + a \otimes d + b \otimes d)$. Now, $H_1([\![q']\!],\partial_0) = H_1([\![a\|c]\!],\partial_0) \oplus H_1([\![b\|c]\!],\partial_0) \oplus H_1([\![a\|d]\!],\partial_0) \oplus H_1([\![b\|d]\!],\partial_0) \oplus (a+b) \oplus (b+c) \oplus (c+d)$. And $H_2([\![q']\!],\partial_0) = \{*\}$. In particular, S($H_2([\![q]\!],\partial_0)$)={a.c+b.c+a.d+b.d}, and S($H_2([\![q']\!],\partial_0)$={0}. Thus q and q' are not bisimulation equivalent.

7 Conclusion

In this paper we have presented a technique for modeling concurrency centered around the description of higher dimensional automata by bicomplexes. From this we derived a notion of homology of HDA and demonstrated the pertinence of homology to concepts like non–determinism and bisimulation. The notion of HDA is taken directly from [1] and although the original definition is intuitively clear we find that the bicomplexes provide a more streamlined presentation of HDA for some purposes. The idea of applying algebraic topology to the study of concurrency was largely inspired by Pratt's article [9] where he introduced the notion of monoidal homotopy in HDA to model "true non–determinism". With this paper we hope to have provided some initial evidence of the usefulness of the related notion of homology. Further work on the subject will consider the restriction and recursion operators of CCS. Restriction is modeled by projection on vector spaces and recursion by limits in a suitable category of bicomplexes. Finally invariance under refinement of action should be related to the independence of choice of triangulation for calculating homology groups.

Acknowledgement. Thanks are due to Vaughan Pratt and Samson Abramsky for encouraging this work, to Chris Hankin and Patrick Cousot for support, and to the referees for helpful comments. Diagrams were drawn using Paul Taylor's tex macros.

References

1. Rob van Glabbeek. Bisimulation semantics for higher dimensional automata. Technical report, Stanford University, 1990.
2. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison–Wesley, 1979.
3. Yves Lafont and Alain Prouté. Church-rosser property and homology of monoids. Technical report, Ecole Normale Supérieure, 1990.
4. Serge Lang. *Algebra.* Addison Wesley, second edition, 1984.
5. Saunders Mac Lane. *Categories for the working mathematician.* Springer-Verlag, 1971.
6. William S. Massey. Homology and cohomology theory. In *Monographs and Textbooks in Pure and Applied Mathematics*, number 46. Marcel DEKKER, INC., 1978.
7. Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.
8. Henri Poincaré. De analysis situ. *Journal de l'Ecole Polytechnique*, 1895.
9. Vaughan Pratt. Modeling concurrency with geometry. In *Proc. 18th ACM Symposium on Principles of Programming Languages.* ACM Press, 1991.

This article was processed using the LaTeX macro package with LLNCS style