Electronic Notes in Theoretical Computer Science ?? (2002)
URL: http://www.elsevier.nl/locate/entcs/volume??.html 11 pages

On the Specification of Higher Dimensional Automata ⋆

Richard Buckland

School of Computer Science and Engineering
University of New South Wales
Sydney, Australia

Michael Johnson

Departments of Mathematics and Computer Science
Macquarie University
Sydney, Australia

Dominic Verity

Division of Information and Communication Sciences
Macquarie University
Sydney, Australia

Abstract

Higher dimensional automata (HDA) have been widely studied as models of concurrent processes. Most current work focuses on developing the directed algebraic topological notions required to analyse HDA to determine their computer scientific properties (deadlock, safety, unreachable states, etc). Instead, this paper is concerned with the software engineering of HDA and details the specification of HDA by process algebra operations. The specifications work for cubical HDA, but are designed to also work for the explicit choice higher dimensional automata (ECHDA) originally proposed by Buckland. We introduce ω-multigraphs, a graphical notion which is easier to use than pasting schemes but more general than cubical complexes, we describe basic process algebra operations on ECHDA as constructions on ω-multigraphs, we discuss the trichotomy of concurrent, conferring, and conflicting choices, and note that the "deadlock choice" can arise from intersecting conflicting choices, and we remark that common software engineering refinements correspond to choice refinement.

⋆ Research partially supported by the Australian Research Council.

1 Introduction

Higher dimensional automata (HDA) are a natural generalization of non-deterministic automata. HDA were introduced by Pratt [14] as an extension of models of concurrent systems based on non-deterministic automata, in order to model also true concurrency. In particular, Pratt needed automata with higher dimensional structure to obtain Birkhoff duality between schedules and automata in the presence of true concurrency.

Since the pioneering work of Goubault [8] HDA have been widely studied. Recent work has focused most on the development of directed algebraic topology which is used for the analysis of HDA to determine important properties such as safety, deadlock, and unreachable states. Examples include further work by Goubault, for example [11] and [10], a series of papers by Gaucher [3], [4], [5], [6], [7] and work by Grandis [12] and Raussen [16]. See also the special issue of *Mathematical Structures in Computer Science*, [9].

This paper is concerned with the specification of HDA via process algebra operations, and to the authors' knowledge it is the first paper to consider this issue in detail.

HDA have been defined variously as n-categories [14], cubical complexes [15], and pasting schemes [4]. In most cases all cells are indeed cubical. So the specification of HDA must be able to deal with operations on some appropriate notion of cubical higher dimensional graph. However, we have designed the operations to work on more general higher dimensional graphs, for the following reason. In [1] the first author argued that *explicit* choices should be incorporated into process models and he proposed a notion which we now call explicit choice higher dimensional automata (ECHDA) [1] based on pasting schemes [13]. While this paper doesn't especially argue the case for ECHDA, it is noteworthy that the extra generality adds little complication to the process algebra operations provided an appropriate notion of higher dimensional graph is used. For the remainder of this paper we will concentrate on ECHDA, but the reader is reminded that HDA are a simple special case of ECHDA, so all that we say applies to HDA too.

Non-deterministic automata may be represented simply as directed graphs in which the nodes are interpreted as states and the edges as events. Since ECHDA (and HDA) make important use of higher dimensional structure they cannot be represented using ordinary directed graphs. One of the principal purposes of this paper is to introduce ω-*multigraphs* which are a relatively simple notion of higher dimensional graph. The ω-multigraphs are used to represent ECHDA in the same way that graphs are used as representations of non-deterministic automata. In addition we show precisely how to construct ω-multigraphs corresponding to basic process algebra terms. Exactly as for non-deterministic automata, the nodes of ω-multigraphs are interpreted as states and the edges are interpreted as events. The higher dimensional faces of ω-multigraphs are used to represent explicit choices

[1] ECHDA is a plural abbreviation. Rather idiosyncratically, when we need to refer to an explicit choice higher dimensional automat*on* we will call it an ECHDon.

and true concurrency.

Interestingly, early drafts of [14] already noted the limitations of pasting schemes and called for a new notion of "n-graph". In some senses ω-multigraphs provide an answer — ω-multigraphs provide an alternative foundation for ECHDA (and hence for HDA, but see also the final paragraph of this paper which points out the main limitation on using ω-multigraphs as a *definition* of HDA).

One of the advantages of the ω-multigraph representation is that the 1-skeleton of an ω-multigraph (the directed graph obtained by taking just the nodes and the 1-dimensional edges) is precisely the usual representation of the non deterministic automaton associated with the system. Thus ω-multigraphs are easily used by software engineers already experienced in the use of non-deterministic automata — the extra structure available in an ω-multigraph is used for the extra specifications (choices and concurrencies) which the engineer can choose to make.

However, ω-multigraphs representing ECHDA can become quite high dimensional and hence difficult to visualize and manipulate. Arguably, when the high dimensionality arises from many processes executing concurrently it is an accurate reflection of the nature of the system. When instead the high dimensionality is a result of nested choices there is an easy solution called *flattening* which reduces the dimensionality of the ω-multigraph. Flattening was suggested in section 6 of [1], but the present paper includes the first full description of a flattening operation. Also in [1] it was suggested that flattening was actually part of the specification process. We now have further experience with ECHDA and a prototype ECHDA CASE tool [2] and it is apparent that for software engineering applications flattening must be employed frequently to reduce complexity. We show in Section 5 how to flatten even in the absence of choice specifications.

In summary ECHDA provide a promising software engineering tool. This paper provides a rigorous mathematical foundation for ECHDA (necessary to continue CASE tool development), shows how the basic process algebra operations are performed on ECHDA, and provides the first formal treatment of the important flattening operation.

2 ω-multigraphs

Pasting schemes provide a foundation for ECHDA, but they have several disadvantages. Firstly they are either unwieldy to define, or they must be defined by inductive construction rather than axiomatically. In either case they take considerable time to introduce in detail. Furthermore it can be quite difficult to prove that a proposed scheme is really a pasting scheme and this makes it hard to define operations on pasting schemes. In this section we define ω-multigraphs, which are a relatively simple, axiomatically described, class of higher dimensional graphs. In the following sections we will define the operations on ω-multigraphs which correspond to sequential composition, selection, concurrency, and flattening.

The ω-multigraphs are not themselves easy to define, since they capture a quite delicate notion of higher dimensional graph. The ω indicates that the multigraphs might be arbitrarily high dimensional (an n-multigraph is n-dimensional).

Definition 2.1 *An* ω-multigraph *is a set X, a subset $\mathcal{U}$ of the power set of X which contains all the singleton subsets, and for each natural number i two functions $s_i, t_i : \mathcal{U} \to \mathcal{U}$ such that*

$$s_i s_j = s_i t_j = s_i \qquad i < j$$

$$t_i s_j = t_i t_j = t_i \qquad i < j$$

$$s_i s_j = t_i s_j = s_j \qquad j \leq i$$

$$s_i t_j = t_i t_j = t_j \qquad j \leq i.$$

Notice that one of each of the pairs of equations is dual to the other, where dual means that it can be obtained from the other by choosing a subscript, say i, and changing all occurrences of s_i to t_i and vice versa. Frequently we will state only half our results, leaving the dual results implicit.

Definition 2.2 *If $s_i\{x\} = \{x\}$, call x an i*-cell.

Lemma 2.3 *An element x is an i cell if and only if $t_i\{x\} = \{x\}$.*

Lemma 2.4 *For any x, $s_i\{x\}$ is an i-cell.*

Definition 2.5 *Write* $\dim x$ *for the least i such that x is an i-cell, and say that x is* $\dim x$-dimensional.

Definition 2.6 *Write* $\dim X$ *for the greatest i such that there exists an $x \in X$ which is i-dimensional, and say that X is* $\dim X$-dimensional.

The above four equations are very basic to the theory and will be referred to as the st-equations. They were first used in the definition of ω-categories in [17]. The letters s and t are chosen as mnemonics for source and target. Each element x of an ω-multigraph will have for each i an i-dimensional source $s_i\{x\}$ and an i-dimensional target $t_i\{x\}$. If x is n-dimensional we will draw it as oriented from its $(n-1)$-dimensional source to its $(n-1)$-dimensional target.

Example 2.7 Here are two 2-dimensional ω-multigraphs:

$$\begin{tikzcd} & q \arrow[dr, "b"] \arrow[d, phantom, "\Downarrow\alpha"] & \\ p \arrow[ur, "a"] \arrow[rr, "x"'] & {} & r \end{tikzcd} \qquad \text{and} \qquad \begin{tikzcd} & q \arrow[r, "u"] \arrow[d, phantom, "\Downarrow\beta"] & r \arrow[dr, "w"] & \\ p \arrow[ur, "v"] \arrow[rrr, "c"'] & {} & & s \end{tikzcd}$$

In the first case α has as 0-dimensional source (0-source) $\{p\}$ and as 1-source $\{a,b\}$, and as 0-target $\{r\}$ and as 1-target $\{x\}$. Similarly x has as 0-source $\{p\}$ and as 0-target $\{r\}$ etc. Likewise, in the second case β has as 1-source $\{u,v,w\}$ and $\{u,v,w\}$ has as 0-source p and as 0-target s etc. Notice that the st-equations are satisfied provided that where a cell z has been drawn as n-dimensional we require it to be an n-cell (ie $s_i\{z\} = \{z\}$ for all $i \geq n$ and dually).

Example 2.8 A multigraph is an ω-multigraph in which every element is a 1-cell: The nodes are the 0-cells, and the multigraph is a graph if and only if $s_0\{x\}$ and $t_0\{x\}$ are singletons for all x.

3 Process algebra operations

The following subsections develop in detail the correspondence between basic process algebra operations and constructions on ω-multigraphs.

Sequential composition

Construction 3.1 *The sequential composition of two ECHDA which are represented by two ω-multigraphs $(X, \mathcal{U}, s_i, t_i)$ and $(Y, \mathcal{V}, s'_i, t'_i)$ is defined when $s_0(X) = t'_0(Y) = z$ say, and X and Y are disjoint apart from z. It has corresponding ω-multigraph given by $(X \cup Y, \mathcal{U} \cup \mathcal{V})$ with inherited source and target functions (notice that $s_i\{z\} = t_i\{z\} = z = s'_i\{z\} = t'_i\{z\}$).*

There may be times during the development of a detailed process specification when one of X or Y will need to be replaced by an isomorphic ω-multigraph in order to achieve the condition $s_0(X) = t'_0(Y)$, or in order to ensure that X and Y are otherwise disjoint. Usually we will pass over such modifications without comment. If we do need to comment we will call them *coherence relabellings*.

Selection (choice)

In non-deterministic automata, represented as directed graphs, a choice between two processes X and Y at state s is represented by an outgoing branch at s with X being one path of the branch, and Y the other. In ECHDA the space between the paths is labelled with a 2-dimensional cell which represents the choice condition used to determine whether X or Y occurs. The choice condition may be an ordinary **if ... else ...** statement, a probabilistically specified choice, an explicitly non-deterministic internal or external choice, an `always` choice (if `FALSE` X else Y — always choose Y), a `never` choice (if `TRUE` X else Y — never choose Y), etc. For example, the first ω-multigraph shown in Example 2.7 would represent the choice between the process a then b, and the process x, with the choice condition specified by the cell α (whose details are not shown, in the same way that the detail of event a is not shown).

Three aspects of the treatment of choice should especially be noted.

Firstly, the choice condition is represented by a directed cell. In most, but not all, cases choice is asymmetric — if B then X else Y is in general different from if B then Y else X, and in ECHDA represented as ω-multigraphs one can be obtained from the other by reversing the direction of the choice condition.

Secondly ECHDA are strongly dualizable. Few people would argue that in order to choose between X and Y there must be a state s from which both X and Y can occur. Frequently in specifications X is built, Y is built, and then although they appear to begin in different states a coherence relabelling is used (often implicitly) to give them a common start state. Alternatively, a new start state s is introduced and two *null-transitions* are introduced to connect s with the start state of X and of Y respectively. In ECHDA we dually require X and Y to have a common final state, either by coherence relabelling, or by introducing a new state t and two null-transitions from the final state of X and of Y respectively to t. Thus choice conditions are always cells bounded by processes like those shown in Example 2.7. They are never just open branches as often appear in non-deterministic automata. Note that this does not introduce a semantic distinction between choices in ECHDA and choices in non-deterministic automata — it makes no difference, and corresponds better with elementary programming practice, to have a state t which represents the completion of (say) the statement if B then X else Y and which occurs after the final state of X and after the final state of Y.

Thirdly, from what we have said already it is clear that a process viewed as an ECHDon might be 2-dimensional, in contrast with processes in non-deterministic automata. If we want to be able to introduce choice cells between arbitrary processes we will need to make a definition which allows for example for a choice between two 2-dimensional ECHDA, yeilding a 3-dimensional ECHDon, or between a 1-dimensional ECHDon and a 3-dimensional ECHDon yielding a 4-dimensional ECHDon, and so on.

Construction 3.2 *A selection P between two ECHDA represented by ω-multigraphs $(X,\mathcal{U},s_i,t_i)$ and $(Y,\mathcal{V},s'_i,t'_i)$ is defined when X and Y are disjoint and when the one of highest dimension is n-dimensional. The corresponding ω-multigraph is constructed as follows. It has as elements $X\cup Y$ together with a new $(n+1)$-dimensional element p, distinct new elements $s_i\{p\}$ and $t_i\{p\}$ for $0\le i<n$, and for each i with $0<i\le n$ four new choice elements*

- *An i-dimensional* `always` *choice C_{i1} with $s_{i-1}\{C_{i1}\}=s_{i-1}\{p\}$ and $t_{i-1}\{C_{i1}\}=s_{i-1}\{X\}$*
- *An i-dimensional* `never` *choice C_{i2} with $t_{i-1}\{C_{i2}\}=t_{i-1}\{p\}$ and $s_{i-1}\{C_{i2}\}=t_{i-1}\{X\}$*
- *An i-dimensional* `always` *choice C_{i3} with $s_{i-1}\{C_{i3}\}=s_{i-1}\{p\}$ and $t_{i-1}\{C_{i3}\}=s'_{i-1}\{Y\}$*
- *An i-dimensional* `never` *choice C_{i4} with $t_{i-1}\{C_{i4}\}=t_{i-1}\{p\}$ and $s_{i-1}\{C_{i4}\}=t'_{i-1}\{Y\}$*

The elements of $X\cup Y\cup\{p\}$ are called executable *to distinguish them from the merely* structural elements *which are introduced in this construction. In the above formulas we have elided some structural elements from the right hand sides involving X and Y for clarity.*

Finally the new $(n+1)$-dimensional element has associated with it the choice condition *P which specifies the circumstances under which its n-source or n-target occur. These choice conditions may be ordinary* **if ... else ...** *statements, probabilistically specified choices, explicitly non-deterministic internal or external choices* *etc.*

Perhaps this warrants some further explanation: Intuitively, the structural elements of the form $s_i\{p\}$ are the "new start states" (see above) at each dimension, the structural elements of the form $t_i\{p\}$ are the "new final states" (see above) at each dimension, and the C_{ik} are the "null-transitions" at each dimension (four of them because there is a start transition and an end transition for each of X and Y).

Concurrency

Concurrency is more easily expressed. It is in fact the topological tensor product of the ECHDA and can be described in the same way as the topological tensor product of pasting schemes first introduced by Street and Johnson. (But beware, it is still delicate — Johnson and Pratt showed that the pasting schemes defined in [13] are *not* closed under this tensor product. This fact was overlooked for some time because the smallest counterexample is six dimensional).

Construction 3.3 *If $(X, \mathcal{U})$ and $(Y, \mathcal{V})$ are ω-multigraphs then the ω-multigraph corresponding to their concurrent composition is given by $(X \times Y, \sum(\mathcal{U} \times \mathcal{V}))$ with*

$$s_i(U,V) = \sum_{j+k=i} s_j U \times s_k V \qquad i \text{ even}$$

and

$$s_i(U,V) = \sum_{j+k=i} s_j U \times t_k V \qquad i \text{ odd}$$

and dually.

These formulas summarize some particularly beautiful combinatorial mathematics and are well worth studying.

4 A trichotomy of choices

The reader will have noted the "etc." at the end of phrases like

> choice conditions may be ordinary **if ... else ...** statements, probabilistically specified choices, explicitly non-deterministic internal or external choices etc.

ECHDA are a model for recording choice information as well as state and transition information, but they make no restrictions on the kinds of choices that can appear. A choice condition may be a precisely specified deterministic choice, or at the other extreme it might merely indicate during a specification that there is a choice between two processes, but the details of how that choice will be made will only be determined later in the specification process.

In fact, in our treatment of ECHDA we view choice even more widely. Historically, the concurrent execution of a and b has sometimes been treated as a choice between interleavings ab and ba. We are happy to view *both* of these as a kind of choice — the first is sometimes referred to as the *concurrent* choice between a and *b* while the second is usually a standard non-deterministic choice. In other words, we are happy to treat the cubical cells of HDA as "choices" with similar status to the other choice cells in ECHDA.

Furthermore, the practice of software engineering can to some extent be viewed as choice refinement. In constructing systems we begin with several outcomes, and gradually provide more and more information about how to choose between those outcomes until we have an implemented system. This view is strengthened if we permit a "choice" between *a* and *b* to include the concurrent choice — we *choose* to do *both a* and *b*, and we explicitly decline to distinguish many possible orderings of executions of parts of *a* and *b* (if they're not viewed as atomic).

But there is an interesting aspect of refinement in the development of concurrent systems. Frequently processes are constructed independently, and then specified as executing concurrently. Only later, as the specification becomes more detailed, are synchronization points, semaphores, or other mutual exclusions introduced between the processes. In HDA the concurrent processes are a cube, and when the synchronizations or mutual exclusions are added the cube is eroded as holes appear which would have contained forbidden traces. This is an evocative metaphor, which we don't wish to lose.

However, if we allow yet another choice condition we can view the erosion as choice refinement. The *conflict* choice between *a* and *b* is viewed as a form of the concurrent choice between *a* and *b* — both *a* and *b* will occur, but with mutual exclusion. So we have refined our concurrent choice between *a* and *b* to explicitly exclude all but two possible orderings of the execution of *a* and *b*. In HDA the conflict choice would be represented as a hole. In ECHDA it can still be represented as a hole, or it can have a choice cell which explicitly says that traces inside the square are prohibited by mutual exclusion. ECHDA, like HDA, support true concurrency, but they also support classical interleaved or explicitly mutually exclusive concurrency.

In summary, we sometimes classify ECHDA choices in a trichotomy

(i) *Concurring choices* in which, at least until further refinement takes place, both processes execute in any temporal order (explicitly including true concurrency sometimes viewed as the inside of the cell)

(ii) *Conflicting choices* in which, usually unchanged by refinement, both processes execute, but in mutual exclusion (so the choice determines which side of the boundary of the cell will occur, and once that initial choice is made the temporal order of the processes is fully determined)

(iii) *Conferring choices* in which, as a result of some "conferring" whether with an external actor, internal state, an oracle, or whatever, a choice is made between two processes. This third is the traditional notion of choice (if then else, etc).

Of course conflicting choices are a special case of conferring choices, but their important properties warrant distinguishing them: We will see below that the interaction between types of choices has important effects during the specification and analysis of systems. The interaction of conflicting choices can be particularly dramatic leading as it might to deadlock (recall the famous Swiss flag example which occurs in many papers including for example [3]).

5 Flattening

If an ω-multigraph is constructed from some given basic processes (1-cells) by repeated sequential compositions and choices it will contain only a single executable i-dimensional element in each i-source and i-target. Flattening is a process which reduces dimension by increasing the number of executable elements occurring within a source or target. We must therefore remark upon how to interpret such sources or targets.

How do we interpret the composition of choices (cells of dimension greater than 1) with common start state? These are treated in the same way as a composition of events $e : S \to T$ and $f : T \to U$ in which f only has the opportunity to occur if we have already reached state T. Similarly in

$$\begin{tikzcd} S \arrow[rr, bend left=60, "e_1"] \arrow[rr, "e_2"] \arrow[rr, bend right=60, "e_3"'] \arrow[rr, phantom, bend left=30, "\Downarrow\alpha"] \arrow[rr, phantom, bend right=30, "\Downarrow\beta"] & & T \end{tikzcd}$$

in state S, β only has the opportunity to choose between e_2 and e_3 if α has already chosen that e_1 will *not* occur.

For $n > 2$ the flattening of an n-dimensional ω-multigraph will be an $(n-1)$-dimensional ω-multigraph whose $(n-1)$-dimensional elements consist of the $(n-1)$-dimensional elements of the ω-multigraph, together with new $(n-1)$-dimensional elements corresponding to the n-dimensional elements of the ω-multigraph. For each q of dimension n, the $(n-1)$-dimensional elements in $s_{n-1}\{q\}$ are reinterpreted as the disjunction of their choice conditions with the choice condition of q, and the $(n-1)$-dimensional elements of $t_{n-1}\{q\}$ are reinterpreted as the conjunction of their choice conditions with the choice condition of q. The new $(n-1)$-dimensional element corresponding to q has as $(n-2)$-source $t_{n-2}\{q\}$ and as $(n-2)$-target $s_{n-2}\{q\}$.

It is interesting to note that while this process would seem to introduce loops we show in a paper in preparation that in all practical situations these loops will be eliminated because they involve unexecutable traces.

6 Future work

Now that we have full formal definitions of the process algebraic operations for ω-multigraphs, we can further develop the CASE tool described in [2]. Meanwhile, there are other important theoretical issues:

(i) ECHDA have been designed to work with whatever choice conditions a software engineer chooses to use. However, in order to implement flattening we need to specify the interaction of choices under conjunction and disjunction and for some choice conditions this interaction is quite delicate. For example, when q above has choice conditions which are not deterministic it is important that we only evaluate those conditions once and substitute them wherever they are required. We have catalogued an algebra and logic of choices, but we need to extend it to include as wide a range of choice condition types as possible.

(ii) The use of flattening at different stages during a specification and refinement cycle generates an equivalence on ω-multigraphs which might be thought of as a kind of bisimulation equivalence. We have only just begun to explore the detailed properties of this equivalence.

Finally, we should note that ω-multigraphs are not the perfect notion of n-graph because they permit "pathological" behaviour which has no standard interpretation in our algebra of processes. Instead, ω-multigraphs provide a framework in which we can easily represent ECHDA and their operations. If ever we are satisfied that we have captured all appropriate operations on ECHDA then the collection of ω-multigraphs which can be constructed from basic "cells" using those operations will presumably be the sought after class of n-graphs. The test for this is a certain freeness property which will be adumbrated elsewhere.

References

[1] R. Buckland, Choice as a first class citizen, in *Intensional Programming I*, eds Orgun and Ashcroft, (1996) 249–259, World Scientific.

[2] R. Buckland and M. Johnson, ECHIDNA: A system for manipulating explicit choice higher dimensional automata, AMAST'96, Lecture Notes in Computer Science **1101** (1996) 587–591.

[3] P. Gaucher, From concurrency to algebraic topology, *ENTCS*, **39** (2000).

[4] P. Gaucher, Homotopy invariants of higher dimensional categories and concurrency in computer science, *Mathematical Structures in Computer Science*, **10** (2000) 481–524.

[5] P. Gaucher, Combinatorics of branchings in higher dimensional automata, *Theory and Applications of Categories*, **8** (2001) 324–376.

[6] P. Gaucher, Investigating the algebraic structure of dihomotopy types, *ENTCS*, **52** number 2, to appear.

[7] P. Gaucher, About the globular homology of higher dimensional automata, *Cahiers de Topologie et Geometrie Differentielle Categorique*, (2002), to appear.

[8] E. Goubault, The Geometry of Concurrency, PhD thesis, ENS, 1995.

[9] E. Goubault (ed), Geometry and concurrency, special issue of *Mathematical Structures in Computer Science* **10** (2000) 409–573.

[10] E. Goubault, L. Fajstrup and M. Raussen, Detecting Deadlocks in Concurrent Systems, CONCUR'98 Lecture Notes in Computer Science **1466** (1998) 332–347.

[11] E. Goubault and M. Raussen, Dihomotopy as a tool in state space analysis, LATIN'02, Lecture notes in Computer Science **2286** (2002) 16–37.

[12] M. Grandis, Directed homotopy theory I: The fundamental category, preprint, 2001.

[13] M. Johnson, The combinatorics of n-categorical pasting, *Journal of Pure and Applied Algebra* **62** (1989) 211–225.

[14] V. Pratt, Modelling concurrency with geometry, *Proceedings of POPL 18*, ACM Press, (1991) 311–322.

[15] V. Pratt, Transition and cancellation in concurrency and branching time, *Mathematical Structures in Computer Science* (2002) to appear.

[16] M. Raussen, On the classification of dipaths in geometric models for concurrency, *Mathematical Structures in Computer Science* **10** (2000) 427–457.

[17] R.H. Street, The algebra of oriented simplexes, *Journal of Pure and Applied Algebra* **49** (1987) 283–335.