Explicit choice higher dimensional automata, ω-multigraphs, and process algebra operations

Richard Buckland and Michael Johnson *

School of Mathematics and Computing, Macquarie University
AUSTRALIA
email: `mike@mpce.mq.edu.au`
phone: +612 850 9583, fax: +612 850 9551

Abstract

Explicit choice higher dimensional automata (ECHDA) have been proposed as models for concurrent processes because of their advantages for software engineering. These advantages include a full treatment of true concurrency, the provision of a consistent model throughout the development process, a natural partitioning of development processes, and the fact that their one dimensional skeletons correspond to non-deterministic automata and so are already familiar to many software engineers. The provision of ECHDA CASE tools depends upon a rigorous treatment of the connections between process algebra and ECHDA. This paper introduces ω-multigraphs, shows how they are used to represent ECHDA, and uses them to provide a rigorous treatment of the process algebra operations for ECHDA. In addition we provide a rigorous treatment of a dimension reduction operation which is important for software and specification visualization.

Topics and Keywords: Non-determinism, concurrency, software refinement, process model, process algebra, parallel programming, program verification, program visualization.

*The authors gratefully acknowledge the Microsoft Research Institute (MRI) and the Australian Research Council (ARC).

1 Introduction

Non-deterministic automata are widely used as models of concurrent systems (for a recent brief introduction see [5]). Vaughan Pratt has argued [6] that higher dimensional automata (HDA) provide a better model since they do not abstract away true concurrency. In his contribution [1] to the book *Intensional Programming I*, Richard Buckland 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) based on pasting schemes [4].

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 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. We show how ω-multigraphs represent ECHDA in the same way that graphs are used as representations of processes in non-deterministic automata and other process algebras. In addition we show precisely how to construct ω-multigraphs corresponding to terms in process algebras. 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 and true concurrency.

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 trained 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 4 how to flatten even in the absence of choice specifications.

In summary ECHDA provide a new and promising software engineering tool. This paper provides a rigorous mathematical foundation for ECHDA (necessary to continue CASE tool development), shows how the basic process algebraic 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 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

$$\begin{aligned} s_i s_j = s_i t_j = s_i & \qquad i < j \\ t_i s_j = t_i t_j = t_i & \qquad i < j \\ s_i s_j = t_i s_j = s_j & \qquad j \leq i \\ s_i t_j = t_i t_j = t_j & \qquad j \leq i. \end{aligned}$$

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 reader to state the dual results.

Definition 2 If $s_i\{x\} = \{x\}$, call x an i-cell

Lemma 3 *An element x is an i cell if and only if $t_i\{x\} = \{x\}$.*

Lemma 4 *For any x, $s_i\{x\}$ is an i-cell.*

Definition 5 Write $\dim x$ for the least i such that x is an i-cell, and say that x is $\dim x$-dimensional.

Definition 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 [7]. 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 7 Here are two ω-multigraphs:

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

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 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

We now present the important technical results.

Sequential composition

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 $s_i\{z\} = t_i\{z\} = z = s'_i\{z\} = t'_i\{z\}$).

Selection (choice)

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 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 $0 < i < n$, and new elements with choice conditions labelled "always" and "never" (as described in [1]). 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.

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.

Concurrency

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 summarise some particularly beautiful combinatorial mathematics and are well worth studying.

4 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

$$\begin{tikzcd}
S \arrow[rr, bend left=60, "e_1"] \arrow[rr, "e_2"] \arrow[rr, bend right=60, "e_3"'] \arrow[rr, phantom, shift left=4, "\Downarrow\alpha"] \arrow[rr, phantom, shift right=4, "\Downarrow\beta"] & & T
\end{tikzcd}$$

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.

5 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:

1. 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.

2. 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.

References

[1] R. Buckland, Choice as a first class citizen, in *Intensional Programming I*, eds Ashcroft and Orgun, 1996, to appear.

[2] R. Buckland and M. Johnson, ECHIDNA: A system for manipulating explicit choice higher dimensional automata, AMAST'96 (Algebraic Methodology and Software Technology), Lecture Notes in Computer Science, to appear.

[3] R. Buckland and M. Johnson, On a tensor product of schemes. in preparation, (1996).

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

[5] F. Moller and D.A. Smolka, On the computational complexity of bisimulation, *Computing Surveys*, **27** (1995) 287–289.

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

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