AAECC (2015) 26:151–164
DOI 10.1007/s00200-014-0243-2

ORIGINAL PAPER

Computing path categories of finite directed cubical complexes

Michael D. Misamore

Received: 20 February 2014 / Revised: 23 April 2014 / Accepted: 26 July 2014 /
Published online: 30 December 2014

Abstract A new method is introduced for simplifying finite directed cubical complexes. This construction is effective in the sense that it can be computed relatively efficiently, and it induces a fully faithful inclusion on path categories. The resulting parallelized family of algorithms enables calculations of morphism sets of path categories of such complexes in many cases of practical interest, and working code is provided as a proof of concept.

Keywords Directed topology · Path category · Simplicial sets

Mathematics Subject Classification Primary 18G30; Secondary 0304

1 Introduction

In [5], Pratt suggested using finite directed cubical complexes as models for higher dimensional automata as an application to concurrency theory: in this framework, computational processes which compete for common resources may not be able to enter certain sets of states simultaneously, and these "forbidden regions" determine "holes" in the directed state space. The problem, then, is usually to classify the possible paths between two given system states which successfully navigate around all such holes, where two such paths are considered equivalent whenever there exist directed homotopies between them. This leads to the study of the path categories $P(|X|)$ of triangulations $|X|$ of finite directed cubical complexes X. Path categories are not standard homotopy invariants [3], but they can nevertheless be analyzed with homotopical tools. In particular, it becomes valuable to know when one can replace X with a subcomplex Y such that the inclusion $Y \subset X$ induces a fully faithful inclusion

M. D. Misamore (✉)
University of Western Ontario, London, Canada
e-mail: m.misamore@gmail.com

$P(|Y|) \hookrightarrow P(|X|)$. Theorem 1 here gives a practical criterion for when this is possible, and it has led the present author to develop software that applies this technique to a significant class of examples. In combination with the previous software implementations of the techniques developed in [3], it is now feasible to easily analyze the behavior of path categories for a large class of finite directed cubical spaces of interest, including many of higher genera.

This paper is organized as follows: the necessary background from homotopy theory is covered in Sect. 2, followed by the theoretical results leading to Theorem 1 in Sect. 3. The development and analysis of some practical algorithms to implement these ideas is discussed in Sect. 4, including results that lead to a parallelizable algorithm for detecting corner vertices (Proposition 2) and an explicit identification of the potential new corner vertices that can occur when such a corner vertex is removed (Proposition 5). The last section (Sect. 5) describes the software implementation of these ideas and the practical accomplishments achieved with it thus far. Of course, concurrency models are not the only possible application for these techniques: finite directed cubical state spaces have potential applications to all locally partially ordered dynamical systems, such as those arising in motion planning problems where it is of interest to classify the possible means of avoiding obstacles. Finally, it is worth pointing out that other approaches exist that attempt to simplify directed cubical complexes while preserving the entire directed homotopy type: [4] is a recent example of what can be done so far in this direction, at least in two dimensions.

2 Background

In posets, a *standard cubical cell* $\mathcal{P}(\underline{n})$ of dimension $n \geq 0$ consists of the poset of subsets of the finite set $\underline{n} := \{1, \ldots, n\}$. For any integer $N \geq 0$, the *ambient directed cubical N-space* C^N given by the product $(\mathbb{N} \cup \{0\})^N$ is also naturally a poset, where $(x_1, \ldots, x_N) \leq (y_1, \ldots, y_N)$ whenever $x_i \leq y_i$ for $1 \leq i \leq N$; observe that this is not a total ordering unless $N = 1$. A *directed (cubical) cell* of dimension n in ambient cubical N-space consists of an injective map of posets $c : \mathcal{P}(\underline{n}) \to C^N$ such that $c(\underline{n}) - c(\emptyset) = (1, \ldots, 1)$ on its image; observe that this is automatically a full and faithful embedding at the level of categories so one may conflate c with its image in C^N. These are the objects of a category $\mathcal{P}/C^N$ whose morphisms are commutative triangles

$$\begin{array}{ccc} \mathcal{P}(\underline{n}) & \xrightarrow{\theta} & \mathcal{P}(\underline{m}) \\ & {\scriptstyle c}\searrow \quad \swarrow{\scriptstyle d} & \\ & C^N & \end{array}$$

where θ is injective since c is injective.

A cell c is a *subcell* of a cell d exactly when there is a morphism $c \to d$. A *finite directed cubical complex* X in C^N for some $N \geq 0$ consists of a finite set of directed cells in C^N which is closed under taking subcells. The present definition of finite directed cubical complexes is preferred to that given in [3] (where the ambient space is $\square^N$) since in practical applications to concurrency models it is preferable for each axis of the ambient space to extend along more than a single unit. In principle this can be emulated within a single cubical cell: for example, modeling a 10-dimensional

finite directed cubical complex with 7 units along each axis would require an ambient cell $\square^N$ with exponent $N \geq 10 \cdot \log_2(8) = 30$. This is equivalent to the number of bits required to represent the corresponding subcomplex in C^{10} with $|C| = 8$, but the embeddings for the latter representation are much more direct to realize in software.

A *top-cell c* of such a complex X is a cell which is contained in no other subcell; top-cells are significant in practice because their presence implies the presence of all of their constituent subcells, thus for algorithms it often suffices to store only the top-cells of a cubical complex at any given point in time, thereby saving a significant amount of memory. Observe that top-cells need not be all of the same dimension, and that every cell in a finite complex is contained in some (not necessarily unique) top-cell.

The simplicial nerve $B\mathcal{P}(\underline{n})$ is usually denoted $\square^n$ and called the *standard cubical n-cell*; its poset $N\mathcal{P}(\underline{n})$ of nondegenerate simplices is just $\mathcal{P}(\underline{n})$ again via the definition of simplicial nondegeneracy. The *triangulation* $|\square^n|$ of $\square^n$ is given by the product $(\Delta^1)^n$ of n copies of the standard 1-simplex. The k-simplices $\Delta^k \to (\Delta^1)^n$ correspond to functors $\underline{k} \to \underline{1}^n$ and from this one can immediately see that certain "diagonal maps" are included among them, yielding additional non-cubical 1-simplices in particular. One may define the triangulation $|Y|$ of an arbitrary cubical set Y via

$$|Y| := \varinjlim_{\square^n \to Y} |\square^n|$$

where the colimit is taken over the cubical cell category of Y (see §3, [2] for a modern introduction to cubical sets).

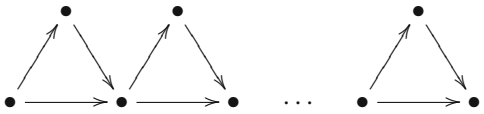
Any simplicial set X has a *path category* $P(X)$ which is generated by the free graph on the vertices and edges of X subject to the relations $d_1(\sigma) = d_0(\sigma) \circ d_2(\sigma)$ for all 2-simplices σ of X; in particular every path $a \rightsquigarrow b$ in X is represented by a uniquely determined path in NX, the non-degenerate simplices of X. Thus $\mathcal{P}(X) = \mathcal{P}(NX)$ where the latter is the free graph on the poset NX subject to the above relations (after observing that degenerate 2-simplices do not generate any non-trivial relations). For a cubical set Y, define $P_\square(Y) := P(|Y|)$, the path category of the triangulation of Y. It is known (Lemma 1.1, [3]) that the cubical 2-skeleton $\mathrm{sk}_2(Y)$ completely determines $\mathcal{P}_\square(Y)$, but on the other hand every non-degenerate 2-simplex of $|\mathrm{sk}_2(Y)|$ factors through the triangulation of some cubical 2-cell (since it has to come from somewhere), so that $\mathcal{P}_\square(Y)$ is characterized as the free graph on the vertices and edges of Y subject to the relations generated by the 2-cells of Y. In the case when Y is a finite directed cubical complex, every cell is non-degenerate (has distinct vertices) so the equivalence relation is generated by non-degenerate 2-cells; two paths $f, g : a \rightsquigarrow b$ in Y will be called *homotopic* in what follows if $[f] = [g]$ in $\mathcal{P}_\square(Y)$.

In posets, an *n-simplex* in ambient directed N-space for $N \geq n \geq 0$ consists of an injective map $\underline{n} \to \underline{N}$ where the poset ordering on each is just the natural number ordering. There is again a category of such objects over $\underline{N}$ where the morphisms are commutative triangles and the morphisms $\underline{m} \to \underline{n}$ over $\underline{N}$ are automatically inclusions, which define *subsimplices*. Simplices are uniquely determined by the images of the objects so that one usually conflates an n-simplex x with its list $(x_0, \ldots, x_n)$ of image vertices in $\underline{N}$. A *finite directed simplicial complex* in $\underline{N}$ then consists of a finite set of simplices in $\underline{N}$ which is closed under taking subsimplices.

In practice, the vertices of a finite directed cubical complex Y in C^N can be taken to consist of tuples $(x_1, \ldots, x_N)$ of natural numbers where $x_i \geq 0$. These tuples admit a *lexicographic ordering* $\leq_l$ which is a total ordering with the property that if $(x_1, \ldots, x_N) \leq (y_1, \ldots, y_N)$ in the usual cubical poset ordering then the same is true for $\leq_l$. Taking this total ordering on the vertices $v(Y)$ of Y thereby induces a poset injection $(v(Y), \leq_l) \hookrightarrow \underline{M}$ by bijectively mapping the total ordering on $v(Y)$ to $\{0, \ldots, M\}$ for sufficiently large M. The non-degenerate simplices of $|Y|$ then lift to representations in $\underline{M}$ so that if Y is finite directed then $|Y|$ can be regarded as a genuine finite directed simplicial complex.

3 Avoiding vertices

Computing the path category $P(X)$ of an arbitrary finite directed simplicial complex is already possible in software (see [3]), but the combinatorial explosion of possible paths through such a complex limits the applicability of such algorithms to those problems in which the complexes are relatively small. For example, the path category of a large wedge sum of $\partial\Delta^2$ such as



is clearly intractable to compute given enough summands. In general, the problem seems to be that knowledge of $P(a, b)$ together with knowledge of $P(b, c)$ is generally insufficient to efficiently compute $P(a, c)$, as in particular there may be non-local "jumps" defined by 2-simplices which are not considered in the computation of the former two sets. This prevents naïve local-to-global approaches from working.

One does know, however, that in practice many of these problems come from triangulations of finite directed cubical sets, as in concurrency problems (e.g. [5]). Unlike directed simplicial complexes, directed cubical complexes come with an obvious notion of locality: edges are uniformly represented as unit increments along single coordinates, and cubical cells are therefore intrinsically bounded in space. It is therefore reasonable to ask if there is a method to directly reduce the complexity of such a complex in a way that preserves $P(a, b)$ for any particular pair (a, b) of interest, ideally via a modification which is "local" in the sense just described. More specifically, given a finite directed cubical complex X and a pair (a, b) of vertices of X, when is it possible to find a proper subcomplex $Y \subset X$ containing a and b such that the induced map $P(|Y|)(a, b) \to P(|X|)(a, b)$ is a bijection?

One answer to this question (communicated to the present author by Rick Jardine) is that any pushout of the form

$$\begin{array}{ccc} \partial\square^n & \longrightarrow & Y \\ \downarrow & & \downarrow \\ \square^n & \longrightarrow & X \end{array}$$

where the left vertical arrow is the standard inclusion induces an isomorphism $P(|Y|) \cong P(|X|)$ of path categories if $n \geq 3$. An easy consequence is that $P_{\square}(X) \cong P_{\square}(\mathrm{sk}_2(X))$ so that the path category is determined by the (cubical) 2-skeleton. However, by passing immediately to the 2-skeleton one loses opportunities to simplify the complex: the triangulation of the directed cubical 2-torus does not contain any 2-simplices which can be replaced by 1-horns Λ^2_1, for example.

Instead, one could read the above result backwards: it does not *hurt* to include n-cells for $n \geq 3$, so they may as well be considered wherever possible. Such a complex can be simplified using a new technique which begins with the following observation:

Lemma 1 *Suppose X is a finite directed cubical complex and let $x \in X$ be a vertex which is contained in a unique top-cell. Then any path $f : a \rightsquigarrow b$ passing through x such that $x \notin \{a, b\}$ is homotopic to a path $g : a \rightsquigarrow b$ which does not pass through x. Furthermore, this homotopy of paths is uniquely determined by f and x.*

Proof Any non-degenerate path through x factors into the form

$$a \rightsquigarrow s \to x \to t \rightsquigarrow b$$

where $a \leq s < x < t \leq b$. Since x is contained in a unique top-cell by assumption, the 1-cells $s \to x$ and $x \to t$ must belong to some common 2-cell. Taking the local poset representation where $s = \emptyset$, $x = \{v_1\}$ and $t = \{v_1, v_2\}$, one has a sequence of the form

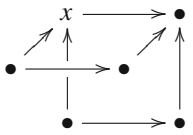
$$\emptyset \to \{v_1\} \to \{v_1, v_2\}$$

which is homotopic to the composite

$$\emptyset \to \{v_2\} \to \{v_1, v_2\}$$

which avoids $x = \{v_1\}$. This representation makes it evident that the homotopy is uniquely determined by f and x. □

There exist examples of cubical complexes X where some vertex $x \in X$ has the property that every non-degenerate path $a \rightsquigarrow b$ through x ($x \notin \{a, b\}$) factors through a 2-cell but x is not contained in a unique top-cell, e.g.



The condition of the Lemma is therefore sufficent but not necessary, but using the unique top-cell criterion nevertheless gives a good geometric interpretation for what is happening.

Lemma 2 *Suppose $K_x \subset |\square^n|$ is the full subcomplex supported on all vertices of $|\square^n|$ except some $x \neq \underline{n}$. Then there is a simplicial homotopy equivalence $K_x \simeq |\square^n|$ and therefore homotopy equivalences $P(K_x) \simeq P(|\square^n|) \simeq *$.*

Proof Observe that any k-simplex of $|\square^n|$ or K_x may be thought of as a flag

$$y_0 \overset{\subseteq}{\to} y_1 \overset{\subseteq}{\to} \cdots \overset{\subseteq}{\to} y_k$$

where the y_i are subsets of $\underline{n}$ (none of which equal x in the case of K_x). The picture

$$\begin{array}{ccccccc} y_0 & \longrightarrow & y_1 & \longrightarrow & \cdots & \longrightarrow & y_k \\ \downarrow & & \downarrow & & & & \downarrow \\ \underline{n} & \xrightarrow{1} & \underline{n} & \xrightarrow{1} & \cdots & \xrightarrow{1} & \underline{n} \end{array}$$

describes a simplicial homotopy sending every such simplex to the vertex $\underline{n}$. The vertical maps exist in K_x since either $y_0 \neq \emptyset$ in which case $[y_i, \underline{n}]$ belongs to $\partial|\square^n|$ or else $y_0 = \emptyset$ in which case $\emptyset \to \underline{n}$ belongs to K_x since it is supported away from x. The latter statement follows from the fact that P sends simplicial homotopies to natural transformations of functors. □

In fact these homotopy equivalences are manifestly via "directed" homotopies in the sense that they respect the poset orderings on vertices. The inclusions $K_x \subset \square^n$ are *not* equivalences on the path categories since they are not essentially surjective; in particular, the free category on $\square^n$ itself contains no nontrivial isomorphisms so x cannot be isomorphic to an object in the free category on K_x. Nevertheless, one has

Lemma 3 *Suppose $K_x \subset |\square^n|$ is the full subcomplex supported on all vertices of $|\square^n|$ except for $x \in |\square^n|$ ($x = \underline{n}$ now permitted). Then the induced functors*

$$i_* : P(K_x) \to P(|\square^n|)$$

are fully faithful.

Proof If $P(|\square^n|)(a, b) = \emptyset$ for $x \notin \{a, b\}$ then certainly $P(K_x)(a, b) = \emptyset$ since K_x is obtained from $\square^n$ by removing cells, thus i_* is trivially bijective on hom-sets in this case. Otherwise $P(|\square^n|)(a, b) = *$ so it suffices to show that $P(K_x)(a, b) = *$. If $P(|\square^n|)(a, b) \neq \emptyset$ then $P(K_x)(a, b) \neq \emptyset$ since any path $a \rightsquigarrow b$ in $|\square^n|$ is equivalent to one in $\square^n$ and by Lemma 1 any such path can always avoid x up to homotopy. Therefore it suffices to show that any two paths in K_x from a to b are homotopic.

Suppose $a \neq \emptyset$ or $b \neq \underline{n}$. Then a path $a \rightsquigarrow b$ in K_x lies in $\partial|\square^n|$ so that the flag maps of the form

$$\begin{array}{ccccccc} a & \longrightarrow & y_1 & \longrightarrow & \cdots & \longrightarrow & y_n & \longrightarrow & b \\ \downarrow & & \downarrow & & & & \downarrow & & \downarrow 1 \\ a & \longrightarrow & b & \xrightarrow{1} & \cdots & \xrightarrow{1} & b & \xrightarrow{1} & b \end{array}$$

establish that any two paths $f, g : a \rightsquigarrow b$ are homotopic in K_x. Finally, observe that any two non-degenerate paths $f, g : \emptyset \rightsquigarrow \underline{n}$ in K_x are homotopic to the unique path $\emptyset \to \underline{n}$ in K_x whenever $\emptyset, \underline{n} \in K_x$. □

Given a vertex $x \in \square^n$, the full *cubical* subcomplex $L_x \subset \square^n$ supported away from x is closely related to K_x: every path of K_x except possibly $\emptyset \to \underline{n}$ belongs to $\partial|\square^n|$ so that if $x \in \{\emptyset, \underline{n}\}$ one has $|L_x| = K_x$.

Lemma 4 *The inclusion $j : |L_x| \subseteq K_x$ induces an isomorphism*

$$j_* : P(|L_x|) \cong P(K_x)$$

of path categories. In particular, j_* is fully faithful so the composite inclusion $|L_x| \subseteq K_x \subset \square^n$ induces a fully faithful functor on path categories.

Proof It suffices to assume that $x \notin \{\emptyset, \underline{n}\}$ since otherwise the result is trivial. Consider any path $a \rightsquigarrow b$ in K_x and observe that this path also belongs to $|L_x|$ unless it equals $\emptyset \to \underline{n}$, in which case there is some vertex $y \neq x$ (since $n \geq 2$ in this case) and a homotopy

$$\begin{array}{ccccc} \emptyset & \longrightarrow & y & \longrightarrow & \underline{n} \\ \downarrow & & \downarrow & & \downarrow \\ \emptyset & \longrightarrow & \underline{n} & \longrightarrow & \underline{n} \end{array}$$

showing that $\emptyset \to \underline{n}$ is homotopic to a path in $|L_x|$; the maps $P(|L_x|)(a, b) \to P(K_x)(a, b)$ are therefore surjective. Any two paths $a \rightsquigarrow b$ in $|L_x|$ are also homotopic since they factor through $\partial|\square^n|$ and are therefore both homotopic to the path $a \to b$, so the map on path sets is also injective, hence bijective. □

The following result is the basis for an algorithm:

Theorem 1 *Suppose there is a pushout*

$$\begin{array}{ccc} L_x & \longrightarrow & Y \\ \downarrow & & \downarrow i \\ \square^n & \xrightarrow{c} & X \end{array}$$

of finite directed cubical complexes where c is a (non-degenerate) top-cell of X, $L_x \subset \square^n$ is the full subcomplex of $\square^n$ on all vertices except some $x \in \square^n$, and $Y \subset X$ is the full subcomplex of X supported on all vertices except x. Then the induced functor

$$i_* : P(Y) \to P(X)$$

on path categories is fully faithful.

Proof The functors $|\cdot|$ and P preserve pushouts and the induced functor $P(|L_x|) \to P(\square^n)$ is fully faithful by Lemma 4. Fritsch and Latch have shown in (Prop. 5.2, [1]) that pushouts in **Cat** preserve fully faithful functors which are inclusions on objects, so one finds in particular that $i_* : P(Y) \to P(X)$ must also be fully faithful. □

A vertex $x \in X$ contained in a unique top-cell of a finite directed cubical complex X will be called a *corner vertex* of X. Given a corner vertex $x \in v(c)$ belonging to the set of vertices $v(c)$ of some top-cell c of X, the Theorem shows that replacing c with the cubical subcomplex $L_x \subset c$ supported on $v(c) - \{x\}$ does not alter $P(X)(a, b)$ for $x \notin \{a, b\}$. In practice one often has a specific a and b in mind, and replacing c with L_x often produces new corner vertices so this process can be iterated until no corner vertices remain in X. Crucially, this process applies to cells of any dimension, not just 2-cells, so one has an opportunity to determine $P(X)(a, b)$ in complexes such as large cubical tori without first passing to the 2-skeleton. This strategy also applies to complexes of higher genus such as cubical surfaces of genus 2 or higher.

4 Algorithmic considerations

On a computer one must choose a truncated subspace $C^N \subset (\mathbb{N} \cup \{0\})^N$, such as that given by $C = \{0, \ldots, 2^8 - 1\}$, in which to embed the finite directed cubical complex of interest; the vertices are then represented as N-tuples $x = (x_1, \ldots, x_N)$ of bytes. A directed n-cell c is uniquely identified by the images $c(\emptyset)$ and $c(\{1, \ldots, n\})$ which will be called the *minimum* and *maximum* vertices of c, denoted by $\min(c)$ and $\max(c)$. The pointwise subtraction $d_c := \max(c) - \min(c)$ defines a vector of bits called the *difference vector* of c; in particular, the count of 1's in d_c determines the dimension of c. It follows that a space-efficient computer representation of a directed cubical cell may be given by the pair $(\min(c), d_c)$ where $\max(c)$ is recovered via the pointwise addition $\min(c) + d_c$.

Let $|X|$ denote the number of top-cells of a finite directed cubical complex $X \subseteq C^N$. Then it is possible to determine all the corner vertices of X by scanning through all top-cells of X, storing their vertices, and then throwing away any vertices which are found to have occurred more than once. This requires $\mathcal{O}(N \cdot |X| \cdot 2^N)$ integer operations to generate the vertices since there are $|X|$ top-cells, and each could have as many as 2^N vertices, each of which is described by N integers. It also requires about $\mathcal{O}(N \cdot |X| \cdot 2^N)$ operations to determine which of these vertices occur only once (using a hash table implementation, say) so the algorithm has overall time complexity $\mathcal{O}(N \cdot |X| \cdot 2^N)$.

The major practical problem with this approach is that the number of vertices of X could be enormous: in the general case $\mathcal{O}(N \cdot |X| \cdot 2^N)$ bytes would be required to store them all and this quickly becomes infeasible for large complexes. Of course, one could instead try to detect if a *single* vertex is a corner vertex: this requires $\mathcal{O}(N \cdot |X| \cdot 2^N)$ time and $\mathcal{O}(N)$ space to detect if the vertex occurs in more than one top-cell of X. The space usage is clearly not a concern in this case, but the problem is that corner vertices can be rare: even for dimension 2 one can consider large rectangular spans containing millions of top-cells but only four corner vertices. Therefore, just guessing at vertices to test is unlikely to yield corner vertices in a reasonable amount of time, even in the presence of parallelization.

A couple of lessons can be learned from these considerations:

1. the number of vertices $\mathcal{O}(|X|\cdot 2^N)$ is a major limiting factor for the time-complexity of any algorithm that must deal with them directly; and
2. attempting to store substructures of top-cells, such as vertices, is a losing game in terms of memory efficiency.

An alternative approach based on these observations begins with

Lemma 5 *Suppose X is a finite directed cubical complex and $K_1, K_2 \subset X$ are subcomplexes consisting of top-cells of X such that $K_1 \cup K_2 = X$. Then if $x \in K_1 \backslash K_2$ then x is a corner vertex of X if and only if it is a corner vertex of K_1.*

Proof If x is a corner vertex of X then it is contained in a unique top-cell of X, so it must be contained in a unique top-cell of K_1 since K_1 consists of top-cells of X by assumption, hence it is a corner vertex of K_1 as well. Conversely, if $x \in K_1$ is a corner vertex then it is contained in a unique top-cell of K_1, but this is the only top-cell of X that can contain x since $x \notin K_2$, hence $x \in X$ must be a corner vertex. □

One therefore seeks subcomplexes $K_i \subset X$ for which it is particularly efficient to determine that a given vertex $x \in K_i$ does not belong to the subcomplex $X \backslash K_i$ of all top-cells of X whose vertices do not all belong to K_i. The subcomplexes K_i must also have a feasible number of vertices so that their corner vertices may be efficiently computed as above. One way to find such subcomplexes of X is via vertex spans. A *vertex span* consists of a cubical subcomplex $s(v, w) \subseteq C^N$ whose cells c are exactly those satisfying

$$v \leq \min(c) \leq \max(c) \leq w$$

in the cubical partial ordering on the vertices of C^N. One sees immediately that any cubical cell c is a vertex span for $(\min(c), \max(c))$, but the structure is more general since a vertex span may contain more than one top-cell.

Proposition 1 *The morphism sets $P(|s(v, w)|)(a, b)$ of the path category of a vertex span $s(v, w) \subseteq C^N$ each contain at most one element. A path $a \rightsquigarrow b$ exists in $s(v, w)$ if and only if $a \leq b$ in the cubical partial ordering on vertices.*

Proof If $w - v = (c_1, \ldots, c_N)$ then the triangulation $|s(v, w)|$ is the simplicial nerve of a poset

$$s(v, w) \cong \underline{c_1} \times \cdots \times \underline{c_N}$$

where $\underline{c_i}$ is the poset $0 \to \cdots \to c_i$ with the total ordering. Thus

$$P(|s(v, w)|) \cong P(\underline{c_1}) \times \cdots \times P(\underline{c_N})$$

since the path category functor preserves products. Given any two objects of the latter category, there is at most one morphism between them since the same is true

of any $P(\underline{c_i})$. If there is a path $a \rightsquigarrow b$ in $s(v, w)$ then clearly one must have $a \leq b$ since the coordinates increment along any such path. Conversely, if $a \leq b$ then $b = a + (c_1, \ldots, c_N)$, in which case one can consider the path

$$a \to \cdots \to a + (c_1, 0, \ldots, 0) \to \cdots \to a + (c_1, c_2, 0, \ldots, 0) \to \cdots \to b$$

which belongs to $s(v, w)$. □

Detecting that a top-cell c of X belongs to a vertex span $s(v, w)$ is easy: one simply tests the above inequalities using $\mathcal{O}(N)$ integer comparisons and constant additional space. Moreover, the vertices in the boundary of any vertex span can be efficiently determined:

Lemma 6 *A vertex $x \in s(v, w)$ lies in the boundary of a vertex span $s(v, w) \subseteq C^N$ if and only if there exists an i, $1 \leq i \leq N$, such that $x_i \in \{v_i, w_i\}$.*

Proof The vertex $x \in s(v, w)$ is interior if and only if $0 < x_i - v_i < c_i$ for all $1 \leq i \leq N$ where $w - v = (c_1, \ldots, c_N)$. □

The following Proposition leads to a more memory-efficient method for detecting corner vertices:

Proposition 2 *Suppose X is a finite directed cubical complex, $s(v, w) \subseteq C^N$ a vertex span consisting of top-cells, and $K_1 := X \cap s(v, w)$ the set of top-cells of X belonging to $s(v, w)$. Then if $x \in K_1$ is a corner vertex of K_1 and $x \notin \partial s(v, w)$, then x is a corner vertex of X.*

Proof Since $x \in s(v, w)$ and $x \notin \partial s(v, w)$, the subcomplex $X \backslash K_1$ of top-cells cannot contain x. Observing that $K_1 \cup (X \backslash K_1) = X$, one has that $x \in X$ is a corner vertex by Lemma 5. □

Given such a vertex span $s(v, w)$, one can determine $K_1 = X \cap s(v, w)$ in $\mathcal{O}(N \cdot |X|)$ time by filtering the top-cells of X for those belonging to $s(v, w)$. Once K_1 is determined it will have some fixed number $|K_1|$ of top-cells. The number of integer operations required to determine the corner vertices of K_1 will therefore be essentially $\mathcal{O}(N \cdot |K_1| \cdot 2^N)$, and $\mathcal{O}(N \cdot |K_1| \cdot 2^N)$ bytes will have been consumed. Among the corner vertices detected in K_1, each of those in $\partial s(v, w)$ can be detected in at most $\mathcal{O}(N)$ time. If the top-cells of X are stored in memory (as must be assumed if top-cells are being deleted and inserted as in the algorithms discussed below) then one is instead looking at a space complexity of $\mathcal{O}(N \cdot |X| + N \cdot |K_1| \cdot 2^N)$ bytes, which will tend to be feasible if $|K_1| \ll |X|$; the idea here is to minimize the size of the exponential term.

Proposition 2 gives sufficient but not necessary conditions for a vertex $x \in X$ to be a corner vertex. For example, if $s(v, w) \subseteq C^N$ is a single top-cell then every vertex of $s(v, w)$ belongs to $\partial s(v, w)$ so the Proposition will have nothing to say about the corner vertices of X. Another problem arises when X itself has some vertices in ∂C^N, in which case said vertices will have no chance of being contained in the interior of any vertex span $s(v, w) \subseteq C^N$. In terms of design this leads to two principles:

1. the vertex spans $s(v, w)$ used to detect corner vertices of X should extend at least three units along every coordinate;
2. X itself should be given an embedding where $X \cap \partial C^N = \emptyset$.

Taking these considerations into account leads to

Proposition 3 *Suppose $X \subseteq C^N$ is a finite directed cubical complex such that $X \cap \partial C^N = \emptyset$ and $C = \{0, \ldots, 2^s - 1\}$ for $s \geq 2$. Then the set S of vertex spans of the form*

$$s((2k_1, \ldots, 2k_N), (2k_1 + 3, \ldots, 2k_N + 3)) \qquad 0 \leq k_i \leq 2^{s-1} - 2$$

cover X, and every vertex $x \in X$ is an interior vertex of at least one such span. If $x \in X$ is a corner vertex then there exists at least one span s among this set which detects *it in the sense that $x \in X \cap s$ is a corner vertex and $x \notin \partial s$.*

Proof Since $2k_i = 0$ for $k_i = 0$ and $2k_i + 3 = 2^s - 1$ for $k_i = 2^{s-1} - 2$, one sees that every possible coordinate value in C^N is covered by at least one of the spans, so X itself is certainly covered since the spans consist of top-cells of C^N. Since $X \cap \partial C^N = \emptyset$, every vertex $(x_1, \ldots, x_N) \in X$ must have $x_i \in [1, \ldots, 2^s - 2]$ for all i, and it is clear that there exist k_i where $x_i \in \{2k_i + 1, 2k_i + 2\}$ so this yields a span $s \in S$ where $x \in s$ is an interior vertex. If $x \in X$ is a corner vertex then one has that it is also a corner vertex of $X \cap s$, and $x \notin \partial s$ so this property is detected on $X \cap s$. □

The advantage here is that there are 2^N vertices interior to each such span. The time complexity for determining the corner vertices of X detected by such a span is $\mathcal{O}(N \cdot 3^N \cdot 2^N)$ by the prior discussion; this improves on the naïve method whenever $3^N < |X|$, as is usually the case in large examples. The space complexity in this case is $\mathcal{O}(N \cdot 4^N)$ bytes for a hash-table approach since the vertex span has $\mathcal{O}(4^N)$ vertices; this improves on the naïve general bound of $\mathcal{O}(N \cdot 3^N \cdot 2^N)$. There are $(2^{s-1} - 1)^N \sim \mathcal{O}(2^{N(s-1)})$ such spans to consider in general. If one were to compute corner vertices for these spans in parallel (as is certainly possible) for $N = 8$, say, then each process would consume about $2 \cdot 8 \cdot 4^8$ additional bytes in a compact representation, or in other words 1 megabyte per process, so this is feasible at least in low dimensions.

The above algorithms work for arbitrary finite directed cubical complexes, but of course one can do much better if X has a special form:

Proposition 4 *The corner vertices of a vertex span $s(v, w) \subseteq C^N$ are exactly the elements of the product*

$$\{v_1, w_1\} \times \cdots \times \{v_N, w_N\}$$

where N is the ambient dimension. In other words, they are the vertices which are either minimal or maximal along every coordinate.

Proof First observe that if x is a vertex of $s(v, w)$ and $1 \leq i \leq N$ then $x - e_i$ and $x + e_i$ must belong to distinct top-cells where e_i is the unit vector along the ith coordinate; this is due to the fact that the embedding $X \subseteq C^N$ must satisfy $c(\underline{n}) - c(\emptyset) = (1, \ldots, 1)$

by definition. In particular, if x has the property that $x - e_i$ and $x + e_i$ both belong to $s(v, w)$ for some i then x cannot be a corner vertex; therefore if x is a corner vertex then it must be minimal or maximal along every coordinate.

Conversely, if x is not a corner vertex of $s(v, w)$ then it is contained in more than one unique top-cell. It follows that there exist vertices a and b in $s(v, w)$ such that $a \leq x \leq b$ and $b - a = (1, \ldots, 1, 2, 1, \ldots, 1)$ where 2 occurs at some index i and $x_i - a_i = 1$. Then $x \in s(a, a + (1, \ldots, 1))$ and also $x \in s(a + (0, \ldots, 0, 1, 0, \ldots, 0), b)$, which are distinct top-cells of $s(v, w)$. In particular, $x - e_i \geq a$ and $x + e_i \leq b$ so x is neither maximal nor minimal along the coordinate i. Therefore if x is minimal or maximal along every coordinate then it must be a corner vertex of $s(v, w)$. □

This ready-made description of corner vertices can be useful when X consists of a cubical span with certain subspans deleted (e.g. applications to concurrency), as in that case one immediately knows at least the corner vertices of the original span. From this description one can also easily recover the *corner cells* of $s(v, w)$, i.e. those top-cells containing corner vertices.

Any algorithm that recursively removes corner vertices from finite directed cubical complexes faces the challenge of identifying any new corner vertices that are created. More precisely, if $c \subset X$ is a top-cell with vertex x and $L_x \subset c$ is the cubical subcomplex supported away from x, one needs to efficiently determine the new corner vertices of X generated by replacing c with L_x. Assuming $X \cap \partial C^N = \emptyset$ and $C = \{0, \ldots, 2^s - 1\}$ for $s \geq 2$, Proposition 3 establishes that there is a vertex span $s_c := s(\min(c) - (1, \ldots, 1), \max(c) + (1, \ldots, 1))$ containing c in its interior, so by Proposition 2 it suffices to compute the corner vertices of $s_c \cap X$. The following result determines the potential new corner vertices in advance:

Proposition 5 *If c is a cubical n-cell and $L_x \subset c$ is the subcomplex supported away from some vertex $x \in c$, then each top-cell of L_x contains a unique corner vertex.*

Proof If c has dimension 0 then L_x is empty so d cannot be a top-cell of Y and the assertion is vacuously true, so suppose c has dimension $n \geq 1$. Choosing local coordinates for c where $c(\emptyset) = (0, \ldots, 0)$ and $c(\underline{n}) = (1, \ldots, 1)$, observe that the two different types of top-cells of ∂c are of the form

$$s((0, \ldots, 0, 1, 0, \ldots, 0), (1, \ldots, 1))$$
$$s((0, \ldots, 0), (1, \ldots, 1, 0, 1, \ldots, 1))$$

so that the top-cells of L_x are exactly those given by vertex spans of the form

$$s_i = s((0, \ldots, 0, 1 - x_i, 0, \ldots, 0), (1, \ldots, 1, 1 - x_i, 1, \ldots 1))$$

for $1 \leq i \leq N$. A vertex $y \in L_x$ belongs to s_i exactly when $y_i = 1 - x_i$, so s_i is the only top-cell of L_x containing y exactly when $y_i = 1 - x_i$ and $y_j = x_j$ for all $j \neq i$. Every top-cell $s_i \subset L_x$ therefore contains a unique corner vertex of L_x given by $y = (x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_N)$. □

Since there are exactly N potential new corner vertices, the above algorithm for corner vertices will require $\mathcal{O}(N \cdot 3^N \cdot 2^N)$ integer operations as usual but can be modified to use only $\mathcal{O}(N^2)$ memory; this can be accomplished by prepopulating a hash table with the known corner vertices and then ignoring any vertices which either hash to an empty bucket or to a bucket containing no vertices equal to it. For purposes of detection, the potential corner vertices that are not actual corner vertices need only occur twice in the hash table; further insertion attempts can also be ignored.

Finally, it is useful to know when a set S of corner vertices can be removed from a finite directed cubical complex X in parallel to produce a new complex X_S (whose "new" corner vertices can then be determined by parallel application of the above algorithm and taking the union of the answers). Here is the basic result:

Proposition 6 *Suppose $X \subseteq C^N$ is a finite directed cubical complex, $x, y \in X$ corner vertices of X belonging to distinct top-cells, and $X_x, X_y \subset X$ the subcomplexes obtained by replacing the top-cells containing x and y with L_x and L_y respectively. Then $(X_x)_y = (X_y)_x$. More generally, if S is a set of corner vertices of X such that no two are contained in the same top-cell, then the subcomplex $X_S \subset X$ supported away from S can be obtained by removing the elements of S from X in any order.*

Proof Since x and y are corner vertices, neither can belong to both top-cells containing x and y, thus the only vertices removed from either top-cell will be x and y, and performing these removals in either order will yield the same subcomplex. The latter result follows by induction. □

This raises the possibility that X could contain too many corner vertices to store in memory, but this is easily mitigated in practice by placing an upper bound on the number of corner vertices of X that are stored at any given time; this leads to a graceful degradation in the effectiveness of the algorithm without yielding incorrect results. A similar concern is that of creating too many new top-cells upon the removal of a corner vertex (since ∂c could consist entirely of new top-cells in the worst case), but again this can be mitigated by placing an upper bound on the total number of top-cells allowed in memory at any given time and disallowing the removal of any corner vertex that would cause this bound to be exceeded. So far these concerns have not manifested as problems in practice.

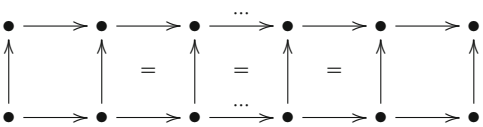
5 Implementation and future work

To test the ideas presented in this paper, the author has implemented software for representing finite directed cubical complexes and reducing them by removing corner vertices. Indeed, these experiments directly led to the formulation of the definition of a "corner vertex" as given here. To give a practical example, there is now a parallelized algorithm for reducing $[1, 2, 3, 4, 5, 6]^5$ to a chain of twenty-five 1-simplices connecting $[1, \ldots, 1]$ to $[6, \ldots, 6]$. On the author's laptop this requires 43 s and 17 megabytes of memory using 8 cores, and a similar example on $[1, \ldots, 10]^3$ requires about 0.5 s and 6 megabytes of memory. The source code is available through Github.

The primary application of interest at the moment is to concurrency theory, where one has a finite directed cubical complex that is effectively a vertex span from which

certain top-dimensional subspans called *forbidden regions* have been deleted. The top-cells of such a complex X are easily computed, and then the algorithm can begin to recursively remove corner vertices from it, producing new top-cells as needed and deleting old ones (for these purposes a pair (a, b) of desired "endpoints" representing the beginning and end of the computation are excluded from the list of corner vertices). Presuming memory bounds are not exceeded, this results in a complex Y without any corner vertices except a and b. In particular, the set of paths $a \rightsquigarrow b$ in Y will usually be much smaller than the corresponding set in X, and it is correspondingly easier to compute $P(|Y|)(a, b)$ than $P(|X|)(a, b)$; code to perform this latter task already exists, including an implementation in C written by the present author.

There are, of course, plenty of examples of finite directed cubical complexes which contain no corner vertices at all. A particularly simple family of examples comes from considering any cubical complex of the form



where the equals signs indicate 2-cells and the remaining squares are boundaries of 2-cells. One can therefore "trap" arbitrarily many top-cells between "empty cells", and similar examples apply to higher-dimensional grids. Calling any such complex *corner-free*, it remains to be seen if another homotopical simplification could be introduced that 1) is easy to detect, compute, and parallelize; 2) applies to at least some corner-free complexes; and 3) is fully faithful on path categories.

Acknowledgments I would like to thank Professor Rick Jardine for introducing me to these applications of homotopy theory to computer science. The anonymous referee is also thanked for providing helpful comments which improved the exposition.

References

1. Fritsch, R., Latch, D.M.: Homotopy inverses for nerve. Math. Zeitschrift **177**(2), 147–179 (1981)
2. Jardine, J.F.: Categorical homotopy theory. Homol. Homotopy Appl. **8**(1), 71–144 (2006)
3. Jardine, J.F.: Path categories and resolutions. Homol. Homotopy Appl. **12**(2), 231–244 (2010)
4. Kahl, T.: Some collapsing operations for 2-dimensional precubical sets. J. Homotopy Relat. Struct. **7**(2), 281–298 (2012)
5. Pratt, V.R.: Modeling concurrency with geometry. In: Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 311–322 (1991). ISBN: 0-89791-419-8