Product Operator on Cell Complexes*

F. Bernardini V. Ferrucci† A. Paoluzzi V. Pascucci

Dip. di Informatica e Sistemistica
Università "La Sapienza"
Via Buonarroti 12, 00185 Roma

Abstract

The intersection of extrusions is a useful tool to generate complex pD models starting from two or more m_iD ($m_i \leq p$) cell complexes. This operation is particularly useful in the context of architectural design, where the 3D model of a building may be automatically generated starting from plan views and sections, both represented as 2D cell complexes [8]. In this paper a more general product operator is introduced which allows for the unification of operations like extrusion, standard intersection, intersection of extrusions and Cartesian product of cell-decomposed polyhedra. A naive implementation of such an operator is a hard task even in 3D, as it seems to require an extensive use of Booleans. In this paper a dimension-independent and efficient solution for this problem is given, using only some linear algebra and the Cartesian product of graphs.

1 Introduction

In this paper a new solid operation on cell-decomposed and dimension-independent polyhedra is introduced. The operation, that we call *generalized product* or simply *product* of polyhedra, takes two polyhedral arguments of possibly different dimensions and produces a polyhedral result of dimension greater or equal to both argument dimensions. The proposed operation is derived from the Cartesian product of cell complexes, and reduces to it as a special case. It is able to unify several different operations which are often useful in solid modeling applications, like extrusion, standard intersection and intersection of extrusions. The various specialized products mainly differ for the way of embedding the arguments, of dimension m and n, respectively, in the coordinate subspaces of the target space of dimension $p \geq \max(m, n)$.

Dimension-independent solid modeling requires representations and operations which uniformly apply to objects of dimension $0, 1, 2, 3, \ldots, n$. This more abstract viewpoint [6] allows to solve in a uniform manner many different geometric problems, like modeling of articulated bodies, piecewise-linear approximation of curved manifolds, graphics representation of multidimensional data and motion encoding. Products of d-dimensional polyhedra times 1-dimensional polyhedra, as well as the intersection of extrusions operation have been introduced in PLASM, a functional design language [8] strongly inspired from FL [1], where the homomorphism between an algebra of polyhedra and an algebra of programs is being currently explored [7].

In PLASM and in several recent approaches to solid modeling [2,9,5] the reference representation is a cell decomposition. In this paper we show that to represent the cells as intersection of halfspaces has several advantages. In particular, it is first shown that with this representation the extrusion operation, which is of great importance in a dimension-independent approach, is a linear operator over the space of linear functions

*This work was partially supported by the Italian National Research Council within the "PF Edilizia" Project under contract nr. 91.02332.64.

†The author acknowledges additional support from IMI.

2nd ACM Solid Modeling '93-5/93/Montreal, Canada
'91-584-4/93/0005/0043...$1.50

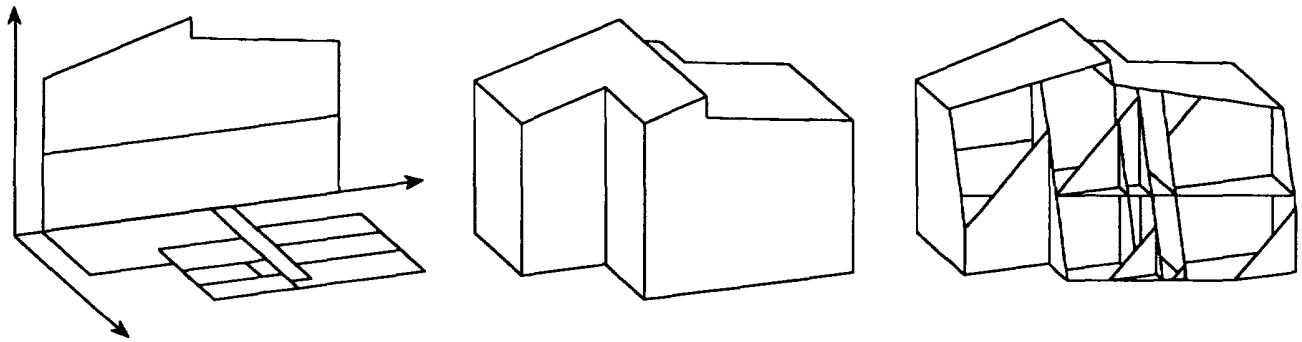

Figure 1: The product operator used as an intersection of extrusions. The generating expression is $@2(A\ {}_{120}\otimes_{102}\ B)$, where A and B are shown on the left and @2() is the extraction operator for the 2-skeleton.

(covectors) associated to the cell faces. Then it is show that an affine transformation may be applied to this representation by multiplying the face covectors for the inverse transformation matrix. Lastly, the computation of the geometry and the topology of the result of the generalized product is discussed.

2 Within the Framework of a Design Language

The product of cell-decomposed polyhedra is one of the more important built-in operators within the functional design language PLASM, which takes a dimension-independent approach to geometry representation and algorithms.

In this language every object is generated by a language expression. The evaluation of the latter produces a polyhedral model which is geometrically consistent since the validity of geometry is syntactically guaranteed. Such an approach allows for a representation which is weaker than usually, allowing for a broader geometric domain encompassing wire-frames, surfaces, solids (and higher dimensional objects), as well as manifolds and non-manifolds, considered as assemblies of pseudo-manifolds. Moreover, with a programming approach a complex design may be hierarchically described and developed with a mixed top-down/bottom-up approach. Using a language the design decisions can be compactly stored on electronic media and quickly transmitted on communication lines, as well as easily recognized and updated through the process of design development and review. Last but not least, it provides a powerful environment for variational geometry, where shapes are completely parameterized.

PLASM can be considered an application shell over FL [1], which is an advanced functional language based on combining forms. FL uses a set of identities—rewriting rules between expressions—for reasoning formally about programs. Simpler equivalent programs can be found at both design and compilation time, with great advantages in style and efficiency of program development. In this context the algebraic properties of the polyhedral operators, e.g. the distributivity of skeleton extractors with respect to the product, can be usefully exploited by the FL optimizing compiler [10].

The model illustrated in Figure 1 was produced by the prototype PLASM interpreter. The dimension-independent operators currently implemented in its geometric nucleus [7] are the generalized product described here, the extractor of the k-skeleton of a d-complex $(0 \leq k \leq d-1)$, affine transformations and boxing and traversal of structures. They work over the domain of non-solid and non-manifold but homogeneously dimensional assemblies.

3 Definition and Examples

In this section, before giving a formal definition of the new operation, we describe three elementary computation steps (extrusion, permutation and intersection); their composition yields the desired product operator. The notation $\mathcal{P}^d$ is used to denote the set of regular polyhedra with dimension d embedded in $\mathcal{A}^d$, the affine space associated to $\Re^d$.

Extrusion. The (non-finite) extrusion is defined as a one-to-many mapping between affine spaces of different

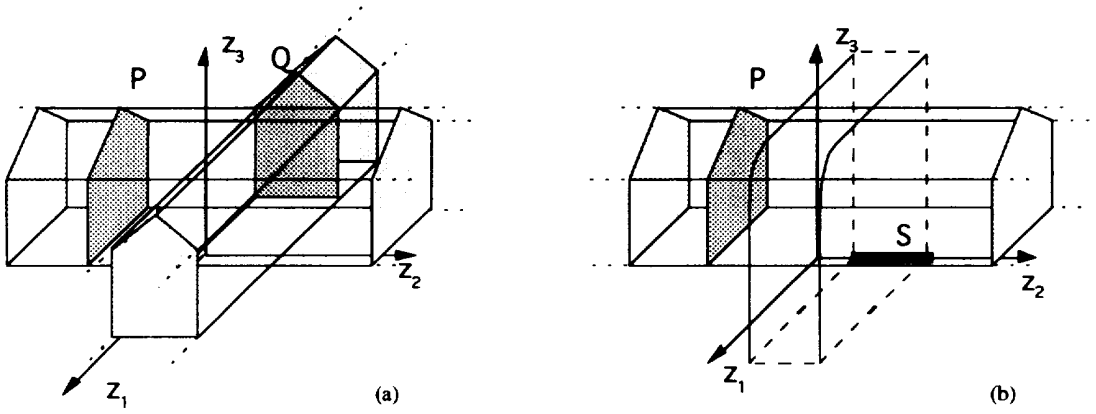

Figure 2: Extrusion of two polyhedra in $\Re^3$.

dimension:

$$E^{p-d} : \mathcal{A}^d \to \mathcal{A}^p : x \mapsto x \times \mathcal{A}^{p-d}, \tag{1}$$

i.e. we have, using coordinate representations

$$x = (x_0, x_1, \ldots, x_d) \mapsto \{(z_0, z_1, \ldots, z_p) = \\ = (x_0, x_1, \ldots, x_d, w_1, \ldots, w_{p-d}), w_i \in \Re\}.$$

It is not difficult to give a coordinate representation, i.e. a matrix representation, of the inverse extrusion mapping, which is a projection:

$$(E^{p-d})^{-1} = \begin{bmatrix} I & 0 \end{bmatrix},$$

where I is a $(d+1) \times (d+1)$ identity matrix and 0 is a $(d+1) \times (p-d)$ zero matrix, but it is not possible to give a similar representation of E^{p-d} if we remain in the affine spaces $\mathcal{A}^d$ and $\mathcal{A}^p$. A matrix representation of the extrusion operator is instead extremely easy to give if we move in their dual spaces, the spaces of the linear functions $\mathcal{A}^d \to \Re$ and $\mathcal{A}^p \to \Re$, as we show in Section 4.

Permutation. Let us denote as π a permutation of the first p integers, i.e. a bijective function on the finite set $\{1, \ldots, p\}$. The permutation mapping

$$\Pi : \mathcal{A}^p \to \mathcal{A}^p \tag{2}$$

such that

$$z = (z_0, z_1, \ldots, z_p) \mapsto \Pi z = (z_0, z_{\pi(1)}, \ldots, z_{\pi(p)})$$

is simply represented by the block matrix

$$\Pi = \begin{bmatrix} 1 & 0 \\ 0 & A \end{bmatrix}_{(p+1)\times(p+1)}, \tag{3}$$

where $A = [a_{i,j}]$ and $a_{i,j}$ is the element $(\pi(i), j)$ of the $p \times p$ identity matrix.

Generalized product operator. The two mappings previously defined can be combined, as the range set of the first coincides with the domain set of the second, so that a composite mapping can be defined:

$$\Theta^{p-d} = (\Pi \circ E^{p-d}) \tag{4}$$

We are finally able to define the desired product operator as a mapping from pairs of polyhedra and pairs of permutations to polyhedra:

$$\otimes^p : \mathcal{P}^m \times \mathcal{P}^n \times \Pi^p \times \Pi^p \to \mathcal{P}^p \tag{5}$$

where Π^p denotes the set of permutations of the first p integers. More useful, as discussed in the following, are the partial functions

$${}_{\pi_1}\otimes^p_{\pi_2} : \mathcal{P}^m \times \mathcal{P}^n \to \mathcal{P}^p \tag{6}$$

such that

$$(P, Q) \mapsto (\Theta^{p-m} P) \cap (\Theta^{p-n} Q).$$

where $\Theta^{p-m} = \Pi_1 \circ E^{p-m}$ and $\Theta^{p-n} = \Pi_2 \circ E^{p-n}$. So, using the product symbol as an infix operator, we can write, by omitting the superscripts

$$P \ {}_{\pi_1}\otimes_{\pi_2} \ Q = \{z \in \mathcal{A}^p \mid z \in \Theta P, z \in \Theta Q\}.$$

where Θ means a proper extrusion and coordinate permutation in $\mathcal{A}^p$ space, depending on the embedding spaces of P and Q and on π_1, π_2, respectively. Notice that to drop the superscripts is allowed because the dimension p of the result is given by the (common) length of permutations.

Further notation. As usual in solid modeling we are mainly interested to the regularized version of the operator, defined as the closure of the interior:

$$P \ {}_{\pi_1}\otimes^*_{\pi_2} \ Q = \mathrm{clos}(\mathrm{int}(P \ {}_{\pi_1}\otimes_{\pi_2} \ Q)).$$

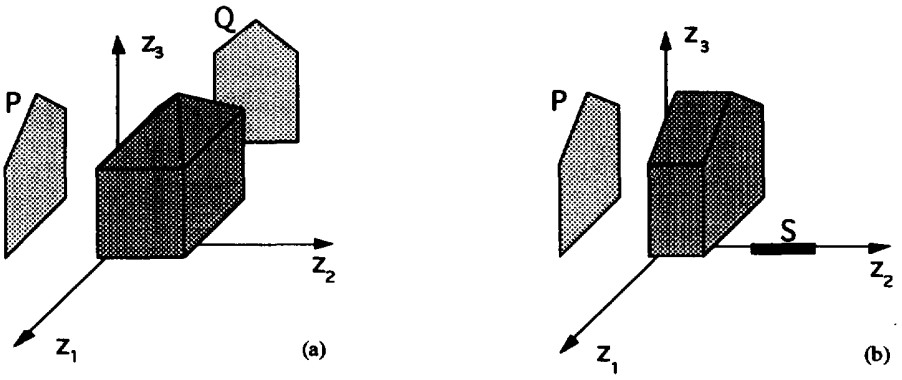

Figure 3: Intersection of two extruded polyhedra. The result shown in the picture can be obtained as $P\ {}_{102}\otimes_{012}\ Q$ (a) and $P\ {}_{102}\otimes_{010}\ ^{c}$ (b).

In the following we will drop the superscript $*$ symbol and the word "regularized", but we will always refer to the regularized operator.

In writing the permutations π_1, π_2, we will use the convention of substituting the indices corresponding to the added coordinates w's with zeroes. E.g., when the point of coordinates $(x_0, x_1, x_2, x_3, w_1, w_2)$ maps to $(x_0, w_1, x_2, x_1, x_3, w_2)$, the corresponding permutation range (42135) is written as 02130. The first coordinate, x_0, never changes its position and is not taken into account in the permutations. The advantage of this notation is that it makes clear along which coordinates the extrusion is performed.

Example 1 *In Figure 2 two examples of extrusion of two polyhedra are depicted. In Figure 2a the resulting polyhedra P' and Q' are obtained as*

$$P' = \Theta^1 P = (\Pi_1 \circ E^1)P, \text{ with}$$
$$E^1 : (x_1, x_2) \mapsto \{(z_1, z_2, z_3) = (x_1, x_2, w_1), w_1 \in \Re\}$$
$$\Pi_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

with $\pi_1 = (132)$, written as 102. Analogously

$$Q' = \Theta^1 Q = (\Pi_2 \circ E^1)Q, \text{ with}$$
$$E^1 : (x_1, x_2) \mapsto \{(z_1, z_2, z_3) = (x_1, x_2, w_1), w_1 \in \Re\}$$
$$\Pi_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

with $\pi_2 = (312)$, written as 012. Notice that it is also

$$P' = \Theta^1 P \cap \mathcal{A}^3 = P\ {}_{102}\otimes_{000}\ o$$

$$Q' = \Theta^1 Q \cap \mathcal{A}^3 = P\ {}_{012}\otimes_{000}\ o,$$

where $o \in \mathcal{P}^0$ is the 0-dimensional polyhedron which has homogeneous coordinate representation (x_0), with $x_0 = 1$ ($\mathcal{P}^0$ is a singleton that contains only o).

Similarly, in Figure 2b the polyhedron P' is obtained as above, whereas S' is given by

$$S' = \Theta^2 S = (\Pi \circ E^2)S = S\ {}_{010}\otimes_{000}\ o, \text{ with}$$
$$E^2 : (x_1) \mapsto \{(z_1, z_2, z_3) = (x_1, w_1, w_2), w_1, w_2 \in \Re\}$$
$$\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

with $\pi = (213)$, written as 010. In Figure 3 the results of the intersections of the extruded polyhedra are shown. □

Specialized products The generalized product previously defined is able to unify several different operations which are often useful in modeling applications. The various operations mainly differ for the way of embedding the polyhedral arguments, of dimension m and n, respectively, in the target space of dimension p:

1. if $p = m = n$ and both π_1 and π_2 are the identity permutation we obtain the usual intersection operation (see Figure 4a);
2. if $p > m = n$ and $\pi_1 = \pi_2$ we obtain the generation of an "indefinite cylinder", i.e. the intersection of the extrusions of P and Q (see Figure 4b);
3. if $p = m + n$ and $\pi_1(k) = 0$ if and only if $\pi_2(k) \neq 0, 1 \leq k \leq p$, we obtain the Cartesian product of polyhedra (see Figure 4c);

4. if $\max(m,n) < p < m+n$ we obtain the intersection of extrusions (see Figure 4d);

5. if $p = \max(m,n)$ and $m \neq n$ one polyhedral argument is intersected with the extrusion of the other;

6. if $p > m+n$ the kind of the result may vary, depending on the permutations. In general the result of the operation is an intersection of extruded solids.

4 Representation of Polyhedra

The used representation scheme is decompositive, where the cells are convex sets resulting from the intersection of halfspaces. A polyhedron is defined as the union of a set of quasi-disjoint cells. This scheme allows to represent regular polyhedra, possibly non-convex and unconnected. Each cell is represented as a set of simultaneous linear inequalities. The representation is completed by the adjacency graph between pairs of cells. Two d-cells are adjacent when they have a common $(d-1)$-dimensional boundary face. The rationale for these design choices is given in the following, together with a more formal description of the assumptions.

An affine hyperplane $ax = 0$, where $a = (a^0, a^1, \ldots, a^d)$ and $x = (x_0, x_1, \ldots, x_d)$, with $x_0 = 1$, can be seen as the subset of points $x \in \mathcal{A}^d$ which is mapped to zero from the linear function $a : \mathcal{A}^d \rightarrow \Re$, i.e. as the as the kernel of the mapping. The space of linear functions a is a vector space of dimension d, which is called the *dual space* of $\mathcal{A}^d$, and denoted as $\mathcal{A}_d$ [3]. Instead of defining a convex set as the convex combination of a convexely independent set of points, we prefer to define a convex cell as the set of simultaneous solutions of a set of linear inequalities. Hence we represent a cell as a set of covectors in dual space, which will be called face covectors. As it is shown later in this section, this allows to represent both extrusion mappings and affine transformations as linear operators on the dual space.

Cell of dimension d, or d-cell, is a convex set of $\mathcal{A}^d$; such a set is the intersection of the halfspaces defined by the affine hyperplanes which support the cell *faces*. A cell may therefore be represented as an ordered set of face covectors, called *cell comatrix* in the sequel.

For each covector a normalized coordinate representation $f = (f^0, f^1, \ldots, f^d)$ is given, which is intended as a row vector where $|f| = 1$. For any point $x = (x_0, x_1, \ldots, x_d)$ internal to a cell, $fx < 0$ holds for any face covector f of the cell. Conversely, for points on the cell boundary $fx = 0$ holds for some f. A point belongs to a face of codimension k (or of dimension $d-k$) if it belongs to $h \geq k$ boundary hyperplanes (k of which are affinely independent). Given the comatrix C, a cell c as a set of points is defined as

$$c = \{x \in \mathcal{A}^d |\ x = (x_0, x_1, \ldots, x_d), x_0 = 1, Cx \leq 0\} \quad (7)$$

A comatrix C is *compatible* when the associated cell c is non-empty. A covector b is *implicated* by a compatible comatrix C when both C and C augmented with row b give the same cell. A comatrix is *non-redundant* when none of its rows is implicated by the others.

Polyhedron of dimension d, or d-polyhedron, is the *union* of a collection of quasi-disjoint d-cells. The intersection of any pair of cells in a polyhedron is either empty or is a face (i-dimensional, $0 \leq i \leq d-1$) for both cells. Note that with such a definition a polyhedron is regular, i.e. homogeneously d-dimensional, and may be non-convex and unconnected.

A face-based decompositive representation of a polyhedron P is a pair (F, C_F) where F is the face covector database and C_F is the cell database. For each cell a list of pairs (covector-pointer, sign) and a list of pointers to $(d-1)$-adjacent cells are stored. The presence of a sign is due to the opposite orientation for the face covector of the two cells $(d-1)$-adjacent along the face. Each vertex may be implicitly represented by the list of faces it belongs to. The solution of the linear system written using the incident covectors will give an explicit representation of the vertex, when necessary.

In the following a polyhedron P will be also represented as a set of cells $\{c_i\}$, each cell c being denoted as a pair $(C, \mathcal{AD}_c)$ where C is the unredundant cell comatrix, and $\mathcal{AD}_c$ is the ordered set of adjacent cells. Given two cells c and b of a polyhedron P, with $c = (C, \mathcal{AD}_c)$, then $\mathcal{AD}_c(i) = b$ means that the cell b is adjacent to c along the face with covector given by the i-th row of the matrix C. We will call this face the i-th face of c. We also write $\mathcal{AD}_c(i) = \perp$ to specify that there is no cell adjacent to c along its i-th face, that is the i-th face of c belongs to the boundary of the polyhedron P.

Dual extrusion mapping In the previous section we have defined the extrusion as the mapping (1), which we were unable to represent as a linear operator from the domain to the range space (remind that this was conversely possible for the inverse mapping). We give such a linear operator for the dual extrusion mapping, defined as a mapping between dual spaces:

$$E_{p-d} : \mathcal{A}_d \rightarrow \mathcal{A}_p : a \mapsto a \times \{0\}^{p-d}, \quad (8)$$

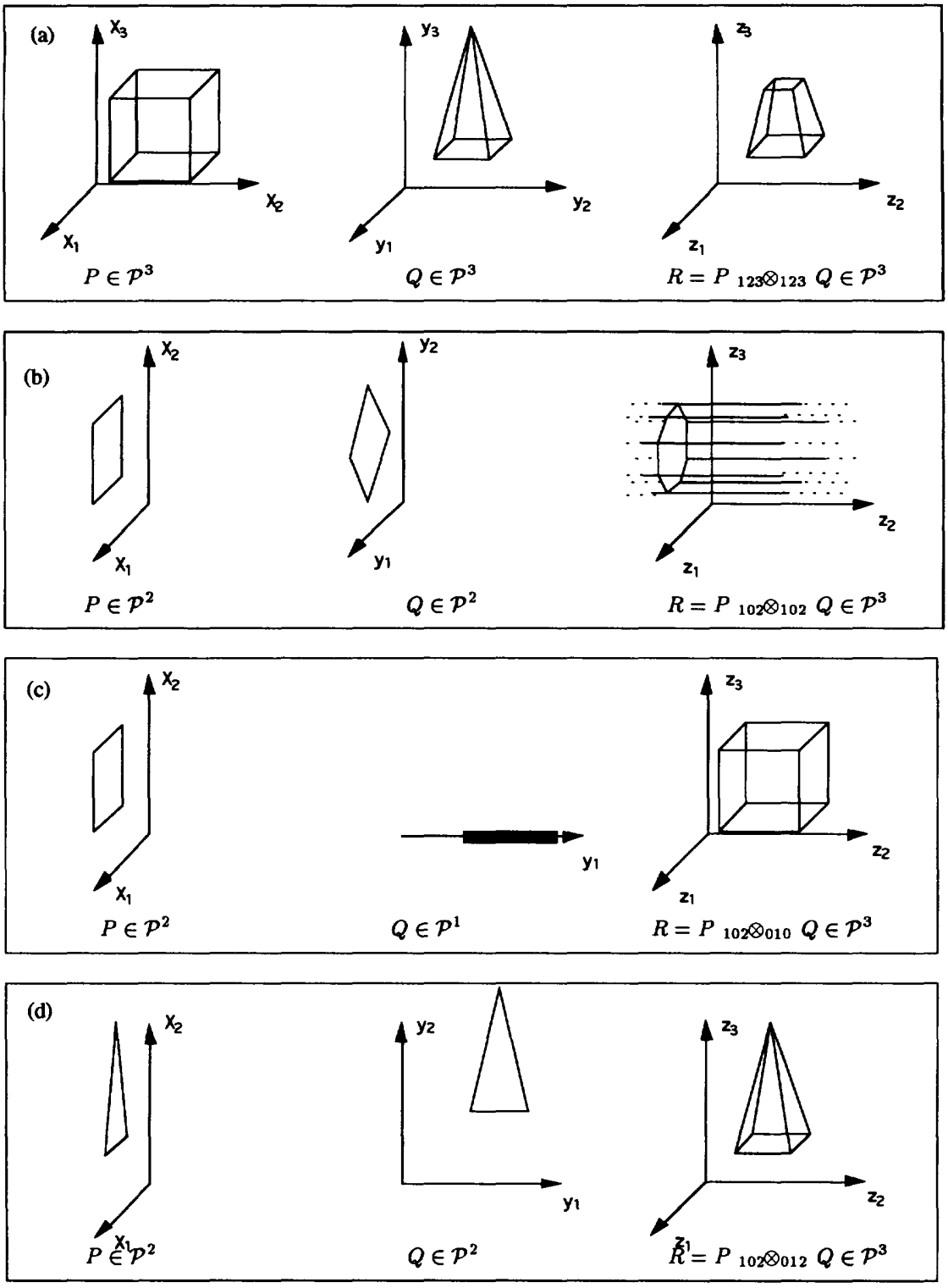

Figure 4: Some applications of the product operator.

i.e., using coordinate representations

$$a = (a^0, a^1, \ldots, a^d) \mapsto (a^0, a^1, \ldots, a^p) = \\ = \quad (a^0, a^1, \ldots, a^d, 0, \ldots, 0).$$

The dual extrusion mapping is a linear function that can be expressed as

$$E_{p-d} = \left[\begin{array}{cc} I & 0 \end{array} \right],$$

where I is a $(d+1) \times (d+1)$ identity matrix and 0 is a $(d+1) \times (p-d)$ zero matrix. Note that

$$(E_{p-d}) = (E^{p-d})^{-1}.$$

Affine transformations Let us consider how the hyperplane $ax = 0$ is transformed according to an affine transformation of the space $\mathcal{A}^d$ where the polyhedra live. As we use homogeneous coordinates, the transformation has a coordinate representation with a $(d+1) \times (d+1)$ invertible real T matrix, so that we write $x^* = Tx$. Hence $x = T^{-1}x^*$, and by substitution in the hyperplane equation we have $aT^{-1}x^* = 0$, finally giving $a^* = aT^{-1}$ for the transformed covector. Notice that $a^*x^* = aT^{-1}Tx = 0$.

An affine transformation T will be therefore applied to a polyhedron $P = (F, C_F)$ by *right multiplication* of each cell comatrix for T^{-1}, or better, by multiplication of each covector in the face database for T^{-1}. With the unusual but useful convention of using matrix multiplication to transform both polyhedra and sets of vectors or covectors, we can write formally:

$$P^* = TP = (FT^{-1}, C_F).$$

5 Product of Cell-Decomposed Polyhedra

In this section we discuss how to compute the generalized product previously introduced for cell-decomposed polyhedra. The following subsections discuss how to compute the geometry and the topology of the result, respectively.

5.1 Computation of Geometry

In Section 3 we have seen that the generalized product of polyhedra can be computed as a sequence of three steps: an extrusion into the target space, a permutation of coordinates and an intersection. Both extrusion and permutation can be expressed as linear operators if we use a convex decomposition with face-based representation. The intersection of convex sets represented as intersection of halfspaces is simply the simultaneous resolution of their constraints. Hence we can easily express the result of the operation in matrix form, starting from the matrices associated to the cells of the operation arguments.

Let be given the polyhedra $P = \{p_h\}$ e $Q = \{q_k\}$, where $p_h = (P_h, \mathcal{AD}_{p_h})$ e $q_k = (Q_k, \mathcal{AD}_{q_k})$, where P_h, Q_k are the cell comatrices of the cells p_h, q_k, respectively.

The product operation between the polyhedra P e Q,

$$P \ {}_{\pi_1}\!\otimes_{\pi_2} Q,$$

gives a polyhedron

$$R = \{r_{hk} |\ r_{hk} = p_h \ {}_{\pi_1}\!\otimes_{\pi_2} q_k,\ (p_h, q_k) \in P \times Q\}.$$

The cell comatrix R_{hk} of the cell r_{hk} can be easily computed by extruding the argument comatrices in dual space (i.e. by computing $P_h E_{p-m}$ and $Q_k E_{p-n}$) and by applying the permutations Π_1^T and Π_2^T to the results, where Π_1 and Π_2 are matrices as defined in (3). The intersection of the extruded cells is then obtained by assembling the results in only one cell comatrix. Hence we have

$$R_{hk} = \left[\begin{array}{c} \tilde{P}_h \\ \tilde{Q}_k \end{array} \right],$$

where

$$\tilde{P}_h = P_h E_{p-m} \Pi_1^T = P_h \Theta_1 \\ \tilde{Q}_k = Q_k E_{p-n} \Pi_1^T = Q_k \Theta_2.$$

Notice that the same matrix Θ_1 (Θ_2) must be applied to all covectors of the polyhedron P (Q), respectively, so that the covector database, seen as a set of covectors, of the resulting polyhedron is simply

$$F(R) = F(P)\Theta_1 \cup F(Q)\Theta_2. \tag{9}$$

The geometry computation of the product $P \otimes Q$ just requires $O(|F(P)| + |F(Q)|)$ matrix multiplications if implemented using the elegant but a bit too expensive linear algebra approach given here. The actual implementation requires only that the covectors database $F(R)$ is written copying covectors from $F(P)$ and $F(Q)$, after insertion of zeroes and a permutation of elements, a $O((|F(P)| + |F(Q)|)p)$ task.

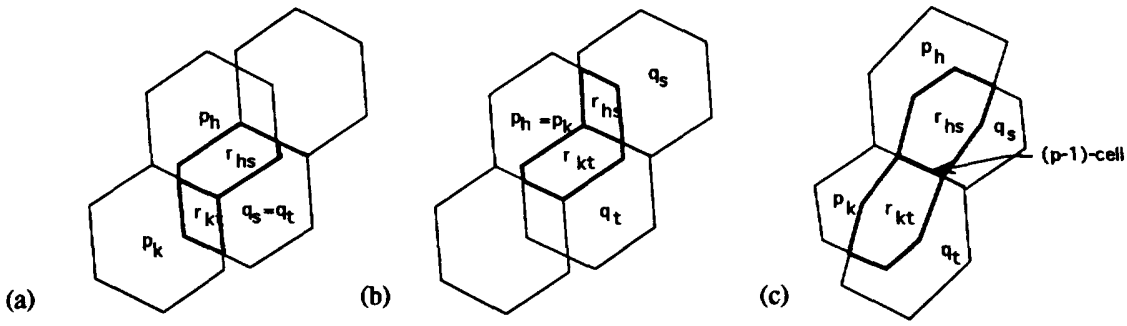

Figure 5: Three different adjacency patterns.

A reduction step of the cell comatrix must be performed for each non-empty or less than p-dimensional cell of the result. In other words, all the $|P| \times |Q|$ comatrices R_{hk} must be checked, to establish non-emptyness and full dimensionality of the associated sets (7). Then, all cells of dimension p must be checked for non-redundancy, i.e. it is necessary to verify that no implicated covectors exist in a comatrix. The covectors in (9) which are not contained in some unredundant comatrix are discarded from the $F(R)$ database.

Example 2 *In Figure 4d the generalized product of two single-cell polyhedra P and Q is displayed. In this case the name of the cell and that of its comatrix coincide. Let*

$$Q = P = \begin{bmatrix} -12 & 4 & 1 \\ 36 & -4 & 1 \\ -8 & 0 & 2 \end{bmatrix}$$

and $\pi_1 = (132) \doteq 102$, $\pi_2 = (312) \doteq 012$. For sake of simplicity face covectors are not normalized. Then we have

$$\tilde{P} = P \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\tilde{Q} = Q \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} -12 & 4 & 0 & 1 \\ 36 & -4 & 0 & 1 \\ -8 & 0 & 0 & 2 \\ -12 & 0 & 4 & 1 \\ 36 & 0 & -4 & 1 \\ -8 & 0 & 0 & 2 \end{bmatrix}$$

□

5.2 Computation of Topology

Two equivalent approaches can be taken for calculating the adjacency graph of a polyhedron resulting from the application of the product operator. A first way is to use a decision rule to recognize cell adjacency. For each cell of the result, the rule yields the appropriate set of edges for the associated graph node. Alternatively, the Cartesian product graph of the adjacency graphs for the operands is built, and then the subgraph describing the topology of lower-dimensional or empty cells, which do not belong to the representation of the result, is deleted and substituted by some suitable new arcs. Lower-dimensional cells can not arise when the operator corresponds to a Cartesian product of polyhedra (see case 3 at the end of Section 3).

An obvious but important observation at the purpose of computing adjacencies for the resulting polyhedron is that the adjacency graphs of the extruded polyhedra are the same as for the operands. In fact, neither the number of cells, nor the number of highest-dimensional faces are affected by the extrusion operation. Moreover, it is easy to see that the extrusion operator neither can disjoint cells which were adjacent in the input polyhedron, nor can make adjacent cells which were not.

Decision rule Consider the (regularized) intersection of $P = \{p_i\}$ and $Q = \{q_j\}$, $P, Q \in \mathcal{P}^p$, obtained by application of the Θ operator. Let $R = P \cap^* Q = \{r_{ij}\}$, $r_{hs} = p_h \cap^* q_s$ and $r_{kt} = p_k \cap^* q_t$ be distinct non-empty cells. We discuss a rule for individuating pairs of cells which have $(p-1)$-dimensional intersection, i.e. that are $(p-1)$-adjacent.

First, suppose $s = t$, and therefore $h \neq k$ (see Figure 5a and clause α of (10)). If p_h is not adjacent to p_k (i.e. $p_h \notin \mathcal{AD}_{p_k}$) then r_{hs} and r_{kt} are not adjacent, because $\dim(p_h \cap p_k) < p-1$ implies $\dim(r_{hs} \cap r_{kt}) < p-1$, since it is $r_{hs} \subseteq p_h$, $r_{kt} \subseteq p_k$. Conversely, if $p_h \in \mathcal{AD}_{p_k}$ and

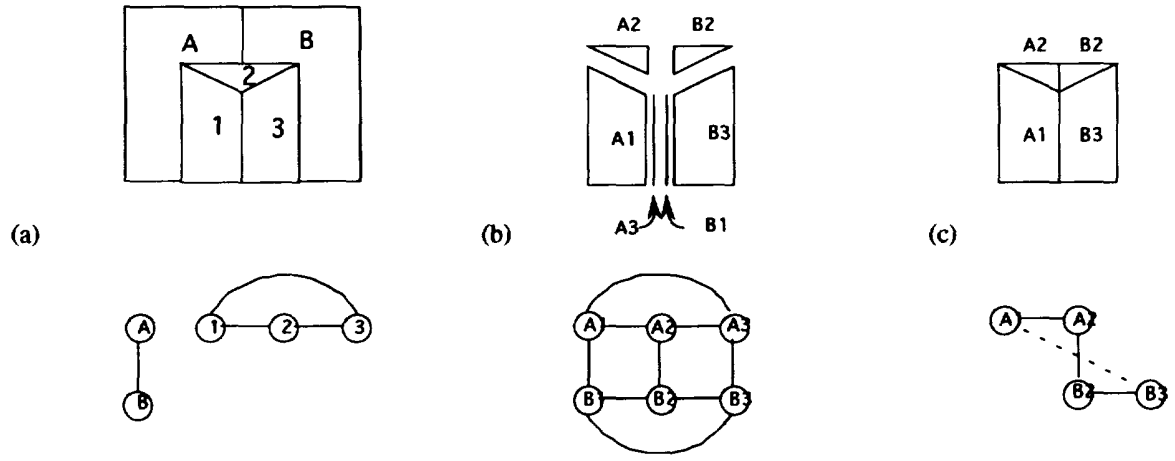

Figure 6: Adjacency reconstruction for $(p-1)$-dimensional degenerate cells. a) The operands $P = \{A, B\}$ and $Q = \{1, 2, 3\}$. b) The 6 cells obtained intersecting cells of P and Q. c) Deletion of empty and degenerate cells, with adjacencies reconstructed by application of the third clause in the rule.

their common $(p-1)$-face is f, r_{hs} and r_{kt} are adjacent if and only if $\dim(f \cap q_s) = p-1$. The case $h = k$, $s \neq t$ is treated analogously (see Figure 5b and clause β of rule 10).

Suppose now that p_h, p_k and q_s, q_t are distinct (see Figure 5c and clause γ of (10)). If $p_h \notin \mathcal{AD}_{p_k}$ or $q_s \notin \mathcal{AD}_{q_t}$, then r_{hs} and r_{kt} are surely not adjacent. Hence, let f be the $(p-1)$-face shared by p_h, p_k and g be the one shared by q_s, q_t. In this case r_{hs} and r_{kt} are adjacent if and only if $\dim(f \cap g) = p-1$, i. e. $\dim(p_h \cap q_t) = p-1$ (and therefore $\dim(p_k \cap q_s) = p-1$).

Formally, the rule for the generation of adjacencies is as follows. Let r_{hs}, r_{kt} be the cells obtained by intersecting (p_h, q_s) and (p_k, q_t), where $p_h, p_k \in P$ and $q_s, q_t \in Q$. Then (see Figure 5):

$$r_{kt} \in \mathcal{AD}_{r_{hs}} \Leftrightarrow \alpha \vee \beta \vee \gamma \qquad (10)$$

where

$$\begin{aligned}
\alpha &= (p_h \in \mathcal{AD}_{p_k}) \wedge (s = t) \wedge \\
&\quad (\dim(p_h \cap p_k \cap q_s) = p-1) \\
\beta &= (h = k) \wedge (q_s \in \mathcal{AD}_{q_t}) \wedge \\
&\quad (\dim(q_s \cap q_t \cap p_h) = p-1) \\
\gamma &= (p_h \in \mathcal{AD}_{p_k}) \wedge (q_s \in \mathcal{AD}_{q_t}) \wedge \\
&\quad (\dim(p_h \cap q_t) = p-1)
\end{aligned}$$

Cartesian graph product The adjacency graph of R is a subgraph of the Cartesian product [4] of the adjacency graphs of the operands, with some new arcs. The final graph is obtained by elimination of the nodes corresponding to empty or lower-dimensional cells from the latter. Accordingly to the chosen representation, only nodes corresponding to p-dimensional cells and edges representing $(p-1)$-adjacencies must be retained. However, $(p-1)$-cells coinciding with the common face of two adjacent p-cells need to be taken into account for adjacency reconstruction. This kind of cells arises when the adjacency pattern is as in Figure 5c. When eliminating such nodes, they can be considered as "bridge nodes", as it is necessary to appropriately maintain adjacency between cells which were connected through the degenerate cell. This is the reason for the third clause in the rule above. Figure 6 illustrates such a situation, where a new edge is introduced between nodes $A1$ and $B3$ when nodes $A3$ and $B1$ are eliminated.

6 Conclusions

In this paper we discussed some properties of a face-based cell-decompositive representation, and introduced a new solid operator, which has been called "generalized product" on dimension-independent polyhedra. In particular we have shown that the new operation is an abstract setting which contains as special cases the (non-finite and finite) extrusion, the regularized intersection, the intersection of extrusions and the Cartesian product of cell complexes. We have shown also that the geometry of the result can be expressed in closed form using standard linear algebra tools (dual spaces and matrix calculus). The topology of the result may be easily derived from the Cartesian product of the adjacency

graphs of the operation arguments.

The techniques discussed here have possible extensions to the (a) curved, (b) non-solid and (c) non-regular case. In the first case a cell is described as a set of implicit non-linear inequalities; in the second one as a set of equalities and inequalities; in the third one the set of cells is of mixed dimensionality. Another useful extension is to consider how the product operator described here can be used as a n-ary operator. We also believe that it may be interesting to study the formal properties of the operation (at least in special cases), and the kind of algebraic structure it defines. E.g. it has been discovered [7] that in this setting the operation of extraction of skeletons distributes over the Cartesian product. Such kind of properties may be usefully exploited within an optimizing compiler for a geometric language.

References

[1] BACKUS, J., WILLIAMS, J., WIMMERS, E., LUCAS, P., AND AIKEN, A. FL language manual, parts 1 and 2. Tech. Rep. RJ7100 (67163), IBM Almaden Res. Center, 1989.

[2] BRISSON, E. Representing geometric structures in d dimensions: Topology and order. In *ACM Symposium on Computational Geometry* (New York, NY, 1989), ACM Press.

[3] CRAMPIN, M., AND PIRANI, F. A. E. *Applicable Differential Geometry.* Cambridge University Press, 1988.

[4] GROSS, J. L., AND TUCKER, T. W. *Topological Graph Theory.* John Wiley, New York, NY, 1987.

[5] MAULIK, A. An efficient intersection algorithm for polyhedral cellular decompositions. In *ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications* (Austin, TX, 1991), J. Rossignac and J. Turner, Eds., ACM Press, pp. 109–118.

[6] PAOLUZZI, A., BERNARDINI, F., CATTANI, C., AND FERRUCCI, V. Dimension-independent modeling with simplicial complexes. To appear on *ACM Transactions on Graphics*, Jan. 1993.

[7] PAOLUZZI, A., PASCUCCI, V., AND VICENTINO, M. More structure and less topology. A programming approach to geometric design. A preliminary version appeared as Tech. Rep. 16–92, Dip. di Informatica e Sistemistica, Università "La Sapienza", Rome, Italy, 1992.

[8] PAOLUZZI, A., AND SANSONI, C. Programming language for solid variational geometry. *Computer Aided Design 24*, 7 (1992), 349–366.

[9] ROSSIGNAC, J., AND O'CONNOR, M. SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In *Geometric Modeling for Product Engineering* (Rensselaerville, NY, Sept. 1988), M. J. Wozny, J. Turner, and K. Preiss, Eds., Proc. of the 1988 IFIP/NSF Workshop on Geometric Modelling, North Holland 1990, pp. 145–180.

[10] WILLIAMS, J. H., AND WIMMERS, E. L. An opmizing compiler based on program transformation. Internal IBM report, IBM Almaden Res. Center, 1991.