Paulo Roma Cavalcanti

Criação e Manutenção de Subdivisões do Espaço

Tese de Doutorado

Departamento de Informática

Rio de Janeiro, 19 de março de 1992

Paulo Roma Cavalcanti

CRIAÇÃO E MANUTENÇÃO DE SUBDIVISÕES DO ESPAÇO

Tese apresentada ao Departamento
de Informática da PUC-Rio como parte dos
requisitos para a obtenção do título de
Doutor em Informática: Ciência da Computação.

Orientador: Paulo Cezar Pinto Carvalho

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 19 de março de 1992

A minha família.

AGRADECIMENTOS (em ordem alfabética)

A Jonas Miranda pelo suporte matemático que proveu a consistência necessária ao tema escolhido.

A Luiz Fernando Martha por permitir a utilização de parte dos seus programas e pelo seu entusiasmo contagiante, que nos deu força para seguir em frente.

A Luiz Henrique Figueiredo pelo suporte em UNIX, VI e TEX (que valeu mais do que qualquer manual).

A Marcelo Gattass que acreditou que este trabalho se realizaria mesmo quando nós próprios ainda não acreditávamos e por ter defendido os nossos objetivos perante a PUC.

A Paulo Cezar Carvalho pela orientação correta e segura e pelo convite para utilizar as instalações do IMPA.

A Bruno Feijó, Marcelo Dreux e Marinho Persiano pelo interesse em avaliar e aperfeiçoar este trabalho.

Ao IMPA pela utilização das suas instalações durante todo o ano de 1991 e pela paz de espírito que o seu ambiente proporciona (que esperamos não mude nunca).

Aos nossos colegas do TeCGraf que, ao longo de todo o desenvolvimento deste trabalho, sempre nos auxiliaram em tudo que foi possível.

Agradecemos também ao suporte financeiro da CAPES, CENPES e CNPq.

RESUMO

Modelagem geométrica é, atualmente, fundamental em sistemas de CAD. No entanto, os modelos tradicionalmente utilizados nem sempre satisfazem todos os tipos de aplicação. Aplicações científicas lidam, muitas vezes, com objetos constituídos por diversos materiais com diferentes propriedades. Relacionamentos de contato, além de comuns, são cruciais em cinemática, planejamento de montagens, robótica e geologia.

Para este propósito, o que se deseja é modelar agregados de objetos (sólidos possivelmente combinados com partes de dimensão inferior), mantendo os relacionamentos de adjacência entre estes objetos. Isto permite a criação de um ambiente de desenvolvimento no qual os diversos aplicativos podem compartilhar, não somente dados, mas, também, algoritmos que processam e alteram estes dados. Uma forma de tratar todos estes tipos de problema, de uma maneira uniforme e coerente, é lidando com subdivisões arbitrárias do espaço, ao invés da divisão clássica em apenas três regiões (interior, fronteira e exterior de um objeto sólido).

Subdivisões planares, embora mais simples, também são relevantes, principalmente nas áreas de geologia e cartografia. A criação de uma subdivisão planar difere da criação de uma subdivisão espacial, especialmente na forma de interagir com o usuário (no processo de modelagem utilizado).

O presente trabalho trata do problema de criar, combinar e manter, em tempo real, subdivisões do espaço Euclidiano bi e tri-dimensional. O objetivo principal é a criação de uma metodologia que permita a construção de subdivisões consistentes topológica e geometricamente. Para isto, foram adicionadas algumas extensões ao conceito de complexo geométrico seletivo, de forma a criar o embasamento necessário para a criação de um esquema de representação flexível que elimina várias restrições impostas pela modelagem tradicional.

O esquema de representação proposto, embora mantenha uma representação explícita do bordo dos objetos, suporta operações similares (porém mais poderosas) às operações *booleanas* para sólidos, sem que nenhuma restrição adicional seja imposta

(além disso, ele propicia a criação de algoritmos geométricos eficientes). Isto cria um elenco poderoso de ferramentas de modelagem, permitindo, futuramente, a criação de uma linguagem para construção interativa de subdivisões. A partir de operações básicas, descritas detalhadamente, que permitem inserir, em uma subdivisão já existente, um novo retalho de superfície ou um novo segmento de curva em tempo linear, é possível criar uma subdivisão completa em tempo quadrático (em relação ao número final de elementos da subdivisão).

ABSTRACT

Geometric Modeling is central to many CAD systems, even though the models traditionally employed do not satisfy all applications. Scientific applications need to deal with objects made of different materials, with distinct properties. Contact relationships are crucial in kinematics, assembly planning, robotic, and geology.

In this context, it is necessary to model aggregates of objects (including solids with lower dimensional parts), keeping track of the adjacency relationships among these objects. This suggests the creation of a development environment where the applications can share not only data but also algorithms that process and transform these data. The way to handle such problems is by supporting arbitrary space subdivisions, instead of the classical subdivision in three regions (interior, boundary, and exterior of a solid object).

Planar subdivisions are simpler, but also important, mainly in geology and cartography. The creation of a planar subdivision differs from the creation of a spatial subdivision, specially by the interaction with the user (by the modeling process).

The aim of this work is to deal with the problem of creating, combining, and maintaining, in real time, subdivisions of the two- and three-dimensional Euclidean space. The main goal is devising a methodology that permits the construction of subdivisions which are topologically and geometrically consistent. This methodology is determined by a modeling process, a mathematical model, and a representation scheme. To achieve this, some extensions were added to the concept of selective geometric complex, creating

the basis for a flexible scheme that eliminates several restrictions imposed by traditional modeling.

The proposed representation scheme maintains an explicit boundary representation and allows one to add boolean operations without any additional restriction (furthermore, it enables the implementation of efficient geometrical algorithms). This provides a powerful set of modeling tools that can be used as the base for the creation of an interactive construction language for space subdivisions. The construction of basic operations which allow the insertion of a surface patch or a curve segment in a given subdivision in linear time, is also described in detail. This means that a complete subdivision can be built in quadratic time with the number of subdivision's elements.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

1 - INTRODUÇÃO

Este trabalho se insere na área conhecida por Modelagem Geométrica. Seu objetivo principal é prover condições para a criação de objetos geométricos que não podem ser modelados pela maioria dos sistemas existentes, pois estes sistemas costumam ser restritivos. Para isto, propõe-se uma conjugação de formas (propostas) de modelagem. Na realidade, pretende-se criar uma metodologia que permita abordar o problema desde um nível abstrato, que corresponde a um modelo matemático, passando pela realização deste modelo, através da proposição de um esquema de representação adequado, até ao nível que permita a definição de uma linguagem para a especificação de um objeto real, a partir de operadores apropriados.

1.1 Histórico

Modelagem Geométrica é uma área que experimentou um grande desenvolvimento nos últimos 20 anos. Como área, ela é extremamente abrangente e foi formada a partir de contribuições oriundas de diversas disciplinas, incluindo matemática, engenharia e informática. Ela fornece a base para os sistemas de CAD atuais, que se tornaram uma ferramenta fundamental para a competição industrial. Curiosamente, a demanda por ferramentas, cada vez mais sofisticadas, cresceu mais rápido do que o correspondente desenvolvimento teórico, o que ocasionou a geração de sistemas complexos, mas que já nasciam com várias inconsistências.

A Modelagem Geométrica pode ser definida como a área que lida com o problema de criar, em um computador, representações não ambíguas para objetos geométricos e de construir os algoritmos que processam estas representações. Historicamente, surgiram diferentes formas de representação dos objetos de interesse, diferindo na quantidade e tipo das informações disponíveis. Uma possível classificação seria [WEIL86]:

- Modelagem por arames (*wireframes*).

É uma das primeiras técnicas de Modelagem Geométrica; representa os objetos por arestas e pontos sobre a sua superfície. Embora simples, os modelos por arames são ambíguos.

- Modelagem por superfícies.

Surgida no início dos anos 60, foi um passo adiante, fornecendo também a descrição matemática das superfícies que delimitam o objeto. Seu problema principal é oferecer poucos testes de integridade do modelo.

- Modelagem sólida.

Surgida no início dos anos 70, contém, implícita ou explicitamente, informações sobre o fechamento e conectividade dos objetos, garantindo a sua realização física.

- Modelagem de dimensão mista.

É a forma de modelagem mais recente e permite representar objetos com estruturas internas ou objetos com elementos pendentes de dimensão inferior. Como o resultado final da modelagem é um sólido delimitado por superfícies que não são, necessariamente, localmente planas, este tipo de modelagem é chamada de *non-manifold*. Surgida no início dos anos 80, seu potencial ainda não foi completamente atingido.

Segundo Mäntylä [MANT88], os principais modelos utilizados na modelagem sólida podem ser classificados em:

- **Modelos de decomposição.**

Descrevem um sólido através da combinação de blocos básicos que são mantidos juntos. *Quadtrees, Octrees* e outros métodos hierárquicos são técnicas importantes nos domínios de visão por computador, processamento de imagem e robótica. *Quadtrees*, por exemplo, representam uma classe de estruturas de dados hierárquicos baseada no princípio de decomposição recursiva. Embora os modelos de decomposição sejam uma representação bastante geral, simples e que permitem a utilização de uma série de algoritmos eficientes, a quantidade de memória necessária na representação, para a obtenção de uma descrição geométrica suficientemente precisa, pode inviabilizá-los. Existem, também, outros problemas quando se trabalha com espaços digitalizados. Por exemplo, operações de rotação não são geralmente invertíveis. Em particular, um quadrado que sofre uma rotação não pode ser representado, com precisão, por uma coleção de quadrados retilíneos. Para uma avaliação mais completa dos métodos hierárquicos, recomendam-se os trabalhos de Hanan Samet [SAME84, SAME88a, SAME88b, SAME89a, SAME89b].

- **Modelos construtivos (CSG).**

Representam um sólido complexo a partir da combinação de sólidos simples, chamados primitivos [REQU77]. Estes primitivos são combinados pelas operações *booleanas* regularizadas de união, diferença e interseção. Uma árvore CSG possui nós internos que correspondem a operações *booleanas* ou movimentos rígidos e nós terminais que correspondem a sólidos primitivos (por exemplo: cubos, cilindros ou toros). Basicamente, os modelos CSG armazenam apenas a árvore de combinação de primitivos, que é avaliada sempre que o sólido tem que ser exibido. É fácil classificar pontos em relação ao sólido (verificar se um ponto está no interior, na fronteira ou no exterior do sólido); porém, um sólido não pode ter uma estrutura no seu interior. Informações topológicas requerem uma avaliação da árvore, o que é bastante caro em termos de tempo de execução. Em contrapartida, o gasto de memória para armazenamento é baixo. Algoritmos de exibição (*rendering*), tipo *scan line*, conseguem, facilmente, gerar uma imagem a partir de uma descrição CSG. O processo de construção do sólido através de primitivos básicos produz uma boa interface para interação com o usuário.

- **Modelos de fronteira (BRep).**

Ao contrário dos modelos anteriores, que tratam sólidos como conjuntos de pontos, modelos de fronteira representam sólidos indiretamente, através da representação das superfícies que os delimitam [BAUM72, WEIL85, MANT88]. Uma BRep é um grafo com nós correspondendo às faces, arestas e vértices que definem a fronteira topológica do sólido. Modelos de fronteira armazenam explicitamente informações topológicas (em certos tipos de aplicação isto pode ser essencial). Para tanto, há um gasto elevado de memória. Operações *booleanas*, típicas de modelos CSG, podem ser implementadas às custas de um processamento adicional para a obtenção do resultado. A principal dificuldade, quando se trabalha com modelos de fronteira, é garantir a consistência geométrica. Modelos CSG podem ser convertidos em modelos de fronteira por um processo chamado de avaliação do bordo [REQU87, ROSS89]. A representação *octree* também pode ser transformada em uma representação de fronteira [KUNI85].

Os modelos mais usados para representar sólidos têm sido CSG e BRep. Ambos são úteis e complementares e, atualmente, os modeladores de sólidos tendem a ser

híbridos e conter ambos os modelos de alguma forma [FISC91]. Estes modeladores, em sua maioria, tratam apenas de objetos que são sólidos num sentido bastante restrito da palavra. Basicamente, são objetos compostos por um único tipo de material, sem estrutura interior e delimitados por uma superfície localmente plana. Estas restrições impedem a modelagem de uma série de objetos característicos de aplicações importantes.

1.2 Definição do Problema

Um esquema de representação possui, em geral, algum tipo de limitação. Assim, quando um novo esquema é proposto, costuma haver um rápido avanço para, em seguida, esbarrar-se nas novas restrições. Os esquemas utilizados na Modelagem Geométrica não escapam a esta regra.

Com o surgimento da modelagem sólida, um passo importante foi dado, pois, pela primeira vez, passaram a existir mecanismos efetivos para garantir a integridade (principalmente topológica) do modelo. Porém, como o nome sugere, modelagem sólida busca representar apenas aquilo que, intuitivamente, se costuma chamar de sólido. Além disso, geralmente, os esquemas de representação utilizados são capazes de lidar apenas com um único sólido, não permitindo que seja representada a interdependência que existe quando diversos sólidos compartilham uma mesma região do espaço.

Este trabalho busca, justamente, **estender o domínio de modelagem** de forma a atender um número maior de aplicações, principalmente, nas áreas científicas. Um dos seus objetivos é mostrar que já é possível romper a barreira imposta pela modelagem sólida tradicional.

As aplicações científicas lidam, muitas vezes, com objetos constituídos por diversos materiais com diferentes propriedades (por exemplo: circuitos de semi-condutores, motores, estruturas de concreto e aviões) ou objetos com diversas regiões (por exemplo: malhas para modelos de elementos finitos). Os relacionamentos de contato, além de comuns, são cruciais em cinemática, planejamento de montagens, robótica, geologia e em muitas outras áreas. O contato existe se as fronteiras dos sólidos se interceptam, mas não o seu interior. Formalmente, dois sólidos estão em contato se houver interseção

entre eles, mas seus interiores forem disjuntos. Uma forma de tratar todos estes problemas, de uma maneira uniforme e coerente, é lidando com subdivisões arbitrárias do espaço.

Subdivisões planares, embora mais simples, também são relevantes, principalmente nas áreas de geologia e cartografia. A criação de uma subdivisão planar difere da criação de uma subdivisão espacial, especialmente na forma de interagir com o usuário (isto é, no processo de modelagem utilizado). Isto porque, as pessoas estão acostumadas a construir objetos bi-dimensionais a partir de uma folha de papel e uma caneta. O raciocínio tri-dimensional é, no entanto, bem mais complicado e a forma de trabalho mais confortável costuma ser baseada em um processo construtivo. Refinam-se formas tri-dimensionais pré-existentes para depois compô-las com outras. O sucesso de sistemas CSG pode ser explicado por este fato.

Várias definições têm sido propostas para estender a noção de sólido. Algumas delas são baseadas nas definições matemáticas de complexos simpliciais ou celulares [SPAN66], e não despertam maior interesse porque estes complexos foram criados para atacar questões topológicas e não as necessidades de Modelagem Geométrica.

A modelagem *non-manifold*, embora criada para suprir esta deficiência, carecia de um embasamento matemático que caracterizasse, precisamente, que tipo de objeto poderia ser modelado. Mesmo assim, surgiram diversas estruturas de dados topológicas [WEIL86, DOBK87, LASZ87, LIEN88] que representam, de forma bastante geral, as informações de adjacência de objetos no $\Re^3$ (não necessariamente homogêneos em dimensão).

Um esquema de representação baseado em *δ-sets*, ou agregados de células — que são conjuntos disjuntos, abertos e regulares do $\Re^3$ (as células são iguais ao interior do seu fecho topológico) — foi proposto por Arbab [ARBA85]. O principal problema desta representação é que ela não é fechada em relação a operação *booleana* de união.

Um esquema geral para representação de objetos $n-$dimensionais, não necessariamente abertos ou fechados, com estrutura interior e fronteira com elementos pendentes,

foi criado por Rossignac & O'Connor [ROSS90] e baseia-se no conceito de Complexo Geométrico Seletivo (SGC). A unidade básica de um SGC também é uma célula, só que esta não precisa ser da dimensão do espaço, tampouco ter fonteira conexa. Seu maior defeito é não tirar proveito de algum tipo de ordenação que possa ser imposto à sua representação.

Rossignac & Requicha [ROSS91] criaram um outro esquema de representação, chamado de Geometria Construtiva Não Regularizada (CNRG), que generaliza os modelos CSG, sendo capaz de representar agregados de objetos (isto é, união de conjuntos de pontos do $\Re^n$ mutuamente disjuntos). Uma característica deste esquema, que pode ser incoveniente em certas aplicações, é a não representação explícita do bordo dos objetos.

1.3 Visão Geral do Trabalho

O objetivo deste trabalho é criar uma metodologia para construir e manter (em tempo real) subdivisões do espaço Euclidiano bi e tri-dimensional. Esta metodologia é composta por um processo de modelagem, um modelo matemático e um esquema de representação [REQU80].

No capítulo 2, são apresentados os conceitos matemáticos básicos utilizados neste trabalho. O capítulo 3 apresenta um modelo matemático formal para subdivisões do espaço e é fundamental para a compreensão dos capítulos subseqüentes. Basicamente, é utilizada a definição matemática de complexos geométricos, sendo explicitada a sua estrutura topológica. Algumas extensões ao conceito de SGC são introduzidas neste capítulo, permitindo a caracterização precisa do tipo de subdivisão do espaço que as estruturas de dados topológicas podem representar.

O capítulo 4 apresenta a metodologia para a criação de subdivisões planares, focalizando principalmente os aspectos necessários à construção de um esquema de representação adequado (estrutura de dados e algoritmos). Subdivisões planares são tratadas num capítulo à parte, não só para introduzir o problema a partir de uma forma mais simples e mais fácil de entender, como, também, por possuírem aplicações características e que necessitam de um processo de modelagem próprio. O capítulo 5 apresenta a me-

todologia para a criação de subdivisões espaciais, generalizando o que foi apresentado no capítulo anterior, mas tratando cuidadosamente os problemas que surgem quando se trabalha no $\Re^3$. Em ambos os capítulos, são propostos algoritmos geométricos eficientes, tal como o da localização de um ponto em uma subdivisão, e feita uma discussão de complexidade. São determinadas, também, que funções genéricas devem ser implementadas para cada tipo de célula, de forma a permitir o tratamento do problema usando a técnica de programação orientada para objeto.

No capítulo 6, é apresentado o processo de modelagem CNRG que é formado por um conjunto de operadores que podem ser usados para construir, interativamente, subdivisões do espaço. Muito embora exista um esquema de representação CNRG e, tendo sido neste contexto que estes operadores foram idealizados, este trabalho mostra como adicionar tais operadores a sistemas que implementam SGCs. A principal vantagem desta abordagem é que mantém-se uma representação explícita do bordo, característica importante às aplicações para as quais este trabalho é dirigido.

Finalmente, no capítulo 7, são apresentadas as conclusões, contribuições e sugestões para futuros trabalhos.

2 - CONCEITOS DE TOPOLOGIA

Este capítulo destina-se a pessoas que não estejam familiarizadas com certos conceitos matemáticos que são largamente empregados na literatura dirigida à Modelagem Geométrica. São apresentadas apenas as principais definições e resultados.

Qualquer pessoa interessada em Modelagem Geométrica certamente já ouviu, uma infinidade de vezes, o termo **topologia**. Ao pé da letra, topologia significa o estudo das formas. Topologia algébrica, topologia combinatória, topologia diferencial e topologia de conjuntos de pontos são alguns dos ramos de uma área relativamente nova (poucos resultados são anteriores a 1850). Para nossos objetivos, interessam apenas a topologia de conjuntos de pontos (ou topologia geral) [LIMA70,NAYL71], que pode ser definida como o estudo das propriedades invariantes por homeomorfismos, e a topologia algébrica [MASS67,HENL79], que estuda o uso de métodos algébricos em topologia.

Intuitivamente, um homeomorfismo pode ser imaginado como uma transformação elástica que preserva adjacências, além de outras propriedades topológicas. Um cubo de borracha, se inflado, transforma-se numa esfera e um cubo vazado, num toro. No primeiro caso, diz-se que o cubo é homeomorfo à esfera e no segundo, ao toro.

2.1 Topologia Geral

Um espaço métrico é um par de objetos: um conjunto X e uma função real $d(x, y)$, chamada métrica ou distância, satisfazendo aos seguintes axiomas:

(1) (positividade): $d(x, y) \geq 0$ e $d(x, x) = 0, \forall x, y \in X$.

(2) (positividade estrita): $d(x, y) = 0$ então $x = y, \forall x \in X$.

(3) (simetria): $d(x, y) = d(y, x), \forall x, y \in X$.

(4) (desigualdade triangular): $d(x, y) \leq d(x, z) + d(y, z), \forall x, y, z \in X$.

O conjunto $(\Re^n,\ d)$ é um espaço métrico com d definida, por exemplo, como:

- $d_2(X, Y) = \{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2\}^{1/2}$;
- $d_\infty(X, Y) = max(|x_1 - y_1|, |x_2 - y_2|, ..., |x_n - y_n|)$.

A métrica d_2 é a métrica usual para o $\Re^n$. O espaço $(\Re^n, d_2)$ é chamado espaço Euclidiano (E^n).

Uma **vizinhança** de um ponto x_0 de um espaço métrico (X, d) é definida como sendo uma bola aberta centrada em x_0:

$$B_r(x_0) = \{x \in X : d(x, x_0) < r\}, r > 0.$$

Um ponto x de um espaço métrico (X, d) é um **ponto de aderência** de um subconjunto $A \subset X$ quando toda vizinhança de x contém pelo menos um ponto de A.

Um subconjunto A de um espaço métrico (X, d) é **fechado** quando A contém todos os seus pontos de aderência.

Um subconjunto A de um espaço métrico (X, d) é **aberto** quando o complemento de A ($\overline{A} = X \backslash A$) é fechado (o que equivale a dizer que todo ponto de A possui uma vizinhança completamente contida em A).

O **fecho** de um conjunto A de um espaço métrico (X, d), denotado por $F(A)$, é o conjunto de todos os pontos de aderência de A.

O **interior** de um conjunto A de um espaco métrico (X, d), denotado por $I(A)$, é o conjunto dos pontos de A que não são pontos de aderência de $\overline{A}$.

A **fronteira** de um conjunto A de um espaço métrico (X, d), denotada por ∂A, é o conjunto dos pontos de X que são pontos de aderência de A e $\overline{A}$.

Um conjunto A de um espaço métrico (X, d) é **conexo** quando ao ser dividido em dois subconjuntos disjuntos B e C com $A = B \cup C$, $B \neq \emptyset$ e $C \neq \emptyset$ então:

- $\exists b \in B$, com $b \in F(C)$;
- $\exists c \in C$, com $c \in F(B)$.

Um conjunto de um espaço métrico é **limitado** quando ele está contido em uma bola de raio r. Um conjunto fechado e limitado do $\Re^n$ é dito **compacto**.

A **regularização** de um conjunto de pontos A, denotada por $R(A)$, é definida por

$R(A) = F(I(A))$. Os conjuntos que satisfazem $R(A) = A$ são ditos **regulares**. Um conjunto regular e limitado é chamado **conjunto-r**.

Um **espaço topológico** (X, T) é um conjunto X, mais um sistema de subconjuntos T, chamados os abertos de X, com a propriedade de que X, $\emptyset$, a interseção finita e a união de conjuntos de T pertencem a T. Todo espaço métrico é um espaço topológico, com uma topologia induzida pela sua métrica. Existem, porém, espaços topológicos cuja topologia não é gerada por uma métrica.

Uma função f, de um espaço topológico (X, T), em um outro espaço topológico (Y, V), é **contínua**, em um ponto p, quando a imagem inversa de cada aberto que contém $f(p)$ em (Y, V) é um aberto que contém p em (X, T). A função f é contínua, se for contínua em cada ponto p de X.

Diz-se que dois espaços topológicos (A, T) e (B, V) são **homeomorfos** quando existe uma função $\Upsilon : (A, T) \rightarrow (B, V)$ contínua, invertível e com inversa também contínua. Υ é dita ser um homeomorfismo de A em B. Ao conjunto das propriedades invariantes por homeomorfismos, chamam-se **propriedades topológicas** do espaço. Dois espaços topológicos A e B são topologicamente **equivalentes** quando há um homeomorfismo de A em B.

Um **subespaço** de um espaço topológico (Y, T) é um espaço topológico (Y', T'), onde: $Y' \subset Y$ e $T' = \{t \cap Y', t \in T\}$. Um **mergulho** [1] de um espaço topológico (X, T), num espaço topológico (Y, V), é um homeomorfismo entre X e um subespaço Y' de Y.

Uma **variedade** é um espaço topológico M tal que existe um inteiro $k > 0$ para o qual todo ponto de M possui uma vizinhança homeomorfa a $\Re^k$. O inteiro k é chamado de **dimensão** de M. No âmbito da Modelagem Geométrica, é usual empregar-se o termo *manifold* para designar uma variedade mergulhada no $\Re^3$. Neste contexto, modelagem *manifold* e estruturas de dados *manifold* referem-se a esquemas de representação capazes de tratar sólidos do $\Re^3$ cujas fronteiras são variedades de dimensão 2. Por outro lado, os

[1] Em inglês, *embedding*.

termos – modelagem *non-manifold* e estruturas *non-manifold* – referem-se a esquemas de representação capazes de tratar tanto sólidos *manifold* como objetos mais gerais cujas fronteiras não são variedades (consulte-se o apêndice para uma discussão mais aprofundada sobre o assunto).

Seja V uma variedade de dimensão 2 e A um conjunto-r do $\Re^3$. Diz-se que A é a **realização** de V no $\Re^3$ quando ∂A e V são topologicamente equivalentes.

Seja $f : \Re^n \rightarrow \Re$ um polinômio da forma:

$$\sum_{i=0}^{m} a_i x_1^{e_{1,i}} x_2^{e_{2,i}} ... x_n^{e_{n,i}}, \quad e_{j,i} \in Z.$$

O conjunto dos pontos $\{p \in \Re^n | f(p) = 0\}$ é chamado de **variedade algébrica** [2] associada a f, e será abreviada por VA.

Seja V um subconjunto do $\Re^n$. Um ponto $x \in V$ é **regular** de dimensão d, quando x possui uma vizinhança $U \subset \Re^n$ tal que $U \cap V$ é uma subvariedade [3] do $\Re^n$, de dimensão d. A dimensão do subconjunto V é definida como a dimensão máxima do conjunto dos pontos regulares de V [CARV92].

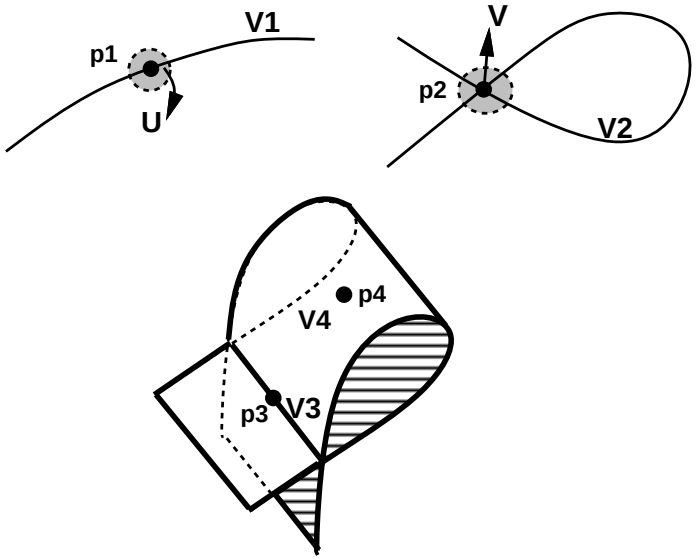


Figura 2.1 - Pontos regulares e singulares de VAs.

[2] Em inglês, *algebraic variety*.

[3] Numa definição geral, exige-se que a subvariedade seja diferenciável. No caso de VAs, isto ocorre sempre.

Os pontos de uma VA f que não são pontos regulares de dimensão máxima são chamados pontos **singulares**. Pelo teorema da função implícita (seção 4.5), ∇f se anula nos pontos singulares.

Na figura 2.1, todo ponto de V_1 é um ponto regular de dimensão 1. O ponto p_2 é um ponto regular de dimensão 0, de V e um ponto singular de V_2. O ponto p_3 é um ponto regular de dimensão 1, de V_3 e um ponto singular de V_4; o ponto p_4 é um ponto regular de dimensão 2, de V_4.

Uma VA não é, necessariamente, uma variedade, no sentido definido anteriormente (fig. 2.2a). Se ela não tiver pontos singulares, então, ela é uma variedade no sentido usual (fig. 2.2b). Existem VAs que só possuem pontos singulares; por exemplo, $x^2 = 0$.

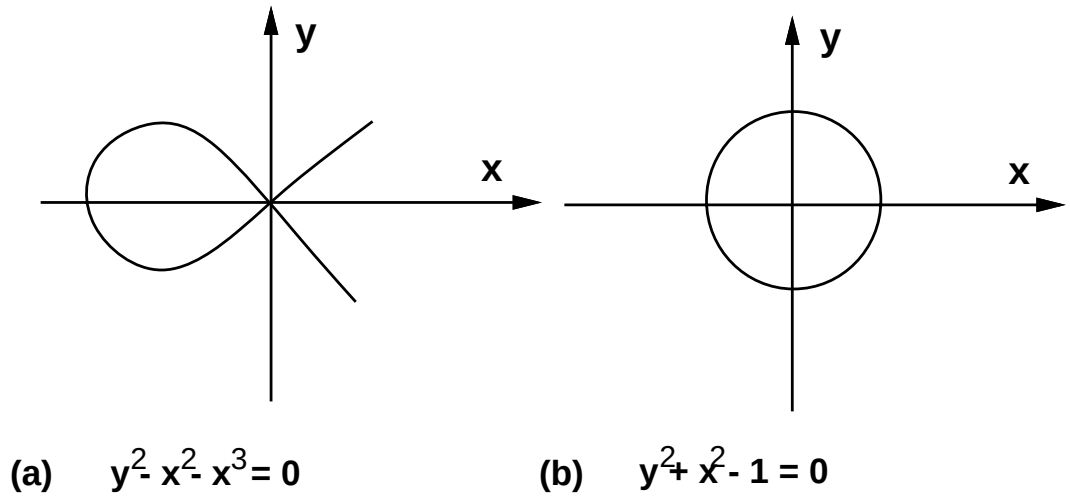


Figura 2.2 - Variedades Algébricas.

O conjunto dos pontos regulares de dimensão máxima de uma VA — daqui em diante, referidos simplesmente como pontos regulares — pode ser decomposto em subconjuntos conexos, contidos em alguma variedade mergulhada no $\Re^n$. Cada um destes subconjuntos é chamado **suporte** (fig. 2.3). A **dimensão de uma VA** é a dimensão dos seus pontos regulares e o seu grau é o grau do seu polinômio.

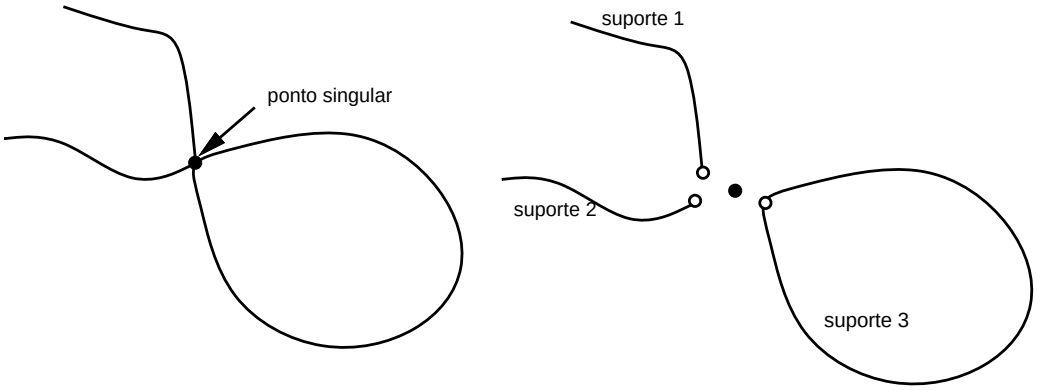


Figura 2.3 - Decomposição de uma VA.

O **fecho convexo** expandido pelos pontos $(p_0, p_1, ..., p_d)$ do $\Re^n$ é o conjunto:

$$[p_0, ..., p_d] = \{\sum_{k=0}^{d} \lambda_k p_k | \lambda_k \geq 0 \text{ e } \sum_{k=0}^{d} \lambda_k = 1\}.$$

Diz-se que $d+1$ pontos do $\Re^d$ são **linearmente independentes** quando os vetores $\{(p_1 - p_0), (p_2 - p_0), ..., (p_d - p_0)\}$ são linearmente independentes. Um **d-simplexo** é o fecho convexo de $d+1$ pontos linearmente independentes (no $\Re^2$ são triângulos e no $\Re^3$ são tetraedros). A **fronteira** de um d-simplexo S consiste de todos os $(d-k)$-simplexos contidos em S, $k > 0$. Cada simplexo na fronteira de S é chamado uma **face** de S.

Um **complexo simplicial** C é um conjunto finito de simplexos satisfazendo às seguintes condições:

- seja S um simplexo de C e seja S' uma de suas faces. Então $S' \in C$.
- sejam S_1 e S_2 dois simplexos de C. Então $S_1 \cap S_2$ é vazia ou é uma face comum.

Um subconjunto do $\Re^n$ é **triangulável** se for homeomorfo a um complexo simplicial. Um conjunto triangulável é chamado **poliedro topológico**. A dimensão de um complexo simplicial é a dimensão máxima dos seus simplexos.

3 - MODELO MATEMÁTICO PARA SUBDIVISÕES DO ESPAÇO

A modelagem sólida é um ramo da Modelagem Geométrica que trata do problema de construir representações não ambíguas para objetos sólidos físicos e os algoritmos que processam estas representações. Boa parte do esforço inicial no seu desenvolvimento foi gasto para encontrar representações válidas e completas e produzir algoritmos corretos. Os sólidos eram modelados, matematicamente, por conjuntos-r, que são subconjuntos limitados, fechados, regulares e semi-analíticos do $\Re^3$. Muitas vezes, as fronteiras dos sólidos tinham que ser homeomorfas a variedades de dimensão 2. A justificativa para estas restrições era a de que um sólido realizável fisicamente sempre as satisfariam. Mas esta justificativa era motivada, muito mais, pelas limitações das representações existentes, do que por uma preocupação com a realização do sólido.

Os esquemas de representação de sólidos mais utilizados foram, e continuam sendo, CSG e BRep. Com o passar do tempo, a evolução do *hardware* e a demanda por novas aplicações, que não se enquadravam em ambas as representações, motivaram a criação de novos esquemas capazes de representar uma subdivisão do espaço.

Por vários anos, subdivisões do espaço foram usadas como uma forma de representação auxiliar, ou em aplicações especializadas, como representação de malhas para modelos de elementos finitos. Atualmente, elas estão sendo utilizadas em aplicações de automação de manufatura para representar, não apenas um, mas um conjunto (agregado) de sólidos. Para este propósito, é necessário que seja possível modelar tais agregados, mantendo os relacionamentos de adjacência entre os objetos (fig. 3.1).

A capacidade de criar subdivisões do espaço também permite modelar efetivamente objetos compostos de diversos materiais. O contato entre estes objetos é representado explicitamente. As estruturas de dados tradicionais, criadas para serem empregadas em modelagem *non-manifold*, são capazes de representar agregados de objetos mas, só recentemente, um tratamento conceitual mais cuidadoso foi desenvolvido.

Várias definições têm sido propostas para estender a noção de sólido. Algumas são baseadas na definição matemática de complexos simpliciais [SPAN66], ou conceitos

relacionados, suportando conjuntos de pontos que não são necessariamente homogêneos em dimensão. Estas extensões procuram representar, em um mesmo modelo, arames (conjuntos de curvas), retalhos de superfícies (conjuntos de faces), regiões (conjuntos de volumes) e, até mesmo, conjuntos de dimensões maiores.

Um modelo matemático bastante geral para representar conjuntos de pontos de dimensão arbitrária, com estrutura interna e com fronteira com elementos pendentes (*dangling*), foi proposto por Rossignac & O'Connor [ROSS90] e baseia-se no conceito de complexo geométrico seletivo.

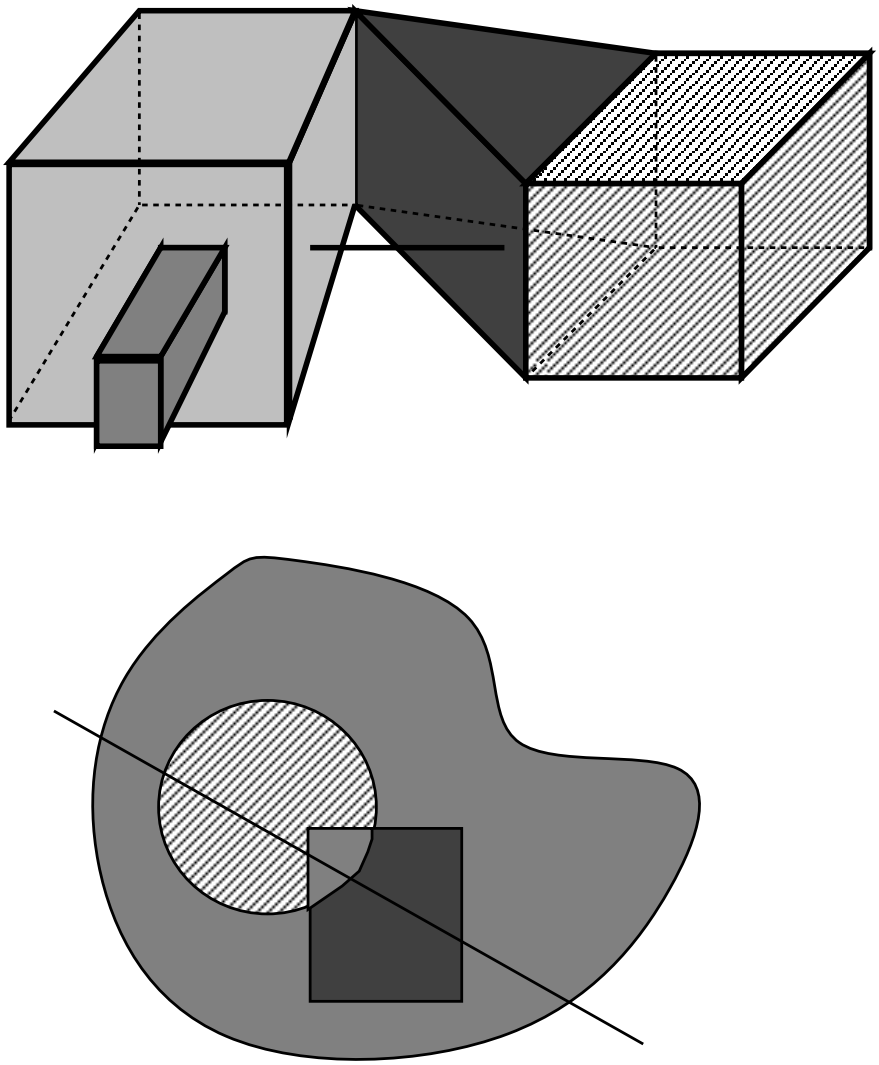


Figura 3.1 - Agregado de objetos.

3.1 Complexos Geométricos

Nesta seção, apresenta-se o conceito de complexo geométrico, criado por Rossignac & O'Connor [ROSS90], que define, com precisão, uma família de subdivisões do espaço $n-$dimensional que é apropriada para modelagem. Na seção 3.2, são apresentadas algumas extensões, desenvolvidas neste trabalho, necessárias à caracterização das subdivisões do espaço que são tratadas nos capítulos subseqüentes.

Um **complexo geométrico**, ou simplesmente complexo, é um conjunto finito de células mutuamente disjuntas, satisfazendo determinadas condições. Uma **célula** é um subconjunto **conexo e relativamente aberto** de algum suporte de uma VA.

A exigência das células possuírem suportes algébricos, se deve a existência de métodos computacionais para tratar esta classe de funções. Além do mais, funções algébricas têm sido utilizadas satisfatoriamente, até hoje, para modelar curvas e superfícies do mundo real. No entanto, qualquer outra classe de funções, de variação limitada, poderia ter sido utilizada; por exemplo, funções analíticas.

O conjunto de pontos representado por uma célula α é o interior de uma região do seu suporte tendo como fronteira $\partial\alpha$, o conjunto dos pontos do fecho de α que não pertencem a α. A cada célula está associado o único suporte, α.suporte, que a contém como um conjunto aberto **relativo**. Formalmente, um complexo K é, por definição, um conjunto **finito** de células $c_j, j \in J$, satisfazendo às seguintes condições:

(1) $i, j \in J \Rightarrow c_i \cap c_j = \emptyset$.

(2) $\forall c \in K$, $\exists l \subset J$ tal que $\partial c = \cup_{i \in l}(c_i)$.

(3) $\forall b \in \text{bdry}(c, K)$, ($b \subset c$.suporte) ou ($b \cap c$.suporte $= \emptyset$).

Os símbolos $\cap$ e $\cup$ denotam a interseção e a união dos conjuntos de pontos correspondentes às células apropriadas. O operador **bdry**(c, K) denota o conjunto de células c_i de K tais que $c_i \subset \partial c$. Diz-se que c é **delimitada** por cada c_i, ou, então, que cada c_i delimita c. O operador **star**(c, K) denota o conjunto de células c_i de K tais que $c \in \partial c_i$. Diz-se que cada c_i é **incidente** a c (fig. 3.2).

Na definição acima, a primeira condição determina simplesmente que os conjuntos

de pontos de duas células quaisquer devem ser disjuntos. A segunda condição é muito importante e é, justamente ela, que garante que toda célula, em um complexo, possui fronteiras "bem comportadas". Esta condição obriga que as fronteiras das células estejam no complexo e sejam formadas por uma ou mais células (fig. 3.2). Por conseguinte, toda célula possui fronteiras algébricas por partes. A terceira condição determina que cada célula, na fronteira de uma outra célula c, deve estar contida no suporte de c, ou, então, sua interseção com o suporte de c deve ser vazia.

A figura 2.2 pode ser encarada como um complexo com quatro células. Embora o ponto singular corresponda a uma célula que está na fronteira de todas as outras, sua interseção com todos os suportes das outras células é vazia.

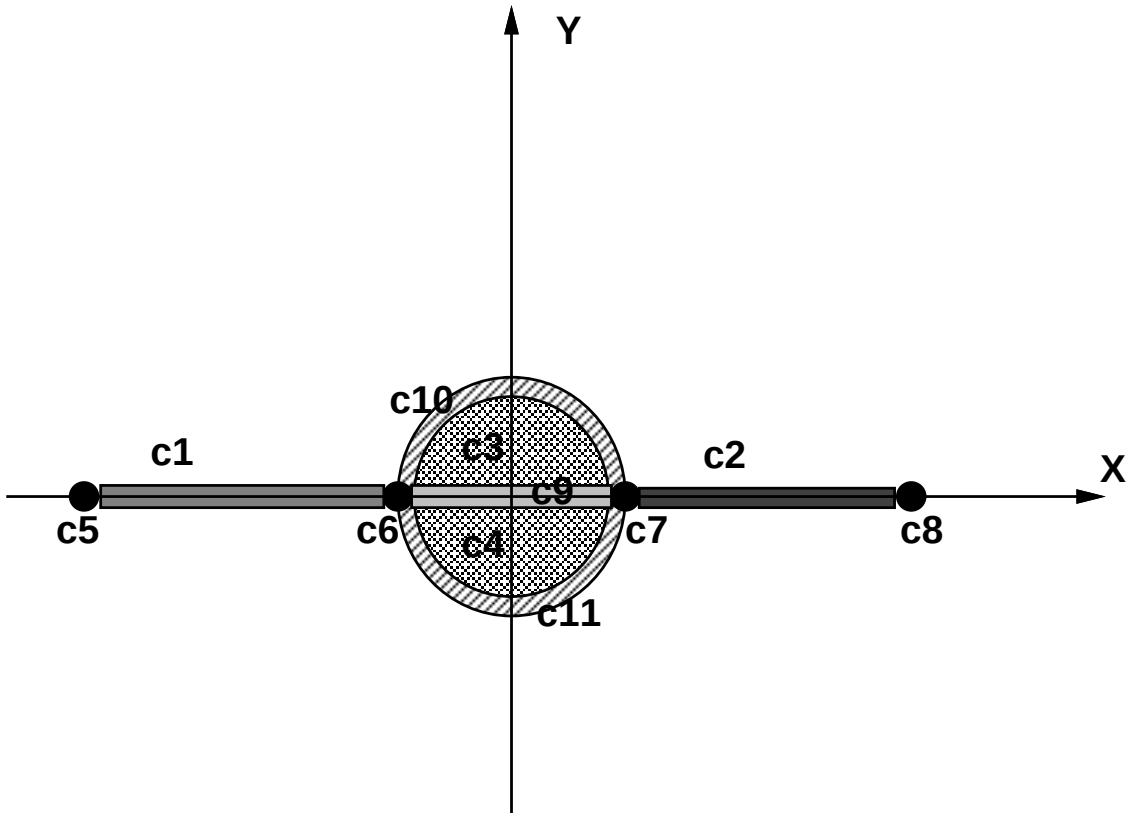
Da definição de complexo, segue que a união de todas as células de um complexo representa um conjunto fechado de pontos e que, dado um conjunto X de células de um complexo K, $X \cup \text{bdry}(X, K)$ é também um complexo. Além disso, células possuem fronteiras algébricas (por partes).

A **dimensão de uma célula** α, denotada por $\alpha.\text{dim}$, é definida como sendo a dimensão do seu suporte, e a dimensão de um complexo é definida como a dimensão máxima das suas células. Denota-se por $K.\text{cells}(d)$ o conjunto de células de dimensão d do complexo K.

Dois complexos A e B são **compatíveis** (fig. 3.4) se:

$$\forall a \in A, \forall b \in B, a \cap b \neq \emptyset \Rightarrow (a = b).$$

Um complexo A é um **refinamento** de um complexo B quando toda célula de B pode ser obtida pela união de células de A.



c_1.suporte $= c_2$.suporte $= c_9$.suporte $\rightarrow y = 0$

c_{10}.suporte $= c_{11}$.suporte $\rightarrow x^2 + y^2 - 1 = 0$

c_3.suporte $= c_4$.suporte $\rightarrow 0x + 0y = 0$ $(\Re^2)$

c_5.suporte $\rightarrow (-3, 0)$, $\quad c_6$.suporte $\rightarrow (-1, 0)$

c_7.suporte $\rightarrow (1, 0)$, $\quad c_8$.suporte $\rightarrow (3, 0)$

$\text{star}(c_6, K) = \{c_{10}, c_1, c_{11}, c_9, c_3, c_4\}$

$\text{star}(c_9, K) = \{c_3, c_4\}$

$\text{bdry}(c_1, K) = \{c_5, c_6\}$

$\text{bdry}(c_3, K) = \{c_9, c_{10}, c_6, c_7\}$

$\text{adj}(c_6, K) = \{c_5, c_7\}$

$\text{adj}(c_9, K) = \{c_2, c_{10}, c_{11}, c_1\}$

Figura 3.2 - Um complexo geométrico.

O **esqueleto** de ordem j de um complexo K, denotado por K.esqueleto(j), é o conjunto de todas as células de K com dimensão menor ou igual a j (fig. 3.3). É fácil verificar que K.esqueleto(j) é um complexo válido e que dois complexos compatíveis possuem todos os esqueletos compatíveis.

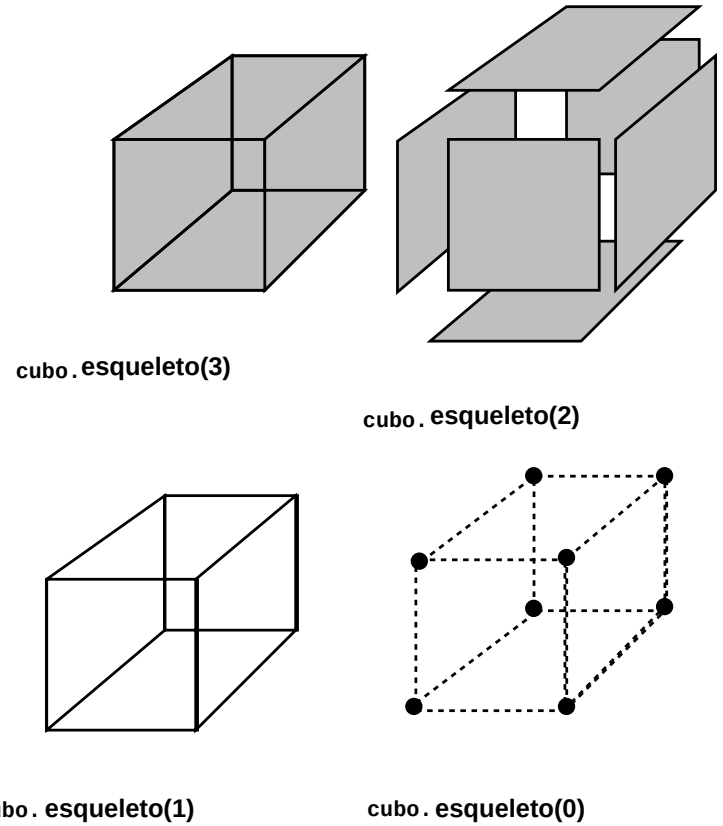


Figura 3.3 - Os esqueletos de um cubo.

3.1.1 Complexos Geométricos Seletivos

Um **complexo geométrico seletivo**, ou SGC, Λ é composto por um complexo, denotado por Λ.complexo, e por um conjunto de atributos associados a cada célula (ou grupo de células). Em particular, existe um atributo, chamado `status`, que especifica se o conjunto de pontos definido pela célula deve ser incluído no conjunto de pontos definido pelo SGC. Conseqüentemente, o conjunto de pontos associado a um SGC Λ, denotado por Λ.pontos, é definido como a união de todos os conjuntos de pontos das células c de Λ.complexo, tais que c.`status` = ACTIVE. O atributo `status` permite que um SGC represente um conjunto de pontos que não é necessariamente fechado.

Dado um SGC Λ, entende-se por **simplificação**, o processo de encontrar um outro SGC Λ' que represente o mesmo conjunto de pontos, mas que tenha o menor número de células possível. O objetivo da simplificação é reduzir o espaço necessário para armazenar um complexo. Basicamente, existem três operações de simplificação (fig. 3.4):

- eliminação - remove uma célula inativa (`status` = INACTIVE);
- colagem - remove uma célula, na fronteira (comum) de duas outras células ativas, contidas no mesmo suporte, criando uma única célula;
- incorporação - remove uma célula que pertence a fronteira de uma única célula.

Uma operação de simplificação só pode ser executada se preservar a validade geométrica do complexo e o conjunto de pontos representado. As operações de simplificação que alterem o conjunto de pontos do SGC devem ser evitadas. Para isto, basta prevenir-se contra:

- eliminação de uma célula que esteja na fronteira de pelo menos uma célula ativa;
- operações de colagem quando as três células envolvidas não estão ativas;
- operações de incorporação quando ambas as células envolvidas não estão ativas.

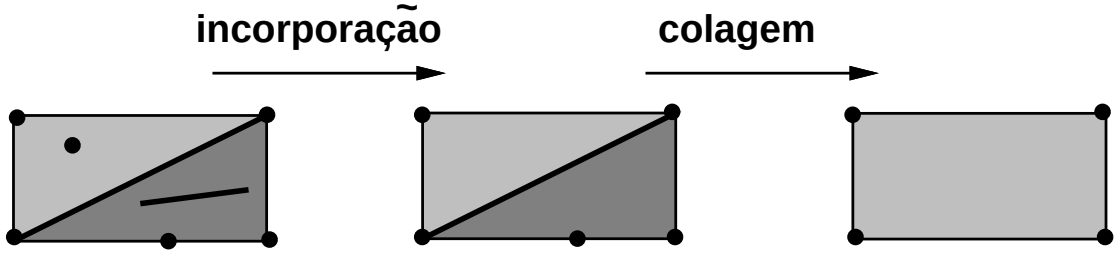


Figura 3.4 - Simplificação.

3.2 Subdivisões do Espaço

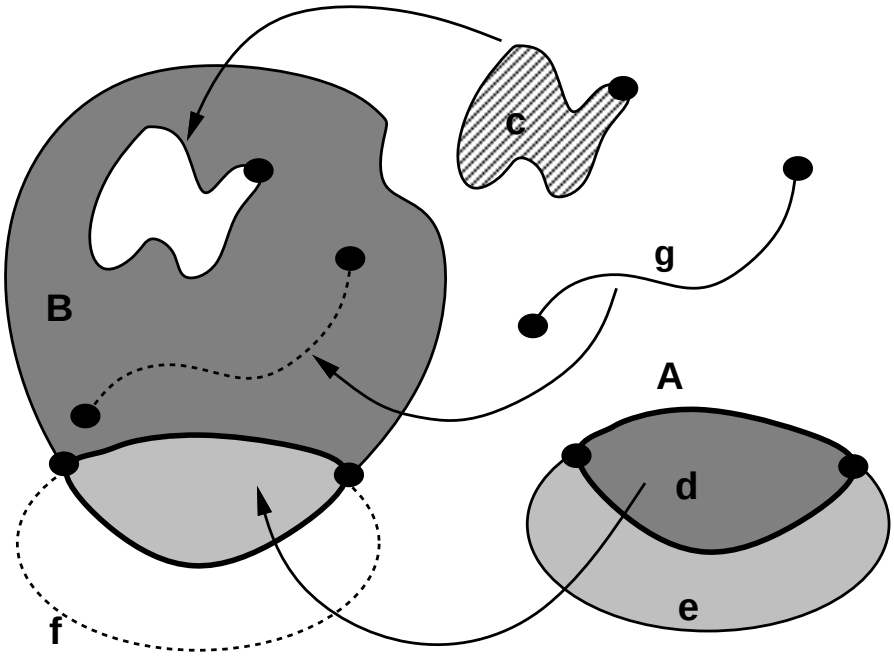
O objetivo desta seção é apresentar algumas extensões, desenvolvidas neste trabalho, do conceito de SGC, de modo a caracterizar exatamente o tipo de objeto que pode ser representado pelas estruturas de dados apresentadas nos capítulos 4 e 5. Para tal, criou-se o conceito de complexo completo e um novo operador, chamado adj. Além disso, como os processos de criação de subdivisões, descritos nestes capítulos, têm como passo fundamental uma etapa de compatibilização dos esqueletos de complexos, criou-se, também, o conceito de encaixe de complexos.

Definição: Um complexo do $\Re^n$ é **completo** quando a união de todas as suas células é o próprio $\Re^n$. Um complexo completo do $\Re^n$ tem, necessariamente, dimensão n.

Conclui-se, então, que dois complexos completos A e B são compatíveis se, e somente se, $A = B$. Esta afirmação é verdadeira porque, se os complexos são completos, a interseção de uma célula qualquer de A com B nunca é vazia. Logo, se A é compatível com B, para cada célula de A, deve haver uma célula idêntica em B e vice-versa.

Definição: Diz-se que um complexo A **encaixa** em um complexo B, ambos de dimensão n, quando A.esqueleto$(n-1)$ é compatível com B.esqueleto$(n-1)$. Uma célula α de A, de dimensão menor que n, encaixa em B quando satisfaz às seguintes condições:

- $\forall c \in \text{bdry}(\alpha, A)$, $c \in B$;
- $\forall c \in B$.esqueleto$(n-1)$, $(\alpha \subset c)$ ou $(\alpha \cap c = \emptyset)$.



As células c, d e g encaixam no complexo B.

A célula e encaixa em B se a célula f pertencer a B.

O complexo A é compatível com o complexo B.

Figura 3.5 - Compatibilidade de complexos e encaixe de células.

Seja c uma célula, de dimensão n, de um complexo K. O operador **adj**(c, K) denota o conjunto de células c_i de dimensão n, onde $c_i \in \text{star}(c_j, K)$ para qualquer $c_j \in \text{bdry}(c, K)$. Costuma-se exigir que c_j tenha dimensão $n-1$. Para $n = 0$, adj(c, K) denota o conjunto de células c_i de dimensão 0, onde $c_i \in \text{bdry}(c_j, K)$ para qualquer $c_j \in \text{star}(c, K)$, c_j de dimensão 1. As células de adj(c, K) são ditas **adjacentes** a c (fig. 3.2).

Existe uma forte relação entre um complexo completo de dimensão n e o seu esqueleto de ordem $(n-1)$. Esta relação fica clara a partir do seguinte teorema:

Teorema: A um complexo geométrico do $\Re^n$ sempre pode ser adicionado um conjunto de células de dimensão máxima de forma que o complexo estendido seja completo.

Prova: Esta afirmação é verdadeira porque o complemento $\overline{A}$ de um conjunto fechado A é vazio, ou é um conjunto aberto com a mesma fronteira de A. Logo, não sendo vazio, cada componente conexa de $\overline{A}$ é um conjunto aberto, cuja fronteira pertence a A. Cada uma destas componentes é de dimensão máxima pois, sendo aberta, contém uma vizinhança de cada um de seus pontos. Assim, basta acrescentar cada uma destas componentes ao complexo para completá-lo. ∎

Corolário: Como conseqüência, todo complexo completo de dimensão n fica inteiramente caracterizado pelo seu esqueleto de ordem $(n-1)$. Em outras palavras, dado um complexo de dimensão n, existe um único conjunto de células de dimensão máxima capaz de completá-lo. ∎

As chamadas representações por fronteira para sólidos representam, justamente, os esqueletos de ordem 2 de alguns complexos de dimensão 3. Como vai-se ver, nos capítulos 4 e 5, uma maneira de criar um complexo completo é através de operadores que constroem o seu esqueleto. Os próprios operadores completam o complexo, automaticamente, à medida que se faz necessário.

Neste trabalho, interessam apenas os SGCs completos e que tenham uma, e somente uma, célula ilimitada, cuja dimensão é máxima. Portanto, vai-se adotar a seguinte definição:

Definição: Uma SE^n é um SGC, de dimensão n, completo e que possui apenas uma célula ilimitada (esta célula tem dimensão n). Uma SE^2 é chamada subdivisão planar ou **SP**. Uma SE^3 é chamada subdivisão espacial ou **SE**.

As células de uma SP são conjuntos relativamente abertos e conexos: células de dimensão 2 ou faces, cujas fronteiras são formadas por um ou mais ciclos (conjuntos ordenados de arestas ou vértices); de dimensão 1 ou arestas (segmentos de curvas uni-dimensionais), cujas fronteiras são constituídas por um ou dois vértices; e de dimensão 0 ou vértices (simplesmente pontos do $\Re^2$).

As células de uma SE[4] são conjuntos relativamente abertos e conexos: células de dimensão 3 ou regiões, cujas fronteiras são formadas por uma ou mais cascas (conjuntos de faces e/ou arestas ou vértices); de dimensão 2 ou faces (retalhos de superfícies bi-dimensionais), cujas fronteiras são formadas por um ou mais ciclos (conjuntos ordenados de arestas ou vértices); de dimensão 1 ou arestas (segmentos de curvas uni-dimensionais), cujas fronteiras são constituídas por um ou dois vértices; e de dimensão 0 ou vértices (simplesmente pontos do $\Re^3$).

Quando Rossignac criou o SGC ele não se procupou em manter na representação, por ele proposta, informações de orientação e ordenação de células. Isto permite que sejam utilizadas estruturas de dados simples, que armazenam, basicamente, apenas o bdry e o star das células.

A orientabilidade é uma propriedade topológica de uma variedade. Quando a variedade está mergulhada, a orientabilidade está relacionada com a propriedade de separação do espaço (uma variedade de co-dimensão 1 separa, localmente, o espaço).

No caso geral, de dimensão n, somente uma pequena parcela dos relacionamentos de adjacência pode ser representada ordenadamente. Apenas as células do bdry de células de dimensão 1 e 2 podem ser representadas de modo ordenado (isto é, vértices que delimitam arestas ou arestas que delimitam faces). Por outro lado, somente o star

[4] Note-se que uma SE é capaz de representar conjuntos de pontos que não são necessariamente homogêneos em dimensão. Por exemplo, sólidos combinados com faces e/ou arestas pendentes.

de células de dimensão $n-2$ e $n-1$ pode ser ordenado. Isto porque, no $\Re^n$, células de dimensão inferior a $n-1$ não são capazes de dividir o espaço. Isto é conseqüência do fato de que apenas uma superfície de co-dimensão 1 pode dividir, localmente, o espaço no qual ela está mergulhada (superfícies mergulhadas no espaço tri-dimensional, curvas mergulhadas em superfícies e pontos em curvas). Por exemplo, arestas que delimitam apenas regiões (chamadas, neste caso, de arames) não podem, em geral, ser ordenadas.

Já nos espaços de dimensão 2 e 3 os relacionamentos ordenáveis correspondem a uma parcela considerável do total dos relacionamentos de adjacência. Especificamente, no $\Re^2$ é possível ordenar as arestas em torno de um vértice e, por conseguinte, as faces (delimitadas por estas arestas) que incidem no vértice. Também é possível orientar uma face, ordenando as células (vértices e arestas) na sua fronteira. No $\Re^3$, ainda é possível manter as faces orientadas e o star de uma aresta (as faces e as regiões delimitadas por estas faces) pode ser ordenado radialmente ao redor da aresta. Porém, não há mais, em geral, uma ordenação possível para as células que incidem nos vértices.

Um outro ponto importante, como vai ser mostrado a seguir, é que um complexo geométrico é um poliedro topológico (ou seja, pode ser triangulado). Isto é o que garante que é possível orientar, coerentemente, a fronteira das suas faces (conforme definido no apêndice).

De certa forma, a orientabilidade estabelece uma estrutura para o mergulho de elementos, puramente topológicos, no espaço Euclidiano, o que permite utilizar estruturas de dados topológicas para representá-las. Esta estrutura tem-se mostrado bastante útil no desenvolvimento de vários esquemas de representação eficientes, como pode ser visto nos capítulos 4 e 5.

Para provar que um complexo é triangulável, é preciso conhecer as propriedades topológicas de conjuntos algébricos. A estrutura topológica de conjuntos analíticos (em particular, algébricos) vem sendo estudada desde a década de 30. Os primeiros trabalhos — onde são apresentadas provas detalhadas a respeito da triangulabilidade destes conjuntos — datam do início da década de 60. Lojasiewicz [LOJA64] mostrou que qualquer coleção finita de subconjuntos do $\Re^n$, localmente semi-analíticos, são trianguláveis

simultaneamente. Os conjuntos semi-analíticos (semi-algébricos) são aqueles que podem ser dados localmente por inequações analíticas (algébricas). A demonstração deste resultado é bastante técnica e foge ao escopo deste trabalho. No entanto, em dimensão 2, pode-se utilizar um processo construtivo simples, descrito a seguir, para decompor um complexo geométrico, de tal forma que as células deste novo complexo sejam homeomorfas ao disco aberto. Neste caso, diz-se tratar de um complexo celular. É um fato conhecido que complexos celulares são trianguláveis.

Teorema: Todo complexo geométrico de dimensão 2 pode ser decomposto formando um outro complexo cujas células são homeomorfas ao disco aberto.

Prova: A demonstração deste teorema utiliza um processo construtivo e será feita considerando o complexo no $\Re^2$. Considere, sem perda de generalidade, que toda aresta é monótona (um segmento de curva é monótono quando ao ser percorrido, de uma extremidade para a outra, sua projeção, no eixo y, produz valores constantes, ou estritamente crescentes ou decrescentes), uma vez que qualquer complexo K pode ser refinado pela criação de vértices nos pontos de cada aresta α de K, onde:

$$\frac{d(\alpha.\text{suporte})}{dx}|_{x=x_0} = 0, \quad x_0 \in \alpha$$

(desta forma, toda aresta de K torna-se monótona).

Corte, então, o complexo por retas horizontais, paralelas ao eixo x, passando por cada um de seus vértices. Sempre que uma destas retas cortar uma aresta, crie um vértice no ponto de interseção dividindo a aresta em duas. Em seguida, para cada face, encontre e ordene os vértices que estejam sobre cada uma das retas. Entre cada par de vértices consecutivos (e não ligados), introduza uma nova aresta. Se, ao ser introduzida uma nova aresta em uma face, dois pontos quaisquer da face não puderem mais ser ligados por um caminho, inteiramente contido na face, sem que este caminho intercepte a nova aresta, então uma nova face foi criada. Adicione esta nova face ao complexo (fig. 3.6).

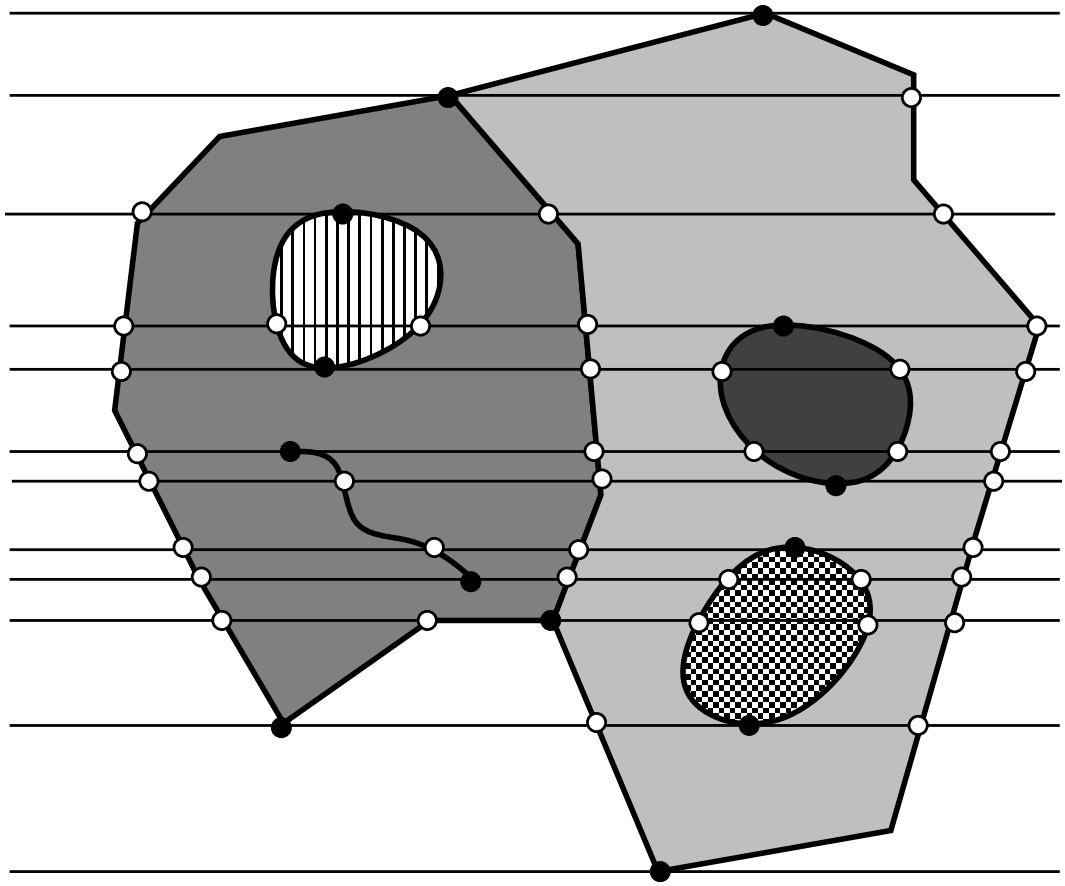


Figura 3.6 - Decomposição de um complexo no $\Re^2$.

Este processo gera uma decomposição do modelo. Esta decomposição produz um novo complexo porque, a cada passo, são respeitadas as condições que definem um complexo (na realidade, o novo complexo é um refinamento do complexo original). Cada face deste novo complexo possui dois, três ou quatro lados (os lados superiores e inferiores formados pelos conjuntos de arestas horizontais e os laterais por segmentos de curvas algébricas). Desta forma, toda face do novo complexo possui, como fronteira, uma curva fechada, conexa, limitada, contínua e sem auto-interseção, ou, então, é uma face ilimitada.

No primeiro caso, pelo teorema de Jordan [HENL79], qualquer uma das curvas divide o plano em duas regiões conexas: uma limitada e homeomorfa ao disco e a outra ilimitada. Como nenhuma face possui buracos (caso contrário, certamente haveria pelo menos um vértice no seu interior e, por construção, passaria uma reta por ele que cortaria a fronteira da face), conclui-se que cada face é homeomorfa ao disco.

No segundo caso, as faces podem ser semi-planos, faixas do plano limitadas apenas por duas retas horizontais ou faixas do plano limitadas em um dos lados por um segmento algébrico. Faixas do plano e semi-planos são trivialmente homeomorfos ao disco. Se a faixa for limitada em um dos lados por um segmento algébrico, aplica-se um homeomorfismo de forma a limitá-la no outro lado por um segmento vertical. Isto cria uma curva de Jordan e vale o argumento aplicado ao primeiro caso. ∎

É possível generalizar parcialmente o princípio da demonstração em dimensão 2, para utilizá-lo em dimensão 3: podem-se fazer cortes com planos paralelos ao plano xz e em seguida com planos paralelos ao plano xy. Cada um destes cortes fica determinado pelos vértices e pontos singulares (máximo local, mínimo local e pontos de sela), de cada face e cada aresta, em relação à respectiva direção. Nos pontos de interseção de um plano com uma aresta são criados novos vértices. As faces devem ser refinadas, conforme descrito no teorema para dimensão 2. Assim, cada região pode ter tido sua fronteira refinada pela inserção de novas arestas. Cada conjunto de novas arestas, na fronteira de uma região, contidas em um mesmo plano, forma a fronteira de uma face plana que deve ser adicionada. Se, ao ser introduzida uma nova face, dois pontos quaisquer da região não puderem mais ser ligados por um caminho, inteiramente contido na região, sem que intercepte a nova face, então uma nova região é formada e adicionada ao complexo.

O primeiro tipo de corte gera regiões delimitadas por superfícies homeomorfas à esfera, ou a um $n-$toro (fig 3.7a). O segundo tipo de corte garante que todas as superfícies são homeomorfas à esfera (fig. 3.7b). Pelo teorema de Jordan-Brouwer [GREE67], cada uma destas superfícies divide o espaço em duas regiões. Uma região ilimitada e outra limitada e homeomorfa à bola aberta, pois a superfície que a delimita é algébrica (por partes).

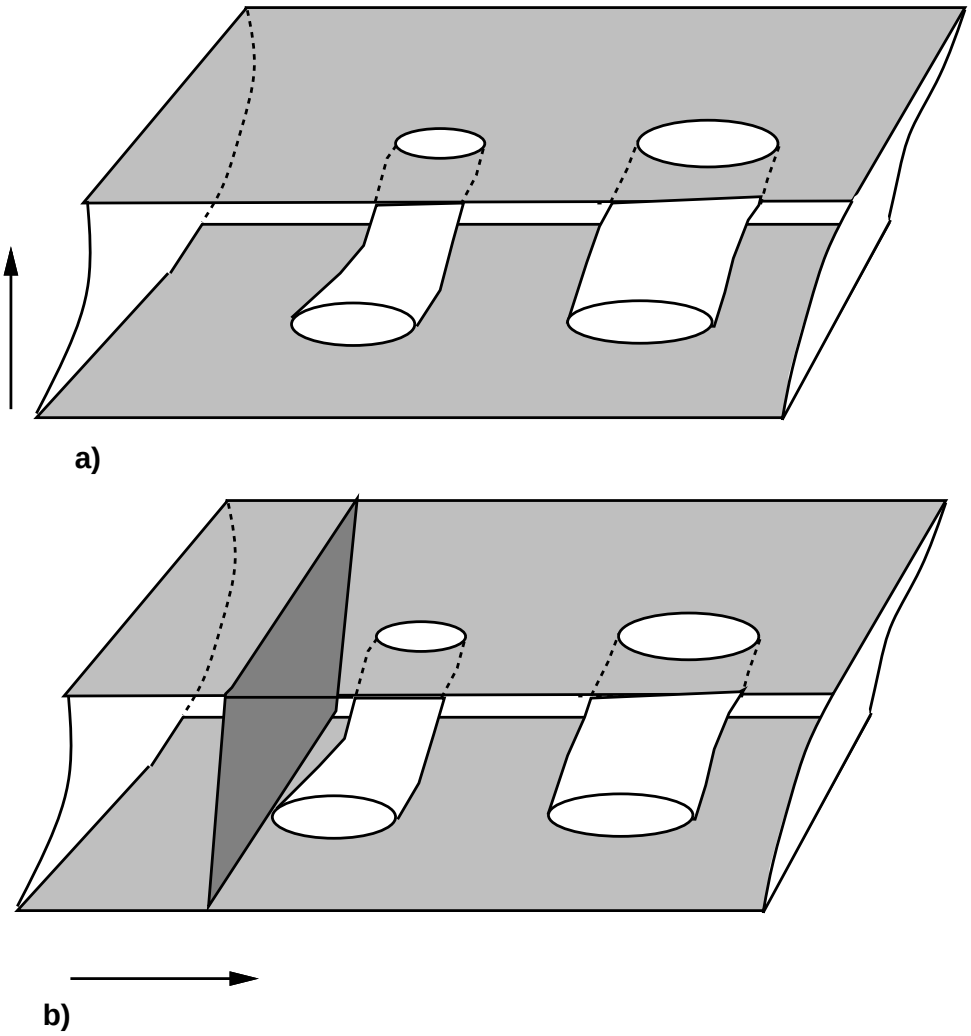


Figura 3.7 - Decomposição de um complexo no $\Re^3$.

Esta técnica, no entanto, não se aplica a qualquer complexo do $\Re^3$. Caso o modelo possua arestas que não estejam na fronteira de uma face, a técnica não pode ser utilizada. Isto porque, no $\Re^3$, uma curva não divide, localmente, o espaço. Neste caso, deve-se utilizar uma decomposição que utilize uma superfície que contenha a curva (o que é sempre possível, pois as curvas são algébricas).

Estes teoremas são importantes porque mostram que um complexo geométrico representa o mesmo tipo de objeto que podia ser representado pelos modelos de decomposição celular. A diferença fundamental é o que constitui uma célula em cada modelo. Um complexo geométrico evita uma fragmentação excessiva do modelo.

4 - SUBDIVISÕES PLANARES

Este capítulo trata do problema de criar e manter subdivisões do $\Re^2$. Uma subdivisão simples está apresentada na figura 4.1. Nesta subdivisão, o plano é dividido em três regiões (ou faces) limitadas (f_1, f_2, f_3) e uma região ilimitada (f_{ext}). Cada região é delimitada por um conjunto de segmentos de curva (ou arestas) que, por sua vez, são delimitados por um par de pontos (ou vértices) não necessariamente distintos.

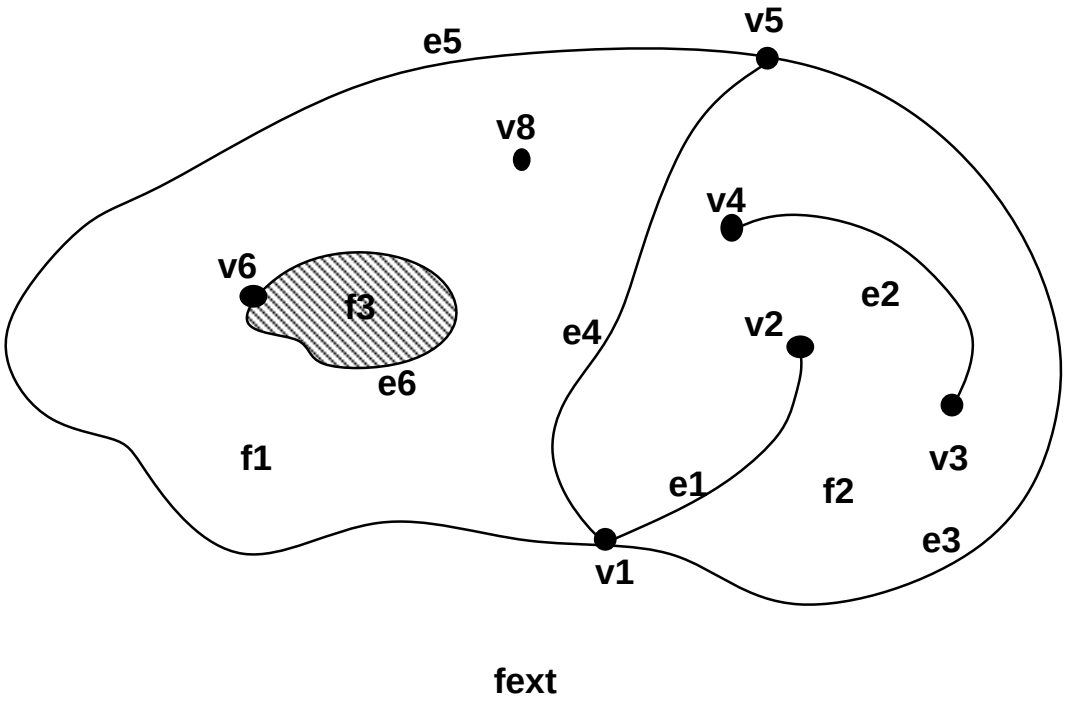


Figura 4.1 - Uma subdivisão do $\Re^2$.

Uma subdivisão do $\Re^2$ pode ser modelada por uma subdivisão planar (SP), apresentada no capítulo 3. Neste caso, as células do complexo, de dimensão 0, são os vértices; as de dimensão 1, as arestas; e as de dimensão 2, as faces.

Uma boa exemplificação de uma SP é um mapa geográfico (fig. 4.13). Um mapa é apresentado, em geral, sobre uma superfície plana que está dividida em uma série de regiões delimitadas por um conjunto de segmentos de curva que constituem as respectivas fronteiras. Existem outras aplicações que necessitam de representações similares; por exemplo: malhas para modelos de elementos finitos para seções planas; perfis geológicos; curvas de nível topográficas; plantas baixas; e grafos.

Um problema de grande interesse consiste em, dado um conjunto de segmentos de curva (ou simplesmente segmentos), obter a SP determinada por estes segmentos. Um segmento de curva é uma curva algébrica, limitada e sem auto-interseção. Para tal, é necessário calcular todos os pontos de interseção entre estes segmentos, criando vértices em cada um destes pontos, arestas entre cada par de vértices ligados por um segmento e faces, sempre que um conjunto de arestas formar uma área fechada.

Neste capítulo é apresentada uma **metodologia para criação de SPs** composta por um modelo matemático, um esquema de representação e um processo de modelagem [REQU80]. Esta metodologia foi utilizada para implementar o EDP (Editor de Diagramas Planares), que é capaz de criar e manter uma subdivisão planar interativamente, permitindo a inserção de novos segmentos em tempo real. Para lidar com a diversidade de geometrias de segmentos necessária às aplicações, foi identificado um conjunto de funções que permitem a geração de bibliotecas de geometrias manipuláveis pelo EDP. A introdução de uma nova geometria para os segmentos pode ser feita através da criação da biblioteca correspondente.

O problema de criar uma subdivisão planar foi abordado sobre três aspectos: o da correção dos algoritmos, inseridos em um mundo ideal e supondo-se aritmética real com precisão infinita; o da robustez dos algoritmos, onde o problema é produzir um resultado aceitável, mesmo lidando com dados imprecisos e trabalhando com aritmética de ponto flutuante; e o da eficiência dos algoritmos adotados.

4.1 Representação para SPs

Uma **representação**, em um computador, consiste em um conjunto de estruturas de dados e num conjunto de procedimentos e operadores que as manipulam. O **domínio** da representação é o conjunto de possibilidades para o qual ela é válida. Uma representação está correta quando é suficiente sobre o seu domínio e os seus operadores não criam configurações que não pertencem ao domínio.

No caso de uma SP, uma descrição geométrica completa pode representar todas as informações sobre as formas geométricas das células e suas respectivas localizações

no plano. Uma representação puramente geométrica, no entanto, não é muito interessante. Especificamente, considerem-se dois segmentos que se interceptam. O ponto de interseção, embora possa ser encontrado a partir da descrição geométrica de cada segmento, é um ponto especial que exige um processamento, em geral complicado, para sua determinação. Uma representação para SPs que contenha, de forma explícita, todas as interseções, não só permite que os cálculos destas interseções sejam feitos uma única vez, no momento da sua criação, mas também evita a propagação de erros numéricos. O ferramental necessário para a construção deste tipo de representação é fornecido pela **Topologia**.

A Topologia fornece abstrações que podem ser utilizadas para especificar as vizinhanças (adjacências) das células de uma SP. Formalmente, Topologia é o estudo das propriedades invariantes por homeomorfismos. Como as adjacências são invariantes por homeomorfismos, pode-se falar em uma topologia de adjacências.

No contexto de SPs, quando se pensa em topologia, normalmente está-se referindo às adjacências entre as células, ou seja, proximidade física e ordenação. Informações de adjacência são referidas, informalmente, como a topologia da SP. A principal vantagem de armazenar, explicitamente, na representação de uma SP, a sua topologia é que isto possibilita a construção de algoritmos geométricos muito mais eficientes [GUNT88]. A nomenclatura utilizada para descrever a topologia de uma SP é a seguinte (capítulo 3):

- Um **vértice** é uma célula, de dimensão 0, que corresponde a um único ponto do $\Re^2$. Não podem existir dois vértices com a mesma localização geométrica.
- Uma **aresta** é uma célula, de dimensão 1, que corresponde a um segmento de curva homeomorfo a um segmento de reta. A fronteira de uma aresta é formada por dois vértices não necessariamente distintos. Estes dois vértices são ditos ligados.
- Um **ciclo** é um subconjunto conexo e ordenado de vértices e arestas alternadamente. Um ciclo pode consistir em um único vértice, sendo chamado, neste caso, de ciclo pontual.
- Uma **face** é uma célula, de dimensão 2, que corresponde a um subconjunto conexo do $\Re^2$. Uma face é orientada e sua fronteira é composta por um ou mais ciclos. Um dos ciclos contém os demais e representa a sua **fronteira externa**. Os outros

ciclos (se existirem) são chamados anéis e representam fronteiras internas.

Em uma SP é possível formar nove tipos de relacionamentos de adjacência que caracterizam as adjacências entre vértices, faces e arestas [WEIL83]. Cada relacionamento é representado por um par de letras; a primeira indica o tipo do elemento de referência; a segunda simboliza o grupo de elementos adjacentes ao elemento de referência. É utilizada a seguinte notação:

- <> indica que o grupo de adjacência pode ser ordenado ciclicamente e {} indica que não há ordenação.
- Letra minúscula especifica uma certa célula e maiúscula um conjunto de células.
- A cardinalidade de um grupo de adjacência é expressa por um expoente.

Cada relacionamento de adjacência permite obter um certo tipo de adjacência entre células (fig. 4.2).

$v < V >$ - denota o conjunto de vértices adjacentes (ligados) a v.

$v < A >$ - **denota o conjunto de arestas que incidem em** v.

$v < F >$ - denota o conjunto de faces que incidem em v.

$f < V >$ - denota o conjunto de vértices que delimitam f.

$f < A >$ - **denota o conjunto de arestas que delimitam** f.

$f < F >$ - denota o conjunto de faces adjacentes a f.

$a\{V\}^2$ - denota os dois vértices que delimitam a.

$a\{A\}$ - **denota o conjunto de arestas adjacentes a** a.

$a\{F\}^2$ - denota as duas faces que incidem em a.

Tabela 4.1 - Tipos de relacionamentos de adjacência.

Na terminologia do capítulo 3, cada grupo de adjacência é um subconjunto de células, de mesma dimensão, de bdry, star ou adj.

bdry(a,SP) $= a\{V\}^2$	bdry(f,SP) $= f < V > \cup f < A >$
star(a,SP) $= a\{F\}^2$	star(v,SP) $= v < A > \cup v < F >$
adj(v,SP) $= v < V >$	adj(a,SP) $= a\{A\}$
adj(f,SP) $= f < F >$	

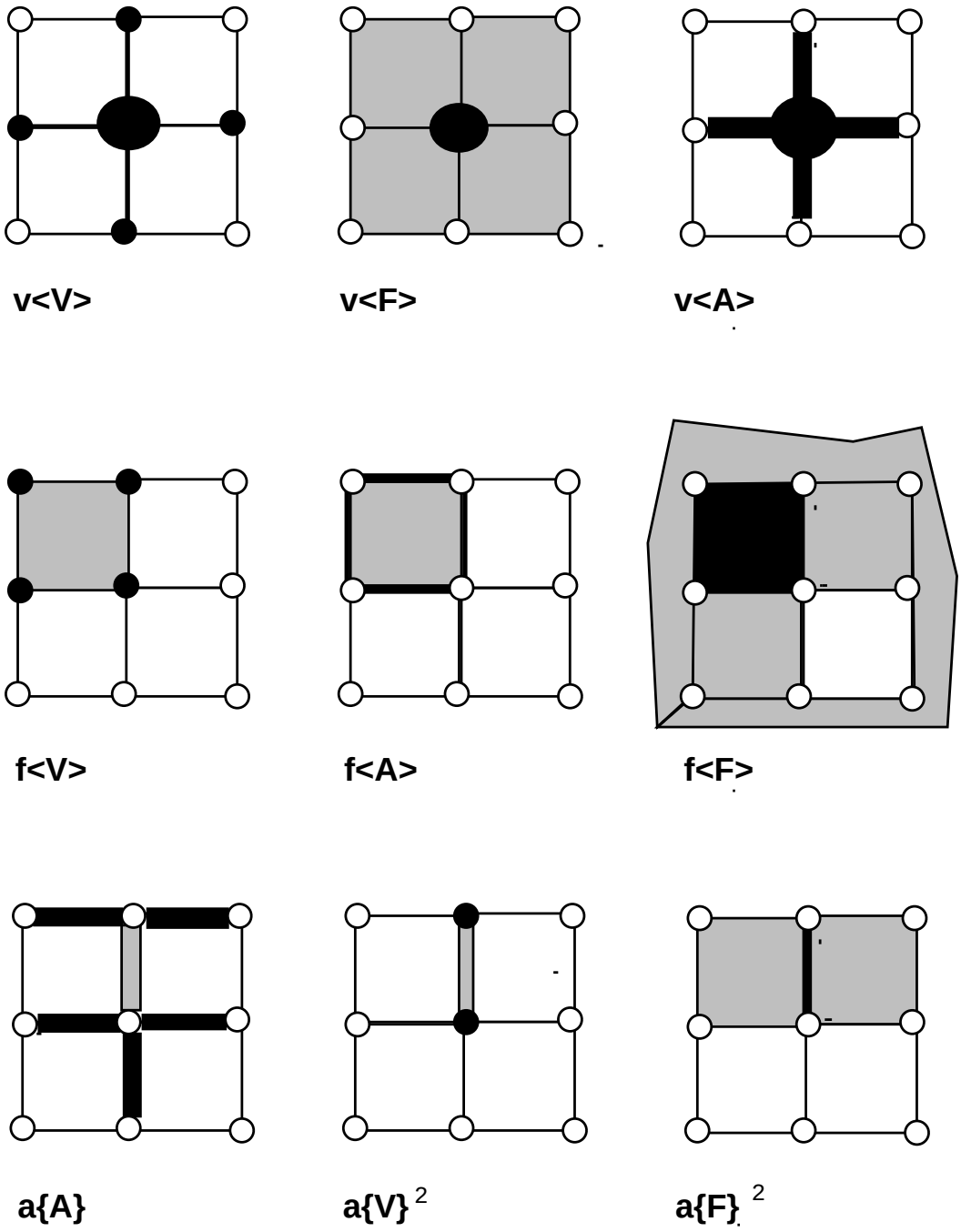


Figura 4.2 - Relacionamentos de adjacência.

A **suficiência topológica** de uma representação é a capacidade de representar, completamente e de forma não ambígua, topologias de adjacência. **Completeza** é a capacidade de gerar todas as informações topológicas a partir da representação. **Não ambigüidade** implica que, para um único conjunto de dados da representação, existe

um único conjunto de informação topológica que pode resultar da interpretação da representação, ou seja, há um mapeamento bi-unívoco entre a representação e a informação topológica.

A **suficência de informação**, de uma representação topológica, é a capacidade de extrair todos os relacionamentos de adjacência entre as células de uma SP. Deve-se frisar que, quando se fala em suficiência de informação, não se está pressupondo o uso de nenhum formato particular para armazenamento dos dados.

Considerando-se uma topologia de adjacência com apenas vértices, faces e arestas, suficiência é a capacidade de recriar todos os nove tipos de relacionamentos de adjacência sem erro ou ambigüidade. No contexto, é a capacidade de um subconjunto dos relacionamentos de adjacência, entre as células de uma SP, prover informação suficiente para reproduzir, única e completamente, a sua topologia. Nesta definição, não é permitido o uso da geometria associada às células para derivação de qualquer informação topológica. Entretanto, não é necessário armazenar informações sobre todos os tipos de relacionamentos de adjacência, uma vez que existem relacionamentos que são individualmente suficientes. Os relacionamentos em negrito na tabela 4.1 são suficientes [WEIL86].

Uma estrutura de dados topológica é suficiente quando a informação nela armazenada permite a derivação de algum relacionamento suficiente, o que significa dizer que todos os outros relacionamentos podem ser obtidos (alguns mais facilmente que outros).

Uma SP tem uma estrutura topológica que permite orientar consistentemente os ciclos de arestas que delimitam as suas faces. Esta estrutura é representada por um grafo dirigido plano $\{V, A, F\}$ formado por um conjunto finito de vértices V, arestas A e faces F. Este grafo satisfaz às seguintes condições:

(1) Cada aresta está na fronteira de exatamente duas faces (não necessariamente distintas). Diz-se, então, que cada aresta possui dois **usos**, um em cada face.

(2) Usos de uma mesma aresta estão orientados em sentidos opostos. Isto induz uma orientação para todas as faces (por exemplo, sentido horário). A face ilimitada (externa) fica orientada no sentido contrário (anti-horário).

4.1.1 Representações Topológicas

Existem várias representações, suficientes topologicamente, que podem ser utilizadas para SPs. Estas representações foram criadas, originalmente, para representar a fronteira de sólidos *manifold* (a definição deste tipo de objeto encontra-se no apêndice). As representações mais populares são baseadas em arestas, no sentido de que todas as informações necessárias para descrever a topologia estão armazenadas nas estruturas relacionadas com arestas. A justificativa para esta preferência é que elas possuem campos de tamanho fixo (uma aresta é delimitada por dois vértices e delimita duas faces) e todos os três relacionamentos suficientes possuem arestas nos respectivos grupos de adjacência.

Para representação de SPs, adotou-se, neste trabalho, uma estrutura de dados que armazena, explicitamente, os dois usos de uma aresta pelas duas faces que a compartilham. Um uso de uma aresta é orientado e representa o lado de uma aresta visto por uma face (fig. 4.3). Cada uso de uma aresta é utilizado uma única vez pelo ciclo na fronteira de uma face e cada um destes usos induz uma orientação na face. Os dois usos de uma aresta possuem sentidos opostos. Esta estrutura de dados é chamada HED e está baseada nas estruturas HALF-EDGE e FACE-EDGE [MANT88, WEIL86]. Abaixo está a declaração C, da parte relativa à aresta, desta estrutura.

```
typedef struct model   Model;
typedef struct face    Face;
typedef struct loop    Loop;
typedef struct edge    Edge;
typedef struct edgeuse Edgeuse;
typedef struct vertex  Vertex;
struct edgeuse
{
 Edgeuse *eu_nxt, *eu_prv; /* o proximo uso e o anterior */
 Loop    *eul_ptr;         /* ciclo delimitado por este uso */
 Edge    *eue_ptr;         /* aresta */
 Vertex  *euv_ptr;         /* vertice inicial deste uso */
 Edgeuse *eueu_mate_ptr;   /* o outro uso da mesma aresta */
};
```

`Edge, Vertex e Face` são estruturas que contêm informações geométricas ou es-

pecíficas da aplicação, evitando a multiplicação das mesmas. As adjacências de arestas ao redor de uma face são armazenadas em uma lista encadeada, circular e ordenada. Esta lista pode ser obtida percorrendo-se, sucessivamente, os campos `eu_nxt`. Há também acesso a um vértice, ao ciclo e ao outro uso da aresta através de ponteiros. O campo `eu_prv` é opcional, existindo apenas por uma questão de eficiência de acesso.

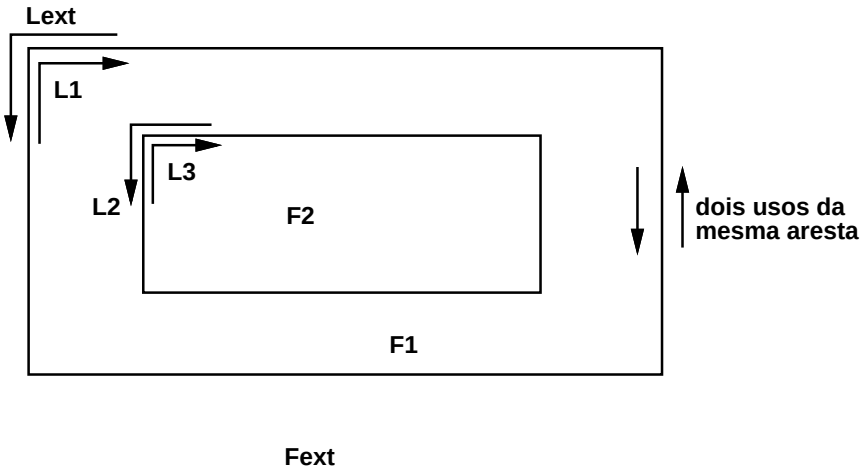


Figura 4.3 - Faces, ciclos e usos de aresta.

Na estrutura HALF-EDGE, o campo para o casamento dos dois usos de uma aresta (`eueu_mate_ptr`) não está armazenado na estrutura `Edgeuse`, mas sim na estrutura `Edge`, que contém ponteiros para os seus dois usos. Já na estrutura FACE-EDGE, não existe o ponteiro para o ciclo, mas sim um ponteiro para a face oposta (adjacente ao outro uso da mesma aresta). A forma apresentada, no entanto, parece ser a mais natural pela experiência na implementação desta estrutura, pelo autor, na PUC-Rio.

A estrutura HED é suficiente, pois o relacionamento $f < A >$ pode ser obtido apenas com os campos `euv_ptr` e `eu_nxt`. Uma face fica determinada percorrendo cada uso de aresta (que representa um lado de uma aresta que delimita a face), utilizando o campo `eu_nxt`, até que se retorne ao ponto de partida. O campo `euv_ptr` é necessário para "costurar" as faces umas às outras. Em uma SP, com arestas retilíneas, $a\{V\}$ determina unicamente uma aresta. Usos de aresta, delimitados pelo mesmo par de vértices, pertencem a mesma aresta. O vértice inicial está armazenado no campo `euv_ptr` e o final, no mesmo campo do próximo uso. A "costura" das faces pode ser

feita por um procedimemnto de identificação topológica. O campo `eueu_mate_ptr`, no entanto, fornece esta informação diretamente.

Na representação completa de uma SP, existem ponteiros para listas circulares de faces, arestas e vértices. A estrutura que armazena uma face contém, por sua vez, um ponteiro para uma lista circular com os seus ciclos e a estrutura para ciclo contém um ponteiro para um dos seus usos de aresta. Esta representação apresenta uma descrição hierárquica da SP, partindo dos níveis mais altos em dimensão (faces), para os mais baixos (vértices) (fig. 4.4). Esta forma de organização tem-se mostrado adequada à maioria das aplicações.

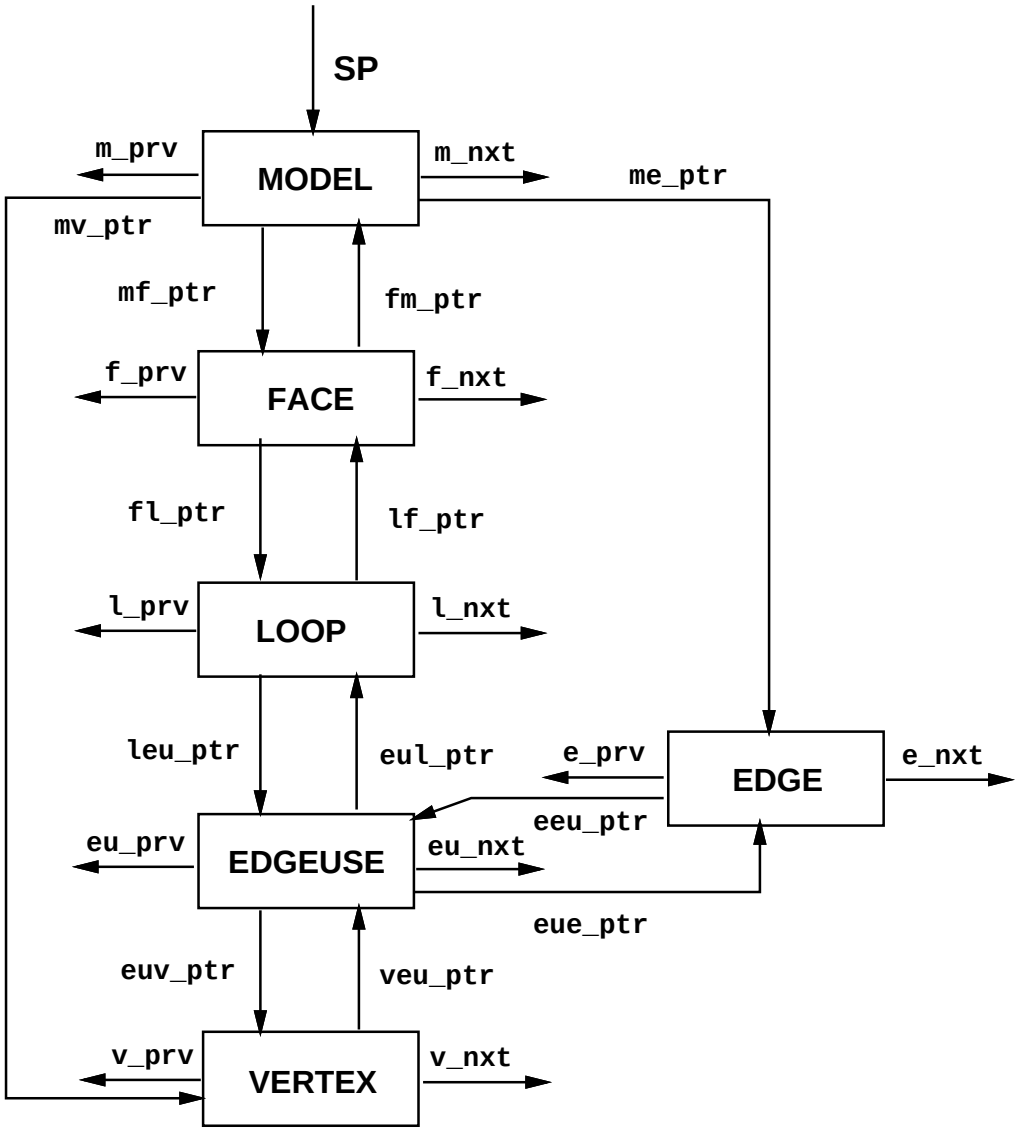


Figura 4.4 - Hierarquia da estrutura HED.

```
struct model
{
 Model  *m_nxt, *m_prv;
 Face   *mf_ptr;        /* cabeca (face externa) da lista de faces */
 Edge   *me_ptr;        /* lista de arestas */
 Vertex *mv_ptr;        /* lista de vertices */
 void   *mattrib_ptr;   /* atributos da subdivisao */
};
struct face
{
 Face  *f_nxt, *f_prv;
 Model *fm_ptr;         /* subdivisao que possui a face */
 Loop  *fl_ptr;         /* cabeca (ciclo externo) da lista de ciclos */
 void  *fattrib_ptr;    /* atributos da face */
};
struct loop
{
 Loop    *l_nxt, *l_prv;
 Face    *lf_ptr;        /* face delimitada pelo ciclo */
 Edgeuse *leu_ptr;       /* um dos usos de aresta do ciclo */
 void    *lattrib_ptr;   /* atributos do ciclo */
};
struct edge
{
 Edge    *e_nxt, *e_prv;
 Edgeuse *eeu_ptr;       /* um dos dois usos da aresta */
 void    *eattrib_ptr;   /* atributos da aresta */
};
struct vertex
{
 Vertex  *v_nxt, *v_prv;
 Edgeuse *veu_ptr;       /* uma aresta incidente no vertice */
 void    *vattrib_ptr;   /* atributos do vertice */
};
```

A estrutura HED representa exatamente o que é necessário para uma SP, ou seja, um complexo **completo** de dimensão 2. A sua utilização no espaço tri-dimensional permite representar apenas o esqueleto de ordem 2 de alguns complexos de dimensão 3. Note-se que esta estrutura permite representar, ordenadamente, as adjacências de arestas (e por conseguinte de faces) ao redor de vértices.

4.1.2 Operadores de Euler

Estruturas de dados topológicas são um tanto complexas para serem manipuladas diretamente. Baumgart [BAUM72] introduziu um conjunto de operadores, chamados EOps, que fornecem um meio, de relativo alto nível, para manipular este tipo de estrutura. Especificamente, EOps servem para construir topologias de adjacência, sem entrar nos detalhes do formato de armazenamento dos dados. Uma outra vantagem é que a sua utilização fornece uma interface de programação que permite que a estrutura de dados utilizada seja trocada sem a necessidade de reescrever toda a aplicação (somente a parte que implementa os EOps). Uma propriedade importante é que qualquer EOp é invertível, possibilitando a criação de operadores inversos que podem restaurar a topologia de uma SP.

Tradicionalmente, os EOps possuem nomes da forma *mxky*, com *m* significando *make* e *k* significando *kill*; *x* e *y* indicam o tipo do elemento topológico criado ou destruído. Embora bastem quatro EOps para construir qualquer topologia de adjacência de uma SP (este resultado é mostrado no apêndice), pode-se utilizar um número maior de EOps para facilitar a escrita de aplicativos. Um conjunto adequado é formado por:

`mvfs`	(*make vertex, face and SP*)	`kvfs`	(*kill vertex, face and SP*)
`mev`	(*make edge and vertex*)	`kev`	(*kill edge and vertex*)
`mekr`	(*make edge, kill ring*)	`kemr`	(*kill edge, make ring*)
`mef`	(*make edge and face*)	`kef`	(*kill edge and face*)
`mvr`	(*make vertex and ring*)	`kvr`	(*kill vertex and ring*)
`split`	(*split edges*)	`join`	(*join edges*)

Este conjunto contém os quatro EOps básicos, e seus respectivos inversos, mais dois EOps adicionais (`mvr` e `split`). O EOp `mvfs` cria uma SP com uma única face (ilimitada) com um único vértice. Os EOps `mekr` e `mef` criam uma aresta entre dois vértices de uma mesma face. O EOp `mef` deve ser usado quando os dois vértices pertencerem ao mesmo ciclo. Neste caso, uma nova face é criada. O EOp `mekr` deve ser usado quando os dois vértices pertencerem a ciclos diferentes. Neste caso, a nova aresta funciona como uma ponte que conecta os dois ciclos. Como conseqüência, um dos ciclos é eliminado. Estes

dois EOps podem ser implementados como um único EOp `me` que, fazendo os testes necessários, executa as ações apropriadas.

O EOp `mev` cria uma aresta e um vértice em uma face. A aresta é delimitada por um vértice já existente e pelo vértice criado. O EOp `split` divide uma aresta em duas, criando um novo vértice entre elas e o EOp `mvr` cria um ciclo pontual em uma face.

Os três EOps que criam arestas podem ser imaginados como procedimentos para inserir um segmento inteiramente contido em uma face (fig. 4.7). Formalmente, EOps permitem inserir uma célula, de dimensão menor que 2, desde que ela encaixe na SP. O EOp `me` completa a SP automaticamente, criando as faces (células de dimensão 2) quando se faz necessário.

É importante não esquecer que os EOps não têm conhecimento da geometria do segmento e por isso podem criar SPs inconsistentes geometricamente. Cabe à aplicação evitar que isto ocorra.

4.2 Processo de Modelagem

Um processo de modelagem deve fornecer, a um usuário típico de um sistema, uma maneira, familiar e eficiente, para a entrada dos dados que permitirão a construção da representação de um modelo.

Uma forma de construção de uma SP, conveniente para várias aplicações, consiste em um **processo incremental**, através do qual cada segmento de curva, que compõe a SP, é especificado interativamente (possivelmente através de uma mesa digitalizadora), sendo a SP atualizada após a inserção de cada segmento. Isto significa que é fornecida a descrição geométrica de um conjunto de segmentos, a partir da qual é criada a representação topológica e geométrica da SP. Esta é a técnica de modelagem considerada neste capítulo. Entretanto, existem outras formas de criar uma SP; por exemplo, fornecendo a lista das suas faces através da especificação (talvez ordenada) dos segmentos que as delimitam (um processo utilizado em aplicações não interativas). Uma outra forma é utilizando operadores CNRG e será apresentada no capítulo 6.

4.3 Inserção de Segmentos

O passo fundamental no processo de criação de SPs, considerado neste capítulo, consiste na inserção de um novo segmento na SP.

Um segmento de curva é uma curva algébrica simples, conexa e com duas extremidades (não necessariamente distintas). Um segmento determina unicamente um complexo completo, de dimensão 2, com uma única aresta e com apenas uma face (ilimitada). A fronteira da aresta corresponde às extremidades do segmento. No que se segue, o termo segmento vai ser empregado para se referir, tanto ao segmento em si, como ao seu complexo associado.

O problema de inserir o novo segmento pode ser visto como o problema de combinar os SGCs representados pela SP existente e pelo novo segmento. Para tal, é necessário refinar o complexo referente ao segmento a ser inserido, de modo a que as suas células possam ser encaixadas na SP existente.

O procedimento que insere um novo segmento em uma SP deve garantir que, após a inserção, a SP continue consistente topológica e geometricamente. A consistência topológica é mantida, naturalmente, se forem utilizados EOps de forma apropriada. Para isto, o segmento deve ser dividido em um conjunto de segmentos inteiramente contidos em faces da SP, chamados segmentos simples. Isto corresponde justamente a refinar o segmento para que ele encaixe na SP. Ao final do refinamento, o segmento pode possuir várias arestas. Cada uma destas arestas determina um segmento simples que pode ser inserido pelos EOps apropriados.

Quando um segmento é inserido ele pode interceptar várias arestas diferentes, sendo que cada aresta pode ser interceptada em vários pontos. Basicamente, existem quatro tipos de situação que devem ser analisadas:

- o segmento é simples.
- o segmento intercepta uma aresta.
- o segmento intercepta um vértice.
- o segmento possui um trecho em comum com alguma aresta.

Em um segmento S, S_i e S_f denotam as suas extremidades inicial e final, respectivamente, e T_i o vetor tangente a S, no ponto S_i. Para inserir um segmento S é necessário descobrir que arestas da SP são cortadas e as coordenadas de cada ponto de interseção, para poder subdividí-lo em um conjunto de segmentos simples. Isto pode ser feito por partes, pois se T_i está contido em uma face F, a primeira interseção, a partir de S_i, ocorrerá obrigatoriamente com uma célula de ∂F. Se esta célula não for um vértice, a aresta cortada por S terá que ser dividida topologicamente. Desta forma ter-se-á sempre um vértice v_i no ponto de interseção. O segmento S é então quebrado em duas partes, uma de S_i até v_i (S_1) e a outra de v_i até S_f (S_2). S_{1f} e S_{2i} passam a estar associados a v_i. S_1 é um segmento simples por construção e o processo deve ser repetido para S_2, até que S_2 também seja um segmento simples (fig. 4.5). Neste processo de simplificação de um segmento o algoritmo mais importante é aquele que calcula a interseção entre dois segmentos de curva (seção 4.5).

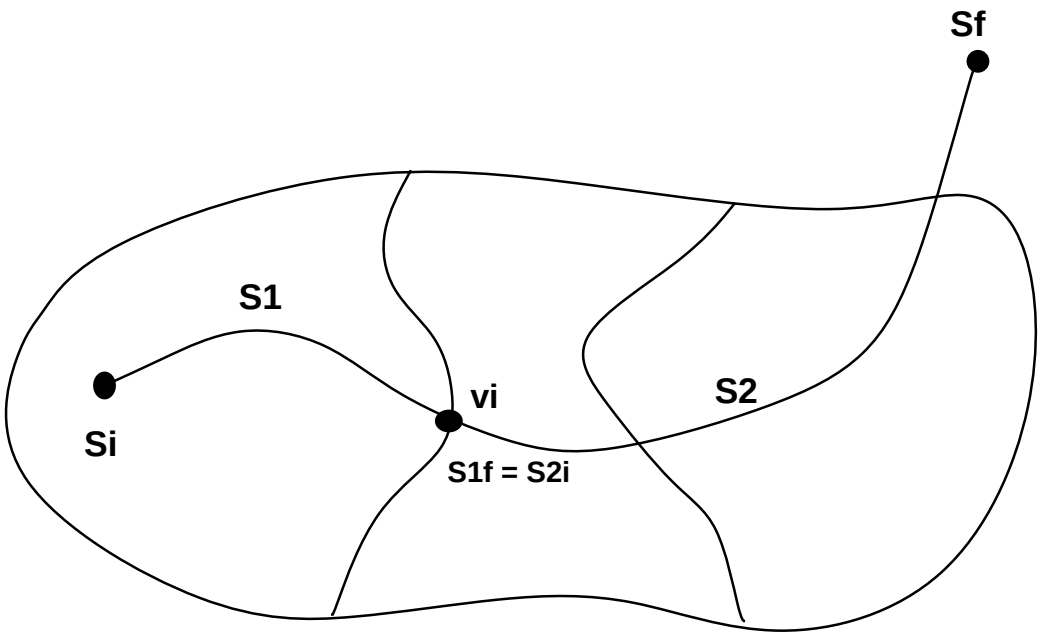


Figura 4.5 - Decomposição de um segmento em segmentos simples.

4.3.1 Inserção de Segmentos Simples

Uma vez que se tenha dividido um segmento em um conjunto de segmentos simples, cada um deles deve ser inserido na SP. O problema se reduz, então, em como inserir um segmento simples S, com extremidades S_i e S_f. Para isto, suponha-se que todas as faces da SP estão orientadas no sentido horário, isto é, a fronteira externa está orientada no sentido horário e as internas (se houver alguma) no sentido anti-horário. É conveniente considerar que a face externa (infinita) não possui fronteira externa.

Para incluir S é necessário determinar:

1) a face F_o que contém S.
2) os vértices v_i e v_f de ∂F_o que serão ligados por S.
3) as duas arestas que sucedem S nos ciclos ordenados de arestas incidentes em v_i e v_f.

O primeiro passo para incluir S é garantir que S_i coincide geometricamente com um vértice v_i da SP (para que possa ser associado a ele). Caso não exista tal vértice, S_i pode estar sobre uma aresta, que deverá ser dividida (surgindo um novo vértice), ou no interior de uma face, devendo, neste caso, ser criado um ciclo pontual nesta face.

Para determinar F_o, é usado o vetor tangente T_i. Existem dois casos triviais: quando S_i está associado a um vértice v_i que é um ciclo pontual, ou quando v_i delimita uma única aresta. Em ambos os casos, S pertence à face que contém S_i. Se v_i delimitar mais de uma aresta, então F_o está delimitada por duas arestas (que podem ter sido criadas pela divisão de uma aresta quando v_i foi criado), cujas tangentes em S_i são dois vetores, um precedendo (formando ângulo orientado máximo com T_i) e o outro sucedendo (formando ângulo orientado mínimo) T_i no sentido anti-horário (fig. 4.6). Estes vetores fornecem também os argumentos para os EOps a serem utilizados, de forma a que o segmento seja inserido corretamente no ciclo ordenado de arestas que incidem em v_i. O processo descrito para S_i deve ser aplicado também a S_f.

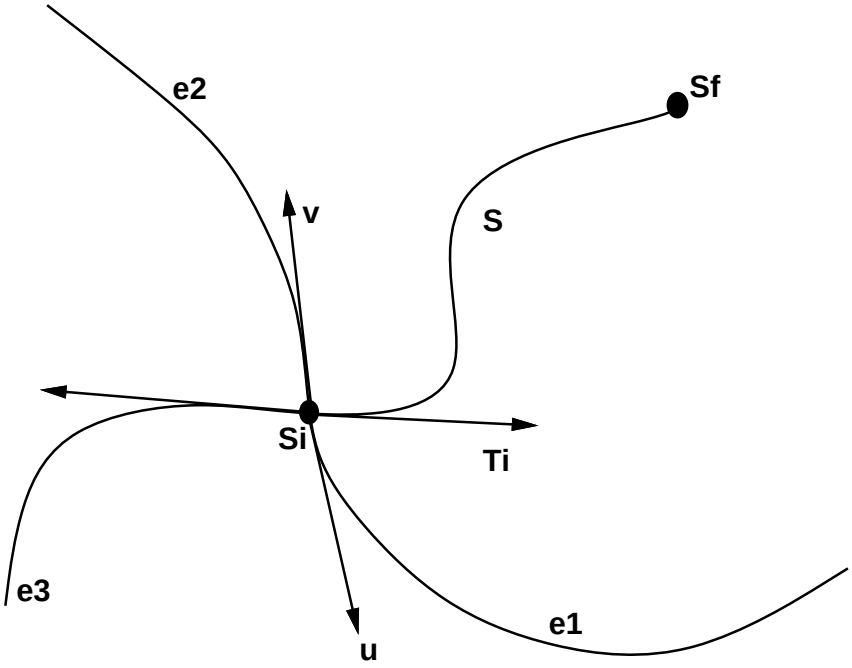


Figura 4.6 - Determinação da face que contém T_i.

Topologicamente, podem ocorrer três situações como conseqüência da inclusão de S. Cada uma delas determina que EOp deve ser utilizado para inserí-lo:

- ser fechada uma nova face, caso S_f toque uma aresta pertencente ao mesmo ciclo que contém S_i (fig. 4.7a).
- ser eliminado um ciclo, caso S_f toque uma aresta pertencente a um ciclo diferente daquele que contém S_i (fig. 4.7b).
- ser simplesmente criada uma aresta, caso S_f não toque nenhuma aresta da SP (fig. 4.7c).

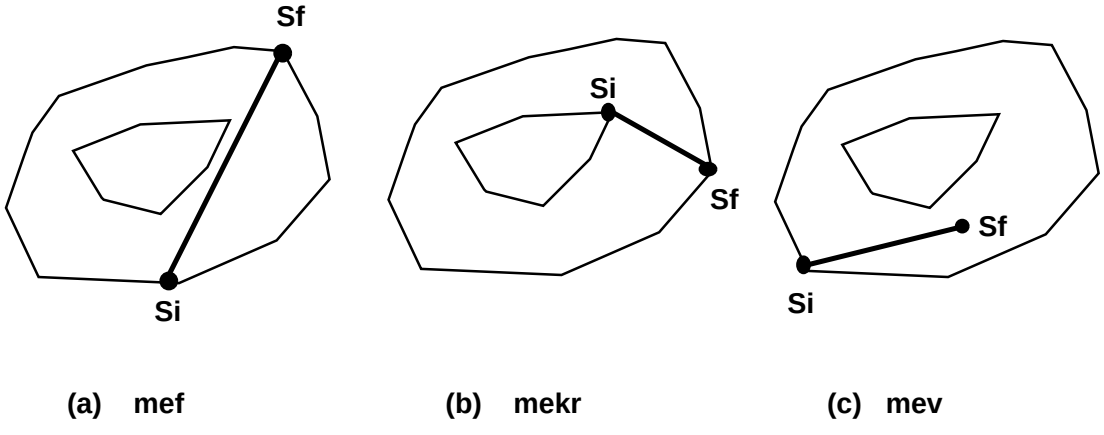


Figura 4.7 - Casos possíveis na inclusão de um segmento simples.

Quando uma nova face F é formada, ela é sempre criada a partir de um ciclo de construção de F_o. Sua área é subtraída da área original de F_o. A simples aplicação do EOp `mef` não garante, a princípio, que a convenção para a orientação das faces seja mantida, uma vez que isto depende de como a face foi fechada (fig. 4.8a) (ou seja, da geometria da aresta). A verificação de que a fronteira externa de F não está orientada no sentido arbitrado indica a necessidade de trocar o ciclo de construção pelo ciclo externo de F (fig. 4.8b).

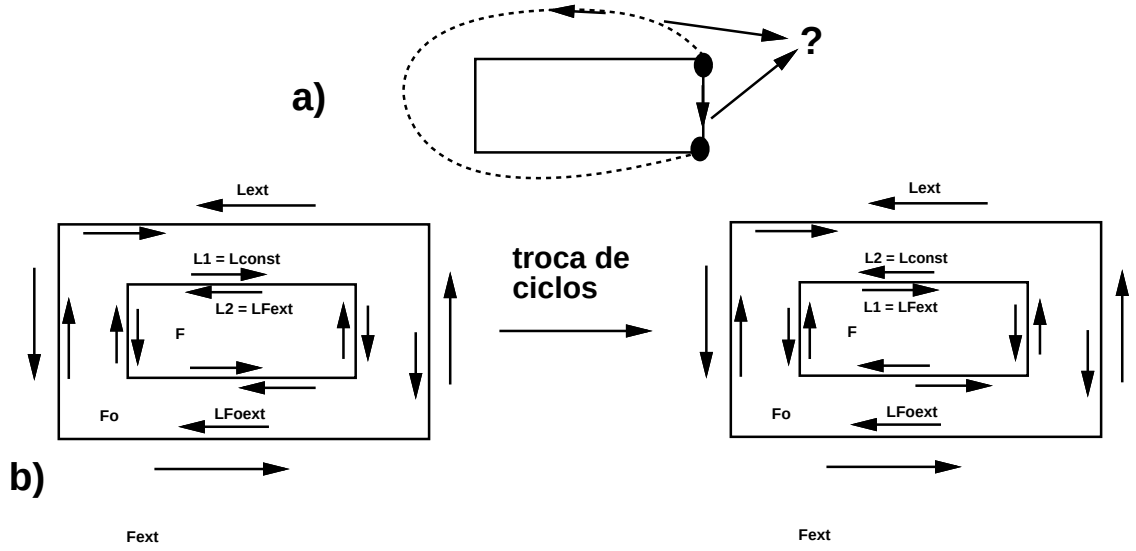


Figura 4.8 - Troca de ciclos.

A verificação da orientação de uma face é feita calculando-se a sua área com sinal. Para o caso de arestas retas, basta calcular o somatório dos produtos vetoriais [5] entre cada par de vetores com origem em um ponto r, no fecho da face (por exemplo, um vértice qualquer), e extremidades em vértices consecutivos (vértices correspondentes a usos de aresta consecutivos) [CARV91]. Com a orientação arbitrada, as áreas de todas as faces, exceto da face externa, são negativas. Arestas com geometrias quaisquer requerem um procedimento que calcule a área delimitada pela aresta e por dois segmentos de reta, com origem em r, e extremidades em cada vértice que a delimita.

[5] O produto vetorial no $\Re^2$ é feito considerando nula a coordenada z dos vetores e como resultado, a norma do vetor produto.

A nova face F pode conter, ainda, alguns ciclos de F_o, que são as suas fronteiras internas. Deve-se verificar que ciclos de F_o passam a pertencer a F. Isto é feito percorrendo a lista de ciclos de F_o (com exceção daquele que forma sua fronteira externa e daquele que delimita F) e verificando se qualquer um de seus vértices está dentro da área delimitada pela fronteira externa de F. Caso esteja, ele será movido para F (isto é, passará a pertencer a ela (fig 4.9). Isto requer um procedimento que verifique se um dado ponto pertence a uma dada face.

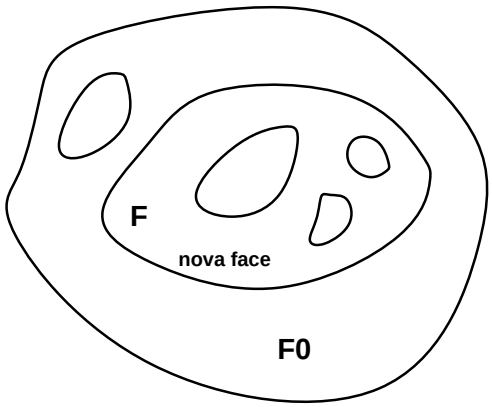


Figura 4.9 - Distribuição de ciclos.

A verificação da inclusão de um ponto em uma face pode ser feita por um algoritmo baseado no teorema de Jordan [HENL79]. Contam-se quantas vezes uma semi-reta qualquer, com origem no ponto, intercepta a fronteira da face. Se interceptar um número ímpar de vezes, o ponto está dentro da face. Deve-se tomar cuidado quando a semi-reta interceptar um vértice, para contar o número correto de interseções [CARV91].

Um algoritmo alternativo baseado em ângulos também pode ser empregado. Para arestas retilíneas, calcula-se o somatório dos ângulos orientados formados por segmentos com origem no ponto e extremidades em vértices consecutivos. Se o valor deste somatório for nulo, o ponto está fora. O resultado não é confiável se o ponto estiver muito próximo da fronteira da face [CARV91]. Este algoritmo pode ser adaptado para tratar arestas com outros tipos de geometria.

4.4 Estimativa para o Tempo de Criação de uma SP

O primeiro passo no processo de inclusão de um novo segmento em uma SP, que consiste em associar a extremidade inicial do segmento a um vértice já existente (ou, caso não exista tal vértice, criando-o primeiro), é o mais demorado. Uma vez associado o vértice inicial, o tempo de inclusão é proporcional ao número de arestas de cada face cortada.

O primeiro passo implica em determinar qual vértice, qual aresta ou qual face contém S_i. O algoritmo implementado para verificar a qual face um dado ponto pertence percorre a lista das faces, verificando se o ponto se encontra dentro da área F_{ext}, delimitada pela fronteira externa da face corrente F_c (inicialmente a primeira face da lista). Caso esteja, só precisam ser consideradas, daí para frente, as faces que estejam completamente contidas em F_c. Para verificar isto, basta tomar um vértice qualquer, de cada face subseqüente, e verificar se ele está contido em F_{ext} (pode-se recusar a face quando o vértice escolhido pertencer também a fronteira de F_c). Se uma face que passou no teste anterior também contiver o ponto, esta será a nova face corrente F_c. O processo termina quando se tiver chegado ao final da lista das faces. F_c indica, então, que face contém o ponto.

O processo descrito acima pode ser acelerado testanto primeiro se o ponto está dentro do retângulo que engloba cada face. De todo modo, obtém-se um algoritmo cujo tempo de execução é proporcional ao número de células da SP. Como o número de faces (F), vértices (V) e arestas (A) em uma divisão planar estão relacionados linearmente pela equação de Euler-Poincaré (consulte-se o apêndice, para uma justificativa deste fato), a complexidade do algoritmo pode ser expressa por $O(A)$, o que significa que uma subdivisão completa pode ser criada em $O(A^2)$, sendo A o número final de arestas na SP.

Deve-se observar que não é possível fazer uma análise da complexidade do algoritmo em função do número de segmentos originais, uma vez que estes segmentos podem ser divididos em um número qualquer de arestas.

A lista de faces pode ser alterada de forma a refletir o aninhamento das faces. Na figura 4.10, as linhas pontilhadas finas ligam faces que estão em um mesmo nível hierárquico, enquanto as grossas ligam uma face ao conjunto de faces que ela contém. Esta hierarquia pode ser mantida com apenas dois ponteiros por face. Desta forma, o algoritmo descrito anteriormente, lançando mão desta hierarquia, já sabe quais são as faces que estão contidas na face corrente.

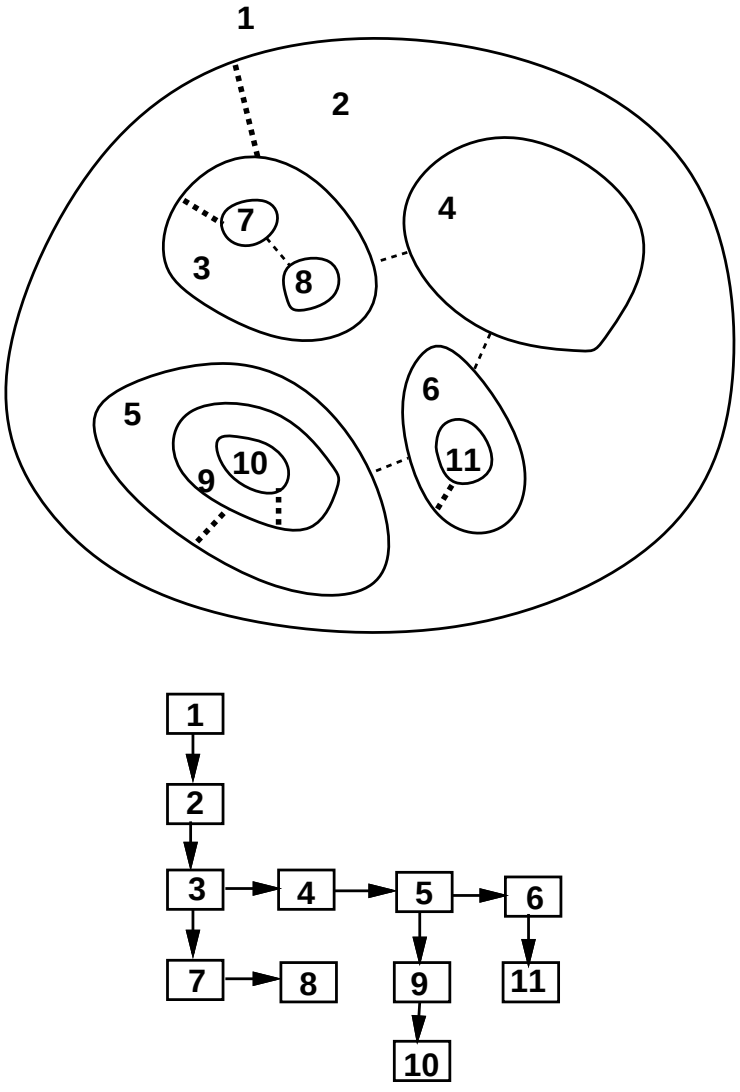


Figura 4.10 - Aninhamento de faces.

4.5 Interseção

Um dos algoritmos mais importantes na construção de uma SP é o que calcula as interseções de dois segmentos de curva. Uma curva no $\Re^2$ fica determinada de **forma implícita** por uma equação do tipo:

$$F(x, y) = 0,$$

onde F é uma função real nas variáveis x e y. A curva é formada pelos pontos do plano que satisfazem esta equação. O termo - implícita - é usado para caracterizar funções que são deduzíveis de uma equação. Em geral, diz-se que uma função $y = f(x)$, $x \in [a, b]$ é definida implicitamente por uma equação $F(x, y) = 0$ se $F(x, f(x)) = 0$, $x \in [a, b]$. Normalmente, tem-se disponível somente uma equação implícita, sendo impossível resolvê-la explicitamente para uma função de x. Por exemplo, numa equação como $y = x^2 - e^x$, y é uma função explícita de x. Já a equação $x^2 + xy + y^3 - 2x^5 + 3y^7 = 0$ não permite definir explicitamente y como função de x. No entanto, mesmo sem a forma explícita, é possível obter diversas informações sobre f, incluindo as suas derivadas, se for assumido que a equação define implicitamente uma função derivável em um intervalo.

$$\frac{dy}{dx} = -\frac{\partial F/\partial x}{\partial F/\partial y}, \quad \text{se } \frac{\partial F}{\partial y} \neq 0.$$

No exemplo:

$$2x + y + xy' + 3y^2y' - 10x^4 + 21y^6y' = 0$$

ou

$$y' = -\frac{2x + y - 10x^4}{x + 3y^2 + 21y^6} \ .$$

As hipóteses de que uma equação em x e y define sempre uma função derivável $y = f(x)$ sobre um intervalo, podem falhar. Por exemplo, a equação $x^2 + y^2 = 0$ é satisfeita somente por $x = 0$ e $y = 0$ e portanto não existe nenhuma função implícita. Na prática, normalmente, tem-se razões para esperar que a equação $F(x, y) = 0$ defina uma função derivável em algum intervalo. Pelo teorema da função implícita, se $F(x, y)$ é definida numa região aberta D do plano xy, possui derivadas parciais contínuas em D e, além disso, $\partial F/\partial y \neq 0$ no ponto (x_0, y_0) em D e $F(x_0, y_0) = 0$, então, pelo menos

numa vizinhança deste ponto, F define implicitamente uma função derivável $y = f(x)$. Se o vetor $\nabla F = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y})$ for nulo no ponto x_0, então, não é possível obter, numa vizinhança de x_0, uma função $y = f(x)$ nem uma função $x = f(y)$. Neste caso, diz-se que o ponto x_0 é um ponto **singular** de F.

Algumas curvas possuem representações da forma:

$$x = f(t), \quad y = g(t),$$

onde f e g são funções contínuas num intervalo. Esta representação é chamada **paramétrica** (ou caminho no plano xy) e é um mapeamento de $\Re$ em uma curva do $\Re^2$. As curvas paramétricas são muito empregadas em modelagem geométrica. Curvas de Hermite, Bezier e Spline utilizam polinômios para f e g e impõem condições sobre as funções e as respectivas derivadas em determinados pontos.

Uma **função algébrica** é uma função contínua num intervalo, definida implicitamente por uma equação polinomial em x e y:

$$a_1 x^{n_1} y^{m_1} + a_2 x^{n_2} y^{m_2} + ... + a_k x^{n_k} y^{m_k} = 0;$$

$(n_1, n_2, ..., n_k$ e $m_1, m_2, ..., m_k)$ são inteiros não negativos. Assim, cada função racional (quociente de dois polinômios) e cada inversa contínua de uma função racional é algébrica. Por exemplo, $y = x^{1/3}$ é uma função algébrica definida implicitamente pela equação $y^3 - x = 0$.

As funções não algébricas são chamadas **transcendentais**. Pode ser verificado que as funções trigonométricas, a função e^x e a função $ln(x)$ são transcendentais. Estas funções e todas as funções não algébricas obtidas delas por adição, subtração, multiplicação, divisão, multiplicação por constantes, composição e formação de funções inversas são todas chamadas **funções transcendentais elementares**.

Ao ultrapassar as funções algébricas para criar funções transcendentais, o passo inicial é um processo de limite. As funções transcendentais elementares podem ser representadas, em intervalos convenientes, por séries de potências, por exemplo:

$$e^x = \lim_{n \to \infty} (1 + x + \frac{x^2}{2!} + ... + \frac{x^n}{n!}).$$

Funções assim representáveis são chamadas **funções analíticas**. As funções analíticas não despertam maior interesse, neste trabalho, pois sua representação em um computador é truncada (ou seja, são aproximadas por funções algébricas).

O cálculo da interseção de duas curvas definidas explicitamente se reduz a resolver uma equação. Por exemplo, encontrar os pontos de interseção de $y = x^2$ e $y = cos(x)$ é equivalente a encontrar as raízes de $x^2 - cos(x) = 0$.

A resolução de equações é um problema básico em matemática. Em álgebra elementar se aprende como resolver equações lineares e quadráticas. Para as equações cúbicas (grau 3) e quádricas (grau 4) existem fórmulas para as raízes em função dos coeficientes [DICK03], mas elas são muito complicadas. Para polinômios de grau maior não há fórmulas deste tipo e tem-se que usar algum método numérico de aproximação de raízes.

O método de Newton, por exemplo, permite encontrar aproximações sucessivas das raízes de uma função f definida e derivável no intervalo $[a, b]$:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad f'(x_n) \neq 0.$$

O método converge desde que se conheça um intervalo apropriado que contenha uma única raíz.

Quando se lida com segmentos de curva, não é interessante calcular pontos de interseção que, embora pertencendo aos suportes dos segmentos, não pertencem a ambos os segmentos, pois isto irá requerer um processamento numérico desnecessário. É importante que existam, então, condições que permitam concluir se dois segmentos se interceptam ou não. Estas condições são baseadas em limites aproximativos interiores e exteriores, formando um envelope que contém o segmento. Para um segmento, considerem-se os limites que satisfazem às seguintes condições:

1) Cada limite é uma seqüência conexa de segmentos de reta.
2) Os dois limites formam a fronteira de um envelope (uma área). Os limites não se cruzam e terminam em um ponto comum.
3) O segmento está inteiramente contido no envelope.

4) Qualquer caminho, contido no interior do envelope, que inicie em um ponto sobre um limite e termine em um ponto sobre o outro limite, intercepta o segmento.

Se os envelopes de dois segmentos se interceptam formando uma área de quatro lados, com a propriedade de que cada par de lados opostos origina-se de ambos os limites do mesmo segmento, então, os segmentos se interceptam e o(s) ponto(s) de interseção está(ão) dentro desta área [TURN88]. Além disso, qualquer par de limites dos segmentos se interceptam o mesmo número de vezes que os segmentos. Isto permite que qualquer uma das quatro combinações de limites sejam usadas como aproximações, topologicamemte válidas, para determinação de um ponto aproximado de interseção dos segmentos.

Na figura 4.11, duas curvas de Bezier (cúbicas) possuem como envelope os fechos convexos dos respectivos pontos de controle. Se os envelopes se interceptam, mas a condição anterior não é satisfeita, então, para o caso de curvas de Bezier, um processo de subdivisão criado por Casteljau [BOEH84] pode ser usado para gerar um novo envelope para cada segmento.

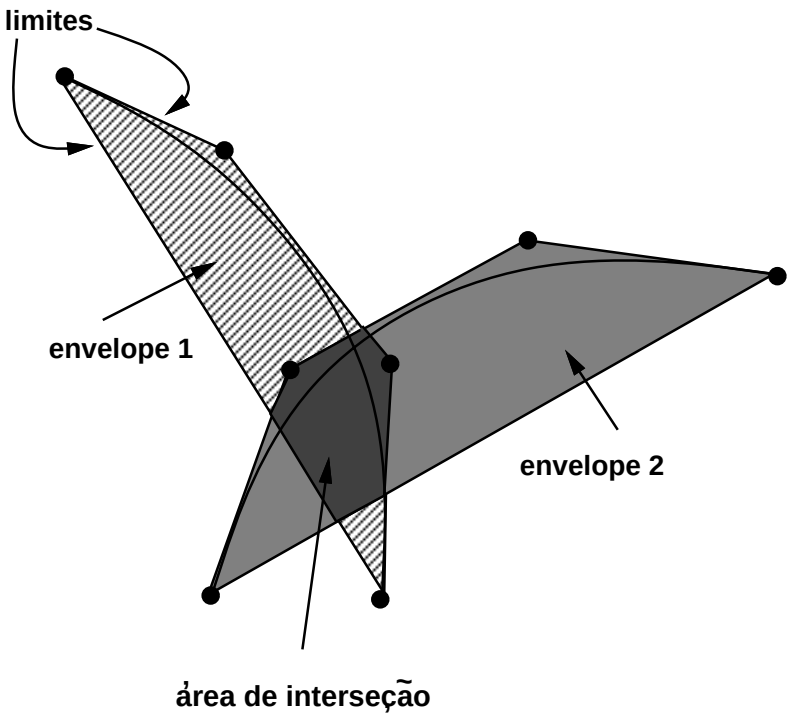


Figura 4.11 - Duas curvas de Bezier com os respectivos envelopes.

Lane e Riesenfeld propuseram a aplicação repetitiva de subdivisões como uma forma de calcular o ponto de interseção [LANE80]. Sederberg e Goldman [SEDE86] propuseram que a interseção de tais curvas seja feita inteiramente por métodos numéricos (Sederberg e Parry [SEDE86] comentam alguns dos problemas numéricos) e Turner [TURN88] propôs um método híbrido. Estes algoritmos podem ser aplicados a curvas B-splines se elas forem convertidas em uma seqüência de curvas de Bezier. Outros tipos de envelope também podem ser utilizados e uma comparação entre eles pode ser encontrada em [SEDE88].

Um algoritmo genérico para calcular a interseção entre dois segmentos compromete a modularidade de um sistema de SP. Para acrescentar uma nova geometria deve-se levar em conta a interação com as já existentes. Portanto, uma abordagem possível é aproximar cada segmento por um limite (isto é, linear por partes). Um processo de refinamento destes limites pode ser feito de forma a produzir um ponto de interseção aproximado.

4.5.1 Tolerâncias

Para obter algoritmos robustos não basta tratar do problema matemático de encontrar a interseção de duas curvas. Isto devido a dois problemas adicionais: o erro introduzido no cálculo dos pontos de interseção quando se utiliza aritmética de ponto flutuante; e a imprecisão inerente a todo processo interativo, quando o usuário não deseja (ou não consegue) especificar posições geométricas com precisão absoluta. Para lidar com estes dois problemas, a fim de obter uma implementação robusta, é necessário introduzir o conceito de tolerância [HOFF89].

O mecanismo proposto considera que a SP foi gerada corretamente e que o segmento deve se adaptar a ela. Cada vértice e cada aresta possui um campo de atração cuja abrangência depende de uma dada tolerância. Este campo de atração fornece espessura às arestas.

Quando um segmento é criado interativamente, ele sofre um pré-processamento. Se uma de suas extremidades cair dentro do campo de atração de um vértice, suas

coordenadas passam a ter os valores das coordenadas do vértice (fig. 4.12a). Caso caia dentro do campo de atração de uma aresta, suas coordenadas passam a ter os valores das coordenadas do ponto determinado pela projeção da extremidade sobre a aresta (fig. 4.12b). Em um segundo passo, o segmento pode ser dividido em dois, caso o seu interior seja atraído por um vértice (fig. 4.12c).

Se a implementação do campo de atração não for feita consistentemente, a alteração na geometria do segmento poderia fazer com que um segmento que não interceptaria nenhuma aresta passasse a interceptar, dependendo da configuração já existente e do valor da tolerância utilizada (fig. 4.12d).

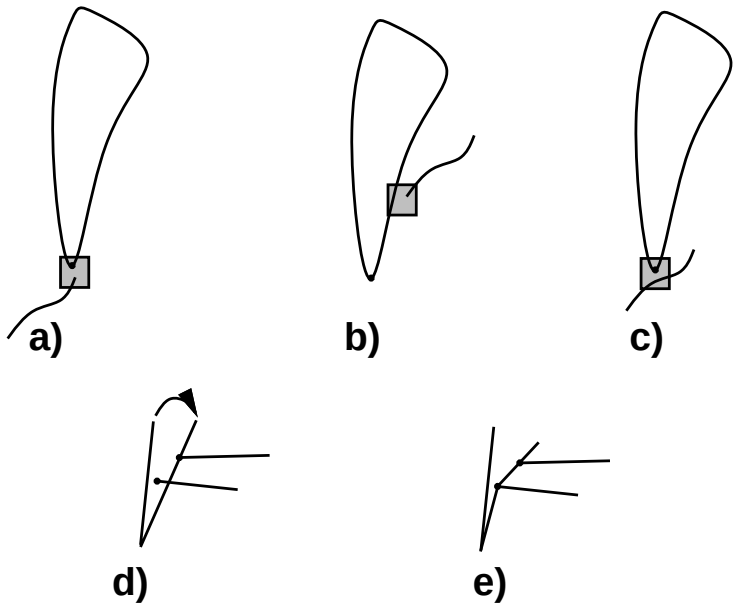


Figura 4.12 - Campos de atração.

Uma propriedade interessante seria garantir que todos os vértices e todas as arestas mantivessem uma distância mínima entre si. Para isto, sempre que um segmento fosse atraído, ele deveria ser reprocessado recursivamente para determinar se, com a alteração na sua geometria, ele passaria a ser atraído primeiro por um outro vértice.

Os valores das tolerâncias dependem, entre outras coisas, do espaço de coordenadas do usuário, no qual os objetos estão descritos. Tipicamente, quando for feita uma ampliação, para obter-se algum detalhe, as tolerâncias devem ser diminuídas proporcionalmente.

4.5.2 Arestas Poligonais

Embora o objetivo seja lidar com arestas quaisquer, um caso particular importante de geometria de arestas é aquele dado por linhas poligonais, pois este é o tipo de geometria mais característico de aplicações ligadas a área de mapeamento geográfico e geológico. Além disso, os algoritmos geométricos envolvidos são simples e uma poligonal pode ser usada para aproximar as curvas de interesse prático em modelagem geométrica.

Uma poligonal é um conjunto (finito) conexo, ordenado e sem duplicatas de segmentos de reta. Ela fica perfeitamente caracterizada dada a seqüência ordenada dos pontos pertencentes a dois segmentos (pontos intermediários) mais os seus pontos inicial e final (chamados de extremidades). Para armazenar uma seqüência ordenada de pontos, basta uma estrutura, chamada L_p, que contenha o número de pontos e um ponteiro para uma lista com as coordenadas dos pontos.

As extremidades de uma poligonal estão sempre associadas ao(s) vértice(s) que delimitam uma aresta (sendo necessário distingüir o vértice inicial do vértice final). Os pontos intermediários, no entanto, são armazenados separadamente. O ponteiro para o atributo geométrico da aresta aponta para a L_p, com os seus pontos intermediários ordenados do vértice inicial para o final. No caso de uma aresta reta, este ponteiro é nulo. Para percorrer a aresta no sentido inverso, deve-se partir do vértice final e percorrer a L_p que contém os pontos intermediários, de trás para frente.

A parametrização de uma aresta poligonal é muito simples: basta computar o número inteiro de segmentos de reta até o ponto desejado e adicionar um número entre zero e um, correspondente ao valor paramétrico no segmento de reta que contém o ponto. Isto corresponde a permitir que o parâmetro tome valores no intervalo $[0..n]$, onde n é o número de segmentos de reta da poligonal. No caso de arestas poligonais, cada ponto intermediário pode ser imaginado como um falso vértice, que também possui um campo de atração. O algoritmo que calcula o ponto de interseção entre duas poligonais verifica se cada segmento de reta da primeira poligonal intercepta algum segmento da segunda. Caso o ponto de interseção caia dentro do campo de atração de um vértice (ou falso vértice) de qualquer uma das duas arestas, são devolvidas as coordenadas do vértice

como sendo as coordenadas do ponto de interseção.

Para garantir que, ao ser atraída, a poligonal não passa a interceptar uma aresta cuja interseção não foi detectada, considera-se que apenas um segmento de reta da poligonal é modificado por vez, sempre alterando o seu ponto final e mantendo o ponto inicial fixo. Desta forma, basta considerar a projeção dos vértices de cada aresta sobre o segmento de reta em questão e retornar a interseção mais próxima do ponto inicial (fig. 4.12e).

Uma maneira de acelerar o tempo de cálculo do ponto de interseção entre dois segmentos de reta é utilizar, como envelopes, os retângulos cujas diagonais coincidem com os segmentos.

4.6 Arquitetura de um Sistema de SP

A metodologia a ser adotada na implementação de um sistema de subdivisão planar deve ser, claramente, a de construir camadas de *software* com funcionalidades bem definidas: uma camada responsável pelo tratamento da topologia utilizando EOps (esta camada desconhece completamente que tipo de geometria está sendo utilizada); uma camada responsável pelo gerenciamento e manipulação de entidades geométricas (**MGG**); e, por fim, uma camada responsável pela interação com o usuário.

O MGG é o módulo mais crítico em tal sistema. Ele tem que garantir a validade geométrica da SP que se deseja construir, lidar com todos os problema numéricos, e ser geral o suficiente para suportar uma vasta gama de geometrias para as células, além de permitir a inclusão de uma nova geometria. Em função disto, ele deve ser capaz de tratar geometria de forma genérica, ou seja, ser construído baseado num conjunto pré-determinado de funções que, ao serem reescritas, permitam a adição de células com uma nova geometria. A forma de se atingir isto é fazendo com que as células tragam consigo uma indicação de que rotinas específicas implementam cada função genérica para o seu tipo de geometria. Desta forma, torna-se possível escrever controladores de geometrias e um programador, de posse da descrição da interface de comunicação com os controladores, seria capaz de incluir uma nova geometria no sistema.

Existem funções, no MGG, encarregadas de iniciar, com valores *defaults*, os atributos do usuário de cada célula criada. Elas alocam uma área de memória para os atributos e fazem com que o ponteiro para atributo do usuário, da célula criada, contenha o seu endereço. Quando uma célula é dividida topologicamente, a situação é um pouco diferente, pois, em geral, deseja-se que a nova célula possua atributos idênticos aos da célula original. Para este caso, existe uma função encarregada de alocar uma área de memória para os atributos da nova célula e preenchê-la com valores idênticos aos da célula original.

4.6.1 Armazenamento de SPs em um Meio Permanente

Uma vez criada uma SP, é natural que seja necessário salvá-la para futuras utilizações ou atualizações. O grande problema nesta tarefa é que uma estrutura de dados topológica, como a HED, é implementada tipicamente com o emprego de ponteiros.

Uma forma elegante de armazenar a estrutura em disco é através do uso de um algoritmo de inversão [MANT88]. Este algoritmo desfaz completamente a SP, eliminando primeiro todas as suas arestas e, em seguida, todos os ciclos pontuais remanescentes. Em disco são gravados, então, os EOps inversos daqueles utilizados na destruição da SP. A execução destes EOps na ordem inversa recria a SP original. Deve-se notar, porém, que, para isto, é necessário salvar também os atributos geométricos e os atributos do usuário de cada célula.

Como os EOps inversos aos que destroem a estrutura devem ser executados na ordem inversa, é aconselhável armazená-los já de acordo com esta ordenação, ao invés de ler o arquivo de trás para frente. Para tal, é utilizada uma pilha. As rotinas que implementam os EOps devem empilhar um registro com a descrição do EOp inverso e seus respectivos parâmetros, sempre que estiver sendo executado o algoritmo de inversão.

O processo se resume, então, em desempilhar e gravar cada registro de descrição do EOp e os registros de atributos geométricos e do usuário, se os respectivos ponteiros não forem nulos. Para recriar a SP basta ler o arquivo, executando cada EOp e ligando as células criadas aos seus atributos.

Com este processo é possível armazenar todas as informações presentes em qualquer SP em um arquivo seqüencial. O único aspecto inconveniente é que a SP é destruída, devendo ser reconstruída, a partir do meio permanente, caso seja necessário reutilizá-la após o salvamento.

4.6.2 Funções Necessárias a um Sistema de SP

Para ser possível lidar com curvas com geometrias quaisquer, foi identificado, nas seções 4.3 e 4.3.1, um conjunto de funções que devem ser escritas para que uma nova geometria possa ser suportada (funções 1-7). Algumas delas podem não ser implementáveis facilmente, para certas geometrias, sendo necessário a utilização de métodos aproximativos. As funções 8-10 existem para permitir o armazenamento permanente e a recuperação de uma SP. A função 11 existe para que seja possível exibir graficamente uma SP.

1) Dado um par de células, de mesma dimensão, verificar se elas são geometricamente idênticas.

2) Dadas duas arestas, determinar se elas se interceptam e, em caso afirmativo, retornar uma lista ordenada com todos os pontos de interseção.

3) Dada uma célula e as coordenadas de um ponto, verificar se o ponto está no seu interior.

4) Dada uma célula, calcular uma medida com sinal da sua extensão. Para uma aresta é o seu comprimento e para uma face a sua área.

5) Dada uma aresta A, para ser inserida em uma SP, e um vértice v que a delimita, retornar a aresta que deve suceder A, no ciclo ordenado de arestas incidentes em v.

6) Dado um par de arestas (faces) C_1 e C_2, que foram produzidas como resultado da divisão de uma aresta (face) C, distribuir os atributos geométricos de C entre C_1 e C_2.

7) Dado um par de arestas (faces) C_1 e C_2, que foram produzidas pela eliminação de um vértice (aresta) $c \in (\partial C_1 \cap \partial C_2)$, combinar os atributos geométricos de C_1 e C_2 para formarem os atributos geométricos da aresta (face) resultante.

8) Dada uma célula, alocar ou liberar a área de memória correspondente aos seus atributos.

9) Dado um arquivo e uma célula, gravar todos os seus atributos e uma indicação de quantos registros foram gravados.

10) Dado um arquivo e a indicação de quantos registros devem ser lidos, ler todos os atributos de uma célula e retornar um ponteiro para a área que os contém.

11) Dada uma célula, desenhá-la.

4.7 Aplicações

Subdivisões planares aparecem naturalmente em uma série de situações. Até mesmo aplicações tradicionais de geometria computacional, como triangulações de Delaunay e diagramas de Voronoy [PREP85], podem empregar a metodologia deste capítulo[6] .

A título de ilustração, cita-se a aplicação para a qual o EDP foi desenvolvido [CARV89, CAVA90], que consiste em gerar mapas cartográficos (fig. 4.13). Os mapas podem ser criados sem nenhuma preocupação com a ordem de digitalização das curvas. Quando uma subregião geográfica (uma face) é formada (por exemplo, um estado), o próprio EDP preenche a sua área com uma determinada cor, o que serve como *feedback* para o usuário. Existe um modo de operação que evita a geração de arestas pendentes quando da criação de uma face. Este modo recusa segmentos que interceptem arestas já existentes, a menos de uma certa tolerância. O mecanismo de atração implementado é aquele descrito na seção 4.5.1, sendo as tolerâncias ajustáveis, pelo usuário, a qualquer momento.

[6] Na realidade, digramas de Voronoy necessitam que sejam representadas arestas semi-infinitas, existindo, não apenas uma, mas várias faces ilimitadas.

As vantagens de criar mapas desta forma vêm a ser que atualizações posteriores podem ser introduzidas sem um impacto maior do que a digitalização de alguns novos segmentos e que a atribuição de atributos, aos elementos geográficos, pode ser feita, simplesmente, pela indicação do elemento na tela e o preenchimento dos campos correspondentes a partir de uma janela aberta pelo EDP.

Figura 4.13 - Mapa do Brasil gerado pelo EDP.

Na análise por elementos finitos, uma seção plana é estudada através da sua subdivisão em um certo número de elementos simples, normalmente, triângulos ou quadriláteros. Estes elementos definem uma subdivisão planar (fig. 4.14). Um modelador, similar ao EDP, pode ser criado para ser um gerador interativo de malhas. O usuário pode definir uma seção plana, decompô-la em regiões, especificar que regiões devem ser subdivididas e que técnica de subdivisão deve ser empregada em cada região [CARV90, CAMP91]. Posteriormente, a malha pode ser refinada em determinadas vizinhanças problemáticas.

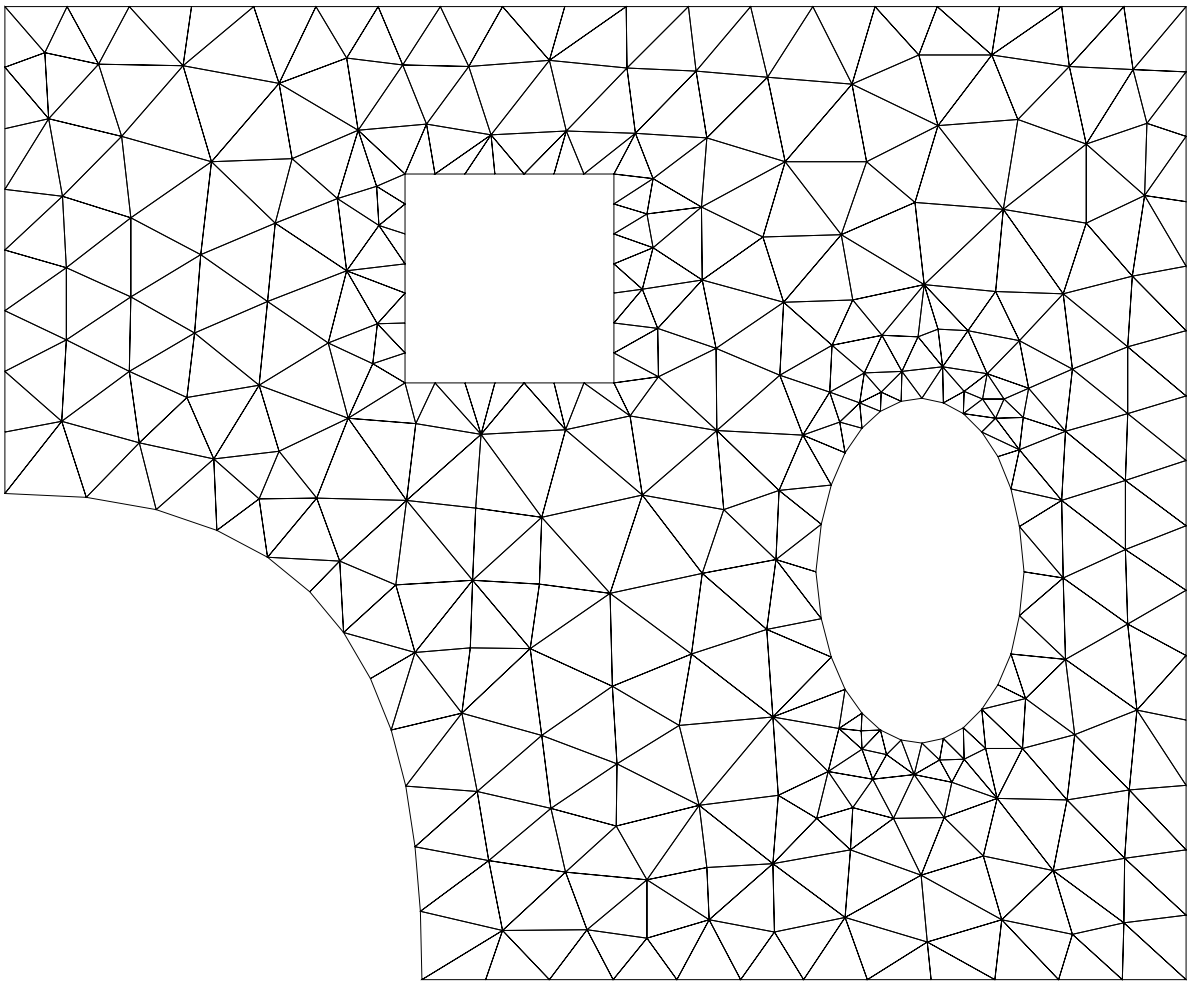


Figura 4.14 - Malha para modelo de elementos finitos.

5 - SUBDIVISÕES ESPACIAIS

Este capítulo generaliza, para o $\Re^3$, o que foi apresentado no capítulo anterior para o $\Re^2$. Seu objetivo é apresentar soluções para os problemas que surgem quando se deseja criar e manter subdivisões do $\Re^3$. Em alguns pontos uma certa repetição é inevitável, para evitar que o leitor tenha que se referir freqüentemente aos capítulos anteriores.

Um exemplo de uma subdivisão espacial simples está apresentada na figura 5.1. Nesta subdivisão, o espaço é dividido em duas regiões limitadas (r_1, r_2) e uma região ilimitada (r_{ext}). Cada região é delimitada por um conjunto de cascas que, por sua vez, são formadas por um conjunto de faces (e/ou arames) encadeadas.

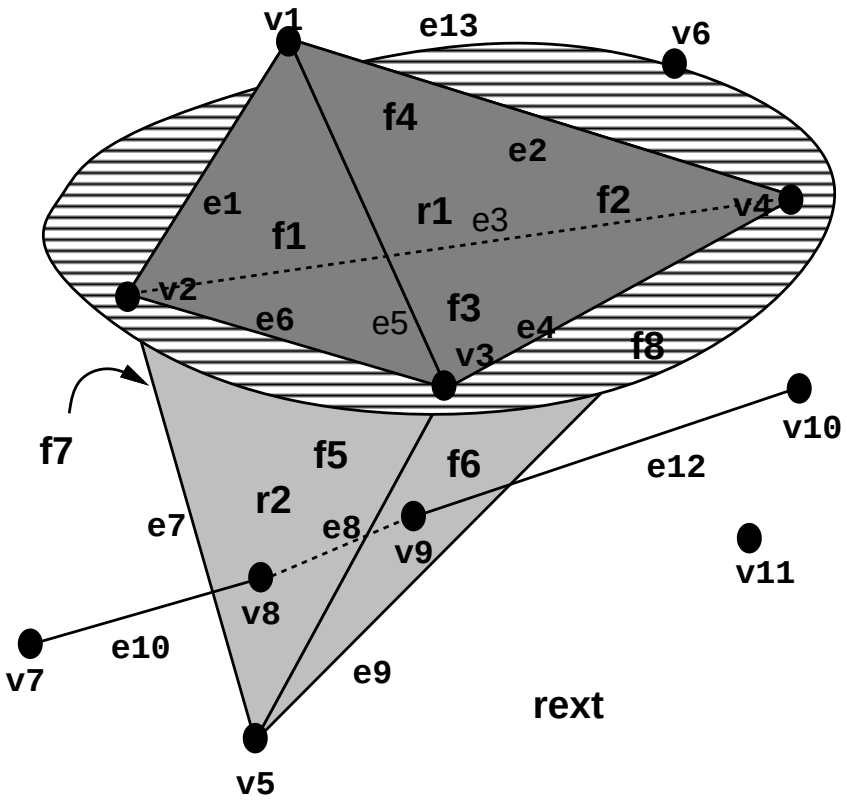


Figura 5.1 - Uma subdivisão do $\Re^3$.

Uma subdivisão do $\Re^3$ pode ser modelada por uma subdivisão espacial (SE), apresentada no capítulo 3. Neste caso, as células do complexo, de dimensão 0, são os vértices; as de dimensão 1, as arestas; as de dimensão 2, as faces; e as de dimensão 3, as regiões.

Um problema de grande interesse, e que generaliza o processo de modelagem do

capítulo 4, consiste em, dado um conjunto de retalhos de superfície (ou simplesmente retalhos), obter a SE determinada por estes retalhos. Um retalho é uma superfície algébrica[7] , orientável, conexa, sem singularidades e com bordo.

No decorrer deste capítulo, vai ser apresentada uma **metodologia para criação de SEs** composta por um modelo matemático, um esquema de representação e um processo de modelagem (para aplicações não interativas) [REQU80]. Um destaque especial vai ser dado ao aspecto relativo à construção de algoritmos geométricos eficientes, em especial, no que diz respeito a localização de um ponto em relação a uma dada subdivisão.

A metodologia foi utilizada para implementar o SSE (Sistema de Subdivisão Espacial) que é capaz de criar e manter uma subdivisão espacial, permitindo a inserção de novos retalhos em tempo real. Para lidar com a diversidade de geometrias de retalhos necessária às aplicações, foi identificado um conjunto de funções que permitem a geração de bibliotecas de geometrias manipuláveis pelo SSE. A introdução de uma nova geometria para os retalhos pode ser feita através da criação da biblioteca correspondente. Isto permite que seja utilizada a técnica de programação orientada para objeto.

5.1 Representação para SEs

Uma representação geométrica é uma estrutura de dados criada para descrever um objeto geométrico. Normalmente, ela contém dados simbólicos que descrevem a topologia do objeto (adjacências e incidências) e dados aritméticos (possivelmente imprecisos); por exemplo: coordenadas de vértices ou pontos de controle de uma curva de Bezier. Uma representação possui um modelo quando existe um objeto, no espaço Euclidiano, que satisfaz a sua parte simbólica. Aos dados numéricos da representação correspondem dados numéricos do modelo, o que pode requerer números com precisão infinita. Se uma representação tiver dados numéricos exatos ela é o seu próprio modelo e é chamada de representação natural.

Assim como para uma SP, embora uma descrição geométrica completa possa re-

(7) Neste trabalho, são consideradas apenas superfícies descritas por variedades algébricas.

presentar todas as informações sobre as formas geométricas das células, e suas respectivas localizações no espaço, é aconselhável combinar, na representação de uma SE, informações topológicas e geométricas. Isto porque, se dois retalhos de superfície se interceptam — embora a curva de interseção possa ser encontrada a partir da descrição geométrica de cada retalho — um processamento, em geral complicado, é exigido para sua determinação. Assim, uma representação para SEs que contenha de forma explícita todas as interseções, não só permite que os cálculos destas interseções sejam feitos uma única vez, no momento da sua criação, mas também evita a propagação de erros numéricos.

No contexto de SEs, quando se pensa em topologia, normalmente está-se referindo às adjacências entre as células, ou seja, proximidade física e ordenação. Informações de adjacência são referidas, informalmente, como a topologia da SE. Como vai ser visto no decorrer deste capítulo, a grande vantagem de armazenar, explicitamente, na representação de uma SE, a sua topologia é que isto possibilita a construção de algoritmos geométricos muito mais eficientes. A nomenclatura utilizada para descrever a topologia de uma SE é a seguinte (capítulo 3):

- Um **vértice** é uma célula, de dimensão 0, que corresponde a um único ponto do $\Re^3$. Não podem existir dois vértices com a mesma localização geométrica.
- Uma **aresta** é uma célula, de dimensão 1, que corresponde a um segmento de curva homeomorfo a um segmento de reta. Um **arame** é uma aresta que não pertence a uma face. A fronteira de uma aresta é formada por dois vértices não necessariamente distintos. Estes dois vértices são ditos ligados.
- Um **ciclo** é um conjunto conexo e ordenado de vértices e arestas (alternadamente) na fronteira de uma face. Um ciclo pode consistir de um único vértice, sendo chamado, neste caso, de ciclo pontual.
- Uma **face** é uma célula, de dimensão 2, que corresponde a uma porção conexa e limitada de uma superfície no $\Re^3$. Sua fronteira é composta por um ou mais ciclos. Um dos ciclos contém os demais e representa a sua fronteira externa. Os outros ciclos (se existirem) são chamados anéis e representam fronteiras internas (furos).

- Duas faces S e Q de uma SE estão **encadeadas** quando existe uma seqüência de faces $\{F_1, F_2, ..., F_n\}$ na SE com:
  (1) $F_1 = S$ e $F_n = Q$;
  (2) $\Gamma_i = (\text{bdry}(F_i,\text{SE}) \cap \text{bdry}(F_{i+1},\text{SE})) \neq \emptyset$, $i \in [1, n-1]$. Se todo Γ_i contiver pelo menos uma aresta, diz-se que S e Q estão fortemente encadeadas.
- Uma **casca** é um subconjunto conexo de células na fronteira de uma região. Ela pode ser formada por um conjunto de faces encadeadas, apenas por arames, ou, até mesmo, por um único vértice, sendo chamada, neste caso, de casca pontual.
- Uma **região** é uma célula, de dimensão 3, que corresponde a um volume conexo do $\Re^3$. Sua fronteira é formada por uma ou mais cascas. Uma das cascas contém as demais e representa a sua fronteira externa. As outras (se existirem) representam fronteiras internas (cavidades).

Em uma SE é possível formar dezesseis tipos de relacionamentos de adjacência que caracterizam as adjacências entre vértices, faces, arestas e regiões [WEIL86]. Cada relacionamento é representado por um par de letras (seção 4.1). Escrevendo estes relacionamentos, na terminologia do capítulo 3, em função dos operadores bdry, star e adj, tem-se que:

$\text{bdry}(r,\text{SE}) = r\{F\} \cup r\{A\} \cup r\{V\}$
$\text{bdry}(f,\text{SE}) = f < A > \cup\ f < V >$
$\text{bdry}(a,\text{SE}) = a\{V\}^2$
$\text{star}(f,\text{SE}) = f\{R\}^2$
$\text{star}(a,\text{SE}) = a < R > \cup\ a < F >$
$\text{star}(v,\text{SE}) = v\{A\} \cup v\{F\} \cup v\{R\}$
$\text{adj}(r,\text{SE}) = r\{R\} \quad \text{adj}(f,\text{SE}) = f\{F\}$
$\text{adj}(a,\text{SE}) = a\{A\} \quad \text{adj}(v,\text{SE}) = v\{V\}$

Situações onde mais de duas faces incidem na mesma aresta, ou vários volumes estão conectados por um único vértice, são comuns em uma SE (fig. 5.2). Por isso, uma representação para SEs deve ser capaz de armazenar, de alguma forma, este tipo de condição.

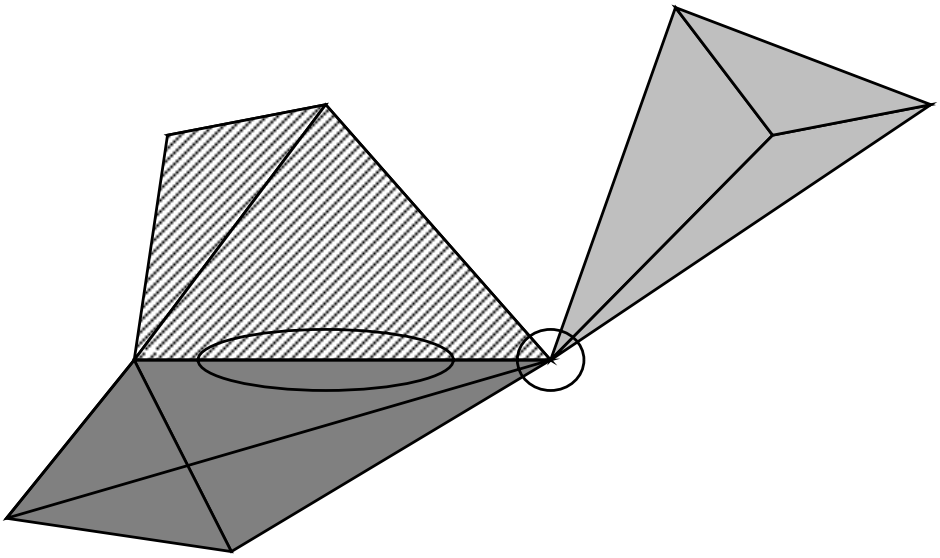


Figura 5.2 - Condições *non-manifold* em um ponto e ao longo de uma curva.

Kevin Weiler [WEIL86] criou uma estrutura de dados capaz de armazenar os relacionamentos de adjacência de uma SE. Esta estrutura de dados é conhecida por *radial edge* ou RED pelo fato de ser armazenada, explicitamente, a lista de faces, ordenadas radialmente, em torno de uma aresta (fig. 5.10). Weiler criou a RED para ser utilizada em modelagem *non-manifold* e provou que ela é completa, ou seja, permite que se extraia qualquer relacionamento de adjacência presente em uma SE.

5.1.1 A Estrutura de Dados RED

A exemplo do que ocorre com a estrutura HED, a RED utiliza o conceito de uso de um elemento topológico para armazenar a topologia de uma SE. Um uso pode ser visto como a ocorrência de um elemento topológico em um relacionamento de adjacência com um elemento de dimensão superior. Assim, a RED armazena explicitamente os dois usos (lados) de uma face pelas duas regiões (não necessariamente distintas) que a compartilham. Cada uso de face é limitado por um ou mais usos de ciclo, que por sua vez são formados por uma seqüência alternada de usos de aresta e usos de vértice (fig. 5.3). Usos de vértice são utilizados para captarem condições *non-manifold* nos vértices, tal como o marcado com um círculo na figura 5.2.

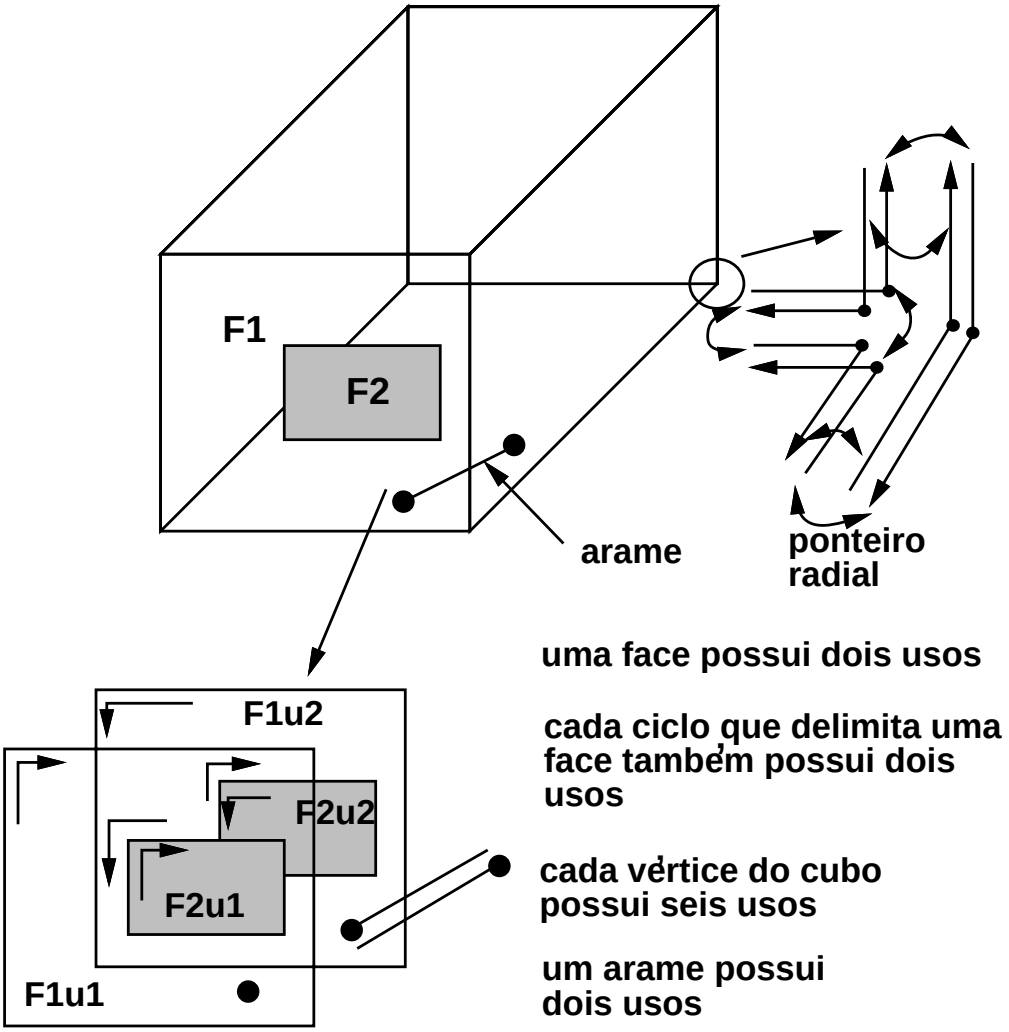


Figura 5.3 - Usos de faces, ciclos e vértices.

A RED apresenta uma descrição hierárquica de uma SE, partindo de níveis mais altos em dimensão (regiões), para os mais baixos (vértices) (fig. 5.4). Todos os elementos topológicos são mantidos em listas circulares duplamente encadeadas e possuem ponteiros para atributos. A convenção adotada para nome de ponteiros é a concatenação dos caracteres que simbolizam *de elemento para elemento* `_ptr`. Os caracteres utilizados são `m`, `r`, `s`, `f`, `l`, `e`, `v`, `fu`, `lu`, `eu` e `vu`. Todos os `next` e `last` são ponteiros de listas circulares duplamente encadeadas.

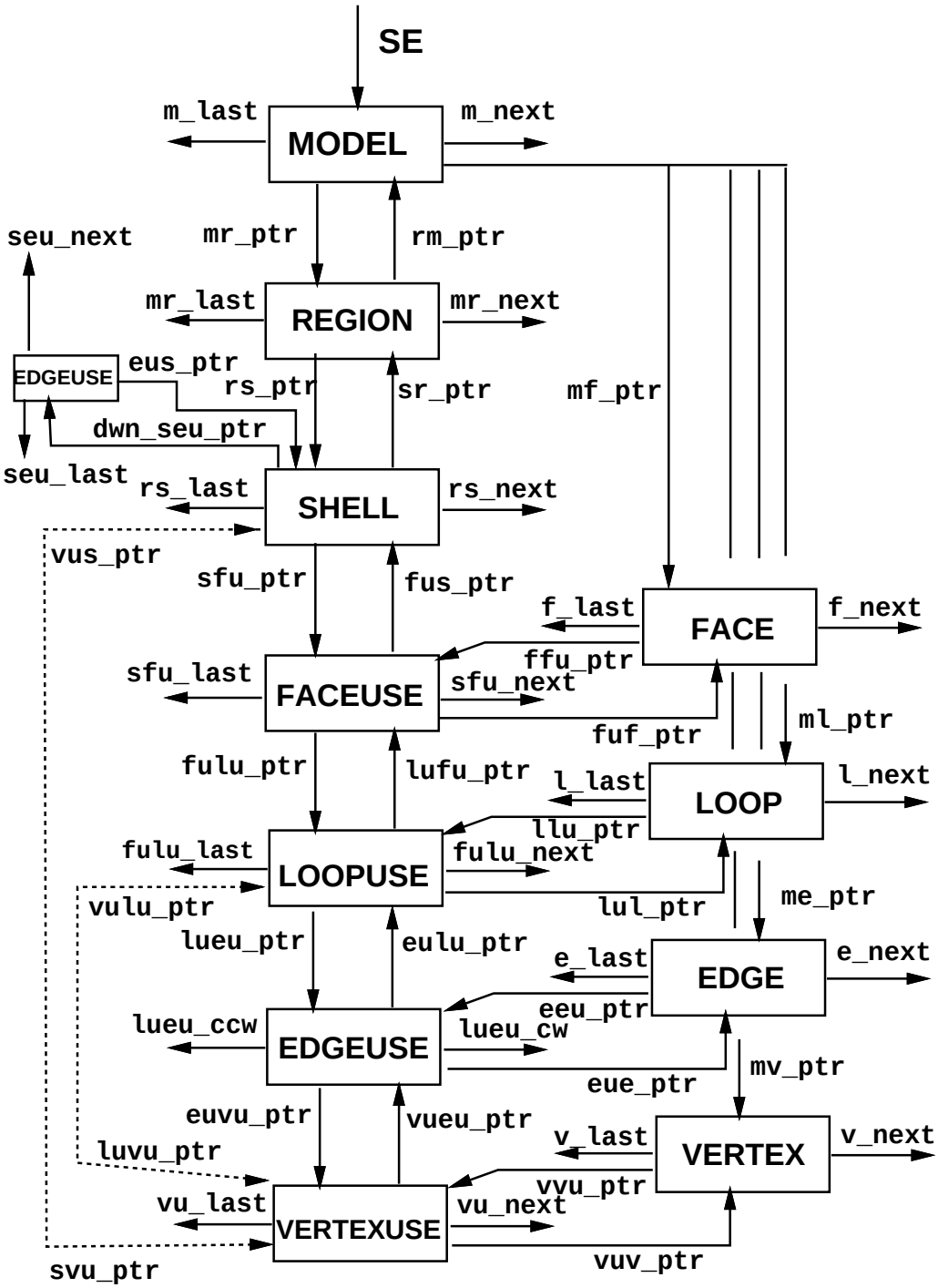


Figura 5.4 - Hierarquia da RED.

A declaração em C de cada estrutura que compõe a RED é apresentada a seguir:

```
typedef struct model     Model;
typedef struct region    Region;
typedef struct shell     Shell;
typedef struct face      Face;
typedef struct loop      Loop;
typedef struct edge      Edge;
typedef struct vertex    Vertex;
typedef struct faceuse   Faceuse;
typedef struct loopuse   Loopuse;
typedef struct edgeuse   Edgeuse;
typedef struct vertexuse Vertexuse;
typedef void   *Attr_ptr;
struct model
{
 int      cnt;      /* counter              */
 Model   *m_next;   /* linked list fwd ptr  */
 Model   *m_last;   /* linked list back ptr */
 Region  *mr_ptr;   /* region list pointer  */
 Face    *mf_ptr;   /* face list pointer    */
 Loop    *ml_ptr;   /* loop list pointer    */
 Edge    *me_ptr;   /* edge list pointer    */
 Vertex  *mv_ptr;   /* vertex list pointer  */
 Attr_ptr a_ptr;    /* model attribute ptr  */
};
```

Uma SE é armazenada na estrutura `Model` que possui ponteiros para listas circulares com cada tipo de elemento topológico da SE (regiões, faces, ciclos, arestas e vértices). Isto permite, por exemplo, que todos os elementos de um determinado tipo sejam encontrados rapidamente e sem repetição.

```
struct region
{
 int      cnt;     /* counter              */
 Model   *rm_ptr;  /* owning model pointer */
 Region  *mr_next; /* linked list fwd ptr  */
 Region  *mr_last; /* linked list back ptr */
 Shell   *rs_ptr;  /* shell list pointer   */
 Attr_ptr a_ptr;   /* region attribute ptr */
};
/*
*/
struct shell
{
 int      cnt;        /* counter              */
 Region  *sr_ptr;     /* owning region ptr    */
 Shell   *rs_next;    /* linked list fwd ptr  */
 Shell   *rs_last;    /* linked list back ptr */
 Attr_ptr a_ptr;      /* shell attributes ptr */
 Desc_type downptr;   /* down pointer type    */
 union
 {
  Faceuse   *sfu_ptr; /* face use list   */
  Vertexuse *svu_ptr; /* vertex use list */
 } dwn;               /* down pointer    */
 Edgeuse    *dwn_seu_ptr; /* (wire) edge use list or NULL */
};
```

A estrutura para região possui um ponteiro para a lista das suas cascas (campo `rs_ptr`) e um ponteiro para a sua SE (campo `rm_ptr`). Uma casca, por sua vez, pode possuir faces e/ou arames ou ser uma casca pontual. O campo `downptr` assume os valores FACEUSE, EDGEUSE ou VERTEXUSE, respectivamente. Se a casca possuir pelo menos um uso de face, o campo `dwn` será um ponteiro para uma lista de usos de face. Se ela for uma casca pontual, `dwn` será um ponteiro para o único uso do vértice. Se a casca possuir arames, o campo `dwn_seu_ptr` será um ponteiro para a lista com o par de usos de cada arame. Caso contrário, será nulo.

```
struct face
{
 int      cnt;     /* counter              */
 Face    *f_next;  /* linked list fwd ptr  */
 Face    *f_last;  /* linked list back ptr */
 Faceuse *ffu_ptr; /* face use list        */
 Attr_ptr a_ptr;   /* face attributes ptr  */
};
/*
*/
struct loop
{
 int      cnt;     /* counter              */
 Loop    *l_next;  /* linked list fwd ptr  */
 Loop    *l_last;  /* linked list back ptr */
 Loopuse *llu_ptr; /* loop use list        */
 Attr_ptr a_ptr;   /* loop attributes ptr  */
};
/*
*/
struct edge
{
 int       cnt;     /* counter              */
 Edge     *e_next;  /* linked list fwd ptr  */
 Edge     *e_last;  /* linked list back ptr */
 Edgeuse  *eeu_ptr; /* edge use list        */
 Attr_ptr  a_ptr;   /* edge attributes ptr  */
};
/*
*/
struct vertex
{
 int        cnt;     /* counter              */
 Vertex    *v_next;  /* linked list fwd ptr  */
 Vertex    *v_last;  /* linked list back ptr */
 Vertexuse *vvu_ptr; /* vertex use list      */
 Attr_ptr   a_ptr;   /* vertex attrib ptr    */
};
```

As estruturas para faces, ciclos, arestas e vértices são mantidas em listas circulares duplamente encadeadas e possuem ponteiros para um dos seus usos. Estas estruturas existem porque os aplicativos lidam com as entidades físicas e não com os seus usos

topológicos. Desta forma, atributos são, geralmente, associados às entidades físicas, evitando a multiplicação de ponteiros para atributos em todos os seus usos.

```
struct faceuse
{
 int       cnt;           /* counter              */
 Shell    *fus_ptr;       /* owning shell         */
 Faceuse  *sfu_next;      /* linked list fwd ptr  */
 Faceuse  *sfu_last;      /* linked list back ptr */
 Faceuse  *fufu_mate_ptr; /* opposite side        */
 Loopuse  *fulu_ptr;      /* list of face loops   */
 Orient    orientation;   /* compared to geom def */
 Face     *fuf_ptr;       /* face descriptor      */
 Attr_ptr  a_ptr;         /* attributes           */
};
```

A estrutura para uso de face possui um ponteiro para o outro uso (lado) da mesma face (campo `fufu_mate_ptr`), um ponteiro para a lista dos seus usos de ciclo (campo `fulu_ptr`) e ponteiros para a sua casca e a sua face.

```
struct loopuse
{
 int      cnt;           /* counter             */
 Faceuse *lufu_ptr;      /* owning face use     */
 Loopuse *fulu_next;     /* linked list fwd ptr */
 Loopuse *fulu_last;     /* linked list back pt */
 Loopuse *lulu_mate_ptr; /* opposite side       */
 Loop    *lul_ptr;       /* loop descriptor     */
 Attr_ptr a_ptr;         /* attributes          */
 Desc_type downptr;      /* type of down ptr    */
 union
 {
  Edgeuse   *lueu_ptr;   /* edge use     */
  Vertexuse *luvu_ptr;   /* vertex use   */
 } dwn;                  /* down pointer */
};
```

A estrutura para uso de ciclo possui um ponteiro para o outro uso do mesmo ciclo (no outro lado da face) (campo `lulu_mate_ptr`), para o uso de face ao qual pertence

(campo `lufu_ptr`) e para o seu ciclo. Um ciclo pode possuir um conjunto de arestas ou ser um ciclo pontual. O campo `downptr` assume os valores EDGEUSE ou VERTEXUSE, respectivamente. Se o ciclo possuir pelo menos um uso de aresta, o campo `dwn` será um ponteiro para a lista circular ordenada de usos de aresta. Caso contrário, será um ponteiro para o seu uso do vértice.

```
struct edgeuse
{
 int        cnt;           /* counter          */
 Vertexuse *euvu_ptr;      /* starting vertex  */
 Edgeuse   *eueu_mate_ptr; /* eu mate          */
 Edge      *eue_ptr;       /* edge descriptor  */
 Attr_ptr   a_ptr;         /* attributes       */
 Orient     orientation;   /* relative to geom */
 Desc_type  upptr;         /* up pointer type  */
 union
 {
  Shell   *eus_ptr;  /* if shell ptr */
  Loopuse *eulu_ptr; /* if loopuse   */
 } up;               /* up pointer   */
 union
 {
  Edgeuse *lueu_cw;  /* clockwise edgeuse */
  Edgeuse *seu_next; /* next edgeuse      */
 } fwd;              /* fwd pointer       */
 union
 {
  Edgeuse *lueu_ccw;        /* ccw edgeuse    */
  Edgeuse *seu_last;        /* last edgeuse   */
 } bck;                     /* back pointer   */
 Edgeuse  *eueu_radial_ptr; /* radial edgeuse */
};
```

A estrutura para uso de aresta desempenha um papel central na RED. É ela que contém as informações que mantêm as faces ligadas umas às outras. Uma aresta pode delimitar uma face ou não (neste caso, trata-se de um arame). O campo `upptr` assume os valores SHELL ou FACEUSE, respectivamente. Se for um arame, o campo `up` será um ponteiro para a sua casca e o campo `fwd` (`bck`) é o uso que o sucede (antecede), na lista de arames da casca. Caso contrário, o campo `up` será um ponteiro para o uso de

ciclo que o utiliza e o campo `fwd` (`bck`) é o uso que o sucede (antecede) na lista circular ordenada de usos de aresta que delimitam este uso de ciclo. O campo `eueu_radial_ptr` é um ponteiro para o outro uso, da mesma aresta, que delimita o próximo uso de face em torno desta aresta (fig. 5.5). Há ainda um ponteiro para o outro uso de aresta na mesma face (campo `eueu_mate_ptr`), um ponteiro para a sua aresta e um ponteiro para o seu uso do seu vértice inicial (campo `euvu_ptr`). Note-se que os ponteiros `eueu_radial_ptr` e `eueu_mate_ptr` fornecem um acesso ordenado (radialmente) para os usos de aresta que delimitam os usos de face que incidem na aresta.

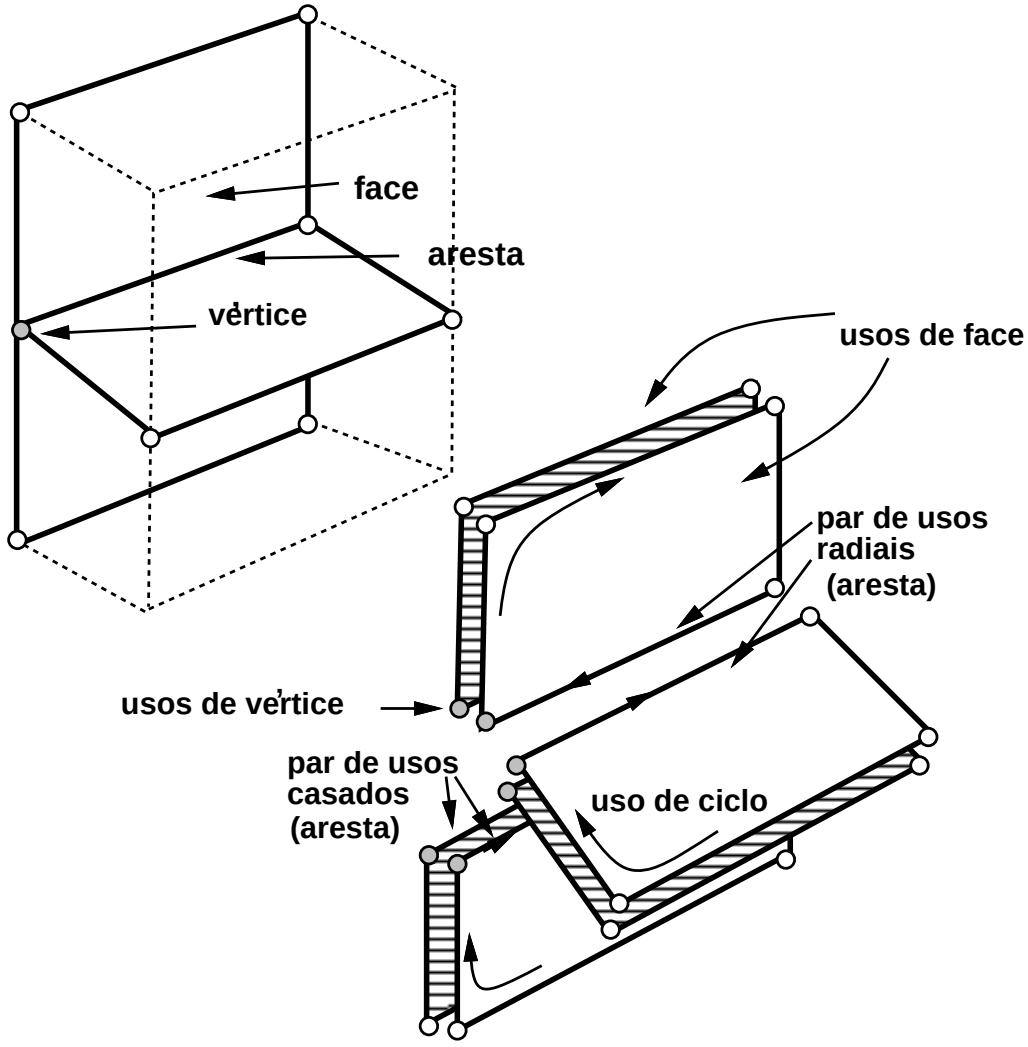


Figura 5.5 - Relacionamento $a < F >$ na RED.

```
struct vertexuse
{
 int        cnt;     /* counter              */
 Vertexuse *vu_next; /* linked list fwd ptr  */
 Vertexuse *vu_last; /* linked list back ptr */
 Vertex    *vuv_ptr; /* vertex descriptor    */
 Attr_ptr   a_ptr;   /* attributes ptr       */
 Desc_type  upptr;   /* up pointer type      */
 union
 {
  Shell   *vus_ptr;  /* shell      */
  Loopuse *vulu_ptr; /* loop use   */
  Edgeuse *vueu_ptr; /* edge us    */
 } up;               /* up pointer */
};
```

Finalmente, a estrutura para uso de vértice contém as informações que mantêm ligadas as faces e arames que se tocam apenas no vértice. Um vértice pode delimitar uma aresta, um ciclo pontual ou uma casca pontual. O campo `upptr` assume os valores EDGEUSE, LOOPUSE ou SHELL, respectivamente. Se delimitar uma aresta, o campo `up` será um ponteiro para o uso de aresta que o utiliza. Se delimitar um ciclo pontual, será um ponteiro para uso do ciclo que o utiliza e se delimitar uma casca pontual, será um ponteiro para a casca. Há ainda um ponteiro para o seu vértice.

Deve-se frisar que a RED representa exatamente o que é necessário para uma SE, ou seja, um complexo geométrico **completo** de dimensão 3. É justamente o conceito de uso de face, mais a ordenação radial de faces em torno de arestas, que permite representar ordenadamente as adjacências das células de dimensão 3. Note-se que, ao contrário do que ocorre numa subdivisão planar, não existe uma ordenação possível para as adjacências de arestas, faces e regiões ao redor de vértices.

5.1.2 Operadores *non-manifold*

Estruturas de dados capazes de armazenar uma SE são muito mais complexas do que as capazes de armazenar apenas uma SP e também não devem ser manipuladas diretamente. Kevin Weiler [WEIL86] introduziu um conjunto de operadores, chamados neste trabalho WOps, que fornecem um meio, de relativo alto nível, para manipular a RED[8] . Estes operadores estão divididos em dois grupos. O primeiro possui operadores que agem exclusivamente sobre as faces de uma SE e são análogos aos apresentados na seção 4.1.2, para SPs. O segundo possui operadores que são capazes de criar arames e adicionar faces que são "costuradas" em arestas ou arames especificados (fig. 5.6).

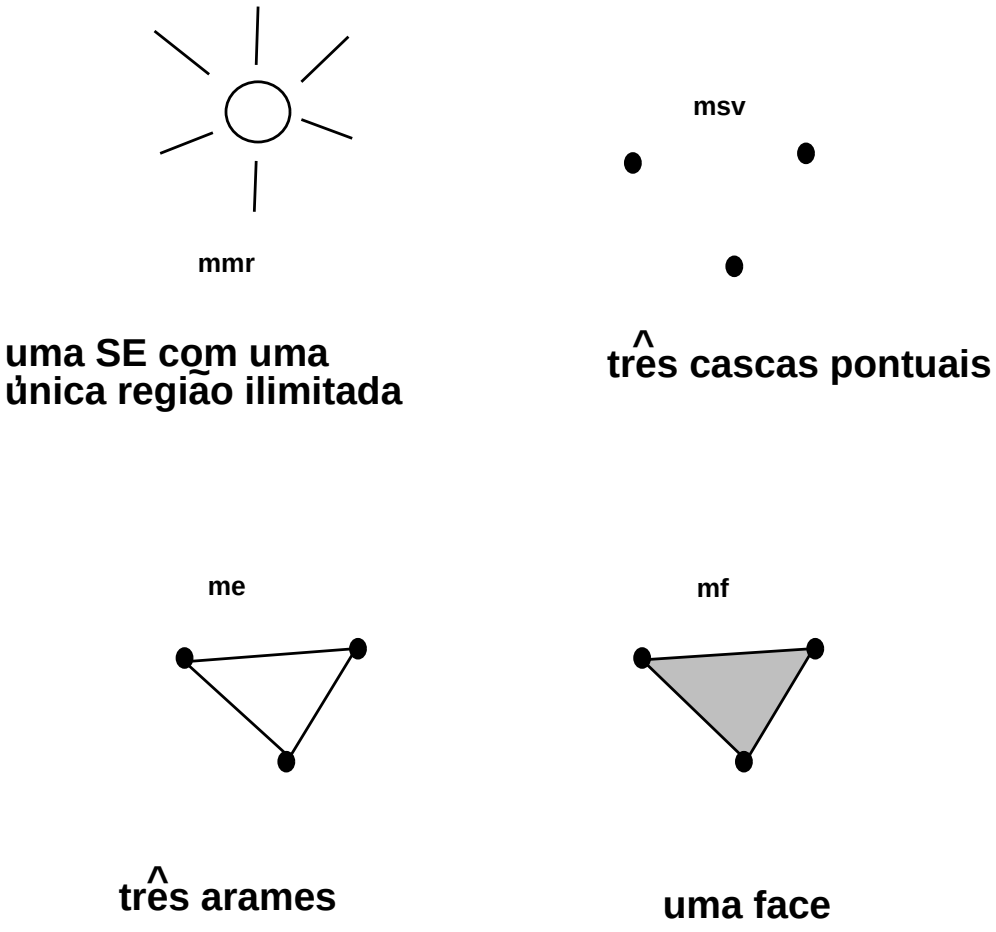


Figura 5.6 - Efeito de WOps.

[8] Embora atualmente já existam trabalhos que apresentam operadores *non-manifold* de uma forma mais rigorosa (por exemplo, [MURA90]), ainda não existe um trabalho definitivo sobre o assunto.

Considerações a respeito de um conjunto mínimo de operadores podem ser encontradas em [TING90]. Os WOps do segundo grupo, e os seus respectivos inversos, são os seguintes:

`mmr`	(*make SE and region*)	`km`	(*kill SE*)
`msv`	(*make shell and vertex*)	`kv`	(*kill vertex*)
`mev`	(*make wire edge and vertex*)		
`me`	(*make wire edge*)	`ke`	(*kill edge*)
`mf`	(*make face*)	`kf`	(*kill face*)
`esplit`	(*split edges*)	`join`	(*join edges*)

O WOp `mmr` cria uma SE com uma única região ilimitada. O WOp `msv` cria uma nova casca pontual em uma dada região. O WOp `mev` cria um arame e um vértice em uma dada região. O arame é delimitado por um vértice já existente e pelo vértice criado. O WOp `me` cria um arame entre dois vértices de uma dada região. O WOp `esplit` divide uma aresta (ou um arame) em duas, criando um novo vértice entre elas.

O WOp `mf` insere uma nova face f, na SE. Os vértices e arestas que formam a fronteira de f devem existir na SE. Este WOp recebe como entrada as arestas a_i de $f < A >$ e, para cada a_i, a face que deve suceder f no ciclo ordenado de faces ao redor de a_i. Através de um algoritmo de caminhamento, este operador detecta o fechamento de uma nova região. Por adjacência de arestas, são percorridos e marcados todos os usos de face que podem ser atingidos a partir de um dos dois usos da face criada. Ao final, se o outro uso não estiver marcado é porque foi formada uma nova região.

Um processo semelhante é usado para, caso seja formada uma nova região, distribuir os usos de face entre a região antiga e a nova. Se as faces que delimitam estas regiões não estiverem fortemente encadeadas (estando conectadas às suas cascas apenas por vértices ou arames), alguns usos de face simplesmente não são percorridos e, conseqüentemente, passam a não pertencer a nenhuma região (fig. 5.7). Em função disto, foi adicionado um parâmetro de saída ao WOp especificado por Weiler. Este novo parâmetro é uma lista (possivelmente vazia) com os usos de face não classificados. Esta classificação deve ser feita, posteriormente, baseada em geometria.

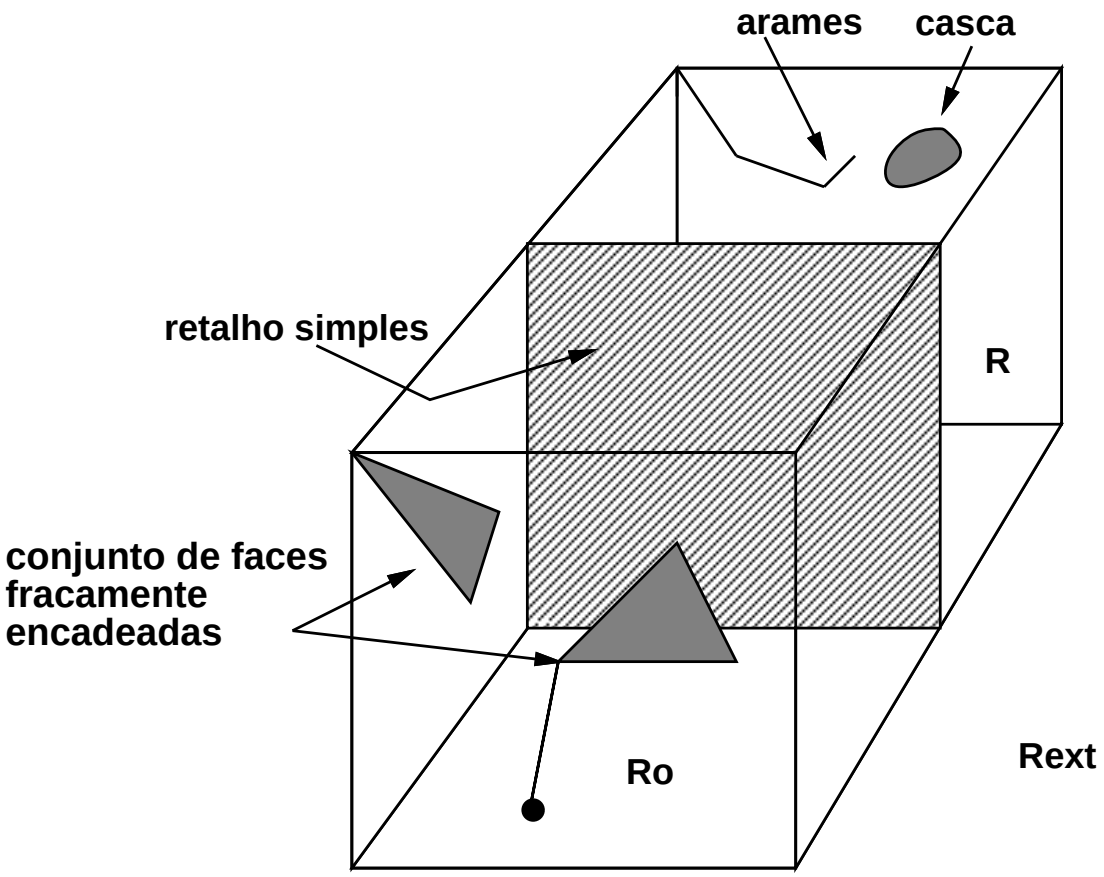


Figura 5.7 - Criação de uma nova região.

O operador `mf` também foi estendido para ser capaz de criar faces com fronteiras desconexas (multi-ciclos). A forma mais simples de fazer isto, ficando o mais próximo possível do que foi especificado por Weiler, é passando o número de ciclos da face e, para cada ciclo, as mesmas informações passadas ao operador original. Para o caso de um ciclo pontual, passa-se o número de arestas (0) para este ciclo e um ponteiro para o vértice. Além disso, deve ser passada a face incidente no vértice (se houver alguma) e uma indicação do seu uso que é adjacente (por região) à região que contém a face que se deseja incluir (fig. 5.8). Esta é a forma de informar ao operador a que casca a nova face deve pertencer, caso ela só esteja conectada pelo ciclo pontual. Este novo operador deve ser capaz de eliminar as cascas que passaram a estar conectadas pela adição da nova face. Permite-se, também, que estes ciclos contenham arestas pendentes.

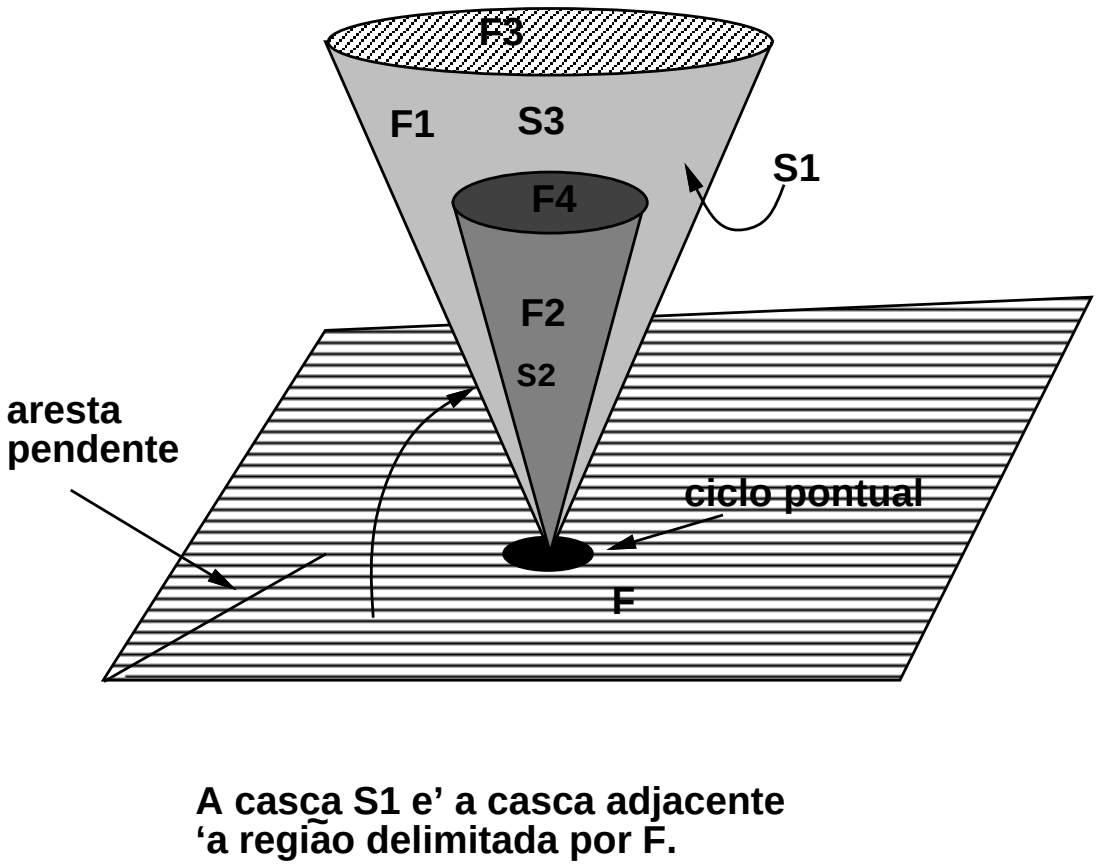


Figura 5.8 - Criação de uma face com um ciclo pontual.

Em cada WOp foi inserido um trecho de código encarregado de chamar um procedimento que registra uma descrição do WOp e os respectivos parâmetros. Isto permite que possam ser executados, posteriormente, algoritmos de *undo*, *redo* ou de inversão (para o armazenamento permanente de uma SE, conforme descrito na seção 5.6.1).

Os WOps podem ser entendidos como uma maneira de construir um complexo completo a partir do seu esqueleto. Especificamente, eles são capazes de adicionar uma nova célula, de dimensão menor que 3, desde de que ela encaixe na SE. O WOp `mf` é o operador encarregado de completar o complexo, criando as regiões (células de dimensão 3) quando se faz necessário.

5.2 Processo de Modelagem

Um processo de modelagem deve fornecer, a um usuário típico de um sistema, uma maneira, familiar e eficiente, para entrada dos dados que permitirão a construção da representação de um modelo. Porém, interação em 3D é uma tarefa intrinsicamente problemática porque os dispositivos de entrada de dados atuais são essencialmente bi-dimensionais e o raciocínio tri-dimensional é pouco exercitado pela maioria das pessoas.

Uma forma de interação adequada a um usuário comum é baseada em um processo construtivo. O modelo é criado a partir de objetos tri-dimensionais pré-existentes que são combinados entre si formando objetos mais complexos. Esta combinação é definida em termos de operações sobre o conjunto de pontos de cada objeto e são chamadas de operações *booleanas*.

Um processo de modelagem CSG baseia-se em operações *booleanas* regularizadas [REQU77] de união, interseção e diferença e é muito popular entre usuários de sistemas de modelagem sólida. Como processo de modelagem para criação de SEs, no entanto, o processo CSG não é adequado por um motivo muito simples: os objetos (CSG), válidos para operações *booleanas*, são sólidos regulares que dividem o espaço em apenas três regiões (interior, fronteira e exterior do sólido). No capítulo 6, será apresentado um processo de modelagem interativo que é uma generalização do processo CSG e pode ser aplicado para construir SEs.

Para aplicações não interativas, o problema de maior interesse consiste em, dado um conjunto de retalhos de superfície (ou simplesmente retalhos), obter a SE determinada por estes retalhos.

Como na criação de SPs, vai-se considerar um processo incremental, no qual cada retalho é inserido na SE, que é atualizada a cada inserção. O retalho a ser inserido determina um SGC completo, que deve ser combinado com o SGC associado à SE já existente. De modo análogo ao caso bi-dimensional, o novo retalho precisa ser refinado, de modo a que possa ser encaixado na SE. Para tal, é necessário calcular todos os segmentos de curva definidos pelas interseções entre retalhos, criando vértices nas in-

terseções destes segmentos, arestas entre cada par de vértices ligados por um segmento e regiões, sempre que um conjunto de retalhos formar um volume fechado.

5.3 Inserção de Retalhos

Um retalho de superfície é uma superfície algébrica, orientável, sem singularidades, conexa e com bordo. Um retalho determina unicamente um complexo completo, de dimensão 3, com uma única face e com apenas uma região (ilimitada). A fronteira da face corresponde ao bordo do retalho. No que se segue, o termo retalho vai ser empregado para se referir, tanto ao retalho em si, como ao seu complexo associado.

Um procedimento para inserir retalhos em uma SE deve garantir que, após a inserção, a SE continue consistente topológica e geometricamente. A consistência topológica é mantida, naturalmente, se forem utilizados WOps de forma apropriada. Para isto, o retalho deve ser subdividido em um conjunto de retalhos inteiramente contidos em regiões da SE, chamados retalhos simples. Isto corresponde a refinar o retalho para que ele encaixe na SE. Ao final do refinamento, o retalho pode possuir várias faces. Cada uma destas faces determina um retalho simples que pode ser inserido pelos WOps apropriados.

Para subdividir um retalho K é necessário descobrir que faces f_i da SE são cortadas e os segmentos de curva determinados por cada interseção (fig. 5.9). Estes segmentos são usados para refinar K e cada f_i. Este refinamento pode ser feito inserindo cada segmento de curva nas faces apropriadas. Como o suporte de uma face é homeomorfo ao $\Re^2$, o método descrito na seção 4.3.1 para inserir um segmento simples em uma face (plana) de uma SP, pode ser adaptado.

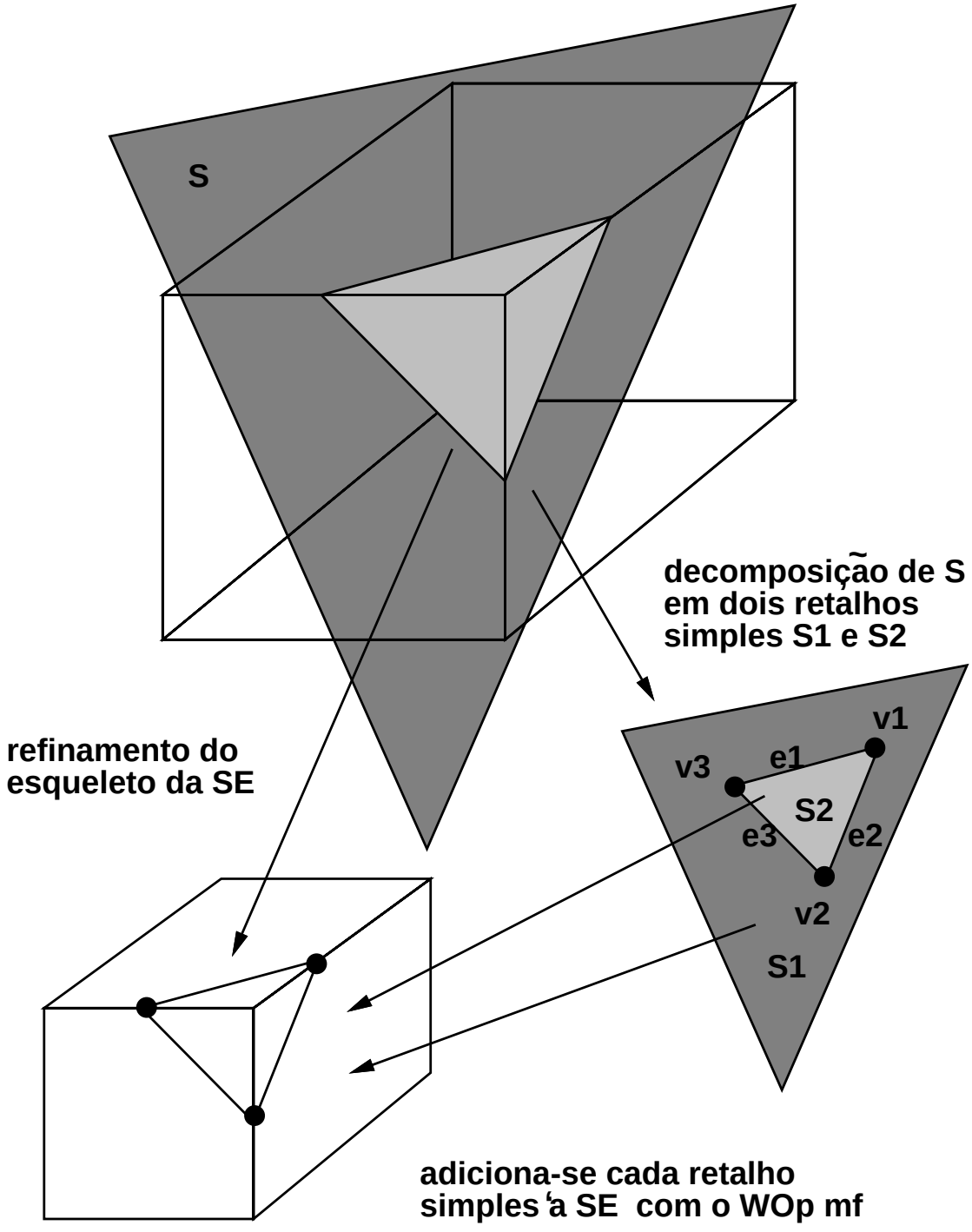


Figura 5.9 - Inserção de um retalho em uma SE.

5.3.1 Inserção de Retalhos Simples

Uma vez que se tenha subdividido um retalho criando um conjunto de retalhos simples, cada um deles deve ser inserido na SE. O problema se reduz, então, em como inserir um retalho simples S. Para isto, suponha-se que todos os usos de face da SE estão orientados no sentido anti-horário, isto é, os usos dos ciclos externos estão orientados no sentido horário (vistos de dentro das regiões delimitadas pelos respectivos usos de face) e os internos (se houver algum) no sentido anti-horário.

Para incluir S é necessário determinar:

1) que vértices v_i e arestas a_i da SE serão "costuradas" ao bordo de S.
2) para cada a_i, encontrar a face que deve suceder S no ciclo ordenado de faces ao redor de a_i.
3) que região R_o contém S.

O primeiro passo é encontrar os vértices da SE que coincidem geometricamente com vértices de S. Estes vértices são armazenados em uma lista V_l. O segundo passo é encontrar as arestas da SE que coincidem geometricamente com arestas de S. Somente arestas que ligam vértices armazenados em posições consecutivas de V_l precisam ser consideradas. Estas arestas são armazenadas em uma lista E_l.

Ao final deste processo, se E_l estiver vazia é porque S não está conectado a SE, ou está conectado a ela apenas por vértices. Em ambos os casos, R_o é a região que contém um ponto qualquer de uma aresta de S. Se E_l não estiver vazia, então para cada aresta a_i, armazenada em E_l, deve ser encontrada a face que sucede S no ciclo ordenado de faces ao redor de a_i. Em um domínio planar, basta considerar cada vetor $V(f)$ perpendicular a a_i e contido em uma face f incidente em a_i. A face f', que sucede S, é aquela cujo $V(f')$ forma ângulo orientado mínimo com $V(S)$. Esta face é armazenada em uma lista F_l. A face f'', que antecede S, é aquela cujo $V(f'')$ forma ângulo orientado máximo com $V(S)$. R_o é a região delimitada por cada face que sucede e antecede S, no ciclo ordenado de faces em torno de cada a_i (fig. 5.10).

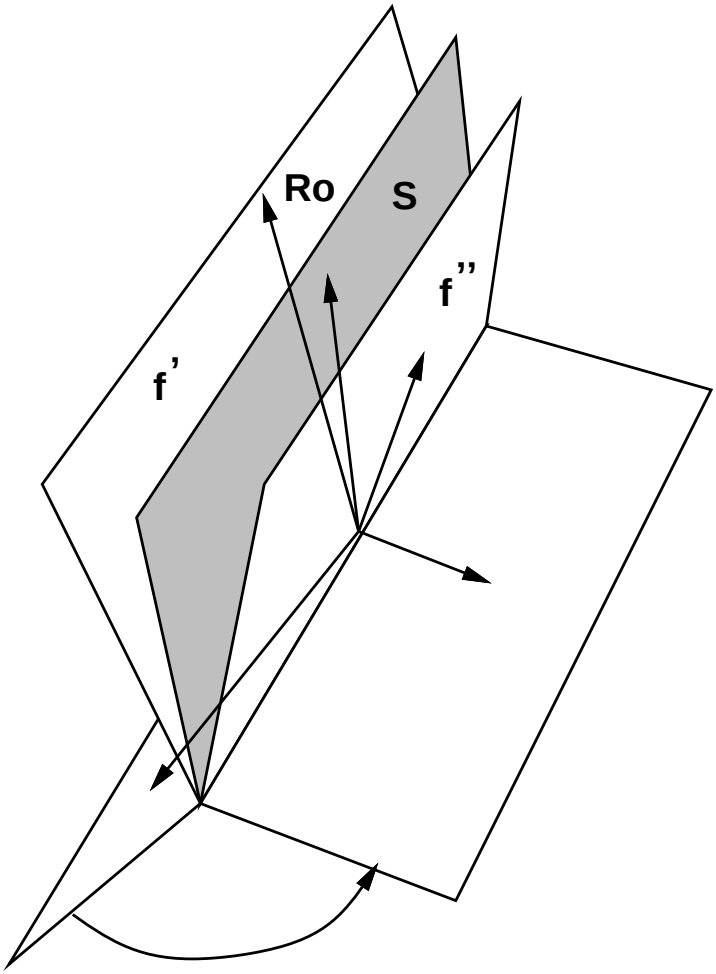


Figura 5.10 - Ordenação radial de faces em torno de arestas.

Para um domínio não planar, este método pode ser adaptado considerando $V(f)$ perpendicular ao vetor tangente à face f em um ponto qualquer de a_i. Isto funciona porque, se duas superfícies algébricas se interceptam ao longo de um trecho de uma aresta, então, a curva de interseção tem como suporte o suporte da aresta.

O útimo passo para inserção de S é adicionar cada v_i e cada a_i não presentes na SE, utilizando os WOps apropriados (`me` ou `mev`), criando arames. As listas V_l e E_l devem ser completadas com os vértices e arames criados. Desta forma, S passa a encaixar na SE e pode ser adicionado com o WOp `mf`. As listas V_l, E_l e F_l permitem determinar os parâmetros necessários.

5.3.2 Criação de Uma Nova Região

A adição de S pode ocasionar a criação de uma nova região R na SE. Uma nova região é sempre criada a partir de uma casca de construção de R_o. Neste caso, R pode (fig. 5.7):

- estar com a fronteira orientada de forma inconsistente com a orientação arbitrada.
- possuir um conjunto Γ de faces (possivelmente vazio), conectadas a ela apenas por vértices ou arames (e por isso não foram acrescentadas a sua casca externa).
- conter algumas cascas ou arames de R_o.

Para cada um destes casos, os seguintes procedimentos devem ser executados para corrigir a representação:

- calcular o volume com sinal de R e, caso o sinal não seja o esperado, trocar a casca de construção, pela casca externa de R. Com a convenção adotada, o volume de todas as regiões é positivo, exceto da região externa, que é negativo.
- verificar, para cada face de Γ, se um ponto qualquer de alguma aresta está contido em R. Se estiver, colocá-la na casca externa de R. Caso contrário, colocá-la na casca externa de R_o.
- verificar se um vértice qualquer, de cada casca de R_o (com exceção da sua casca externa e da casca de construção), está contido em R. Se estiver movê-la para R.
- verificar se um ponto qualquer de cada arame, da casca externa de R_o, está contido em R. Se estiver movê-lo para R.

Para obter o volume com sinal de uma região R, basta percorrer cada face da fronteira externa de R e calcular o somatório dos volumes de cada pirâmide determinada pela face em relação à origem.

Para testar a inclusão de um ponto p em uma região R, sugere-se o seguinte algoritmo: verifica-se, em primeiro lugar, se p está em ∂R. Se não estiver, projeta-se cada face de ∂R na esfera de raio um, centrada em p, e calcula-se o somatório de todas as áreas destas projeções. Se o valor absoluto do resultado for 4π, p está no interior da região e se for nulo, está no exterior. Este algoritmo pode produzir um resultado equivocado, se p estiver muito próximo de ∂R. O método apresentado no final da seção 4.3.1

para determinar se um ponto pertence a uma face, a partir do número de interseções de uma semi-reta, pode ser adaptado, mas exige um tratamento complicado quando a semi-reta interceptar um vértice.

5.4 Estimativa para o Tempo de Criação de uma SE

A princípio, considere-se que a SE é criada a partir de retalhos simples. Isto significa que deve-se verificar para cada novo retalho S, quais dos seus vértices já existem na SE. Supondo-se que S possui nv_s vértices e a SE possui nv_{se} vértices, então, na pior das hipóteses, se farão $(nv_s * nv_{se})$ comparações. A identificação das arestas de S que coincidem com arestas da SE é mais simples. Consideram-se apenas vértices consecutivos de S que foram identificados com vértices da SE e, para cada um deles, verificam-se quais são as arestas da SE que incidem nos vértices identificados. Aquela que incidir em ambos os vértices é a aresta procurada (num domínio planar). Esta busca é proporcional ao número de arestas incidentes nos vértices, que, tipicamente, não costuma ser um número muito grande. Feita a identificação das arestas, resta ainda determinar qual a região que contém S. Utilizando a informação de faces ao redor de arestas (ordenadas radialmente), esta determinação é proporcional ao número de faces ao redor de uma aresta, o que, também, não costuma ser um número elevado.

Note-se que, se os retalhos simples estão fortemente encadeados, pode-se executar este processo apenas para o primeiro retalho S_i. Os próximos retalhos devem ser obtidos por adjacência de aresta com S_i. Desta forma, já se tem antecipadamente uma aresta da SE coincidente com uma aresta de cada novo retalho. Isto permite que, para a identificação dos vértices, a busca seja feita apenas entre os vértices da região que contém cada novo retalho.

Se os retalhos não forem simples, cada um deles pode ser processado contra todas as faces da SE, resultando um algoritmo cuja complexidade pode ser expressa por $O(nf)$, sendo nf o número de faces da SE. Isto significa que uma subdivisão completa pode ser criada em $O(F^2)$, sendo F o número final de faces na SE. Claramente, esta não é a forma mais eficiente.

Para melhorar o tempo de inserção, observe-se a figura 5.11. Suponha-se que seja conhecido um ponto do retalho K (que se deseja inserir), pertencente a uma região qualquer R da SE. Neste caso, sabe-se que se K interceptar um conjunto de faces da SE, obrigatoriamente, alguma face na fronteira de R pertencerá a este conjunto. Além disso, se K tiver um ponto contido em uma outra região qualquer da SE, K terá que ser subdividido no seu processo de compatibilização com as faces que delimitam R (se K não foi subdividido significa que estava inteiramente contido em R). Isto sugere o seguinte algoritmo (recursivo) para compatibilizar um complexo K, com uma única face f, com uma SE:

(1) Encontre um ponto de f que esteja no interior de uma região R da SE.

(2) Compatibilize R e f. Se f interceptar alguma outra região da SE, obrigatoriamente, durante o processo de compatibilização de R e K, pelo menos uma nova face é criada em K. Crie, então, uma lista L e coloque nela f e as faces criadas em K.

(3) Para cada face de L, tome um ponto qualquer no seu interior e classifique-o em relação a R. Se o ponto estiver no interior de R é porque a face está inteiramente contida em R. Neste caso, retire-a de L, marque-a e coloque-a em uma lista M.

(4) Para cada face f de M, percorra cada aresta que a delimita. Se a aresta delimitar uma face de K não marcada, encontre, a partir da aresta, a nova região R e vá para (2) recursivamente.

Este algoritmo tem a virtude de só testar a interseção com as regiões que efetivamente podem ser cortadas pela face. O passo mais demorado é o passo 1, cuja complexidade é $O(F)$. Ele pode ser utilizado também para combinar duas SEs. Uma delas é vista, durante o processo, realmente como uma SE. A outra é vista como um conjunto de faces (fortemente) encadeadas. Isto significa que vão ser utilizadas apenas as adjacências entre as células do seu esqueleto de ordem 2. O retalho K, ao invés de conter inicialmente uma única face, contém agora um conjunto de faces encadeadas. Escolhe-se uma face qualquer como sendo a face inicial e dispara-se o processo. Este procedimento permite que o passo 1 do algoritmo só seja executado uma vez para cada conjunto de faces fortemente encadeadas.

Se a lista de regiões estiver organizada conforme explicado na seção 4.4, para a lista de faces, pode-se conseguir uma melhora expressiva, em alguns casos, para o passo 1.

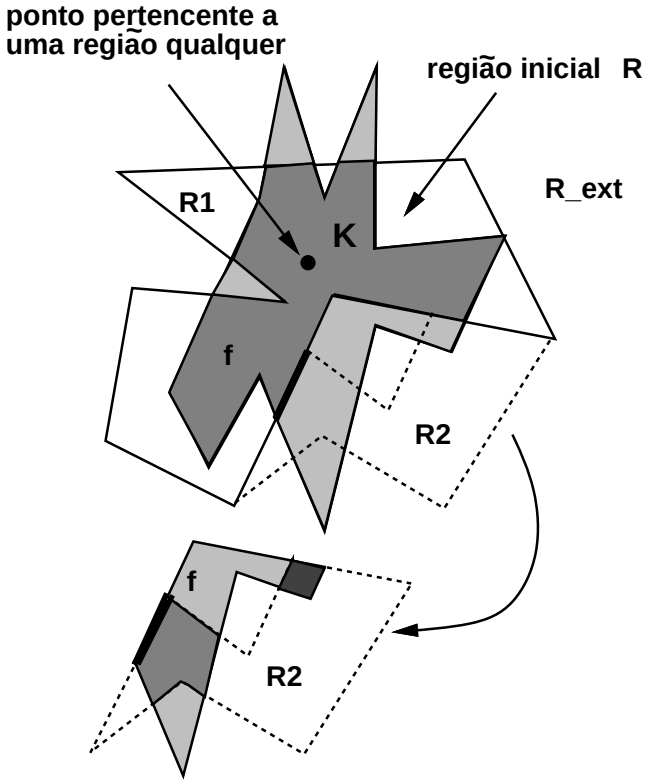


Figura 5.11 - Interseção eficiente.

5.5 Interseção

Quando se combinam duas SEs, a avaliação da interseção de duas superfícies é uma operação que é executada repetidamente e por isso a eficiênca desta avaliação é crítica.

Uma superfície pode ser dada por uma equação $z = f(x, y), f : \Re^2 \rightarrow \Re$ definida e derivável numa região D ou, implicitamente, por uma equação $F(x, y, z) = 0$. Se F é diferenciável numa vizinhança de um ponto $p_0 : (x_0, y_0, z_0)$, $F(x_0, y_0, z_0) = 0$ e $\partial F/\partial z \neq 0$ em p_0, então, pelo menos numa vizinhança de p_0, existe uma função derivável $z = f(x, y)$, definida implicitamente por F, tal que $F(x, y, f(x, y)) = 0$. Além disso:

$$\frac{\partial f}{\partial x} = -\frac{\frac{\partial F}{\partial x}}{\frac{\partial F}{\partial z}}, \qquad \frac{\partial f}{\partial y} = -\frac{\frac{\partial F}{\partial y}}{\frac{\partial F}{\partial z}}.$$

Uma curva no espaço pode ser especificada implicitamente como a interseção de duas superfícies:

$$F(x,y,z) = 0 \quad \text{e} \quad G(x,y,z) = 0.$$

Supondo-se que as duas superfícies contêm o ponto $p_1 : (x_1, y_1, z_1)$ e que ambas possuem planos tangentes em p_1, então, próximo a p_1, ambas as superfícies podem ser aproximadas pelos seus planos tangentes em p_1. Se estes planos não forem coincidentes, eles se interceptam segundo uma reta, que é a tangente à curva de interseção em p_1. A condição para que os planos tangentes não coincidam é que os seus vetores normais $\nabla F, \nabla G$ sejam linearmente independentes em p_1. Assim, ∇F X ∇G é um vetor tangente ao longo da curva de interseção. Se, por exemplo, $\partial(F,G)/\partial(x,y) \neq 0$, então, podem-se resolver ambas as equações para x e y em função de z, em torno de z_1: $x = f(z), y = g(z)$. As equações

$$x = f(t), \quad y = g(t), \quad z = t \quad \text{onde} \quad |t - \delta| < z_1$$

são as equações paramétricas para a curva de interseção nas proximidades de p_1.

A avaliação da interseção de duas superfícies não é um problema simples e continua sendo um tópico de pesquisa. A justificativa para este fato é que uma boa técnica de avaliação lida com três objetivos conflitantes: eficiência, robustez e precisão. Algoritmos baseados em aritmética exata são bastante robustos e precisos, mas costumam ser lentos. Já os algoritmos numéricos, embora eficientes, não são completamente robustos e vários deles falham em alguns casos.

Um algoritmo numérico para encontrar a curva determinada pela interseção de duas superfícies, executa, repetidamente, as seguintes operações conceituais: em um ponto p, da curva de interseção, é construída uma aproximação local (usando, por exemplo, a tangente em p). Percorrendo sobre a curva aproximada uma certa distância (passo), obtém-se uma estimativa do próximo ponto da curva, que é refinado por um método iterativo.

Em [HOFF89] é descrito um algoritmo numérico para avaliar a interseção de duas superfícies implícitas: $F(x,y,z) = 0 \quad \text{e} \quad G(x,y,z) = 0.$

A aproximação utilizada em um ponto p da curva é a expansão (truncada) de Taylor em p. O passo é determinado adaptativamente e a estimativa do próximo ponto é refinada pelo método de Newton-Raphson. Este algoritmo, embora robusto e preciso, falha nas vizinhanças dos pontos singulares da curva de interseção (existem técnicas que permitem contornar esta deficiência do algoritmo). A aproximação de Taylor é encontrada resolvendo um sistema linear de equações, cujos coeficientes dependem das derivadas parciais de ambas as superfícies.

Para o caso de duas superfícies paramétricas (superfícies descritas por três equações da forma $x = f(u,v), y = g(u,v), z = h(u,v)$) encontrar a sua interseção equivale a resolver três equações algébricas com quatro variáveis.

O ponto inconveniente em um algoritmo de interseção genérico, no caso de ser capaz de operar sobre qualquer classe de superfície, é que ele compromete a modularidade de um sistema de SE. Para acrescentar uma nova geometria deve-se levar em conta a interação com as já existentes. Para evitar este problema, pode-se aproximar cada face curva por um conjunto de faces planas e lidar apenas com esta geometria simples. Este processo não é tão imediato como parece, porque as aproximações podem não ser válidas topologicamente, ou seja, o número de interseções, quando se consideram as faces aproximadas, pode não ser igual ao obtido quando são consideradas as faces (curvas) originais. As aproximações devem ser refinadas até que a validade seja alcançada.

Em [TURN88] é descrito um método, baseado em aproximações poliédricas, para avaliação da interseção de superfícies, necessária, por exemplo, às operações *booleanas* entre BReps. Os resultados são comparáveis aos obtidos com geometrias curvas (todas as superfícies são algébricas). A base do método é uma operação de remoção de faces, aplicada a uma aproximação poliédrica, que cria uma estrutura topológica simplificada. A validade topológica do resultado é verificada por meio de um limite poliédrico interior e outro exterior. São estabelecidas também as condições para validade topológica das aproximações de duas superfícies, para determinação da curva de interseção. Os vértices obtidos de forma aproximada são projetados sobre as superfícies incidentes e estas projeções são usadas como valores iniciais para o método de Newton-Raphson. Desta

forma, obtêm-se vértices precisos (significando que cada um deles está na fronteira real das faces incidentes) e uma aproximação da curva de interseção pode ser obtida por interpolação [TURN85].

A imprecisão nos cálculos numéricos pode produzir incorreções topológicas. Em particular, na determinação do modo como duas arestas de dois objetos se interceptam (fig. 5.12a) [HOFF89]. A interseção da aresta e_1 com a face superior do cubo leva a conclusão de que ela está suficientemente próxima da aresta e_2, por exemplo, devido a perturbações posicionais causadas pelo ângulo com o qual e_1 intercepta a face superior. Porém, na interseção de e_1 com a face lateral, pode-se obter um ponto mais preciso, levando a conclusão de que e_1 não intercepta e_2. A forma de evitar este tipo de situação catastrófica é propagando topologicamente cada ponto de interseção, encontrado sobre uma aresta, às faces incidentes. Para isto, basta criar um vértice em cada aresta (fig. 5.12b). Automaticamente, este vértice existe na fronteira de cada face delimitada pela aresta. Este cuidado, e a adoção de um campo de atração adequado, para vértices e arestas, permite manter este tipo de situação sob controle.

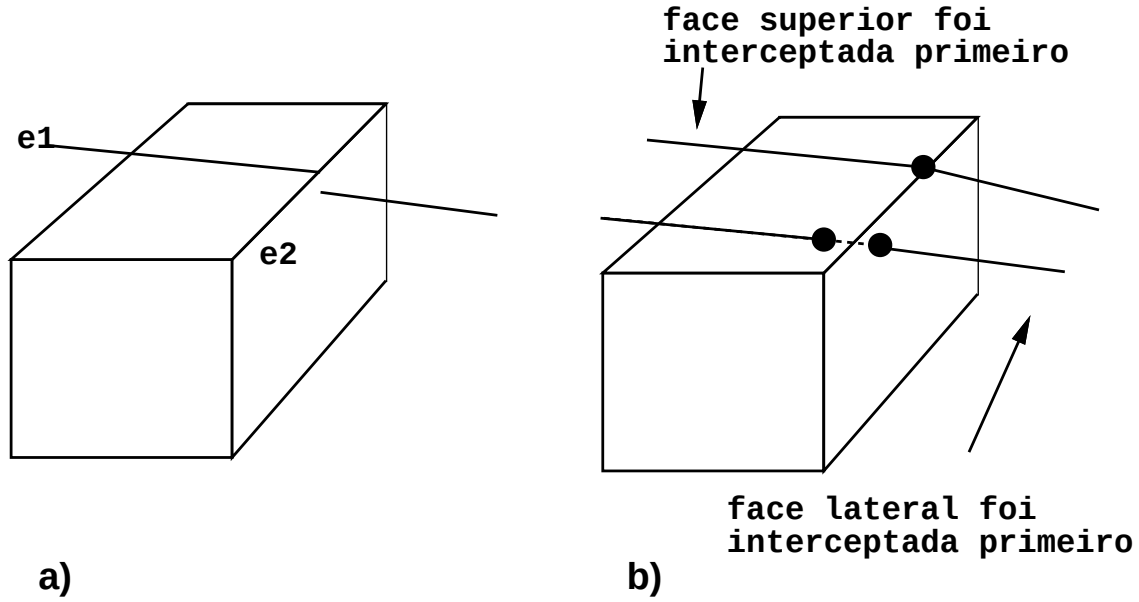


Figura 5.12 - Inconsistência topológica.

5.5.1 Faces Planas

Embora o objetivo seja lidar com faces quaisquer, um caso particular importante para a geometria das faces é aquele que corresponde a faces planas e arestas retilíneas. Os algoritmos geométricos envolvidos são simples. Além disso, qualquer superfície orientável pode ser triangulada consistentemente, o que permite aproximá-la por um conjunto de faces planas. Um algoritmo para calcular interseção de duas faces planas foi desenvolvido e é descrito detalhadamente a seguir.

Dadas duas faces A e B, cujas fronteiras são formadas por um ou mais ciclos, o objetivo é determinar que vértices devem ser criados (em A e B) e entre que pares de vértices devem ser criadas novas arestas, de forma a compatibilizar as faces.

Suponha-se, a princípio, que as duas faces não estão contidas em um mesmo plano. Neste caso, todos os pontos de interseção estão contidos na reta A.suporte $\cap$ B.suporte, denotada por L. O primeiro passo é encontrar um plano P, que contenha L, para projetar ambas as faces, e montar a matriz de transformação que muda o sistema de coordenadas para um sistema ortogonal de eixos com: eixo x coincidindo com L, eixo y contido em P e eixo z perpendicular a P. Neste novo sistema de coordenadas, todas as interseções ocorrem no eixo x (fig. 5.13).

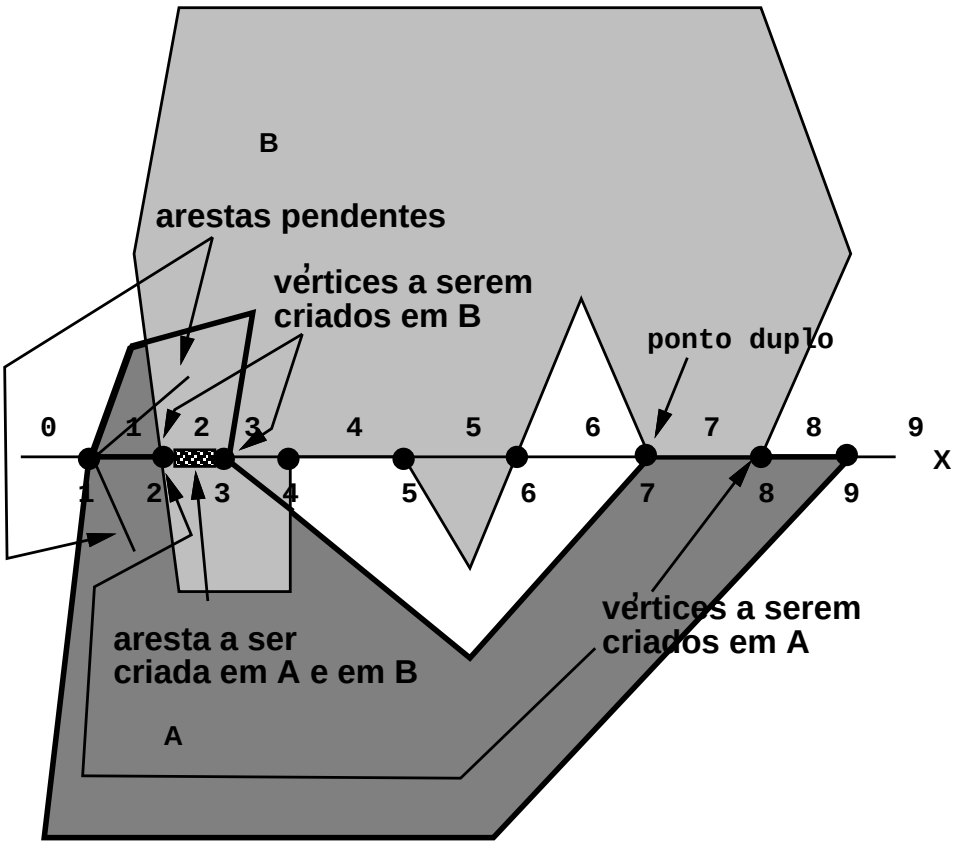


Figura 5.13 - Interseção de duas faces A e B.

O algoritmo percorre cada (uso de) aresta a, da fronteira de cada face, e verifica se ela intercepta L (arestas pendentes são percorridas duas vezes, uma em cada sentido). Para cada interseção, é armazenado em uma lista ordenada I_R, de acordo com a coordenada x de cada ponto, um único registro que a descreve. Se ambas as faces interceptam L, em um mesmo ponto, o registro gerado pela face B, precede o registro gerado pela face A, e possui uma indicação que se trata de um ponto duplo. Cada registro contém as seguintes informações:

- que face o gerou.
- coordenada x no ponto de interseção.
- número de interseções neste ponto (geradas pela mesma face).
- se ambas as faces interceptam L, neste ponto (interseção dupla).
- se o ponto coincide com um vértice de ∂a (guarda-se o vértice).
- se o ponto pertence a a (guarda-se a aresta).
- se a tangencia L, à direita (tangente à direita).
- se o ponto gera um novo vértice em cada face.

I_R contém todas as informações necessárias para verificar se o segmento determinado por cada par de pontos de I_R, está contido ou não em cada face. Cada vez que a interseção corresponder, por exemplo a A, significa que L cruzou ∂A. Se o número de interseções for ímpar significa que, se estava dentro de A, saiu, e vice-versa. Se o ponto foi gerado por uma tangente à direita, significa que o segmento está sobre L. Para cada aresta que está acima (mas que incide em um vértice que está sobre L) ou que cruza o eixo x, conta-se uma interseção. Para as arestas abaixo do eixo x, conta-se zero interseções. Note-se que o número de interseções de uma aresta pendente é naturalmente multiplicado por dois. A classificação de cada um destes segmentos é guardada em duas listas, S_A e S_B, uma para cada face. O preenchimento de cada uma destas listas pode ser feito da seguinte forma:

```
Classifica_Segmento ( face, S, I_R )
{
 int
     s_velho = OUT, s = OUT, i = 1;
 S[0] = OUT;
 Para ( cada registro de I_R ) faca
       {
        Se ( o ponto foi gerado pela face )
           {
            Se ( o numero de intersecoes e' impar )
                  s_velho = NOT s_velho; ( IN -> OUT ou OUT -> IN)
            Se ( o ponto corresponde a uma tangente a direita )
               s = ON;
            Senao
               s = s_velho;
           }
        S[i] = s;
        i = i + 1;
       }
}
```

Na figura 5.13, a face (F) que gerou cada ponto (P), o número de interseções de cada face em cada ponto e a classificação de cada segmento (S) em relação a cada face estão colocados na tabela a seguir.

(F)	(P)	A	B	(S)	A	B	
-	0:	-	-	0:	OUT	OUT	
A	1:	3	-	1:	IN	OUT	
B	2:	-	1*	2:	IN	IN	
A	3:	1	-	3:	OUT	IN	
B	4:	-	0,tgd	4:	OUT	ON	
B	5:	-	tg,0	5:	OUT	IN	
B	6:	-	1*	6:	OUT	OUT	* aresta cruzou o eixo
A, B	7:	0,tgd	1,tgd	7:	ON	ON	
B	8:	-	tg,1	8:	ON	OUT	
A	9:	tg,0	-	9:	OUT	OUT	

Para determinar que pontos de I_R correspondem a vértices que devem ser criados em cada face, existem dois casos a considerar: Se o i-ésimo ponto de I_R não é duplo e foi gerado por B, ele tem que estar contido em A ($S_A[i]$ = ON ou IN). Neste caso, ele gera um novo vértice em A, e gera um novo vértice em B, se ele foi gerado por uma aresta de B que cruzou o eixo x (ponto número 2, na figura 5.13). O mesmo vale trocando-se A por B; Se ele é duplo e foi gerado por uma aresta de B que cruzou o eixo x, ele gera um novo vértice em B. Neste caso, o próximo registro de I_R corresponde ao mesmo ponto (só que gerado por A). Se ele foi gerado por uma aresta de A que cruzou o eixo x, ele também gera um vértice em A.

Para determinar que pares de pontos consecutivos de I_R (exceto pontos duplos) correspondem a arestas que devem ser criadas em cada face, o segmento determinado pelo par deve estar dentro de uma face e não estar fora da outra (por exemplo: $S_A[i]$ = IN e $S_B[i]$ = ON ou IN). Na figura 5.13, o segmento 2 é o único que satisfaz esta condição (classificado como IN IN).

Para cada face, devem-se retornar duas listas. A primeira com as coordenadas de cada novo vértice a ser criado na face e, caso o vértice tenha que ser criado em uma aresta, qual a aresta. A segunda lista especifica pares de vértices que determinam arestas a serem criadas na face. Os vértices são especificados como vértices já existentes

ou como índices da lista anterior.

Quando as duas faces estão contidas em um mesmo plano, a interseção não mais ocorre sobre uma reta. Neste caso, interessa apenas criar vértices e arestas, no interior de cada face, nos pontos de interseção. O algoritmo apresentado pode ser utilizado com pequenas alterações.

5.6 Arquitetura de um Sistema de SE

A metodologia a ser adotada na implementação de um sistema de subdivisão espacial é análoga à adotada em um sistema de subdivisão planar. Existem camadas de *software* com funcionalidades bem definidas: uma camada responsável pelo tratamento da topologia utilizando WOps (esta camada desconhece completamente que tipo de geometria está sendo utilizada); uma camada responsável pelo gerenciamento e manipulação de entidades geométricas (**MGG**); e por fim uma camada responsável pela interação com o usuário.

5.6.1 Armazenamento de SEs em um Meio Permanente

Uma forma elegante de armazenar uma SE em disco é adaptando o algoritmo de inversão utilizado para armazenar uma SP. O objetivo é desfazer completamente a SE, eliminando primeiro todas as suas faces, em seguida eliminando todos os seus arames e por fim todas as cascas pontuais[9] . Em disco são gravados, então, os WOps inversos daqueles utilizados na destruição da SE. A execução destes WOps na ordem inversa recria a SE. Deve-se notar, porém, que, para isto, é necessário salvar também os atributos geométricos e os atributos do usuário associados a cada célula.

Como os WOps inversos aos que destroem a SE devem ser executados na ordem inversa, é aconselhável armazená-los já de acordo com esta ordenação, ao invés de ler o arquivo de trás para frente. Para tal, é utilizada uma pilha. Cada rotina que implementa um WOp deve empilhar um registro com a descrição do WOp inverso e seus respectivos

[9] Utilizando o WOp `kf`, pode-se reduzir qualquer SE a uma única região que contém apenas um conjunto de arames.

parâmetros, sempre que estiver sendo executado o algoritmo de inversão (semelhante ao descrito na seção 4.6.1).

5.6.2 Funções Necessárias a um Sistema de SE

Para ser possível lidar com curvas e superfícies com geometrias quaisquer, foi identificado, nas seções 5.3 e 5.3.1, um conjunto de funções que devem ser escritas para que uma nova geometria possa ser suportada (funções 1-8). Algumas delas podem não ser implementáveis facilmente, para certas geometrias, sendo necessário a utilização de métodos aproximativos. As funções 9-11 existem para permitir o armazenamento permanente e a recuperação de uma SE. A função 12 existe para que seja possível exibir graficamente uma SE.

1) Dado um par de células, de mesma dimensão, verificar se elas são geometricamente idênticas.

2) Dadas duas faces, determinar se elas se interceptam e, em caso afirmativo, retornar uma lista com os vértices e arestas a serem criados em cada face.

3) Dada uma célula, retornar as coordenadas de um ponto no seu interior.

4) Dada uma célula e as coordenadas de um ponto, verificar se o ponto está no seu interior.

5) Dada uma célula, calcular uma medida com sinal da sua extensão. Para uma aresta é o seu comprimento. Para uma face, a sua área e para uma região, o seu volume.

6) Dada uma célula C, de dimensão n(1 ou 2), para ser inserida em uma SE K e uma célula c de K, de dimensão $n-1$ em ∂C, retornar a próxima célula, em relação a C, no ciclo ordenado de células ao redor de c.

7) Dado um par de células C_1 e C_2, de mesma dimensão, que foram produzidas como resultado da divisão de uma célula C, distribuir os atributos geométricos de C entre C_1 e C_2.

8) Dado um par de células C_1 e C_2, com o mesmo suporte e uma célula $c \in (\partial C_1 \cap \partial C_2)$, combinar os atributos geométricos de C_1 e C_2 para formarem os atributos geométricos da célula resultante da eliminação de c.

9) Dada uma célula, alocar ou liberar a área de memória correspondente aos seus atributos.

10) Dado um arquivo e uma célula, gravar todos os seus atributos e uma indicação de quantos registros foram gravados.

11) Dado um arquivo e a indicação de quantos registros devem ser lidos, ler todos os atributos da célula e retornar um ponteiro para a área que os contém.

12) Dada uma célula, desenhá-la.

5.7 Aplicações

São várias as aplicações do que foi apresentado neste capítulo. Uma das mais importantes é a construção do núcleo de um sistema de modelagem geométrica que não se limite a modelar apenas sólidos homogêneos em dimensão. Como vai ser visto no capítulo 6, as operações *booleanas* podem ser implementadas sem a necessidade de um tratamento especial em alguns casos, como havia na modelagem sólida tradicional.

O SSE, por exemplo, foi criado para aplicações não interativas, onde um conjunto de retalhos de superfície é gerado por um meio qualquer (talvez por um programa) e fornecido através de um arquivo. A idéia é que os retalhos possam ser fornecidos em qualquer ordem e, possivelmente, com interseção. Duas saídas típicas estão na figura 5.14. Na primeira delas, três tetraedros, com algumas faces adicionais, podem ser vistos. Na segunda, três cubos (um dentro do outro) são cortados por um plano.

A criação de um gerador interativo de malhas para modelos tri-dimensionais de elementos finitos pode ser facilitada adotando-se a metodologia deste capítulo. Martha [MART89], por exemplo, criou um sistema integrado tri-dimensional de elementos finitos, baseado na RED, para simulação de fraturamento de sólidos. Neste sistema, o processo de geração de malhas é feito através de uma decomposição hierárquica do

domínio, onde retalhos de superfície são adicionados interativamente para subdividir regiões existentes. Este trabalho foi estendido para modelar estruturas de concreto armado, onde, além da geração de malhas, a criação das barras de aço mergulhadas no concreto é feita com a adição das interfaces entre os dois materiais [POTY90].

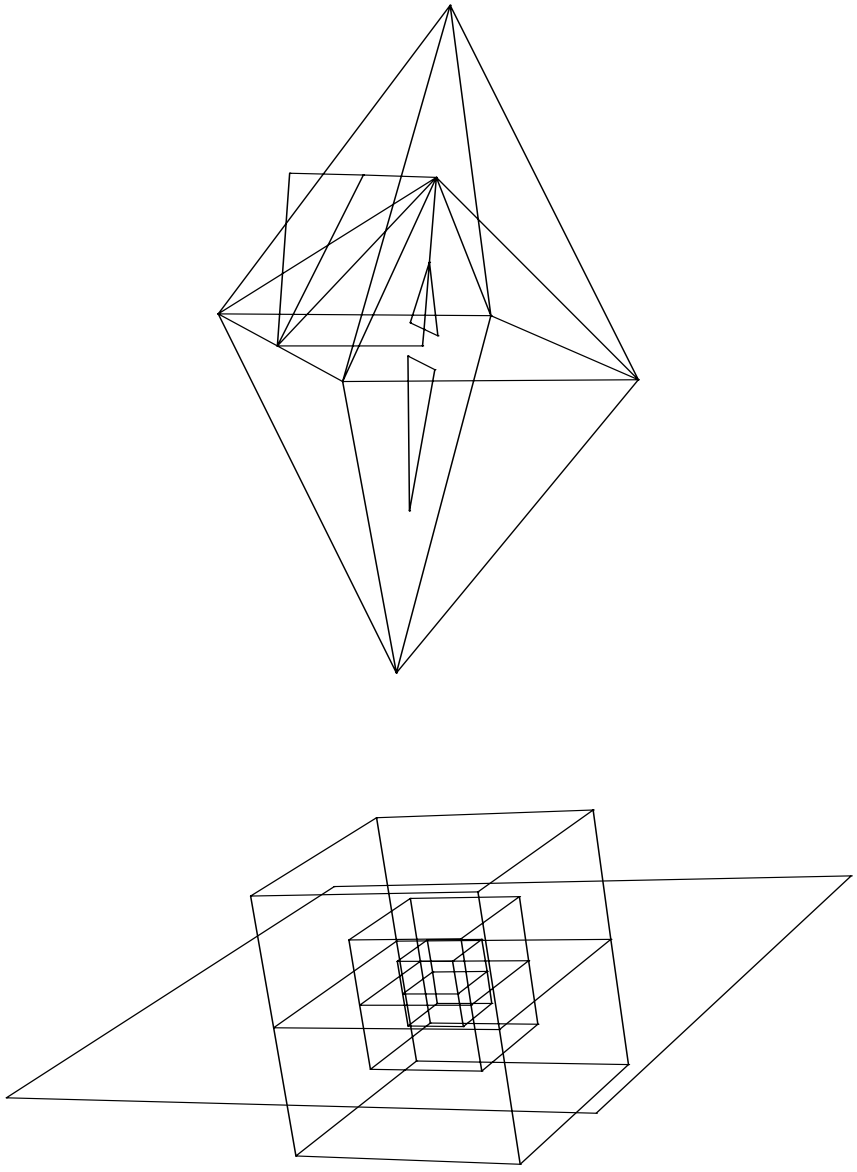


Figura 5.14 - Duas subdivisões simples geradas pelo SSE.

6 - PROCESSO DE MODELAGEM

Nos capítulos anteriores, foi proposto um esquema de representação, baseado no esquema SGC, que utiliza uma estrutura de dados topológica para possibilitar o aproveitamente da ordenação que pode ser imposta a alguns dos relacionamentos de adjacência presentes em subdivisões do espaço. Este esquema visa, em primeiro lugar, manter uma representação explícita do bordo dos objetos e, em segundo lugar, oferecer a possibilidade de implementação de algoritmos geométricos eficientes. O elemento que está faltando para concluir a metodologia é um processo de construção de subdivisões do espaço que seja adequado às aplicações interativas.

Este capítulo sugere um processo de modelagem baseado em operadores que agem sobre subdivisões do espaço. Antes de apresentá-los, vai ser feita uma breve discussão sobre o que se entende por um processo de modelagem, de forma a que o leitor possa avaliar a importância deste tópico. O processo de modelagem talvez seja o ponto mais importante para o uso, na prática, de qualquer sistema interativo de modelagem.

Um **processo de modelagem** é uma forma de, a partir de um modelo inicial M_0, e através de refinamentos sucessivos, produzir modelos intermediários $M_1, M_2, ..., M_{n-1}$ até que se atinja o modelo (satisfatório) final M_n. O ideal é que M_0 esteja o mais próximo possível de M_n e que as operações de refinamento sejam as mais poderosas possíveis, de forma a minimizar o número de modelos intermediários. A pergunta a ser respondida é: qual a forma mais apropriada de obter, interativamente, cada modelo, a partir do seu antecessor?

No capítulo 4, foi apresentado um processo de modelagem bastante razoável para uma aplicação bi-dimensional. Lá, uma SP era construída a partir de segmentos de curva gerados, por exemplo, por uma mesa digitalizadora. No entanto, em modelagem tri-dimensional, como na construção de uma SE, não existe um processo de modelagem tão simples e ao mesmo tempo satisfatório. O processo de modelagem apresentado no capítulo 5 não é próprio para aplicações interativas, porque ninguém deseja (ou tem paciência para) construir um objeto especificando retalhos de superfície.

O processo de modelagem CSG [REQU77] tem sido muito utilizado em modeladores de sólidos, porque apresenta mecanismos de refinamento familiares à maioria das pesssoas. Através de operações *booleanas* regularizadas de união, interseção e diferença, determinados primitivos básicos são combinados produzindo objetos cada vez mais complexos. Porém, para construção de SGCs, este processo não funciona porque as operações *booleanas* são definidas para sólidos regulares (homogêneos em dimensão) que dividem o espaço em apenas três regiões: interior, fronteira e exterior do sólido.

Para suprir esta deficiência, Rossignac & Requicha [ROSS91] criaram uma nova classe de objetos, chamados CNRG, e um conjunto de operadores que agem sobre estes objetos. Embora esta abordagem possa ser vista como um novo esquema de representação para subdivisões do espaço, neste trabalho, interessa apenas utilizar os operadores CNRG como um processo de modelagem. Estes operadores permitem definir operações topológicas e *booleanas* para SGCs e, conseqüentemente, podem ser empregados para construir subdivisões do espaço.

6.1 Geometria Construtiva Não Regularizada (CNRG)

Um objeto CNRG é um conjunto de componentes **mutuamente disjuntas** (talvez desconexas). Uma componente é um conjunto de pontos do $\Re^n$ (possivelmente não regular). O conjunto de pontos pA, de um objeto CNRG A, é a união dos conjuntos de pontos das suas componentes.

Uma árvore CNRG é um grafo dirigido acíclico com uma raiz e representa um objeto. As folhas da árvore são os primitivos CNRG que podem ser compostos por mais de uma componente. Nós internos representam objetos CNRG intermediários obtidos pela aplicação de operações *booleanas*, topológicas, de simplificação ou de filtragem nos objetos representados pelos seus filhos. Considera-se que uma componente está contida em um objeto quando ela está contida no seu conjunto de pontos.

Os operadores definidos sobre um objeto CNRG são: agregação, unificação, soma, produto, subtração, complemento, interior, fecho, fronteira e regularização. Para distingüí-los dos operadores que agem sobre conjuntos de pontos, são utilizados símbolos

diferentes. Por exemplo, $\cap, \cup$ e $\setminus$ são usados para interseção, união e diferença de conjuntos de pontos e $*, +$ e $-$ são usados para os equivalentes CNRG. A seguir, apresentam-se os operadores CNRG (fig. 6.1 e 6.2) criados por Rossignac [ROSS91]:

- **Agregação**

Dados n conjuntos de pontos A_i, a operação de agregação, denotada por $|$, cria o objeto CNRG composto pelos n conjuntos de pontos: $A = \{A_1|A_2|...|A_n\}$.

- **Unificação**

A unificação uA cria um objeto com uma única componente: o conjunto de pontos de A ($uA = \{pA\}$).

- **Complemento**

O complemento cA é um objeto composto de uma única componente formada pelo conjunto de pontos complementar a pA ($cA = \{\overline{pA}\}$).

- **Soma**

A soma $A+B$ é um agregado de componentes formado por: $A_i \cap B_j$, $A_i \setminus Bj$ e $B_j \setminus A_i$ para toda combinação de componentes A_i de A e B_j de B. A soma produz uma subdivisão de $(pA) \cup (pB)$ que é compatível com a decomposição de A e B em componentes. Claramente, $p(A + B) = (pA) \cup (pB)$.

- **Produto**

O produto $A*B$ é o agregado de todas as componentes do tipo $A_i \cap B_j$. Cada componente A_i de A é truncada para $A_i \cap pB$ e é subdividida (compatibilizada) de acordo com a subdivisão de B em componentes. Conseqüentemente, $p(A * B) = (pA) \cap (pB)$.

- **Subtração**

A subtração $A - B$ é o agregado de todas as componentes do tipo $A_i \setminus pB$. Claramente, $p(A - B) = (pA) \setminus (pB)$.

- **Interior**

O interior topológico iA de A no $\Re^n$ é o agregado de componentes $A_i \cap I(pA)$ que são a interseção das componentes de A com o interior de A.

- **Fecho**

O fecho kA de A é o agregado composto de todas as componentes de A mais uma única

componente definida como a diferença entre o fecho de pA e pA ($kA = A|F(pA)\backslash pA$). Note-se que $p(kA) = p(iA) \cup p(bA)$.

- **Fronteira**

A fronteira bA de A é definida como: $bA = kA - iA$. Note-se que $p(bA) = \partial(pA)$.

- **Regularização**

A regularização rA de A é definida como $k(iA)$. Note-se que $p(rA) = R(pA)$.

Estes operadores propiciam um processo de modelagem bastante eficiente na interação com o usuário, particularmente em dimensão 3. Quando seguidos da operação de unificação eles se comportam exatamente como os respectivos operadores que agem sobre conjuntos de pontos. Entretanto, sem a unificação, eles retornam um agregado de componentes mutuamente disjuntas.

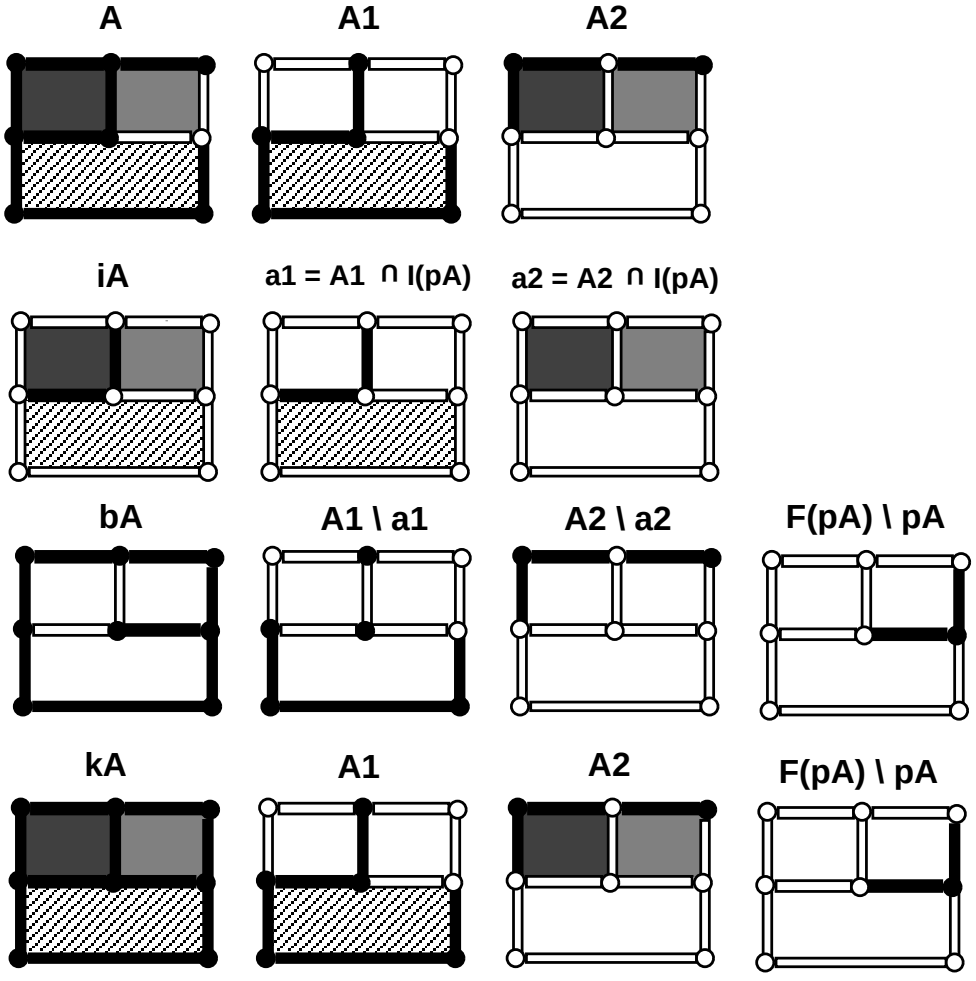


Figura 6.1 - Operações CNRG unárias (adaptado de [ROSS91]).

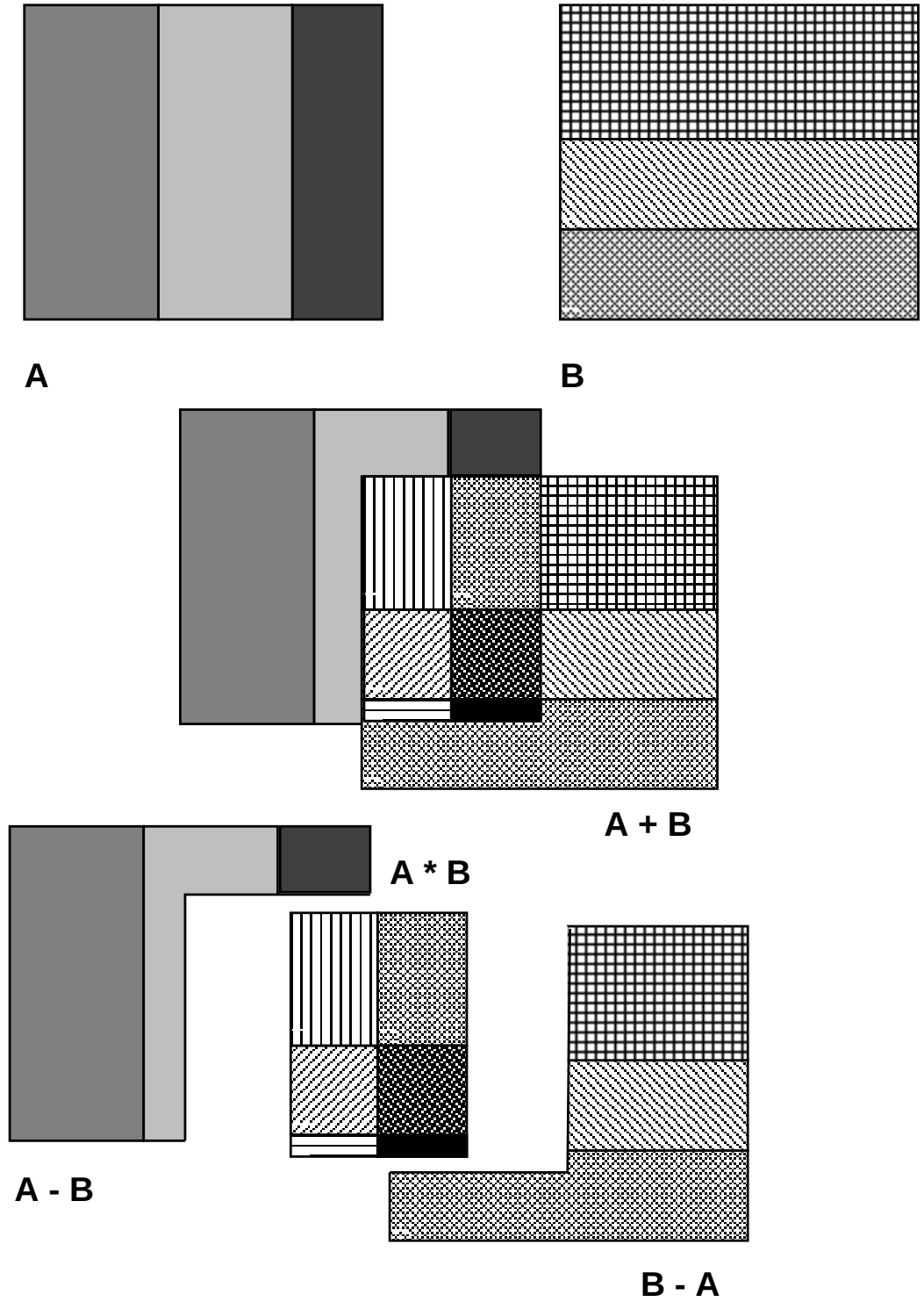


Figura 6.2 - Operações CNRG binárias.

6.2 Combinação CNRG de SGCs

Embora o conceito de objeto CNRG seja mais geral do que o conceito de SGC, para propósito de processo de modelagem, um SGC pode ser imaginado como possuindo três componentes: um complexo geométrico, uma seleção de células e uma **partição** do conjunto de células ativas (conjuntos de células de várias dimensões considerados como uma unidade). Para aplicar operadores CNRG em SGCs é suficiente considerá-los como sendo objetos CNRG cujas componentes são as suas células ativas. Esta é a decomposição de um SGC em componentes considerada daqui em diante.

Uma das principais propostas deste trabalho é que o processo de modelagem CNRG seja implementado, pelo menos para dimensões 2 e 3, tendo como esquema de representação objetos SGC completos. Não existe problema algum em representar objetos, ou subdivisões finitas (incompletas) do espaço, por SGCs completos pois, nestes casos, a região externa ilimitada é uma célula inativa. Da mesma forma, uma cavidade em um sólido também pode ser uma célula inativa.

Assim, os operadores CNRG binários e unários são implementados, em duas e três dimensões, a partir da **combinação** de SGCs completos, conforme descrito a seguir, utilizando a metodologia apresentada nos dois capítulos anteriores. No decorrer deste capítulo, o termo SGC será empregado para se referir a objetos SGC completos.

Um operador CNRG permite criar SGCs pela combinação de SGCs pré existentes. A combinação de dois SGCs A e B significa gerar um novo SGC C, compatível com A e com B. Os operadores binários (soma, produto e diferença) necessitam que cada célula tenha um atributo `origem` que assume um entre quatro valores: I, A, B ou AB (para célula inativa, oriunda só de A, oriunda só de B ou oriunda de A e B, respectivamente). O selecionamento das células apropriadas de C produz o resultado da operação desejada.

Para criar um SGC C, compatível com dois SGCs A e B, é necessário fazer com que A encaixe em B, refinando cada célula de A.esqueleto(2) cuja interseção com alguma célula de B.esqueleto(2) não é vazia (e vice-versa). Em seguida, deve-se adicionar a C cada célula de A e de B. Os seguintes passos devem ser executados (fig. 6.3):

- refine A e B, para que eles encaixem um no outro.
- crie um complexo C, idêntico a A (já refinado), e coloque no atributo `origem`, de todas as células ativas de C, o valor A. Nas inativas coloque o valor I.
- adicione a C cada faceta (face mais fronteira) de B, conforme descrito no capítulo 5. Durante a inclusão de uma faceta F, para cada célula ativa de F, que já existir em C e estiver ativa, coloque na sua `origem` o valor AB. Para as células ativas que não existirem ou estiverem inativas em C, coloque o valor B. As regiões criadas neste processo estão inativas.
- percorra cada região de C verificando se ela está contida em alguma região ativa de A e/ou de B e coloque na sua `origem` o valor apropriado. Esta verificação pode ser feita por topologia.

O processo descrito acima classifica cada célula de C, de acordo com a sua `origem`. A partir daí, o selecionamento das células apropriadas, de acordo com esta classificação, produz o resultado esperado.

- Soma: devem ser ativadas todas as células com `origem` diferente de I.
- Produto: devem ser ativadas apenas as células com `origem` AB.
- Subtração ($A - B$): devem ser ativadas as células com `origem` A.

Os operadores unários agem sobre um único objeto CNRG. Considerando-se que as suas componentes são as células ativas de um SGC completo A e que o seu complemento é formado pelo conjunto de células inativas de A, então, cada operador pode ser descrito pelo selecionamento de determinadas células:

- Interior: para cada célula inativa c, desative as células de bdry(c, A.complexo).
- Fecho: para cada célula ativa c, ative as células de bdry(c, A.complexo).
- Fronteira: são as células que pertencem ao fecho e não pertencem ao interior de A.
- Regularização: são as células que pertencem ao fecho do interior de A.
- Unificação: simplifique A.

Assim, mostrou-se que qualquer sistema que implemente SGCs completos pode fazer operações CNRG, com uma ressalva: quando se representa um objeto CNRG por um SGC, a operação de unificação não produz um objeto com uma única componente. Neste caso, **unificar significa simplificar o SGC** (fig 6.4) (o que não produz uma única célula).

A simplificação de um SGC pode ser executada pelas operações de eliminação, colagem e incorporação (seção 3.1.1). Eliminar uma célula inativa é removê-la, caso ela não esteja na fronteira de uma célula ativa.

Para colar duas células ativas c_1 e c_2 de dimensão n, eliminando no sentido explícito uma célula ativa c de dimensão $n-1$, deve-se verificar se $c \in (\partial c_1 \cap \partial c_2)$ e se c_1.suporte $= c_2$.suporte.

No caso de uma SE, representada pela estrutura RED, o problema de incorporar uma célula ativa pode ser dividido em três casos:

- A célula é um vértice. Neste caso, deve-se observar se ele é uma casca pontual (na fronteira de uma região ativa), ou um ciclo pontual que só é usado por uma única face (que deve estar ativa).
- A célula é uma aresta. Neste caso, ou ela é um arame (na fronteira de uma região ativa), ou ela é usada duas vezes na mesma face (que deve estar ativa) que é a única face no ciclo radial de faces em torno da aresta.
- A célula é uma face. Neste caso, ela deve ser usada duas vezes pela mesma região (que deve estar ativa).

Em certos tipos de aplicação pode não haver sentido em simplificar o SGC, pois as células redundantes podem ter algum significado. Simplificar um SGC, ou não, é uma decisão que deve ser tomada de acordo com cada aplicação.

As operações CNRG descritas neste capítulo fornecem um elenco poderoso de ferramentas de modelagem para criação e modificação de SGCs. Para implementar tais operações é suficiente que o sistema seja capaz de compatibilizar SGCs. Uma vez compatibilizados os SGCs, todos os operadores se resumem em ativar e desativar células.

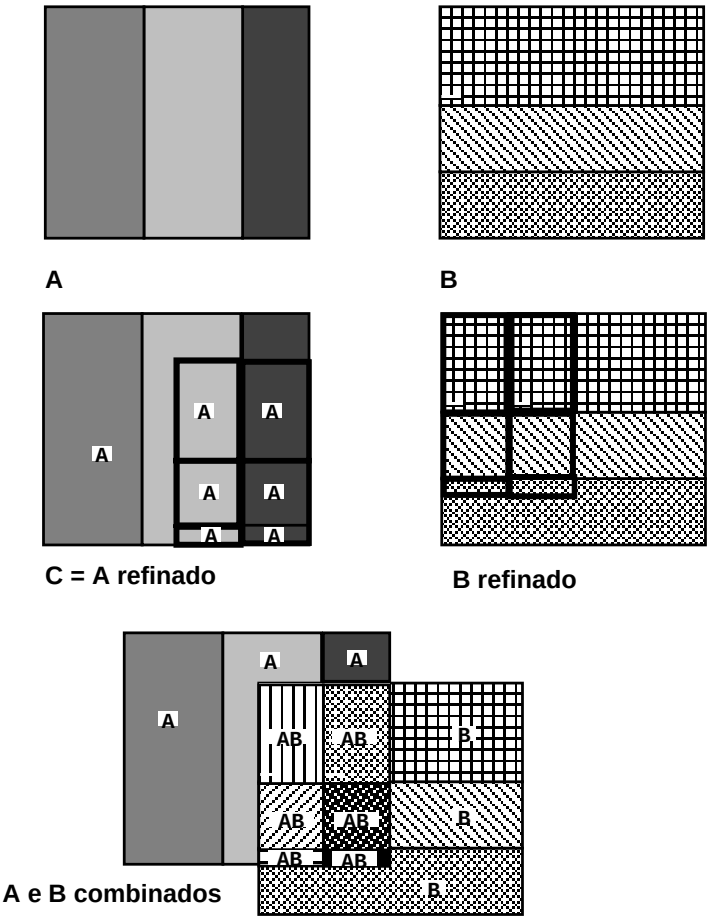


Figura 6.3 - Combinação de SGCs.

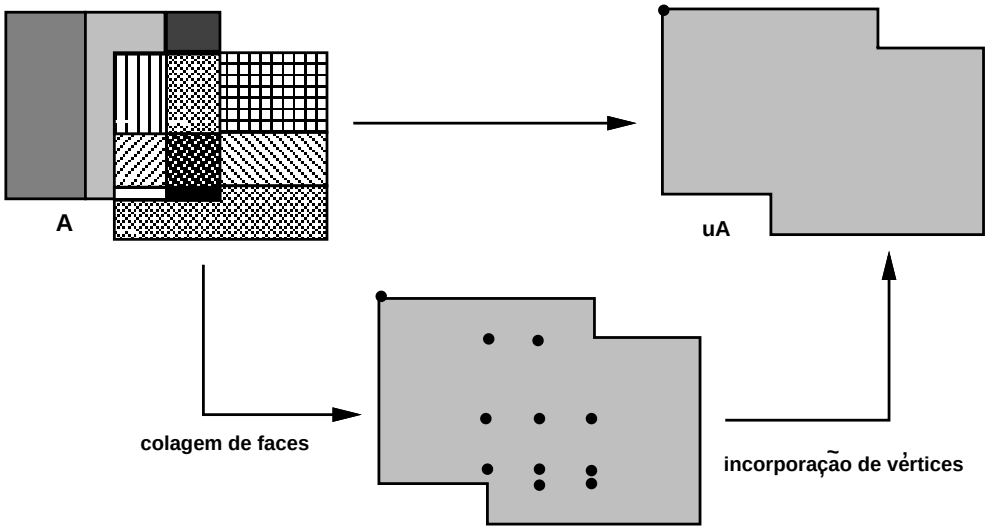


Figura 6.4 - Unificação de um objeto CNRG.

6.3 Modelagem de Objetos Compostos de Diversos Materiais

Os operadores CNRG descritos na seção anterior são especialmente úteis para a modelagem de objetos compostos por diversos materiais. Considere-se, por exemplo, o objeto da figura 6.5, constituído por uma barra de aço parcialmente cravada em uma barra de concreto. Tal objeto é naturalmente modelado por um SGC com três regiões: a correspondente à barra de ferro, a correspondente à barra de concreto e a região ilimitada. Além disso, tal objeto é facilmente descrito por operadores CNRG. Se A representa a barra de concreto (sem a inserção da barra de aço) e B representa a barra de aço, o objeto da figura 6.5 é dado por $(A - B) + B$.

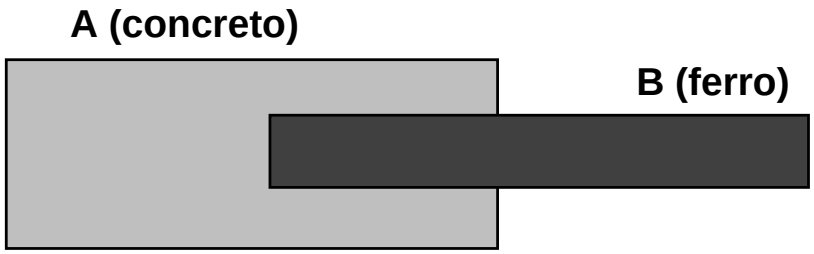


Figura 6.5 - Objeto composto por dois materiais.

Um outro exemplo é o da figura 6.6 que apresenta uma estrutura composta por quatro colunas de concreto, quatro vigas de ferro e uma lage, também de concreto, construída a partir de paralelepípedos, devidamente posicionados no espaço, gerados pelo modelador de sólidos GeneSys [FISC91].

A criação da estrutura pode ser feita pela operação (CSG) de união, mas o resultado é um objeto com uma única região (fig. 6.6). Utilizando a operação (CNRG) de soma, obtém-se um total de 21 regiões distintas (fig. 6.7). Cada uma destas regiões deve ser constituída, claramente, por um único material. A dúvida na composição surge nas regiões que correspondem a interseção de regiões constituídas por materiais diferentes.

Um possível esquema para que a composição de uma região possa ser determinada de forma automática, consiste em atribuir à região um atributo, chamado **fator de dominância** fd, que estabeleça qual material predomina na operação de interseção. Assim, por exemplo, se o ferro tiver $fd = 5$ e o concreto $fd = 3$, os três blocos destacados

na figura 6.7 serão constituídos por ferro. Note-se que, mesmo que o resultado desejado seja uma estrutura composta por uma única região, ele pode ser obtido pela operação de unificação. Esta unificação é feita eliminando-se as faces que não são usadas pela região externa e as arestas (vértices) que delimitam faces (arestas) com o mesmo suporte.

Uma vez gerada a representação de um objeto CNRG, pode-se percorrer cada uma das suas regiões (exceto a externa) e, regularizando-as (eliminando faces pendentes, arames, cascas pontuais), torná-las variedades de dimensão 3 fechadas (com fronteira). Assim, cada casca (sendo uma variedade de dimensão 2) é assimilável por qualquer modelador de sólidos *manifold* que possua uma representação por fronteira. Isto permite decompor o resultado de uma operação CNRG em uma união disjunta (agregados) de objetos *manifold*, proporcionando um meio de incorporar tais operações a modeladores de sólidos já existentes, mesmo que eles possam representar apenas sólidos *manifold*. Tal extensão foi incorporada ao GeneSys.

Figura 6.6 - União CSG.

Figura 6.7 - Soma CNRG.

7 - CONCLUSÕES

Este capítulo apresenta um resumo conclusivo do trabalho, enfatizando as principais contribuições para a área de Modelagem Geométrica. Em linhas gerais, este trabalho propõe uma metodologia para construção e manutenção de subdivisões do espaço Euclidiano bi e tri-dimensional, que sejam consistentes topológica e geometricamente. Mas, mais do que isto, esta metodologia permite a **integração** de diferentes propostas para a subdivisão do espaço.

Modelagem Geométrica não é uma área trivial. Em parte, porque os trabalhos na área costumam apresentar apenas uma visão pontual e especializada, sobre algum tópico; por exemplo: métodos para *rendering*, um tipo de estrutura de dados, como interceptar curvas de Bezier, etc. Em parte, porque as comunidades interessadas na área usam linguagens diferentes. Por exemplo, alguns conceitos, comuns entre matemáticos, não são familiares a pessoas com formação em engenharia ou informática. Como em geral, quem implementa os sistemas não são os matemáticos, isto pode levar uma pessoa, menos experiente, a concluir que determinada aplicação é extremamente complicada, quando não é, ou então a conclusão oposta, que as coisas parecem mais simples do que são na realidade.

A metodologia proposta contém um modelo matemático, um esquema de representação e um processo de modelagem, que podem ser utilizados em diversas aplicações. Em particular, naquelas que não podem utilizar modelos *manifold*; por exemplo, modelos de subdivisão utilizados em métodos numéricos (malhas para modelos de elementos finitos). Em alguns casos, até mesmo modelos *non-manifold* não fornecem exatamente o que é necessário ao tipo de objeto geométrico de interesse, como: objetos compostos por diversos materiais; objetos com estruturas internas; objetos não homogêneos em dimensão; e objetos em contato.

Para este propósito, o que se deseja é modelar agregados de objetos (sólidos possivelmente combinados com partes de dimensão inferior), mantendo os relacionamentos de adjacência entre estes objetos. Isto permite a criação de um ambiente de desenvolvimento no qual os diversos aplicativos podem compartilhar, não somente dados, como,

também, algoritmos que processam e alteram estes dados. Um exemplo que caracteriza bem este intercâmbio são sistemas CAD/CAM, que podem utilizar o mesmo esquema de representação para modelar o objeto a ser analisado e, possivelmente, produzir uma malha para modelos de elementos finitos.

Tendo isto em mente, procuraram-se, dentro da literatura, as ferramentas disponíveis mais apropriadas que pudessem ser integradas, permitindo, assim, a construção de um sistema cujo objetivo é: representar, topológica e geometricamente, agregados de objetos no espaço Euclidiano bi ou tri-dimensional. Em última análise, isto é o que é necessário à maioria das aplicações gráficas. Seguindo esta filosofia, e a tendência atual que vai na direção de sistemas híbridos [FISC91], adotou-se uma representação por fronteira, utilizando um processo de modelagem construtivo.

Embora as estruturas de dados *non-manifold* sejam capazes de representar agregados de objetos, só recentemente, um tratamento conceitual mais cuidadoso foi desenvolvido. O SGC de Rossignac & O'Connor [ROSS90] fornece um modelo matemático adequado para a representação de subdivisões do espaço. Uma **contribuição** deste trabalho foi mostrar quais os tipos de relacionamentos de adjacência que podem ser representados ordenadamente em um SGC (e as vantagens de se tirar proveito desta ordenação) e que um **SGC é um complexo celular**, cujas células foram grupadas de forma a evitar a fragmentação desnecessária do modelo. Este resultado é importante porque mostra que o domínio de modelagem permaneceu o mesmo, tendo sido criada apenas uma decomposição mais econômica.

Com a criação, neste trabalho, do conceito de **complexo completo** foi possível obter uma caracterização precisa do tipo de objeto que pode ser representado por certas estruturas de dados topológicas que têm sido utilizadas, com sucesso, em vários sistemas de modelagem atuais.

A estrutura de dados *radial edge* [WEIL86] adotada, embora seja genérica e versátil, até hoje foi pouco utilizada na prática. Embora ela seja comumente citada na literatura, a impressão que se tem é de que o seu potencial ainda não foi suficientemente explorado. Talvez isto se deva à dificuldade de implementar os operadores *non-manifold* para

manipulá-la, enquanto os operadores que manipulam estruturas *manifold* são simples, concisos e relativamente fáceis de implementar. Mas, certamente, o principal empecilho é a falta de operadores geométricos de mais alto nível, que permitam a geração automática da representação. Uma outra **contribuição** deste trabalho foi criar uma base para a implementação destes **operadores geométricos**.

Como processo de modelagem, sugere-se a utilização dos operadores CNRG, que definem operações de conjunto, análogas às CSG, para agregados de objetos [ROSS91]. Em relação a esta parte, a principal **contribuição** foi mostrar como **adicionar tais operadores** a sistemas que implementam SGCs, de forma a obter-se um esquema de representação que, mantendo uma representação explícita do bordo, pode ser construído a partir de operações sobre o conjunto de pontos que define o objeto.

Espera-se que, com a leitura deste trabalho, obtenha-se uma compreensão razoável do esforço e dificuldades envolvidas no problema de subdividir o espaço. O estado da arte atual está em uma fase muito parecida com aquela fase quando surgiram os primeiros sistemas gráficos independentes de dispositivo. Há uma série de aplicações, com as mesmas necessidades, utilizando métodos e representações diversas, quando já é possível adotar uma abordagem unificada.

Uma das vantagens de utilizar a metodologia proposta vem a ser que, ao final do processo, há uma estrutura de dados topológica que possibilita encontrar todos os relacionamentos de adjacência de forma eficiente. Com uma biblioteca de consulta apropriada, rapidamente se extrai qualquer um destes relacionamentos (isto é, determina-se a vizinhança de qualquer célula). Por exemplo, dado um vértice, pode-se determinar que arestas são incidentes a ele, ou dada uma face, que faces compartilham cada uma de suas arestas. Com estruturas de dados convencionais, alguns destes relacionamentos não podem ser encontrados de forma eficiente.

Por vários anos, subdivisões espaciais foram usadas como uma forma de representação auxiliar, em aplicações especializadas, como para modelos de elementos finitos. Atualmente, elas estão sendo utilizadas em aplicações de automação de manufatura para representar, não apenas um, mas um conjunto (agregado) de sólidos. A capaci-

dade de criar subdivisões espaciais permite modelar efetivamente objetos compostos de diversos materiais. O contato entre estes objetos é representado explicitamente, o que é importante para uma série de aplicações, como: planejamento de montagens, robótica, geologia e no desenvolvimento de sistemas tri-dimensionais para elementos finitos.

A capacidade de criar subdivisões planares, de forma interativa ou não, também possui várias aplicações práticas. Sistemas geográficos e de elementos finitos desempenham estas tarefas, porém, tradicionalmente, usando estruturas de dados mais primitivas que as usadas neste trabalho.

Num sistema geográfico típico, para que se obtenha uma estrutura de dados — onde alguns dos relacionamentos de adjacência estão presentes de forma explícita — são necessárias três etapas: digitalizar um mapa; executar um procedimento que marque determinados pontos especiais como sendo pontos de entroncamento de segmentos (em geral, havendo um número máximo de segmentos por entroncamento); e, a partir dos entroncamentos e dos segmentos, gerar as faces (regiões). Pior ainda, as faces do mapa devem ser digitalizadas sempre em um mesmo sentido e as arestas que não pertencem à face externa são digitalizadas duas vezes, uma em cada sentido. Em alguns sistemas, até os pontos de entroncamento devem ser indicados em tempo de digitalização. Com a metodologia, as três etapas são executadas simultaneamente e em tempo real. Não há qualquer ordem imposta para a digitalização. Quando uma face é fechada, pode-se preencher a sua área com uma determinada cor, o que serve como *feedback* para o usuário.

Uma outra aplicação onde a utilização de estruturas de dados topológicas para subdivisões planares tem-se mostrado bastante eficiente é no desenvolvimento de sistemas bi-dimensionais de elementos finitos. Wawrzynek [WAWR87] utilizou, com excelentes resultados, a estrutura *winged-edge* [BAUM72] no desenvolvimento de um sistema integrado (envolvendo pré-processamento, análise numérica e pós-processamento) de elementos finitos para simulação de propagação de trincas em duas dimensões. A utilização de estruturas de dados topológicas para subdivisões planares também possibilita um ambiente adequado para a geração interativa de malhas para modelos de elementos

finitos bi-dimensionais, de boa qualidade [CARV90, CAMP91].

Esta experiência bi-dimensional foi estendida, com sucesso, para três dimensões e um sistema integrado de elementos finitos, baseado na estrutura *radial edge*, para simulação de fraturamento de sólidos, foi desenvolvido [MART89].

A utilização da metodologia proposta propicia uma nova forma de trabalho que, espera-se, venha a ser incorporada nos sistemas de CAD futuros, automatizando tarefas que, nos sistemas atuais, são executadas manualmente ou ineficientemente.

7.1 Comentários Sobre a Implementação

O desenvolvimento deste trabalho foi acompanhado de uma implementação que permitiu avaliar e comparar os algoritmos apresentados. O SSE (sistema de subdivisão espacial) permite criar uma SE a partir de retalhos de superfície. Ele utiliza uma variante dos WOps implementados, anteriormente, por Martha [MART89]. Dentre as modificações, incluem-se a **alteração do operador** *make face*, especificado por Weiler, para retornar as faces não classificadas, quando do fechamento de uma região, e a extensão para criação de faces com fronteiras desconexas. Foi criado, também, um **mecanismo de inversão** para estruturas de dados *non-manifold*, de forma a armazená-las em um meio permanente.

Embora o SSE seja capaz de criar, também, uma SP, ele utiliza uma estrutura de dados muito mais complexa do que a necessária. Para aplicações bi-dimensionais, foi implementado o EDP (editor de diagramas planares) que constrói uma SP a partir de segmentos de curva. Em ambos os sistemas, foram criados **algoritmos eficientes** para inserção de um novo retalho de superfície ou um novo segmento de curva, evitando que o cálculo de interseção seja feito levando em conta todas as faces ou arestas da subdivisão.

Todo o *software* implementado foi organizado de forma a haver uma separação real entre topologia e geometria, separação esta conseguida através de estruturas de dados topológicas. A organização do *software* permite a inclusão de novas geometrias para faces e arestas. A partir da **identificação de um conjunto de funções padrão**, inerentes ao problema de subdividir o espaço, foi possível isolar os algoritmos geométricos.

Cada face e cada aresta está associada com o conjunto de funções adequadas ao seu tipo de geometria. Isto permite que a adição de uma nova geometria possa ser feita, simplesmente, reescrevendo o conjunto de funções padrão.

Deve-se ressaltar que isto não resolve os problemas matemáticos e numéricos de interseção de células, especialmente para células com geometria curva. Isto apenas localiza o problema dentro da implementação do *software*.

O *software* está disponível na forma de uma biblioteca, permitindo a integração com outros programas. Em particular, ele pode ser integrado com modeladores de sólidos *manifold* que utilizem representação por fronteira, tornando-os capazes de lidar com agregados de objetos. Isto permite que os operadores CNRG sejam acrescentados a tais modeladores. Dado um conjunto de sólidos *manifold*, os objetos CNRG podem ser definidos baseados em atributos. Os resultados das operações CNRG podem ser fornecidas como a união disjunta de sólidos *manifold*.

7.2 Sugestões para Futuros Trabalhos

Tudo o que foi produzido, neste trabalho, visa, em primeiro lugar, uma utilização em algum sistema de modelagem real. Até hoje, a grande maioria dos sistemas de modelagem não são capazes de lidar com agregados de objetos, tampouco com objetos constituídos por diversos materiais.

A metodologia para construção de SPs já foi utilizada com sucesso na PUC-Rio, em aplicações na área de geologia (reconstituição de movimentos geológicos). Por requerer um investimento maior e, por sua aplicação, em geral, ser mais complexa, a metodologia para construção de SEs ainda não foi utilizada numa aplicação real. O autor acredita que a sua utilização seja tão bem sucedida quanto a sua versão bi-dimensional. No entanto, resta aplicá-la concretamente.

Um ponto que não foi abordado aqui diz respeito à construção de algoritmos para tratar geometrias curvas. Embora a metodologia seja aplicável a qualquer tipo de geometria, resta ainda a tarefa de implementar as funções padrão para vários tipos de geometria utilizadas na prática. A principal dificuldade reside em determinar interseções.

O autor também acredita que a melhor forma de tratar geometria curva seja através de aproximações poliedrais, seguida de uma etapa de refinamento do resultado obtido, através de métodos numéricos, conforme apresentado nas seções 4.5 e 5.5. Seria bastante interessante comparar os resultados obtidos com esta técnica com aqueles obtidos com um tratamento geométrico direto.

Um outro ponto interessante é o desenvolvimento de técnicas que permitam a visualização de objetos com estruturas internas, ou compostos por diversos materiais. Estas técnicas poderiam valer-se de cortes feitos em determinados pontos do objeto, de transparência, ou de modelos explodidos.

Por fim, sugere-se a aplicação na construção de malhas para modelos de elementos finitos tri-dimensionais, utilizando algoritmos para triangulação espacial.

REFERÊNCIAS BIBLIOGRÁFICAS

[ARBA85] Arbab, F. - *Set Models and Boolean Operations for Solids and Assemblies.* Technical Report CS-88-52, Computer Science Department, University of Southern California, L.A., July 1985.

[BAUM72] Baumgart, B. - *Winged-Edge Polyhedron Representation.* Stantford Artificial Inteligence, Report no. CS-320, Oct 1972.

[BOEH84] Boehm, G.F & Farin, G. & Kahmann, J. - *A Survey of Curve and Surface Methods in CAGD.* Computer Aided Geometric Design, Jul 1984, pp 1-60.

[BRAI80] Braid, I.C. & Hillyard, R.C. & Stroud, I.A. - *Stepwise Construction of Polyhedra in Geometric Modeling.* Mathematical Methods in Computer Graphics and Design, Academic Press, 1980.

[CAMP91] Campos, J.A.P. - *Geração de Malhas de Elementos Finitos Bi-dimensionais Baseada em Estruturas de Dados Topológicas.* Dissertação de Mestrado, PUC-Rio, Dep. Eng. Civil, Março 1991.

[CARV89] Carvalho, Paulo C.P. - *Map Making Based on Geometrical Modeling Techniques.* SIAM Conference on Geometric Modeling, 1989.

[CARV90] Carvalho, P.C. & Gattass, M. & Martha, L.F. - *A Software Tool Which Allows Interactive Creation of Planar Subdivisions and Applications to Educational Programs.* Proceedings of the International Conference on Computer Aided Training in Science and Technology, Barcelona, July 1990. Edited by: E. Oñate et.al., Pinerage Press, 1990, pp 201-207.

[CARV91] Carvalho, P.C. & Figueiredo, L.H. - *Introdução à Geometria Computacional.* Publicação do IMPA, 1991.

[CARV92] Carvalho, P.C. & Gomes, Jonas de M. & Velho, L.C. - *Space Decompositions: Theory and Techniques.* Preprint, IMPA, 1992.

[CAVA90] Cavalcanti, P.R. & Carvalho, P.C.P. & Martha, L.F. - *Criação e Manutenção de Subdivisões Planares.* Anais do IV SIBGRAPI, São Paulo, julho de 1991, pp 13-24.

[DICK03] Dickson, L.E. - *Introduction to the Theory of Algebraic Equations.* John Wiley and Sons, New York, 1903. Também encontrado em *Congruence of Sets and Other Monographs.* Edited by Sierpiński et.al., Chelsea Publishing Company, Bronx, N.Y.

[DOBK87] Dobkins, D.P. & Laszlo, M.J. - *Primitives for the Manipulation of Three-dimensional Subdivisions.* Third ACM Symposium on Computational Geometry, Waterloo, Canada, Jun 1987, pp 86-99.

[FISC91] Fischer, Rolf - *GeneSys - Sistema Híbrido para Modelagem de Sólidos.* Dissertação de Mestrado, Dep. Informática, PUC-Rio, Agosto 1991.

[GREE67] Greenberg, Marvin - *Lectures on Algebraic Topology.* W.A.Benjamin Inc., N.Y., 1967.

[GUNT88] Gunther, O. - *Efficient Structures for Geometric Data Management.* Springer-Verlag, Lecture Notes in Computer Science, 337, 1988.

[HENL79] Henle, M. - *A Combinatorial Introduction to Topology.* W.H.Freeman and Company, San Francisco, 1979.

[HOFF89] Hoffmann, Christoph M. - *Geometric & Solid Modeling: An Introduction.* Morgan Kaufmann Publishers, Inc., 1989.

[KUNI85] Kunii, T. & Satoh, T. & Yamaguchi, K. - *Generation of Topological Boundary Representations from Octree Encoding.* IEEE CG&A, Mar 1985, pp 29-38.

[LANE80] Lane, J.M. & Riesenfeld, R.F. - *A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces.* IEEE Trans. Pattern Analysis and Machine Intelligence, Jan 1980, pp 35-46.

[LASZ87] Laszlo, M.J. - *A Data Structure for Manipulating Three-dimensional Subdivisions.* Ph.D. Thesis, Report CS-TR-125-87, Department of Computer Science, Princeton University, Aug 1987.

[LIEN88] Lienhardt, P. - *Extension of the Notion of Map and Subdivisions of a Three-dimensional Space.* Cinquieme Symposium sur les Aspects Thèoriques de L'Informatique, STACS 88, Bordeau, Feb 1988.

[LIMA70] Lima, Elon Lages - *Elementos de Topologia Geral.* Ao Livro Técnico, R.J., 1970.

[LIMA87] Lima, Elon Lages - *Meu Professor de Matemática.* Sociedade Brasileira de Matemática, 1987.

[LOJA64] Lojasiewicz, S. - *Triangulation of Semi-Analitic Sets.* Annali della Scuola Normale Superieure di Piza, Vol 18, 1964, pp 450-474.

[MANT82] Mäntylä, Martti - *An Inversion Algorithm for Geometric Models.* Computer Graphics, Vol.16, number 3, Jul 1982.

[MANT83] Mäntylä, Martti & Tamminen, M. - *Localized Set Operations for Solid Modeling.* Computer Graphics, Vol.17, number 3, Jul 1983.

[MANT84] Mäntylä, Martti - *A Note on the Modeling Space of Euler Operators.* Computer Vision, Graphics and Image Processing 26, 1984, pp 45-60

[MANT86] Mäntylä, Martti - *Boolean Operations of 2-Manifolds Through Vertex Neighborhood.* ACM Transactions on Graphics, Vol.5, No.1, Jan 1986, pp 1-29.

[MANT88] Mäntylä, Martti - *Solid Modeling.* Computer Science Press, 1988.

[MART89] Martha, Luis Fernando - *Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three Dimensions.* Ph.D. Thesis, Cornell University, Ithaca, N.Y., 1989.

[MASS67] Massey, Wilians S. - *Algebraic Topology: An Introduction.* Harcourt, Brace & World, N.Y., 1967.

[MURA90] Murabata, S. & Higashi, M. - *Non-Manifold Geometric Modeling for Set Operations and Surface Operations.* Rensselaer Polytechnic Institute, Jun 1990.

[NAYL71] Naylor, Arch W. & Sell, George R. - *Linear Operator Theory.* Holt, Rinehart and Winston, Inc., Jul 1971.

[POTY90] Potyondy, David - *Toward the Simulation of Three Dimensional Reinforced Concrete Subassenblages.* M.S. Thesis, Conell University, May 1990.

[PREP85] Preparata, Franco P. & Shamos, M.I. - *Computational Geometry.* Springer-Verlag, 1985.

[REQU77] Requicha, A. & Voelcker, H. - *Constructive Solid Geometry.* University of Rochester, Production Automation Project, Technical Memo. 25, Nov 1977.

[REQU80] Requicha, A. - *Representations of Solid Objects: Theory, Methods and Systems.* ACM Computing Surveys 12(4), Dec 1980, pp 437-464.

[REQU83] Requicha, A. & Voelcker, H. - *Solid Modeling: Current Status and Research Directions.* University of Rochester, Production Automation Project, IEEE CG&A, Vol.3, 1983, pp 25-37.

[REQU85] Requicha, A. & Voelcker, H. - *Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms.* Proceedings of the IEEE, Vol.73, No.1, Jan 1985, pp 30-44.

[REQU87] Requicha, A. & Vandenbranden, J. - *Automatic Process Planning and Part Programming.* Institute for Robotics and Intelligent Systems, Report IRIS 217, University of Southern California, Los Angeles Ap, 1987.

[ROSS89] Rossignac, Jarek R. & Voelcker, H.B. - *Active Zones in CSG for Accelerating Boundary Evaluation, Redundancy Elimination, Interference Detection and Shading Algorithms.* ACM Trans. Graphics, Vol.8, No.1, 1989, pp 51-87.

[ROSS90] Rossignac, Jarek R. & O'Connor, Michael A. - *SGC: A Dimension-independent Model for Pointsets with Internal Structures and Incomplete Boundaries.* Geometric Modeling for Product Engineering, North Holland 1990, pp 145-180.

[ROSS91] Rossignac, Jarek R. & Requicha, A.G. - *Constructive non-regularized Geometry.* Computer Aided Design, Vol.23, No.1, 1991, pp 21-32.

[SAME84] Samet, H. & Rosenfeld, A., & Shaffer, C.A. & Webber, R.E. - *A Geographic Information System Using Quadtrees.* Pattern Recognition, 17(6), Nov/Dec 1984, pp 647-656.

[SAME88a] Samet, H. & Webber, R.E. - *Hierarquical Data Structures and Algorithms for Computer Graphics, Part I.* IEEE CG&A, 8(3), May 1988, pp 48-68.

[SAME88b] Samet, H. & Webber, R.E. - *Hierarquical Data Structures and Algorithms for Computer Graphics, Part II.* IEEE CG&A, 8(4), Jul 1988, pp 59-75.

[SAME89a] Samet, H. - *Design and Analysis of Spatial Data Structures: Quadtrees, Octrees and other Hierarquical Methods.* Addison-Wesley, Reading, Mass, 1989.

[SAME89b] Samet, H. - *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS.* Addison-Wesley, Reading, Mass, 1989.

[SEDE86] Sederberg, T.W. & Goldman, R.N. - *Algebraic Geometry for Computer-Aided Geometry Design.* IEEE CG&A, Jun 1986, pp 52-59.

[SEDE86] Sederberg, T.M. & Parry, S.R. - *Comparison of Three Curve Intersection Algorithms.* Computer Aided Design, Jan/Feb 1986, pp 58-63.

[SEDE88] Sederberg, T.M. & White, S.C. & Zundel, A.K. - *Fat Arcs: A Bounding Region with Cubic Convergence.* Computer Aided Geometric Design, 1988, pp 205-218.

[SPAN66] Spanier, E.H. - *Algebraic Topology.* Spring Verlag, New York, 1966.

[TING90] Ting, Wu Shin - *Considerations About a Minimal Set of Non-Manifold Operators.* Technische Hochschule Darmstadt - GRIS Wilhelminenstrabe, 7 DA-6100, FRG, Jun 1990.

[TURN85] Turner, Joshua U. - *Precise Solid Modeling with a Faceted Modeler.* IBM Tech. Report No. TR 00.3365, Poughkeepsie, N.Y., 1985.

[TURN88] Turner, Joshua U. - *Accurate Solid Modeling Using Polyhedral Approximations.* IEEE CG&A, 1988, pp 14-28.

[WAWR87] Wawrzynec, Paul A. - *Interactive Finite Element Analysis of Fracture Process.* Department of Structural Engeneering, Report of Cornell University, Feb 1987.

[WEIL83] Weiler, Kevin - *Adjacency Relationships in Boundary Graph-Based Solid Models.* General Electric internal report, Jun 1983.

[WEIL85] Weiler, Kevin - *Edge-Based Data Structures for Solid Modeling in Curved Environments.* IEEE CG&A, 1985, pp 21-40.

[WEIL86] Weiler, Kevin - *Topological Structures for Geometric Modeling.* Ph.D. Thesis, Rensselaer Polytechnic Institute, Troy, N.Y. Aug 1986.

[WILS85] Wilson, P.R. - *Euler Formulas and Geometric Modeling.* IEEE CG&A, Aug 1985, pp 24-36.

[ZEEM79] Zeeman, C.E. - *Uma Introdução à Topologia das Superfícies.* Monografia de Matemática, IMPA, 1979.

APÊNDICE

REPRESENTAÇÃO POR FRONTEIRA

Uma representação por fronteira (BRep) caracteriza um sólido apenas pela sua fronteira, considerada como sendo composta por um conjunto de faces, coladas umas às outras, formando o envoltório fechado do sólido. A descrição geométrica da superfície que o delimita, mais uma orientação topológica que permita estabelecer, para cada ponto da superfície, de qual lado o interior do sólido está, constitui uma representação não ambígua. Esta descrição tem duas partes: a descrição topológica da conectividade e orientação de vértices, faces e arestas e uma descrição geométrica para mergulhar estes elementos de superfície no espaço.

Boa parte dos trabalhos desenvolvidos, até hoje, requerem que a superfície, representada por uma BRep, seja uma variedade de dimensão 2, orientada, fechada e mergulhada no $\Re^3$. Uma variedade é orientável quando é possível distingüir entre cada um dos seus dois lados. As propriedades topológicas das variedades são bem conhecidas, existindo uma teoria matemática bastante rica para descrevê-las. Entretanto, sua utilização, em um esquema de representação, é recente. Mais recente, ainda, são as extensões para representações *non-manifold* necessárias, particularmente, pelo fato de uma operação *booleana* regularizada, entre dois objetos *manifold*, poder produzir um objeto *non-manifold* como resultado.

A estrutura de dados mais antiga utilizada para armazenar a topologia de um poliedro foi desenvolvida, no início da década de 70, por Baumgart [BAUM72] e é conhecida como *winged-edge*. Ela descreve objetos poliédricos *manifold* por três tabelas, que armazenam informações a respeito da conectividade entre vértices, faces e arestas. A nomenclatura utilizada, neste apêndice, está descrita a seguir:

- um **modelo** é um espaço topológico tri-dimensional de modelagem composto por uma ou mais regiões.
- uma **região** é um volume espacial. Existe pelo menos uma região em um modelo e apenas uma região ilimitada.

- uma **casca** é um conjunto de faces que delimita uma região. Uma casca pode ser formada apenas por arames, ou, até mesmo, por um único vértice, sendo chamada, neste caso, de casca pontual.
- uma **face** é uma porção conexa e limitada de uma superfície. Ela está orientada e sua fronteira é formada por um ou mais ciclos. Um dos ciclos representa a sua fronteira externa e os demais (se existirem) são chamados anéis e representam furos.
- um **ciclo** é um conjunto conexo e ordenado de vértices e arestas (alternadamente). Um ciclo pode consistir de um único vértice, sendo chamado, neste caso, de ciclo pontual.
- uma **aresta** é um conjunto (não ordenado) de dois vértices (não necessariamente distintos).
- um **arame** é uma aresta que não pertence a uma face.
- um **vértice** é um ente topológico associado a um ponto do $\Re^3$.
- quando uma aresta pertence a exatamente uma face diz-se tratar de uma aresta **livre**.
- duas faces S e Q de um objeto P estão **encadeadas** quando existe uma seqüência de faces $\{F_1, F_2, ..., F_n\}$ de P com:
  (1) $F_1 = S$ e $F_n = Q$;
  (2) $F_i \cap F_{i+1}$ é uma aresta comum a F_i e F_{i+1}, para $i = 1, n-1$.
- um ciclo B de um objeto P é um **bordo** quando puder ser formado, unicamente, pelas arestas livres de um subconjunto de faces de P.
- um objeto **aberto** possui arestas livres, enquanto um **fechado**, não.

A.1 Fórmulas de Euler-Poincaré

Os dados armazenados em uma BRep são informações simbólicas, ou seja, podem descrever topologias inconsistentes, caso não haja um sólido *manifold* cujos vértices, faces e arestas satisfaçam as relações de adjacência especificadas. Para evitar isto, e garantir que apenas dados topológicos consistentes estão armazenados, foram encontradas algumas fórmulas que relacionam o número de vértices, faces e arestas numa topologia válida. Estas fórmulas estabelecem condições necessárias, mas não suficientes. Condições suficientes são abordadas na seção A.2.

A base para todas estas fórmulas é o teorema de Euler que, descoberto em 1758, afirma que para todo poliedro com V vértices, A arestas e F faces vale: $V - A + F = 2$. Esse teorema é demonstrado por indução em [LIMA87].

Um **poliedro** P é a reunião de um número finito de polígonos simples chamados de faces de P. Os lados destes polígonos são as arestas de P e os vértices das faces são os vértices de P. No entanto, o teorema de Euler só é válido para poliedros que satisfaçam às seguintes condições:

- cada aresta pertence a exatamente duas faces.
- duas faces quaisquer estão encadeadas.
- todo ciclo é um bordo.

Se o poliedro for homeomorfo a uma esfera, estas condições estarão satisfeitas (mas isto é um resultado profundo de topologia). Em particular, isto vale para poliedros convexos, que são poliedros que sempre se situam do mesmo lado de qualquer plano que contenha uma de suas faces (único tipo de poliedro considerado por Euler em sua demonstração).

Foi Poincaré em 1893, o primeiro matemático a compreender que o teorema de Euler é um teorema de topologia, e não de geometria, ao notar que o número $V - A + F$ é um invariante topológico do poliedro P.

Poincaré mostrou que se um poliedro P com V vértices, A arestas e F faces é homeomorfo a um poliedro P' com V' vértices, A' arestas e F' faces, então:

$$V - A + F = V' - A' + F' = \chi(P) \quad \text{(característica de Euler-Poincaré)}.$$

Para todo número inteiro n, existe um poliedro P com $\chi(P) = n$. Por exemplo, os poliedros homeomorfos à esfera possuem característica igual a 2 e os homeomorfos a 1 ou 2-toros, característica igual a 0 ou -2, respectivamente.

Considerando a possibilidade de que os sólidos tenham buracos, mas que continuem limitados por uma única superfície *manifold* conexa, deve-se utilizar uma outra fórmula. É sabido que estes sólidos são topologicamente equivalentes a esferas com zero ou mais alças (ou $n-$toros) [ZEEM79]. O número de alças é chamado de **genus** da superfície.

Generalizando, com genus G, o número de vértices, faces e arestas estão relacionados pela fórmula de Euler-Poincaré:

$$V - A + F - 2(1 - G) = 0.$$

Para considerar sólidos que tenham bolhas internas, separadas por superfícies *manifold* fechadas, chamadas cascas, e que suas faces, embora ainda conexas, possuam fronteiras desconexas formadas por vários ciclos, existe a fórmula de Euler-Poincaré estendida. Seja S o número de cascas, L o número total de ciclos e G a soma dos genus de cada casca, então:

$$V - A + F - (L - F) - 2(S - G) = 0, \quad \text{para objetos fechados.}$$

$$V - A + F - (L - F) - (S - G) = 0, \quad \text{para objetos abertos.}$$

A.2 Sólidos *manifold*

O objetivo desta seção é fornecer condições, necessárias e suficientes, para caracterizar um sólido. A apresentação segue de perto a abordagem apresentada em [HOFF89].

Até agora, ainda, não foi dada uma definição formal do que se considera um sólido. O objetivo é caracterizar um sólido como um poliedro topológico, ou seja, uma variedade de dimensão 3 e com fronteira compacta. Esta definição caracteriza um sólido como um objeto tri-dimensional. Em uma BRep, no entanto, tem-se apenas a descrição da sua fronteira. Deve ser encontrada, então, uma conexão entre a estrutura do sólido e a estrutura topológica da sua fronteira, o que requer o conceito de orientabilidade.

Para caracterizar variedades, de dimensão 2 e 3, em termos de complexos simpliciais, vão-se usar os seguintes teoremas [HOFF89]:

- Uma variedade de dimensão 2 sem fronteira é homeomorfa a um complexo simplicial C de dimensão 2, satisfazendo às seguintes restrições:
  1) todo 1-simplexo de C delimita exatamente dois 2-simplexos.
  2) todos os simplexos incidentes a um 0-simplexo de C formam uma triangulação do círculo.

- Uma variedade de dimensão 3 sem fronteira é homeomorfa a um complexo simplicial C de dimensão 3, satisfazendo às seguintes condições:
  1) todo 2-simplexo de C delimita exatamente dois 3-simplexos.
  2) todos os simplexos incidentes a um 0-simplexo de C formam uma triangulação da esfera. Para incluir variedades com fronteira, basta permitir que possam formar, alternativamente, uma triangulação do disco.
- A fronteira de uma variedade de dimensão 3, conexa, com fronteira e mergulhada no $\Re^3$ é uma variedade de dimensão 2, orientável, sem fronteira e mergulhada no $\Re^3$.
- Uma variedade é orientável se, e somente se, qualquer uma das suas triangulações for orientável. Logo, a orientabilidade de uma variedade de dimensão 2 pode ser estabelecida pela orientação do complexo simplicial que a triangula.

Orienta-se um 2-simplexo, ordenando-se ciclicamente os seus vértices. Em geral, um $d-$simplexo possui exatamente duas orientações.

Sejam S e S' dois 2-simplexos, de um complexo simplicial C, adjacentes a uma mesma 1-face S''. Então, S e S' estão **orientados coerentemente** se as orientações de S'', induzidas por S e S', são opostas.

Seja C uma triangulação de uma variedade de dimensão 2, conexa. Então, C é **orientável** se todos os seus 2-simplexos adjacentes puderem ser orientados coerentemente.

Um ponto p, não pertencente a uma variedade M de dimensão 2, está no interior de M quando existe uma triangulação C de M satisfazendo às seguintes condições:

1) existe um 2-simplexo S de C, orientado como (p_0, p_1, p_2), cujos pontos correspondentes de M são vistos, a partir de p, no sentido anti-horário.
2) existe um ponto q, na imagem de S, tal que o segmento (p, q) só intercepta M em q.

Com esta definição, é possível mostrar que M particiona o $\Re^3$ em dois conjuntos abertos; o primeiro consistindo dos pontos interiores a M e o segundo, consistindo dos pontos que não estão no interior ou sobre M (conjunto dos pontos exteriores).

Agora, é possível apresentar uma definição completa para um sólido *manifold*:

Definição: Um **sólido** *manifold* A, com uma única casca, é uma variedade de dimensão 3, conexa, com fronteira e mergulhada no $\Re^3$. A fronteira ∂A de A é uma variedade de dimensão 2, compacta, sem fronteira, mergulhada no $\Re^3$, com variação limitada e está orientada de tal forma que A consiste do conjunto de pontos interiores a ∂A, juntamente com os pontos de ∂A.

A.2.1 Operadores de Euler (EOps)

Conceitualmente, EOps são as ferramentas básicas para criar e modificar, consistentemente, a topologia (da fronteira) de objetos *manifold*. Eles podem criar superfícies fechadas e modificar estas superfícies adicionando ou eliminando vértices, faces ou arestas. Tradicionalmente, os EOps possuem nomes da forma *mxky*, com m significando *make* e k significando *kill*; x e y indicam o tipo do elemento topológico criado ou destruído.

Reescrevendo a fórmula de Euler-Poincaré estendida, com $R = (L - f)$, tem-se que:

$$V - A + F + 2G - R - 2S = 0.$$

Esta fórmula pode ser considerada como a equação de um hiperplano de dimensão 5 em um espaço de dimensão 6 (na realidade, trata-se de um reticulado, onde cada ponto representa uma topologia válida para um objeto):

$$\nabla Z.X = 0; \qquad X = (V, A, F, G, R, S), \quad Z(X) = V - A + F + 2G - R - 2S.$$

Com este ponto de vista, os EOps podem ser imaginados como vetores de transição no reticulado do subespaço de dimensão 5, definido pela fórmula de Euler-Poincaré estendida [WILS85].

Bastam cinco EOps [BRAI80] para descrever todos os objetos que satisfazem a fórmula de Euler-Poincaré estendida. Estes cinco EOps formam o conjunto mínimo de operadores necessários para criar qualquer objeto *manifold* e são uma base para o plano Euleriano.

$$A = \begin{array}{c} \\ mev \\ mef \\ mvfs \\ kfmrg \\ kemr \\ \end{array} \begin{array}{c} \begin{array}{cccccc} V & A & F & G & R & S \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 1 & 2 & -1 & -2 \end{pmatrix} \end{array},$$

$$A^{-1} = \frac{1}{12} \begin{pmatrix} 9 & -5 & 2 & -2 & 3 & 1 \\ 3 & 5 & -2 & 2 & -3 & -1 \\ -3 & 7 & 2 & -2 & 3 & 1 \\ -6 & 2 & 4 & 8 & -6 & 2 \\ 3 & 5 & -2 & 2 & 9 & -1 \\ -6 & -2 & 8 & 4 & -6 & -2 \end{pmatrix}.$$

Cada transição no plano Euleriano pode ser representada como uma combinação linear dos cinco EOps escolhidos. Seja $\overline{p}$ a transição desejada e $\overline{q}$ o número de aplicações necessárias de cada EOp, então, $\overline{p} = \overline{q}A$ ou $\overline{q} = \overline{p}A^{-1}$. Um valor diferente de zero para q_6 indica que o vetor $\overline{p}$ não representa, de fato, uma transição no plano Euleriano. Um valor negativo em q_i indica que deve ser aplicado o EOp inverso (kev, kef, ...). Como exemplo, considere o tetraedro com 4 vértices, 6 arestas, 4 faces e 1 casca. Neste caso, para descobrir o número de transições necessárias, basta fazer:

$$(4, 6, 4, 0, 0, 1).A^{-1} = (3, 3, 1, 0, 0, 0),$$

que corresponde a 3 mev, 3 mef e 1 mvfs.

Os EOps servem para realizar operações topológicas que irão modificar a topologia de um objeto. O número de EOps, e a função de cada um, num sistema de modelagem geométrica, é arbitrário. Deve-se frisar que todo EOp possui um EOp inverso que cancela a operação executada. Os EOps formam um conjunto completo de primitivas de modelagem para sólidos *manifold* [MANT84].

A.3 Sólidos *Non-Manifold*

Um sólido *non-manifold* pode ser considerado como a **imersão** de vários sólidos *manifold*, ou seja, permite-se que as variedades se auto-interceptem no $\Re^3$, mas restringem-se as interseções às dimensões 0 ou 1. Os objetos são variedades topológicas, mas o seu mergulho no $\Re^3$ permite a coincidência geométrica de estruturas topológicas distintas.

Para identificar os diversos sólidos *manifold*, triangula-se o sólido *non-manifold* e consideram-se os pontos interiores do complexo simplicial resultante. Estes se decompõem em vários conjuntos abertos e conexos, cada um dos quais pode ser entendido como o interior de uma variedade de dimensão 3, com fronteira e imersa no $\Re^3$.

Revertendo o processo, define-se um sólido *non-manifold* como sendo homeomorfo a um complexo simplicial de dimensão 3, cujos pontos interiores são uma variedade de dimensão 3, mergulhada no $\Re^3$ e com variação limitada. O fecho do conjunto de pontos cria uma fronteira compacta e de variação limitada. Em particular, isto implica que a fronteira de um sólido *non-manifold* pode ser triangulada sem conter 2-simplexos degenerados (com área nula).