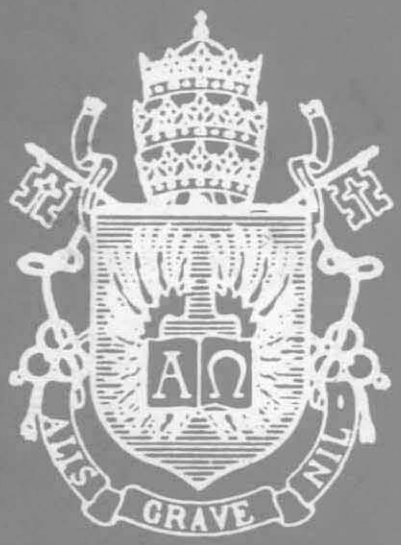


PUC

JORGE ALBERTO PRADO DE CAMPOS

GERAÇÃO DE MALHAS DE ELEMENTOS FINITOS BIDIMENSIONAIS BASEADA EM UMA ESTRUTURA DE DADOS TOPOLÓGICA

DISSERTAÇÃO DE MESTRADO

Departamento de Engenharia Civil
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Rio de Janeiro, Março de 1991

JORGE ALBERTO PRADO DE CAMPOS

GERAÇÃO DE MALHAS DE ELEMENTOS FINITOS BIDIMENSIONAIS BASEADA EM UMA ESTRUTURA DE DADOS TOPOLÓGICA

Dissertação apresentada ao departamento de Engenharia Civil da PUC-Rio como parte dos requisitos para obtenção do título de Mestre em Ciências em Engenharia Civil.
Ênfase : Estruturas.

Orientador: Luiz Fernando Campos Ramos Martha

Departamento de Engenharia Civil

Pontifícia Universidade Católica Do Rio de Janeiro

Rio de Janeiro, Março de 1991

A Teca, Lucas

e Rodrigo.

AGRADECIMENTOS

A Teca, Lucas e Rodrigo, pela paciência e compreensão.

Aos meus Pais, responsáveis por minha formação.

Ao Professor Luiz Fernando Martha, pela orientação deste trabalho, amizade e por todos os conhecimentos transmitidos.

Ao Professor Marcelo Gattass, pela ajuda na definição do tema desta tese.

Aos colegas da PUC, pela amizade e colaboração.

Aos professores e funcionários do Departamento de Engenharia Civil da PUC-Rio, pelo agradável convívio.

Ao CNPq pelo auxílio financeiro.

RESUMO

O presente trabalho consiste em um pré-processador gráfico interativo para a discretização numérica de modelos bidimensionais a serem analisados pelo método dos elementos finitos.

O sistema desenvolvido tem o suporte de uma sofisticada estrutura de dados topólogica e está baseado em uma representação explícita da geometria do problema e em uma forte interação com o usuário. A definição da geometria é feita utilizando-se conceitos de modelagem geométrica e, através de um refinamento progressivo, obtém-se a discretização da estrutura em elementos finitos. O fato de ter-se uma representação do modelo baseada em informações geométricas e topológicas permite que a definição de atributos, como carregamentos, condições de apoio e outros atributos físicos, não se altere durante o processo de discretização do mesmo.

O ambiente desenvolvido mostrou-se bastante propício para a implementação de algoritmos de geração de malhas baseados na técnica de mapeamento transfinito, não restringindo o uso de outras técnicas.

ABSTRACT

The present thesis involves the development of an interactive, computer graphics program for finite element numerical discretization of two dimensional models.

The system relies on a sophisticated topology-based data structure with explicit model geometry representation and on heavy user interaction. Model definitions is performed using geometric modeling concepts. Finite element discretization is performed using a technique of progressive hierarchical refinement which relies on the underlying topological representation. Because loads, support conditions, material properties, and other physical attributes are attached to geometric and topological entities, they need not be reassigned during the process of meshing and remeshing.

This environment is ideally suited for the use of mapping algorithms for meshing generation but it does not constrain the usage of other algorithms.

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

A difusão e popularização dos programas de análise por elementos finitos ou de contorno, de crescente interesse nos meios de engenharia, tem sido dificultada por dois grandes problemas. O primeiro é relativo à geração dos dados. O segundo está ligado à análise dos resultados. Ambos, por serem tarefas exaustivas, consumirem horas de trabalho e estarem sujeitos a erros dos mais diversos tipos, estabelecem barreiras entre o usuário e os referidos programas.

O desenvolvimento de programas de pré e pós-processamento, com uso intensivo da Computação Gráfica e da Modelagem Geométrica, vem diminuindo tal barreira, por propiciar uma interface mais natural homem-máquina e conferir maior rapidez e confiança nos dados gerados. O uso da Computação Gráfica, não obstante o formidável apelo visual, não é suficiente para um programa que prentenda ser interativo. O usuário necessita, também, de rapidez na obtenção das respostas e manipulação dos dados. Estes aspectos estão diretamente relacionados à estrutura de dados adotada.

A estrutura de dados convencional, utilizada nos programas de análise numérica pelo método dos elementos finitos, é eficiente e satisfaz os procedimentos requeridos

para tais programas. Contudo, é muito pobre no que se refere às informações de adjacências entre as entidades topológicas envolvidas, tornando pouco eficiente seu uso em programas de pré e pós-processamento.

A tentativa de inserir informações adicionais na estrutura de dados convencional, objetivando resolver problemas específicos, tem produzido programas relativamente eficientes, porém muito pouco modularizados e gerais.

1.1 - ESCOPO E OBJETIVO

O objetivo deste trabalho é criar um ambiente capaz de representar uma subdivisão planar arbitrária (geometria curva) para servir como a estrutura básica de um gerador de malhas bidimensionais de elementos finitos. Desta forma, as diversas etapas de pré-processamento podem ser vistas como sendo um refinamento da subdivisão planar. No contexto deste trabalho, estas etapas envolvem a definição da geometria, a decomposição em sub-regiões consistentes com os algoritmos de geração de malhas desejados, a discretização dos contornos dessas regiões definindo o grau de refinamento das malhas a serem geradas e, finalmente, o modelo discretizado em elementos finitos. Aqui, este processo é denominado de *definição e decomposição hierárquica do domínio.*

A necessidade de se criar uma subdivisão planar implica no uso de estruturas de dados mais sofisticadas, que sejam capazes de definir eficientemente todas as relações de adjacências entre as entidades topológicas envolvidas (faces, arestas, vértices, etc.). Devido a esta característica, estas

estruturas de dados são chamadas topológicas [WEIL85].

O fato de se manter a representação de uma subdivisão planar a níveis tão baixos quanto a discretização em elementos finitos, onde cada elemento finito é visto como a face de um poliedro delimitada por arestas, que por sua vez são delimitadas por vértices (nós dos elementos finitos), garante uma total consistência dos dados gerados em qualquer fase das diversas etapas envolvidas no processo.

O uso de estruturas de dados topológicas traz ainda outras vantagens, tais como:

- A capacidade de se alterar localmente a estrutura de dados, garantindo a sua integridade global.
- Maior velocidade na consulta e manipulação dos dados. O tempo de resposta a qualquer consulta é, na pior das hipóteses, proporcional ao número de entidades envolvidas.
- A separação entre informações topológicas e geométricas. Isto minimiza ao máximo os problemas geométricos decorrentes de uma aritmética de ponto flutuante. Aqui, como em trabalhos anteriores [WEIL86][MART87][MART89], a geometria é vista como atributo das entidades topológicas.
- A fácil manipulação da topologia das regiões para geração da malha de elementos finitos. Qualquer mudança na topologia é rápida e com um impacto mínimo na malha existente.
- A implementação de diversos níveis de operadores contribui para minimizar a complexidade da manipulação

da estrutura de dados, conferindo ao programa uma arquitetura aberta, onde a implementação de novos procedimentos, como um novo algoritmo de geração de malhas, é feita de maneira simples e concisa.

O principal problema da implementação de estruturas de dados topológicas é a necessidade de grandes espaços de memória para o seu armazenamento.

1.2 GERAÇÃO DE MALHAS

O enfoque adotado para a geração de malha de elementos finitos no presente trabalho está baseado em uma manipulação interativa da geometria e topologia do modelo a ser analisado. Não é o objetivo deste trabalho o desenvolvimento de novos algoritmos de geração de malhas, embora a arquitetura do sistema permita que novas técnicas de geração de malhas sejam introduzidas facilmente. Na verdade, esta é uma das principais caractéristica do sistema: a criação de um ambiente de subdivisão planar genérica possibilitando a utilização do algoritmo de geração de malha mais adequado para cada região da subdivisão.

As técnicas de geração de malhas de elementos finitos, de uma forma geral, se dividem em três grupos: mapeamento transfinito [HABE81], enumeração espacial (quadtree/octree) [YERR84][YERR85][KELA87] e algoritmos de tecelagem [SHAW78] [CAVE85][SCHR88][CHEW89].

Algoritmos de mapeamento transfinito são os mais rápidos e geralmente produzem malhas com elementos de forma regular apropriada. A desvantagem desta técnica é que só pode ser

aplicada a regiões com topologia triangular ou quadrangular, e que também atendam a algumas restrições quanto à subdivisão de suas fronteiras. Isto implica na necessidade de dividir o domínio de geração de malha em regiões que sejam isomorfas a estas formas mais simples.

Técnicas de enumeração espacial são atualmente muito populares para uma geração de malha de forma completamente automática. A vantagem destes algoritmos é que eles são robustos para regiões de forma arbitrária. Um problema com estas técnicas é que elas só são adequadas para regiões com valores grandes de área de superfície. A aplicação destes algoritmos para regiões com pequena área de superfície (por exemplo no caso de uma pá de turbina) resulta em um excessivo número de elementos, além de elementos com formas irregulares.

Os demais algoritmos de geração de malha são classificados como arbitrários ou de tecelagem. Tipicamente [SHAW78], estes algoritmos começam dos contornos das regiões e os "contraem" tecendo uma malha de triângulos até que as fronteiras contraídas recaiam em triângulos simples. Por isso são denomidados de algoritmos de tecelagem. Estes algoritmos têm a vantagem de trabalhar bem em regiões com formas complexas, em particular regiões que incluam furos e cavidades. A principal desvantagem é que estes algoritmos são lentos, tipicamente de ordem quadrática com o número de elementos gerados. Esta ordem pode ser melhorada, mas envolve etapas adicionais de pré-processamento [PREP85].

O modelo presente para geração de malha baseia-se em informações de adjacência topológica e computação gráfica

interativa para subdividir o domínio em sub-regiões, manipulá-las de forma simples e, finalmente, decompor as sub-regiões em elementos finitos. Este ambiente é bastante propício para a utilização de algoritmos de mapeamento, mas não restringe o uso de outros algoritmos. Devido ao rápido acesso às informações asssociadas ao contorno das sub-regiões e à modularidade dos operadores de consulta e manipulação da estrutura de dados, o programa fornece uma conveniente plataforma onde diversos esquemas de geração de malhas podem ser facilmente implementados.

Atualmente estão implementados no sistema alguns algoritmos de mapeamento transfinito discreto [HABE81] e um algoritmo de tecelagem triangular [SHAW78]. Os algoritmos de mapeamento considerados são o mapeamento linear (uma só direção) e bilinear, para regiões com topologia quadrilateral, e bilinear com um dos lados degenerados para um ponto e trilinear, para regiões com topologia triangular.

1.3 - ORGANIZAÇÃO DO PROGRAMA E DA TESE

A Figura 1.1 ilustra a organização geral do programa de geração de malhas.

O nível inferior mostrado nesta figura contém a estrutura de dados topológica e os atributos das entidades topológicas. A estrutura de dados é descrita no Capítulo 2. Os atributos que armazenam as informações utilizadas para o gerenciamento da decomposição hierárquica do domínio são descritos no Capítulo 3. Os atributos que contém as informações geométricas, assim como a manipulação de dados geométricos, estão descritos

no Capítulo 4.

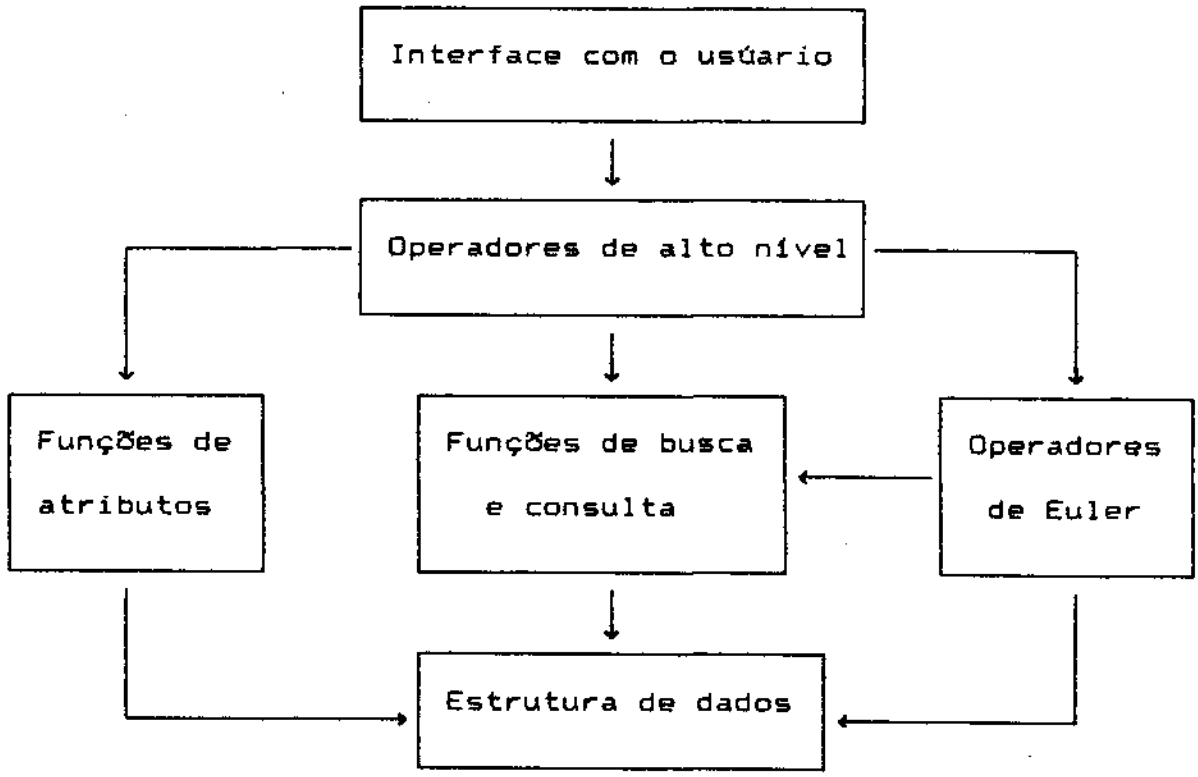


Figura 1.1 - Organização do programa.

Em um nível acima têm-se os operadores que manipulam a estrutura de dados topológica, chamados de operadores de Euler, as funções de atributos e as funções de busca e consulta à estrutura de dados. Os operadores de Euler estão listados no Apêndice A. As funções que manipulam os atributos não são descritas explicitamente nesta tese pois isto corresponde a um mero problema implementacional. No entanto, fica clara, a partir da descrição dos atributos nos Capítulos 3 e 4, a finalidade destas funções. Também por constituírem apenas um problema de implementação, as funções de busca e consulta à

estrutura de dados não são descritas aqui.

Os operadores de alto nível indicados na Figura 1.1 são responsáveis pelo tratamento dos aspectos geométricos e pela formalização dos comandos fornecidos pelo usuário. Estes operadores estão listados no Apêndice B.

Os operadores de alto nível também são responsáveis pelo acionamento dos algoritmos de geração de malhas. A adição, na estrutura de dados, de elementos finitos gerados pelos algoritmos também é feita pelos operadores de alto nível. Como os algoritmos implementados no presente programa já constituem literatura publicada há vários anos e são de conhecimento geral da área, eles estão descritos apenas sucintamente neste trabalho, o que é feito no Capítulo 3.

Na organização da tese, as principais vantagens e características inovadoras do presente enfoque para geração de malhas de elementos finitos são demonstrados, através de exemplos, no Capítulo 5.

Finalmente, no Capítulo 6 são descritas as principais conclusões deste trabalho e fazem-se sugestões para futuras implementações.

CAPÍTULO 2

ESTRUTURAS DE DADOS TOPOLÓGICAS

As estruturas de dados topológicas são capazes de representar as superfícies externas de poliedros tridimensionais [WEIL85]. Neste trabalho, a utilização destas estruturas em duas dimensões faz com que se interprete uma subdivisão planar como o mapeamento contínuo da superfície de um poliedro no plano completado.

Estas estruturas de dados são ditas topológicas porque possibilitam encontrar eficientemente todas as relações de adjacência entre as entidades topológicas.

Existem diversas maneiras de se armazenar estas informações. O interesse deste trabalho está em estruturas de dados topológicas suficientes e que suportam a representação de superfícies 2-*manifold* (variedade dois) ou, simplesmente, *manifold*. Define-se como superfície *manifold* um espaço onde cada ponto tem uma vizinhança que é topologicamente equivalente a um disco aberto [MANT88]. Este é o caso da superfície de um poliedro.

Uma estrutura de dados topológica é dita suficiente se ela é capaz de reproduzir, de maneira única, as nove relações de adjacências entre as entidades topológicas [WEIL86]: face [f], aresta (*edge*) [e] e vértice [v] (Figura 2.1).

Na Figura 2.1, a primeira letra indica a entidade topológica de referência e a segunda indica o grupo de entidades topológicas adjacentes à primeira.

V V	V E	V F
E V	E E	E F
F V	F E	F F

Figura 2.1 - Relações de adjacências entre as entidades topológicas vértices(v), arestas(e) e faces(f).

Como exemplo, considere-se a relação VE. Esta relação fornece todas as arestas adjacentes à entidade de referência vértice. Entende-se como arestas adjacentes a um vértice todas as arestas que possuem este vértice como um dos seus extremos.

Outra importante característica da representação topológica *manifold* é que ela suporta a implementação de um conjunto de operadores chamados de operadores de Euler. Estes operadores são procedimentos que, juntos com as funções de busca e consulta, estão credenciados a acessar diretamente a estrutura de dados. Os operadores de Euler são assim denominados por garantir, a qualquer tempo na manipulação da estrutura de dados, a consistência da representação topológica e como consequência a validade da fórmula de Euler que relaciona as entidades topológicas de um sólido com fronteira 2-*manifold*. No caso de um poliedro sem furos ou cavidades esta fórmula é expressa por:

(V - E + F = 2)

2.1 ESTRUTURAS DE DADOS TOPOLÓGICAS BASEADAS EM ARESTAS

Dentre as diversas possibilidades de estruturas de dados topológicas, destaca-se um grupo de especial interesse, aquele onde a entidade topológica de referência é a aresta. Estas estruturas, também chamadas de representações baseadas na aresta (*edge-based representation*), apresentam algumas propriedades bastante interessantes. A principal delas é que o número de entidades topológicas adjacentes a uma aresta em uma representação *manifold* é limitado. Isto é, o número de vértices e faces adjacentes a uma aresta é no máximo dois, e o número de arestas adjacentes a uma aresta é no máximo quatro.

Os vértices adjacentes a uma aresta são aqueles que delimitam esta aresta, as faces adjacentes a uma aresta são aquelas que têm esta aresta como parte do grupo de arestas que as delimitam e, por fim, as arestas adjacentes a uma aresta são todas aquelas que compartilham um vértice e uma face com a aresta de referência. Esta característica é muito importante numa implementação computacional, visto que o tamanho dos registros da estrutura de dados é limitado e conhecido.

A Figura 2.2 ilustra as relações de adjacências em um sólido com fronteira 2-*manifold*. Considerando-se a aresta de referência e_1, tem-se como vértices adjacentes as entidades v_4 e v_6. As faces adjarentes são as entidades f_1 e f_2 e as arestas adjacentes são e_2, e_4, e_5 e e_7. Nota-se que a aresta e_{10}, embora compartilhe um vértice com a aresta de referência, não é

considerada adjacente à mesma, pois esta entidade não possui uma face adjacente comum à aresta de referência.

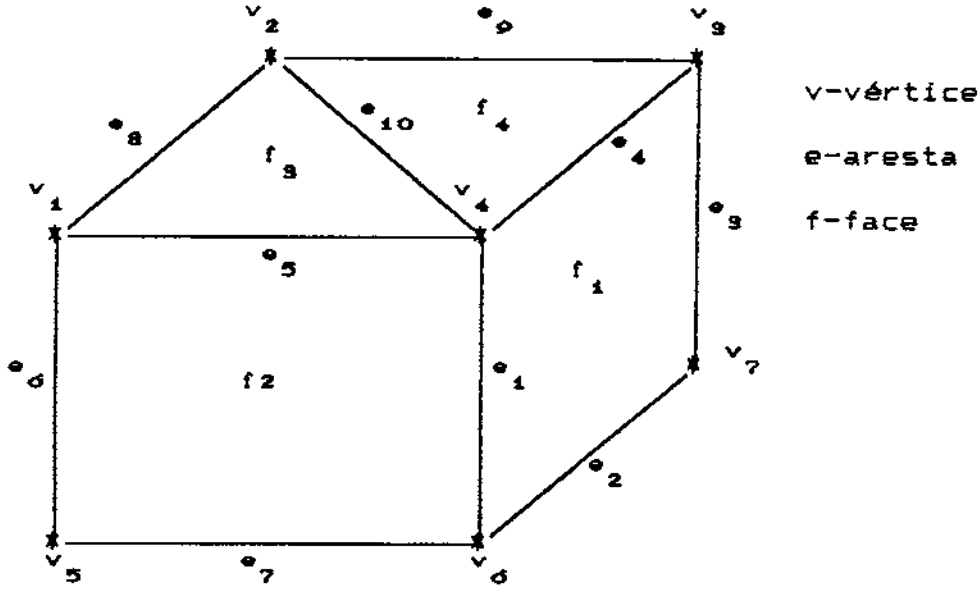


Figura 2.2 - Relações de adjacências em um sólido com fronteira 2-*manifold*.

Apresentam-se, a seguir, algumas estruturas de dados baseadas na aresta. O objetivo, aqui, não é abordar todas as estruturas existentes, mas sim apresentar alguns aspectos das que considerou-se mais importante para o desenvolvimento deste trabalho. Também não é o objetivo, aqui, demonstrar a sufiência destas estruturas. A prova da suficiência de algumas das estruturas apresentadas pode ser encontrada no trabalho de Weiler [WEIL86].

A estrutura de dados arestas aladas (*winged edge*), concebida por Baumgart [BAUM75] com a finalidade de representar superfícies de poliedros com faces planas, foi inicialmente utilizada na pesquisa de reconhecimento de padrões aplicados à

robótica. As principais relações de adjacência armazenadas explicitamente na estrutura de dados arestas aladas são todas as faces, arestas e vértices adjacentes a uma aresta de referência (Figura 2.3). Além disso, cada vértice e cada face aponta para uma aresta adjacente. Isto permite que as relações de adjacências centralizadas em vértices e faces possam ser obtidas também de forma eficiente.

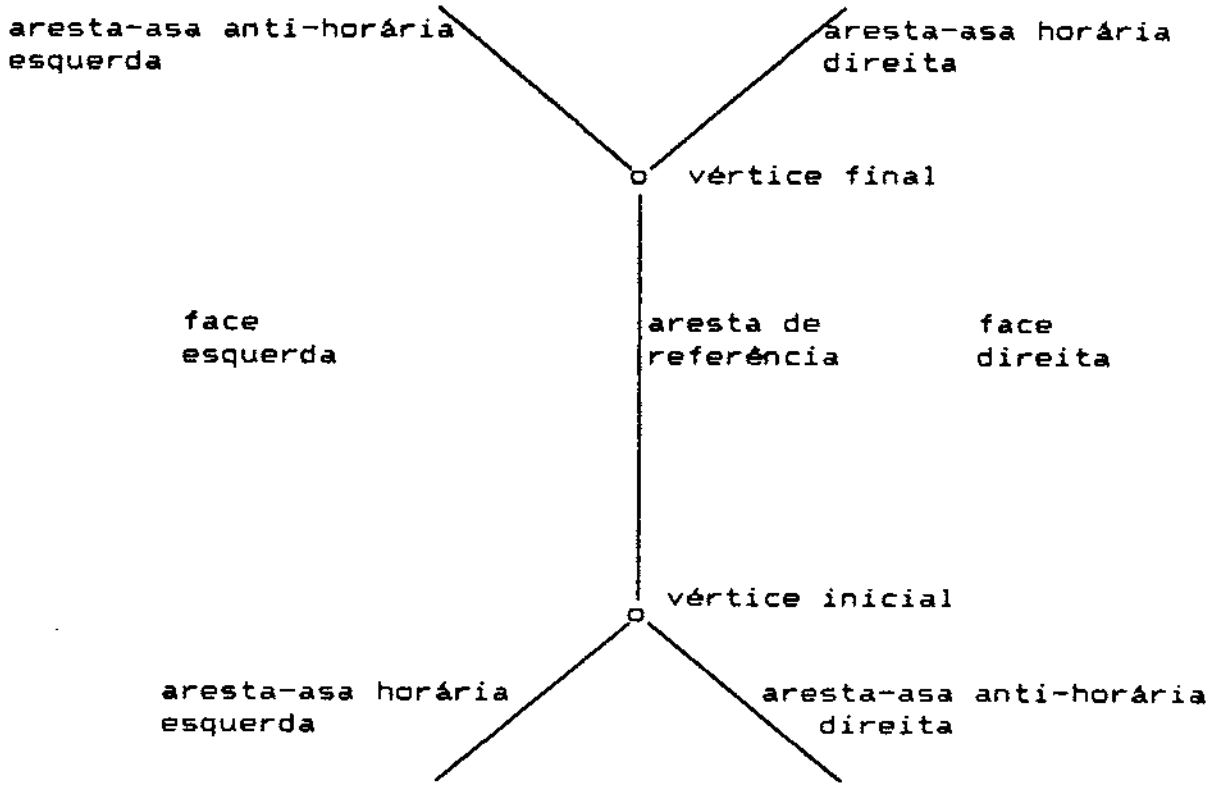


Figura 2.3 - Estrutura de dados arestas aladas.
Elemento topológico aresta.

A principal desvantagem da estrutura arestas aladas está na sua incapacidade de tratar faces com fronteiras desconexas (Figura 2.4-a). Este problema pode ser contornado com a

introdução das chamadas arestas "fictícias" (Figura 2.4-b).

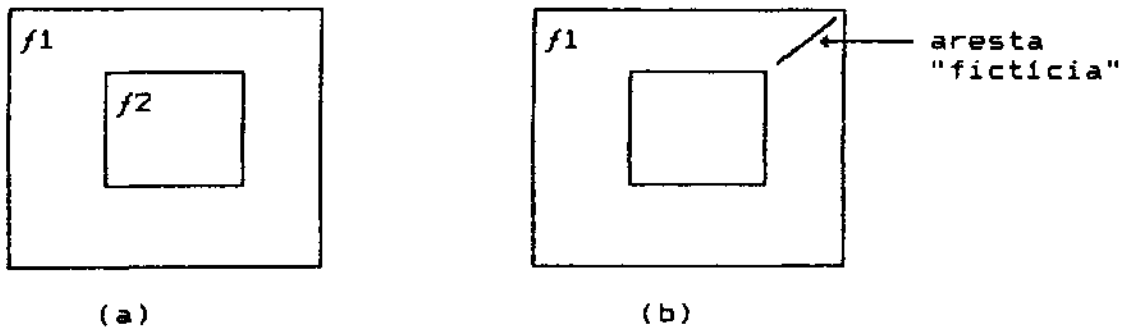


Figura 2.4 - Simulação de faces desconexas com arestas "fictícias".

O termo aresta "fictícia" não implica, contudo, na não existência desta aresta, pois ela é necessária para manter a consistência do modelo, mas sim devido ao fato dela não pertencer ao modelo que se deseja representar.

A introdução de arestas "fictícias" em programas interativos, onde a manipulação do modelo é uma constante, acarreta diversos problemas de ordem prática. A criação e eliminação destas arestas têm que ser feitas de forma automática e completamente transparente ao usuário do programa de aplicação. Isto implica na elaboração de códigos mais extensos e pouco eficientes. Outro grande problema é a representação gráfica dessas arestas. Ao se optar em não representá-las, dá-se ao usuário uma visão mais próxima do modelo, mas, por outro lado, a visualização do mesmo não guarda uma relação de um para um com as informações armazenadas na estrutura de dados.

A introdução de mais informações, na estrutura original

arestas aladas, tem contribuído para torná-la mais eficiente.

Uma contribuição particurlamente interessante, feita por Weiler [WEIL85], originou a chamada estrutura de dados arestas aladas modificada (*modified winged edge*). A diferença consiste, basicamente, na consideração explícita da direção das arestas adjacentes. Esta pequena e importante modificação reduz a complexidade dos algoritmos que consultam e manipulam a estrutura de dados. Contudo, esta estrutura também não consegue representar faces com fronteiras desconexas.

Outra importante estrutura de dados baseada na aresta é a estrutura semi-aresta (*half-edge*) [MANT88] ou face-aresta (*face-edge*) [WEIL85]. Pode-se referir a qualquer uma das duas, indistintamente, pois estas estruturas são conceitualmente idênticas, diferindo somente na nomenclatura e procedimentos sugeridos pelos autores.

A grande diferença da estrutura face-aresta e a arestas aladas é a substituição da entidade topológica aresta por duas entidades, chamadas aresta-uso (*edgeuse*). A entidade aresta-uso, que será discutida mais detalhadamente na seção seguinte, pode ser vista como sendo a divisão das informações contidas na aresta da estrutura arestas aladas, indicando agora, também, o "uso" destas arestas pelas faces adjacentes (Figura 2.5).

O fato de se considerarem, explicitamente, as relações de adjacência entre as arestas-uso no lugar da aresta propriamente dita, simplifica os algoritmos de busca, aumenta a velocidade no acesso e manipulação da estrutura de dados e permite ainda um tratamento mais consistente no caso de arestas curvas. Além

disso, a inclusão do elemento topológico ciclo (Figura 2.5), que representa um circuito de arestas-uso, permite, como explicado no restante deste capítulo, a representação de faces com fronteiras desconexas.

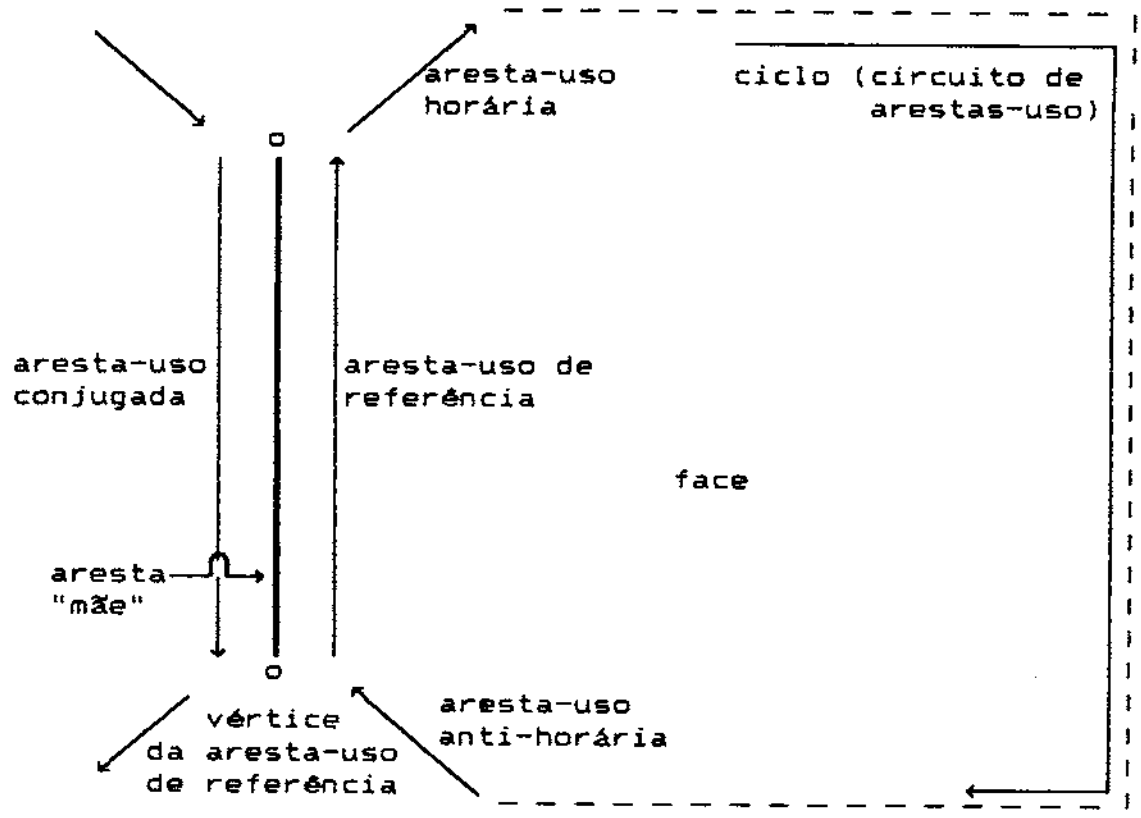


Figura 2.5 - Estrutura de dados face-aresta.
Elemento topológico aresta-uso.

A estrutura de dados face-aresta modificada - FEDm (*modified face-edge data structure*) [MART87], utilizada no presente sistema, é semelhante à já mencionada estrutura face-aresta, exceto pela inclusão da entidade topológica vértice-uso (*vertexuse*).

Esta entidade foi originalmente introduzida por Weiler na apresentação da estrutura arestas radiais (*radial-edge*) [WEIL86] que suporta a representação tridimensional de topologias *non-manifold*. As razões que levaram a consideração da entidade vértice-uso bem como uma descrição mais detalhada da estrutura FEDm estão relacionadas no restante deste capítulo.

2.2 - ESTRUTURA DE DADOS FACE-ARESTA MODIFICADA

A representação explícita da entidade vértice-uso é redundante na maioria das aplicações bidimensionais (*manifold*), uma vez que as informações de adjacências adicionais poderiam ser obtidas diretamente na estrutura face-aresta original.

Uma das razões que levou à consideração deste elemento na presente estrutura é que ela serviu de base na elaboração de um protótipo bidimensional para um sistema aplicado a discretização numérica e simulação de fraturas arbitrárias em três dimensões [MART89]. Os algoritmos então desenvolvidos para o protótipo puderam ser facilmente adaptados ao problema tridimensional, que utiliza a estrutura arestas radiais. Sob este ponto de vista, a estrutura FEDm pode ser interpretada como uma simplificação da estrutura arestas radiais.

Contudo, existem outras razões que justificam a manutenção deste elemento no presente sistema, visto que o interesse aqui não era evoluir até um modelador tridimensional.

O fato de algumas informações, embora redundantes, estarem armazenadas explicitamente na estrutura de dados aumenta a velocidade em algumas funções de busca e manipulação

dos dados. Esta é uma propriedade muito importante em programas interativos.

A última, e decisiva razão, que levou à manutenção do elemento vértice-uso, reside no fato do presente programa de geração de malhas suportar a representação das chamadas arestas de auto-ciclo (*self-loop edge*). Estas arestas são caracterizadas por terem seus vértices geometricamente coincidentes e tratados como uma única entidade topológica. O elemento vértice-uso introduz a unicidade necessária às relações de adjacências das arestas de auto-ciclo.

O problema da aresta de auto-ciclo é que ela sozinha define dois ciclos, um interno e outro externo (Figura 2.6). Como associada a cada aresta só existe um vértice, não existem informações suficientes para identificar de maneira única os ciclos, tornando-se necessária a consideração do elemento vértice-uso para resolver esta ambiguidade. O único vértice desta aresta tem associado a ele dois vértices-uso. Adotando-se uma convenção que sempre relaciona o primeiro vértice-uso com o ciclo interno e o segundo com o ciclo externo, tem-se uma completa definição das relações de adjacências. A Figura 2.6 ilustra as relações de adjacências das arestas de auto-ciclo.

Uma importante característica desta implementação é que as entidades aresta-uso e vértice-uso só são conhecidas no nível dos operadores de Euler e de algumas funções de busca e consulta. Esta característica confere ao programa uma maior flexibilidade quanto ao tipo de estruturas de dados que pode ser implementada, permitindo a troca, sem maiores transtornos e com as devidas limitações inerentes a cada estrutura, por uma

estrutura de dados do tipo arestas aladas.

Os operadores de Euler que manipulam a estrutura FEDm estão descritos no Apêndice A.

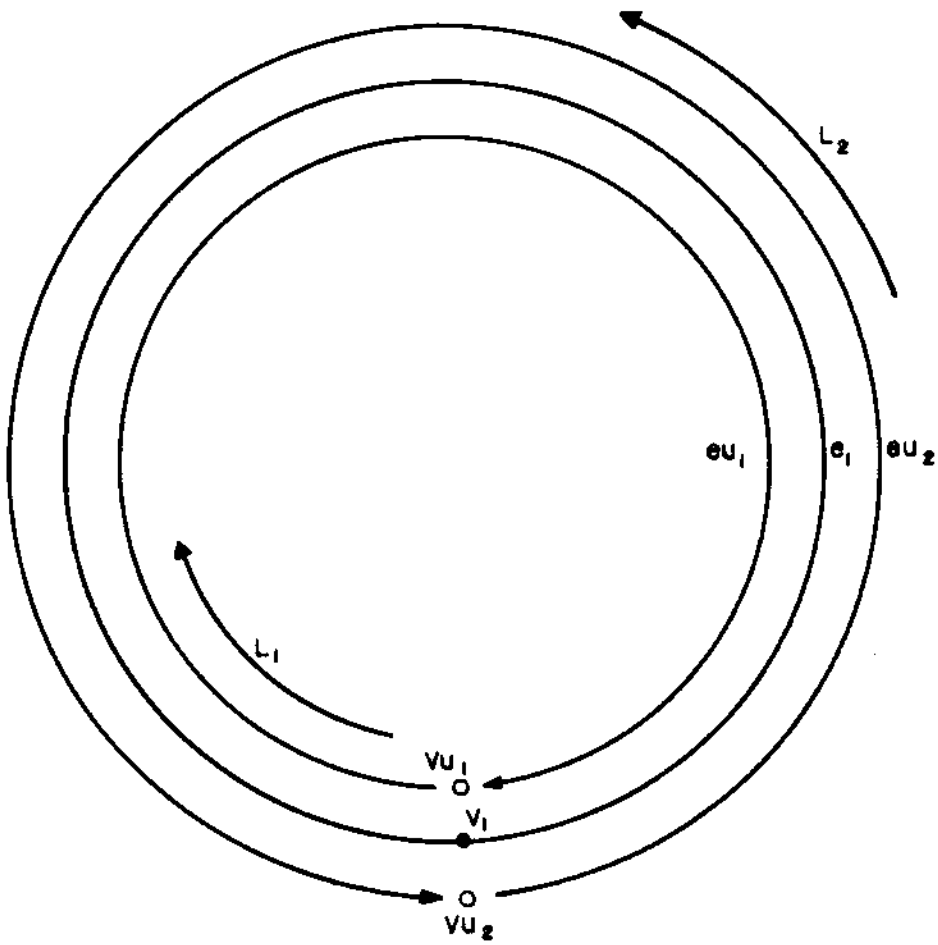


Figura 2.6 - Relações de adjacências das arestas de auto-ciclo.

2.2.1 - ENTIDADES TOPOLÓGICAS

As entidades topológicas presentes na FEDm, modelo, casca (*shell*), face, ciclo (*loop*), aresta (*edge*), vértice, aresta-uso (*edgeuse*) e vértice-uso, serão aqui representadas pelas letras m, s, f, l, e, v, eu e vu respectivamente.

A descrição das entidades topológicas é feita com auxílio do exemplo mostrado na Figura 2.7. A Figura 2.7-a mostra a representação de um polígono utilizando as entidades básicas: face, arestas e vértices. A Figura 2.7-b mostra a representação do mesmo polígono com as entidades presentes na FEDm.

O modelo representa toda a estrutura, e não apenas uma entidade topológica.

A casca é uma superfície orientada que define o contorno de uma região no espaço. No caso presente, onde se tem uma representação bidimensional, a casca é uma subdivisão do plano.

Existe uma relação de um para um entre a casca e o modelo, ou seja, cada modelo tem apenas uma casca, que é a superfície sobre a qual será feita a modelagem.

A representação explícita desta entidade foi mantida por motivos puramente conceituais, não sendo necessária na prática para problemas bidimensionais.

A face representa a parcela da casca que ela cobre, sem incluir o seu contorno. Os contornos das faces são formados pelas entidades arestas e vértices.

A face que cobre toda a casca, exceto as faces das regiões já modeladas, é denominada de face livre. No exemplo da Figura 2.7 existem duas faces: a face f_1 e a face livre f_2.

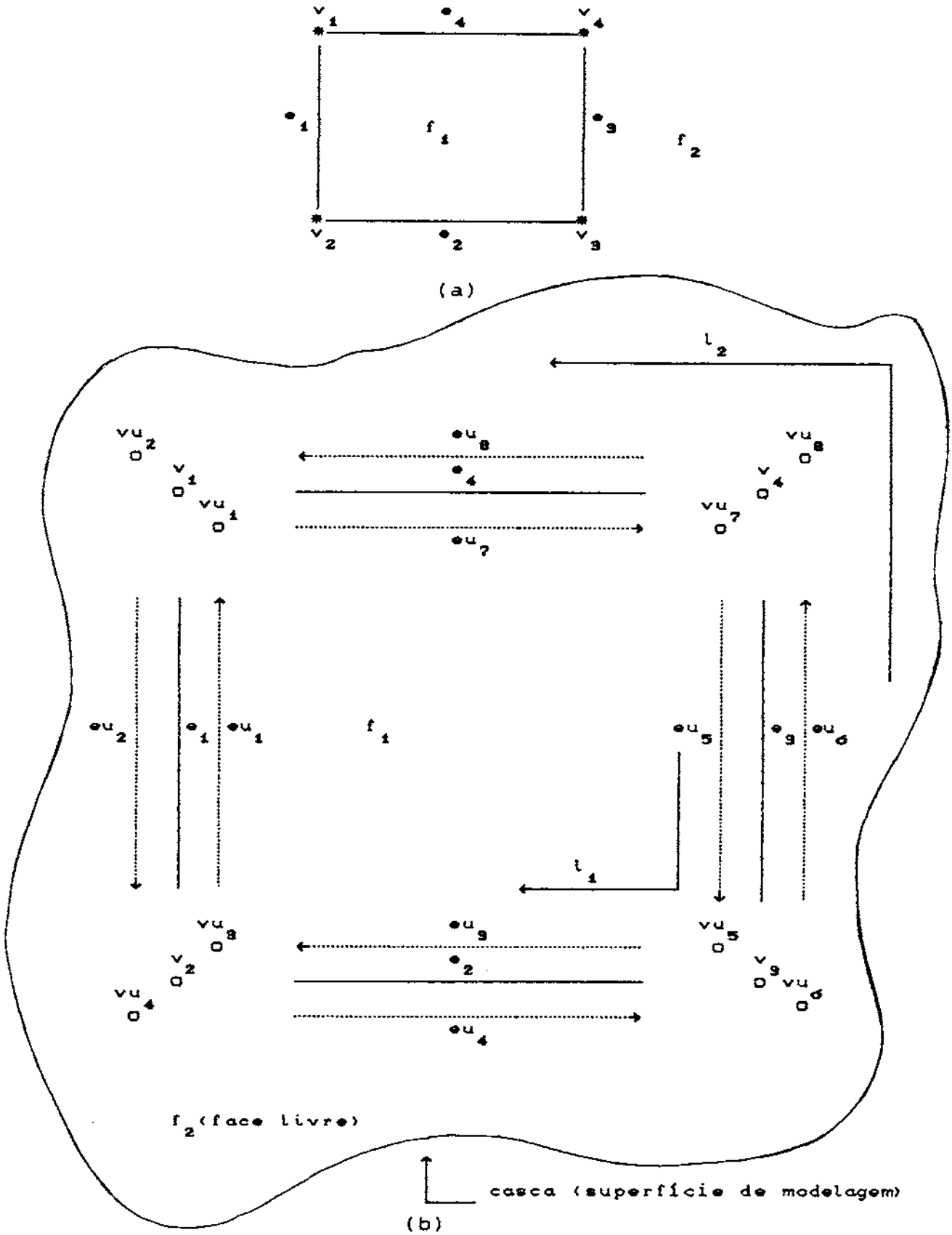


Figura 2.7 - Representação de um polígono na FEDm.

O ciclo é um bordo contínuo e orientado de uma face. Cada face pode possuir vários ciclos, um externo e os demais internos, com exceção da face livre que possui apenas ciclos internos. Na Figura 2.7, a face interna f_1 possui um ciclo externo (l_1) e a face livre f_2 possui um ciclo interno (l_2).

Nesta implementação, os ciclos externos estão orientados no sentido horário e os internos no sentido anti-horário. Seguindo esta orientação, os ciclos são armazenados como uma sequência orientada de arestas-uso. Existe um caso especial de um ciclo degenerado em um vértice. Neste caso, o ciclo é considerado como um ciclo interno da face que o possui.

A entidade **aresta-uso** é um dos segmentos orientados que definem um ciclo e representam o uso de uma aresta neste ciclo. Na Figura 2.7, o ciclo l_1, por exemplo, é formado pelas arestas-uso eu_1, eu_7, eu_5 e eu_9.

Na representação de topologias *manifold* existem exatamente duas arestas-uso associadas a cada aresta. Esta entidade contém a maior parte das informações armazenadas na estrutura de dados.

A entidade **vértice-uso** representa o uso de um vértice por uma aresta-uso. Por exemplo, na Figura 2.7, vu_1 representa o uso do vértice v_1 pela aresta-uso eu_7.

A **aresta** é um segmento do contorno entre duas faces. As arestas são delimitadas, em seus extremos, por dois vértices, que, no caso de arestas auto-ciclo (Figura 2.6), é o mesmo vértice.

O **vértice** é, simplesmente, um ponto no espaço. No caso bidimensional, corresponde a um ponto sobre o plano de

construção (casca). Somente um vértice pode ocupar uma determinada posição no espaço.

2.2.2 - RELAÇÕES DE ADJACENCIAS E REGISTROS DE DADOS

As entidades topológicas estão organizadas em uma representação tipo "de cima para baixo" (*top-down*). A Figura 2.8 mostra a representação gráfica das informações topológicas básicas, presentes na FEDm, segundo a referida representação.

Existem algumas vantagens em se manter uma representação hierárquica deste tipo, onde a descrição dos dados é feita do elemento de maior dimensão topológica (modelo) para o de menor dimensão (vértice-uso).

A principal é que, para determinados algoritmos, pode-se economizar tempo, descartando-se, a *priori*, um grande número de entidades que seriam desnecessariamente processadas.

Esta eliminação é feita através da consulta às informações contidas nas entidades de dimensão superior, quantitativamente em menor número. Com rápidos testes comparativos é possivel identificar um grupo mais restrito de elementos, que serão finalmente testados.

A opção por listas duplamente encadeadas visa obter uma maior flexibilidade e rapidez na manipulação das mesmas.

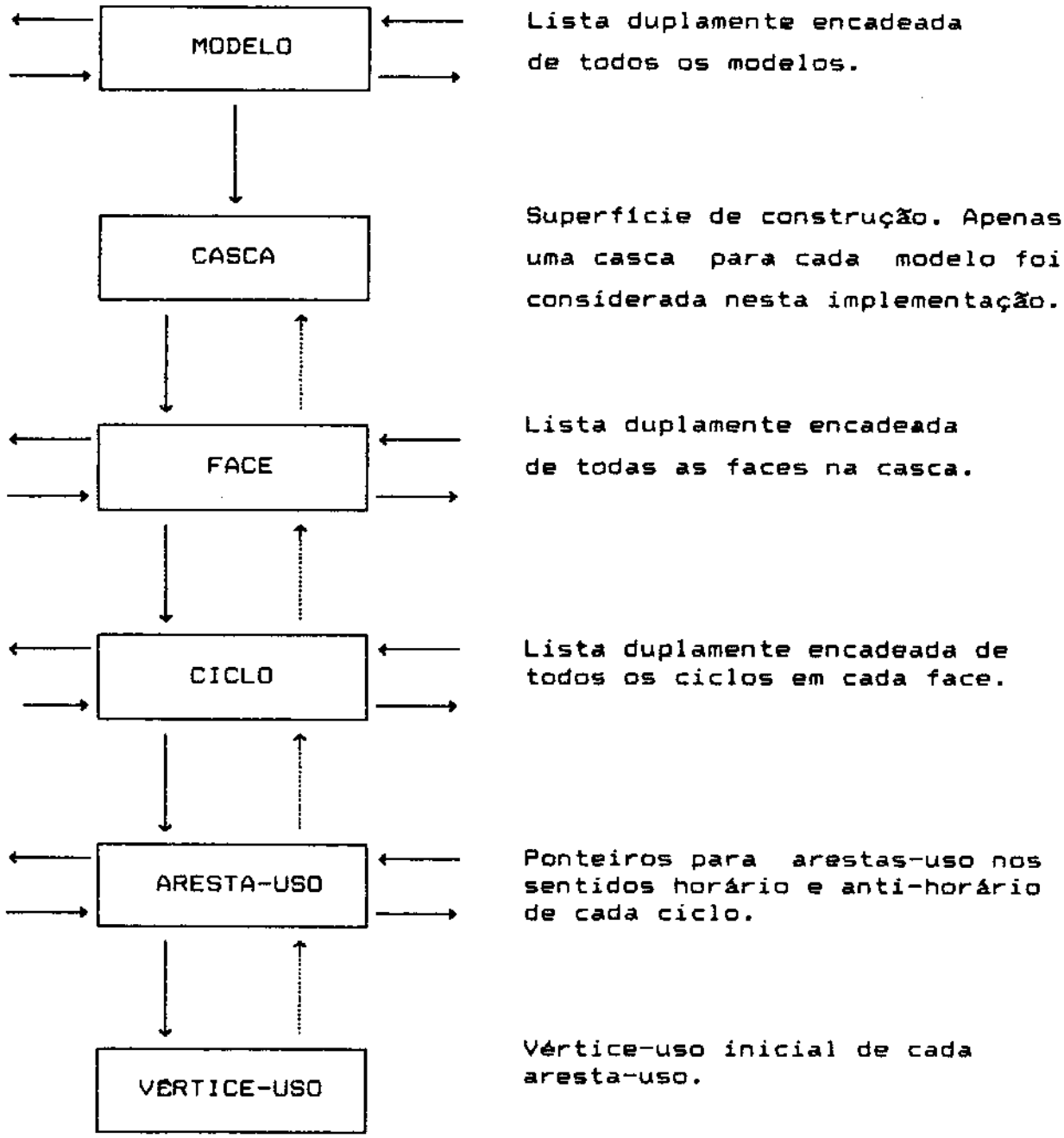


Figura 2.8 - Representação hierárquica das entidades topológicas na FEDm.

Para a descrição dos registros dos dados, uma convenção para a denominação dos campos nos registros das entidades topológicas é adotada. Esta convenção, sugerida por Weiler [WEIL86], é basicamente a seguinte :

a) As entidades topológicas se referenciam entre si através de ponteiros. O nome desses campos são assim descritos:

(da entidade tipo)(para entidade tipo)_ptr

b) Listas duplamente encadeadas de entidades de menor dimensão, mantidas por entidades de dimensão superior, são assim descritas :

(tipo de dimensão superior)(tipo de dimensão inferior)_next(last)

c) Informações não topológicas (geometria p.ex.) são armazenadas em áreas específicas, denominadas áreas de atributos. Os campos que apontam para estas áreas foram denominados de a_ptr.

Por uma questão de uniformidade e prevendo futuras implementações, manteve-se a representação dos campos para área de atributos em todos os registros, mesmo nos quais não se fazia necessário o armazenamento de informações adicionais.

2.2.2.1 - REGISTRO DE DADOS DO MODELO

O registro da entidade modelo (Figura 2.9) contém ponteiros para o modelo anterior, posterior e para sua respectiva casca (ms_ptr).

m_next	ponteiro para o modelo anterior na lista.
m_last	ponteiro para o modelo posterior na lista.
ms_ptr	ponteiro para a casca de cada modelo.
me_ptr	ponteiro para a lista de arestas do modelo.
mv_ptr	ponteiro para a lista de vértices do modelo.
a_ptr	ponteiro para área de atributos do modelo.

Figura 2.9 - Registro da entidade modelo.

Esta entidade aponta ainda para uma lista duplamente encadeadas de arestas (me_ptr) e vértices (mv_ptr) (Figura 2.10). Estas relações são particularmente interessantes, pois permitem, entre outras coisas, executar de maneira rápida e eficiente a atualização do desenho do modelo.

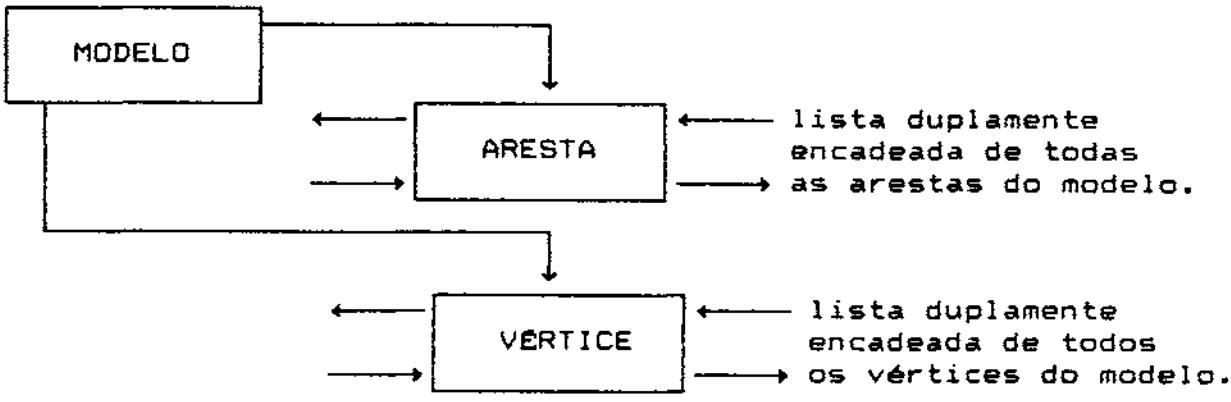


Figura 2.10 - Relações de adjacência modelo-aresta e modelo-vértice.

2.2.2.2 - REGISTRO DE DADOS DA CASCA

De acordo com a estrutura hierárquica da Figura 2.8, a casca aponta para uma lista de faces no modelo (sf_ptr). Por conveniência da implementação, a face livre é também armazenada no registro da casca (free_face). O registro de dados da entidade casca está mostrado na Figura 2.11.

sf_ptr	ponteiro para a lista de faces da casca.
free_face	ponteiro para a face livre.
a_ptr	ponteiro para área de atributos da casca.

Figura 2.11 - Registro da entidade casca.

2.2.2.3 - REGISTRO DE DADOS DA FACE

As faces estão, também, armazenadas em listas duplamente encadeadas. Cada face aponta para a casca à qual pertence (fs_ptr) e para a lista de ciclos que as delimitam (fl_ptr). O registro de dados de uma face está mostrado na Figura 2.12.

fs_ptr	ponteiro para a casca.
sf_next	ponteiro para a face anterior na lista.
sf_last	ponteiro para a face posterior na lista.
fl_ptr	ponteiro para a lista de ciclos da face.
a_ptr	ponteiro para área de atributos da face.

Figura 2.12 - Registro da entidade face.

Exemplos das relações de adjacências entre faces e ciclos estão mostrados na Figura 2.13.

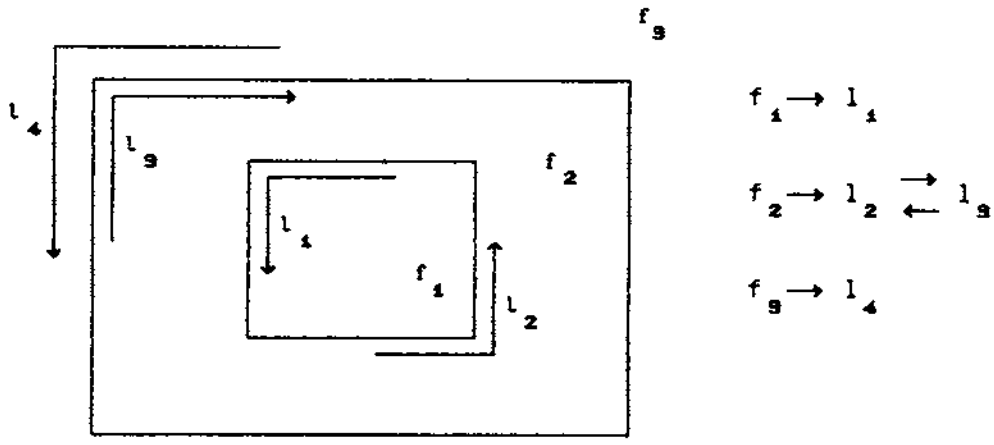


Figura 2.13 - Relação de adjacência face-ciclo.

Informações adicionais, como por exemplo se a face será considerada um furo, quais são as suas coordenadas mínimas e máximas e o material especificado para cada face, estão armazenados na área de atributos.

2.2.2.4 - REGISTRO DE DADOS DO CICLO

Continuando na hierarquia global da estrutura FEDm (Figura 2.8), um ciclo, por sua vez, aponta para a face a que pertence (lf_ptr) e para uma das arestas-uso que compõem o circuito fechado que o define (leu_ptr). O registro do ciclo está mostrado na Figura 2.14.

O caso especial de um ciclo degenerado em um vértice é também considerado. Neste caso, o ciclo aponta para um vértice-uso, que é o único uso do vértice correspondente (lvu_ptr).

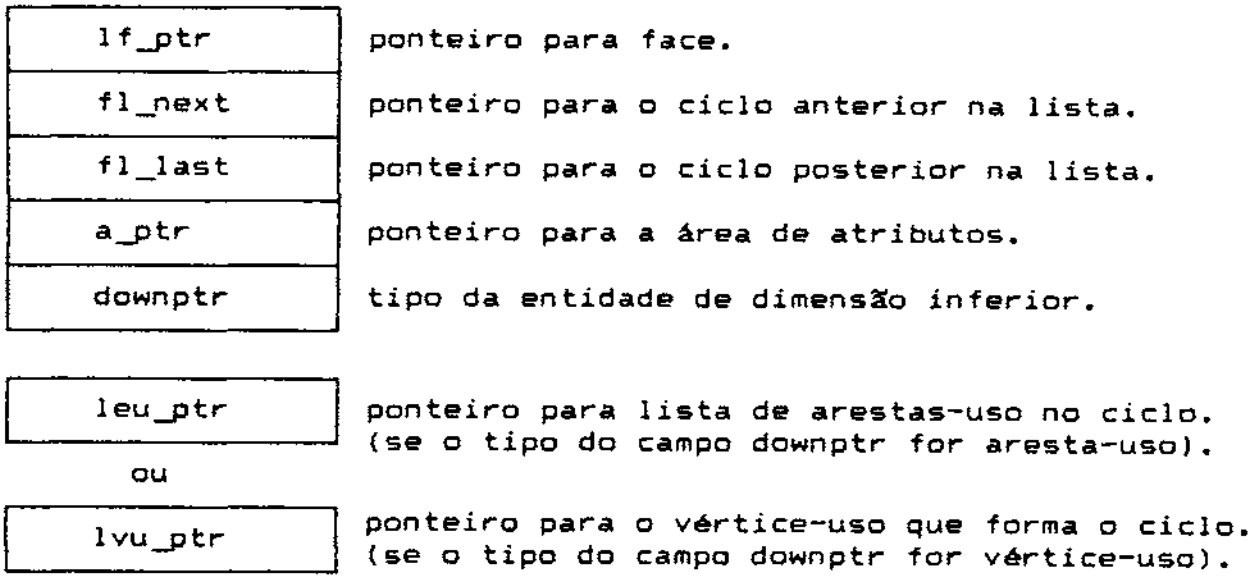


Figura 2.14 - Registro da entidade ciclo.

2.2.2.5 - REGISTRO DE DADOS DA ARESTA-USO

Seguindo a estrutura hierárquica global da Figura 2.8 encontra-se a entidade aresta-uso. O registro de dados desta entidade está mostrado na figura 2.15.

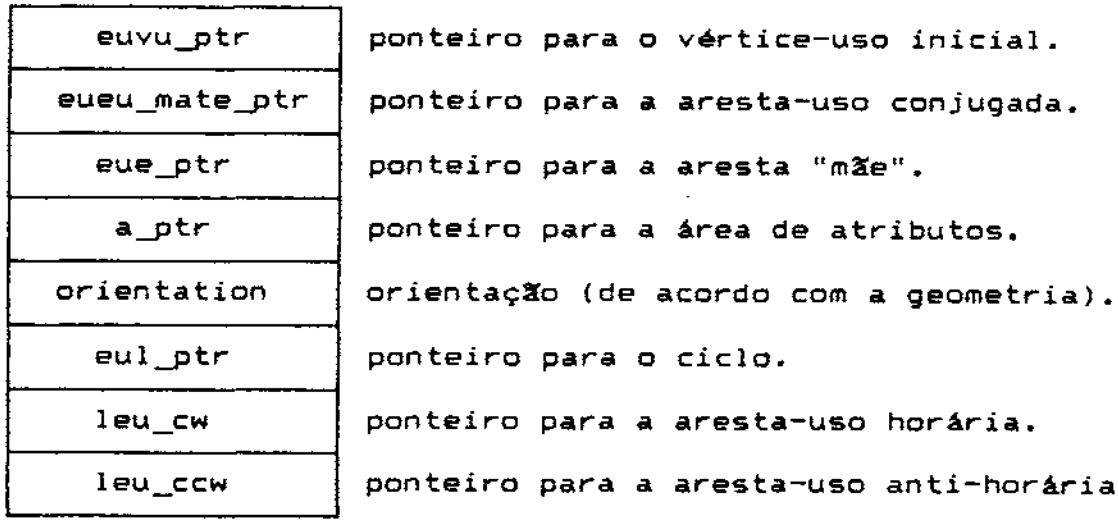
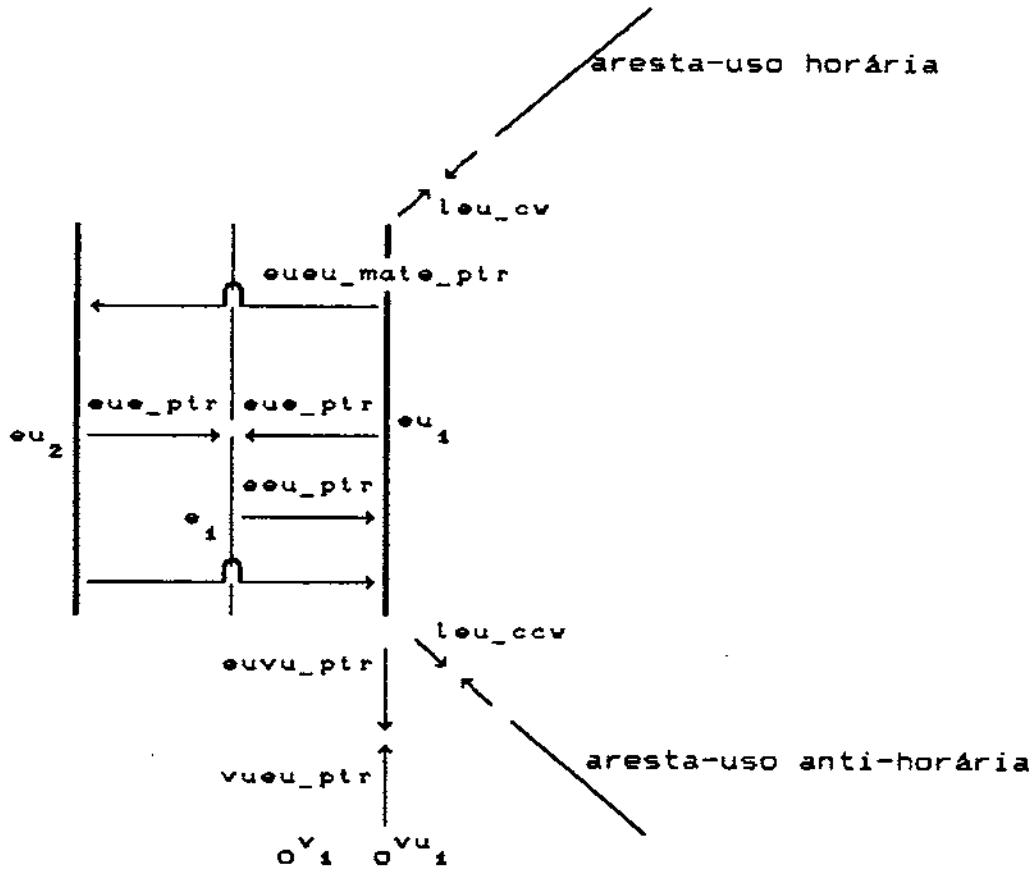


Figura 2.15 - Registro da entidade aresta-uso.



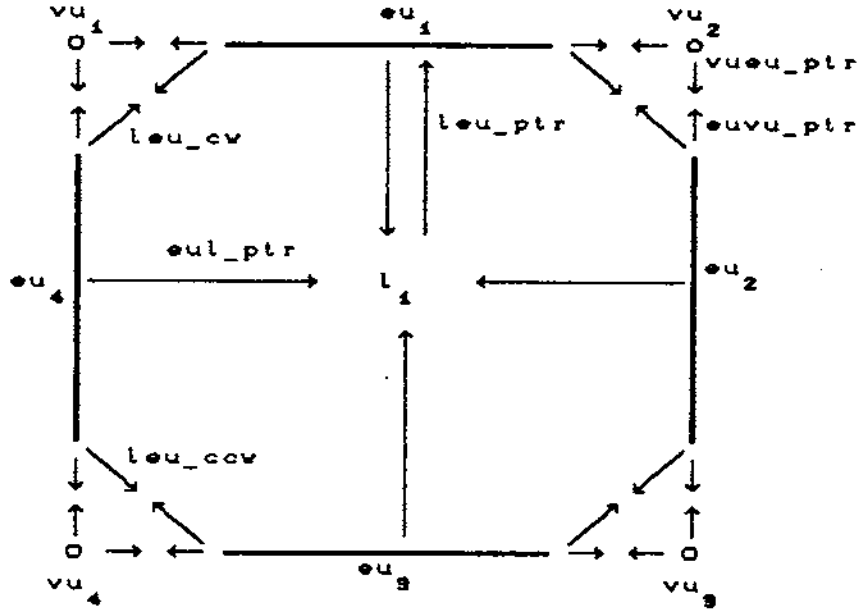


Figura 2.16 - Relações de adjacências da aresta-uso.

A aresta-uso contém ponteiros para o vértice-uso inicial (euvu_ptr), o ciclo a que pertence (eul_ptr), a aresta que a originou (eue_ptr), aresta-uso conjugada (eueu_mate_ptr) e as arestas-uso adjacentes nos sentidos horário (leu_cw) e anti-horário (leu_ccw). A Figura 2.16 ilustra estas relações de adjacência.

Entende-se como aresta-uso conjugada a entidade que juntamente com a referida aresta-uso define uma aresta.

Existe, ainda, um campo que define a orientação da aresta-uso (orientation). Este campo especifica se o sentido que a aresta-uso é percorrida no ciclo é o mesmo que o definido para a geometria da aresta-mãe.

2.2.2.6 - REGISTRO DE DADOS DO VÉRTICE-USO

O vértice-uso está no nível mais baixo da hierárquia global da estrutura FEDm (Figura 2.8). O registro de dados está mostrado na Figura 2.17.

vu_next	ponteiro para o vértice-uso anterior na lista.
vu_last	ponteiro para o vértice-uso posterior na lista
vuv_ptr	ponteiro para o vértice "mãe".
a_ptr	ponteiro para a área de atributos.
upptr	tipo da entidade de dimensão superior.
vul_ptr	ponteiro para o ciclo definido por este vu. (se o tipo do campo upptr for ciclo).
ou	
vueu_ptr	ponteiro para a aresta-uso que gerou este vu. (se o tipo do campo upptr for aresta-uso).

Figura 2.17 - Registro da entidade vértice-uso.

Os vértices-uso estão armazenados em uma lista duplamente encadeada para cada vértice. Cada vértice-uso aponta para o vértice "mãe" (vuv_ptr) e para a aresta-uso que o originou (vueu_ptr) (Figura 2.18).

O caso degenerado de um ciclo formado somente por um vértice-uso é aqui tratado (Figura 2.17), fazendo com que esta entidade aponte para o ciclo que ela define (vul_ptr).

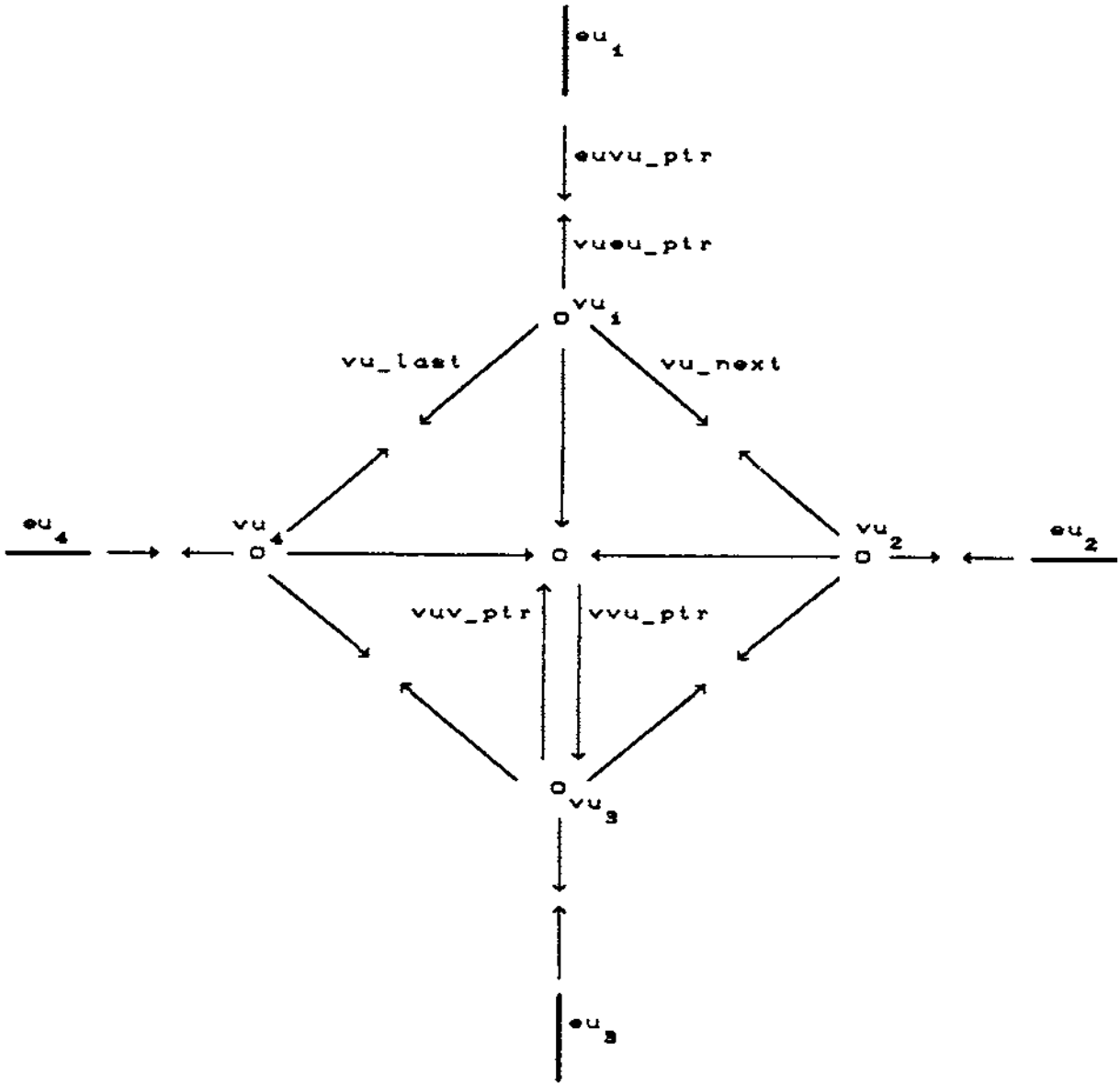


Figura 2.18 - Relações de adjacência vértice - vértice-uso e aresta - aresta-uso.

2.2.2.7 - REGISTRO DE DADOS DA ARESTA

Embora a aresta não pertença à hierarquia global da estrutura FEDm (Figura 2.8), ela também é tratada como uma entidade topológica da estrutura. Nesta implementação, a principal função do elemento topológico aresta é interfacear informações geométricas.

O registro de dados está mostrado na FIgura 2.19. Os dois primeiros campos do registro são usados como encadeamento da lista de arestas do modelo.

Na área de atributo são armazenadas informações como tipo e descrição geométrica das arestas (linha, círculo ou curva Bezier) e ponteiros para as funções genéricas.

As funções genéricas são procedimentos que executam tarefas específicas para os diversos tipos de arestas. Estas funções serão discutidas mais detalhadamente em capítulo à parte.

e_next	ponteiro para a aresta anterior na lista.
e_last	ponteiro para a aresta posterior na lista.
eeu_ptr	ponteiro para uma das arestas-uso da aresta.
a_ptr	ponteiro para área de atributos da aresta.

Figura 2.19 - Registro da entidade aresta.

A entidade aresta guarda ainda relações de adjacência com as entidades que definem seus dois "usos". Ela aponta para uma das suas arestas-uso (eeu_ptr). Estas relações estão mostradas nas Figuras 2.16 e 2.20.

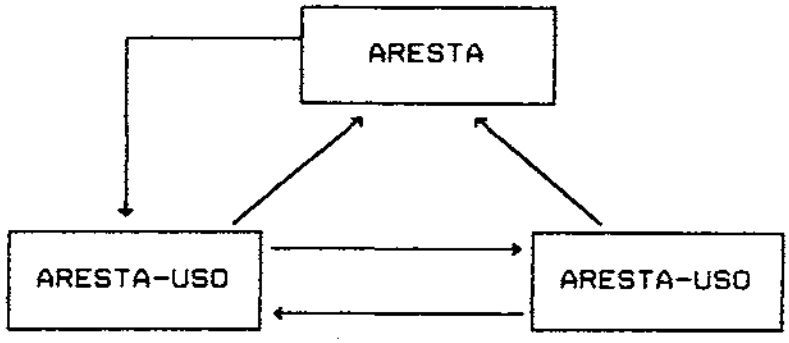


Figura 2.20 - Relação de adjacência aresta - aresta-uso.

2.2.2.8 - REGISTRO DE DADOS DO VÉRTICE

O vértice também é considerado como uma entidade topológica na estrutura FEDm. Seu registro de dados está mostrado na Figura 2.21.

v_next	ponteiro para o vértice anterior da lista
v_last	ponteiro para o vértice posterior na lista
vvu_ptr	ponteiro para um dos vértices-uso do vértice
a_ptr	ponteiro para área de atributos do vértice

Figura 2.21 - Registro da entidade vértice.

Os dois primeiros campos são utilizados como encadeamento da lista de vértices do modelo. A área de atributos, apontada pelo campo a_ptr, é utilizada, por exemplo, para armazenar as coordenadas do vértice.

O vértice também guarda relações de adjacência com seus "usos". Ele aponta, através do campo vvu_ptr (Figura 2.21), para uma lista de vértices-uso que o define. Estas relações

estão ilustradas nas Figuras 2.18 e 2.22.

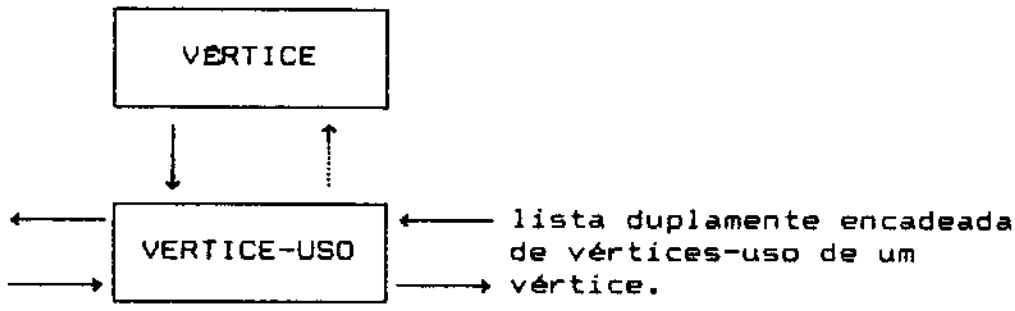


Figura 2.22 - Relação de adjacência vértice - vértices-uso.

CAPÍTULO 3

DECOMPOSIÇÃO HIERÁRQUICA DO DOMÍNIO

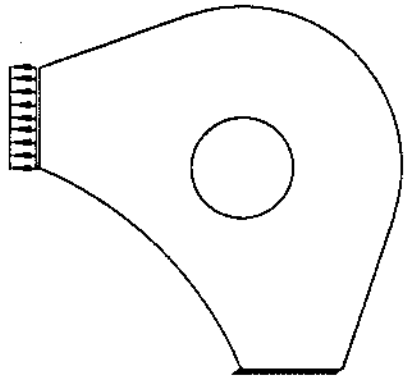
A idéia central do corrente programa de geração de malhas para elementos finitos era a de desenvolver uma ferramenta gráfica e interativa que, através de etapas distintas e bem definidas, fosse capaz de oferecer um ambiente didático e amigável mesmo a usuários pouco experimentados.

Voltado, essencialmente, para a formação acadêmica prática dos estudantes do método dos elementos finitos, o referido programa constitui, ainda, uma poderosa ferramenta para usuários mais experientes.

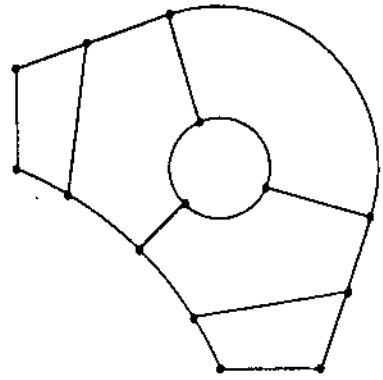
O suporte necessário para a manutenção de um ambiente deste tipo é dado pela já discutida estrutura de dados FEDm, associada a uma subdivisão do domínio em modelos distintos. Cada modelo é, agora, responsável por uma série de procedimentos específicos no processo. Assim, como primeiro passo, tem-se que definir a geometria do problema e, através de uma sequência natural de tarefas, evolui-se, gradativamente, até a obtenção do modelo discretizado em elementos finitos.

Existe na presente implementação a consideração de quatro modelos distintos (Figura 3.1), que serão discutidos em detalhes nas seções subseqüentes. Estes modelos estão associados ao procedimento adotado para o processo de geração

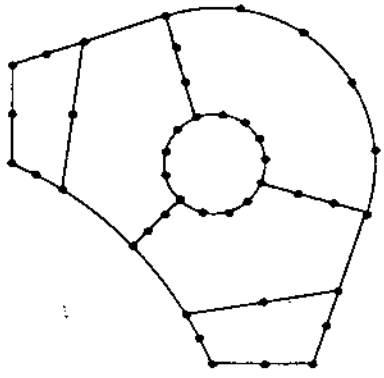
de malha: definição da geometria do problema, decomposição em sub-regiões consistentes com os algoritmos de geração de malhas desejados, discretização dos contornos dessas regiões que definem o grau de refinamento das malhas a serem geradas e, finalmente, discretização do modelo em elementos finitos.



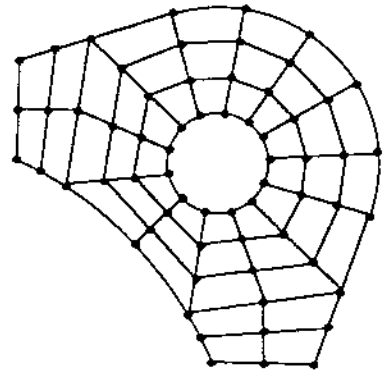
a) modelo da geometria



b) modelo da sub-região



c) modelo da sub-divisão



d) modelo da malha

Figura 3.1 - Representação dos modelos.

3.1 - DISPOSIÇÃO HIERÁRQUICA DOS MODELOS

Os modelos estão organizados em uma representação tipo "de cima para baixo" (*top-down*) (Figura 3.2).

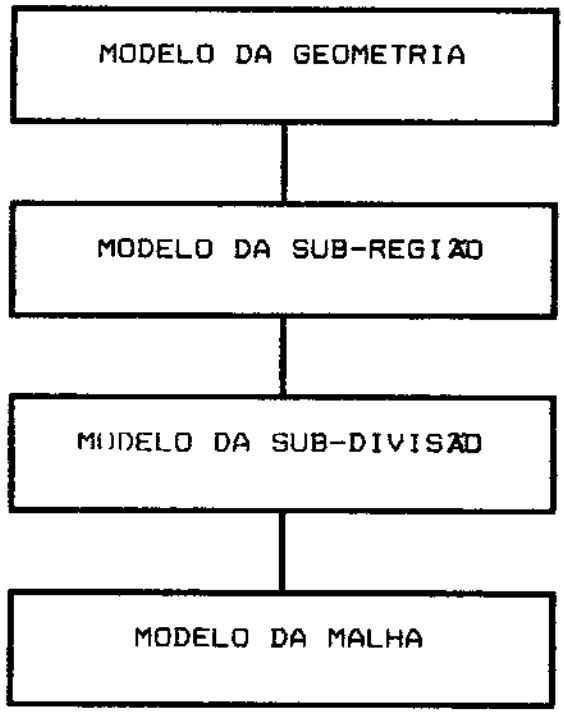


Figura 3.2 - Representação hierárquica dos modelos.

Cada modelo é descrito com uma estrutura de dados FEDm independente. Este fato permite que procedimentos específicos, associados a cada modelo, sejam sempre executados de maneira rápida e consistente.

Neste tipo de representação, os modelos são limitados pelos modelos hierarquicamente superiores. Assim, quando um elemento topológico é adicionado a um modelo ou eliminado de um modelo, todos os modelos, hierarquicamente abaixo, criam ou eliminam, automaticamente, uma instância deste elemento.

Não é permitida a eliminação de elementos que tenham sido gerados em níveis superiores. Por exemplo, uma aresta criada no nível da geometria será automaticamente reproduzida nos demais níveis. Esta aresta, contudo, não pode ser eliminada no nível da sub-região ou da malha, pois tem uma instância em um nível hierárquicamente superior.

Se, ainda, a introdução de um novo elemento em um dado nível gera uma inconsistência topológica nos níveis inferiores, estes níveis são localmente alterados, de maneira a acomodar a introdução do novo elemento. Por exemplo, se o usuário decide adicionar uma aresta no nível da sub-região com a finalidade de dividí-la em duas e tentar melhorar a malha gerada, esta aresta será introduzida nos níveis inferiores e a malha da sub-região inicial será automaticamente eliminada.

Outro aspecto de grande importância nesta implementação é o fato de se ter uma descrição do modelo baseada na sua geometria. Assim, uma vez modelada a geometria do problema, definidos os carregamentos, condições de apoio, propriedades dos materiais ou quaisquer outros tipos de atributos físicos, o usuário não necessita redefinir os referidos atributos toda vez que houver alterações nos níveis inferiores.

O nível de abstração conseguido neste ambiente, além de facilitar a definição destes atributos, permite, ainda, que os carregamentos e condições de contorno tenham suas direções consistentemente definidas de acordo com a descrição geométrica do modelo.

Este tipo de ambiente representa, ainda, uma evolução sobre programas de geração de malhas convencionais, onde os

atributos físicos são especificados a nível de cada elemento finito ou nó da malha e a descrição geométrica do modelo está implicitamente definida através da discretização numérica do mesmo.

3.1.1 - MODELO DA GEOMETRIA

A descrição geométrica do modelo está armazenada no nível da geometria. Como as informações armazenadas neste nível independem da configuração discretizada em elementos finitos, não há necessidade de redefiní-las durante o processo de geração de malhas. Estas informações só são perdidas quando ocorre alguma alteração na geometria do problema, e, ainda assim, somente nas regiões afetadas pela modificação.

A definição da geometria é feita através da adição e eliminação de arestas. Este processo é feito, simultaneamente, em todos níveis hierárquicos.

A adição de uma aresta é feita de maneira simples e interativa. A definição de cada aresta está relacionada com o seu tipo geométrico : reta, círculo ou curva do tipo Bezier.

Para a definição de uma reta é necessária, somente, a especificação dos seus vértices extremos. O arco de círculo necessita, como informação adicional, a posição do centro do círculo que o define. Já, para o círculo completo é necessário, somente, especificar o centro e um ponto sobre o mesmo.

A representação da curva Bezier, por ser uma curva cúbica, necessita a definição de quatro pontos. A definição da curva pode ser feita de duas maneiras distintas, através dos pontos de controle ou de pontos de interpolação.

Cada região fechada (face) deste modelo pode ter um tipo de material associado ou um indicador se a mesma é considerada como um furo no modelo.

Outras informações típicas deste nível, como condições de apoio e carregamentos, não foram implementadas na versão atual do programa.

3.1.2 - MODELO DA SUB-REGIÃO

O objetivo deste modelo é definir as regiões onde distintos algoritmos de geração de malhas serão aplicados. A vantagem de se trabalhar com uma estrutura de dados topológicas é evidente neste caso. A topologia das regiões para geração de malhas é facilmente identificada, pois estas regiões são as faces de um modelo topológico. Desta forma, a consistência entre os algoritmos de geração de malhas e a topologia das regiões pode ser eficientemente testada.

A definição de sub-regiões tem procedimentos semelhantes aos do nível da geometria. Este nível suporta a representação de todos os tipos geométricos de arestas presentes na geometria e suas definições utilizam a mesma interface com o usuário.

O nível da sub-região, contudo, é mais flexível do que o da geometria, no que se refere à manipulação destas arestas, pois permite que as arestas já definidas possam ser divididas em dois ou mais segmentos, mesmo que tenham sua instância definida no nível da geometria.

A divisão de uma aresta pode ser feita diretamente pelo usuário através da especificação de um ponto sobre a mesma ou, ainda, quando uma nova aresta é adicionada ao modelo e tem um

dos seus vértices suficientemente próximo da aresta que é dividida. Em ambos os casos, será criado um novo vértice que será a *fronteira comum* à aresta original e à aresta resultante da divisão da aresta original.

A função de dividir uma aresta tem a sua inversa. A fusão de duas ou mais arestas obtidas por um processo de divisão pode, igualmente, ser feita diretamente pelo usuário ou, de forma automática, com a eliminação da aresta que causou a divisão.

Faz-se, aqui, uma descrição dos aspectos topológicos envolvidos no processo de divisão e fusão de arestas. Os aspectos geométricos serão discutidos no capítulo seguinte. As informações topológicas, necessárias para a implementação destes procedimentos, estão armazenadas na área de atributos dos vértices criados através de um processo de divisão de arestas (Figura 3.3). Somente os vértices da sub-região possuem informações deste tipo.

O campo *split_flag* indica se este vértice foi criado devido à divisão de uma aresta. Os campos *se0_ptr* e *se1_ptr* apontam para as duas arestas resultantes desta operação.

Assim, toda vez que uma aresta é adicionada ao modelo no nível da sub-região, seus pontos extremos são testados contra as demais arestas. Se pelo menos um destes pontos estiver suficientemente próximo a uma aresta, será criado um novo vértice nesta posição e a referida aresta será então dividida (Figura 3.4-b). O campo *split_flag* é marcado como verdadeiro e os campos *se0_ptr* e *se1_ptr* carregados com os endereços das arestas resultantes.

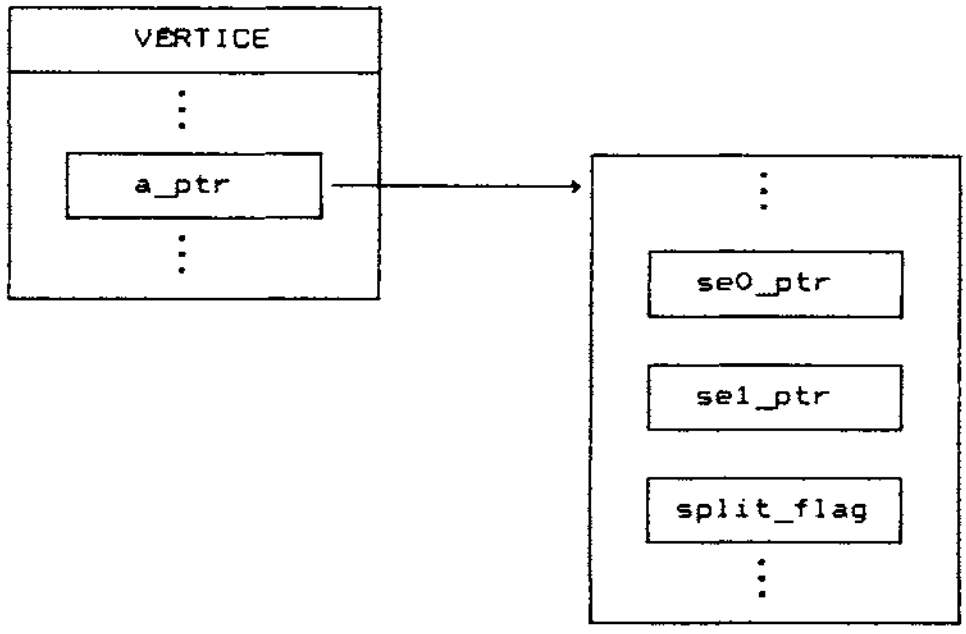


Figura 3.3 - Atributos associados a vértices criados em processo de divisão de arestas.

No caso da eliminação de uma aresta, cada um de seus vértices é testado para saber se o campo *split_flag* associado a ele é verdadeiro. Em caso positivo, a referida aresta é comparada com as arestas originárias da divisão causada por este vértice, cujos ponteiros estão nos campos *se0_ptr* e *se1_ptr*. Se a aresta a ser eliminada corresponde a uma destas arestas, ela é eliminada, e o campo *split_flag* do vértice marcado como falso. Se a aresta a ser eliminada não corresponder a nenhuma das arestas apontadas pelo vértice, e constatando-se que, além da própria aresta, só existem duas outras arestas adjacentes a este vértice, estas arestas serão fundidas e o vértice eliminado (Figura 3.4-c). Caso existam mais de duas arestas adjacentes a este vértice, o processo de fusão, e a consequente eliminação do vértice, não pode ser

executado, sob pena de se gerar uma inconsistência topológica (Figura 3.4-d).

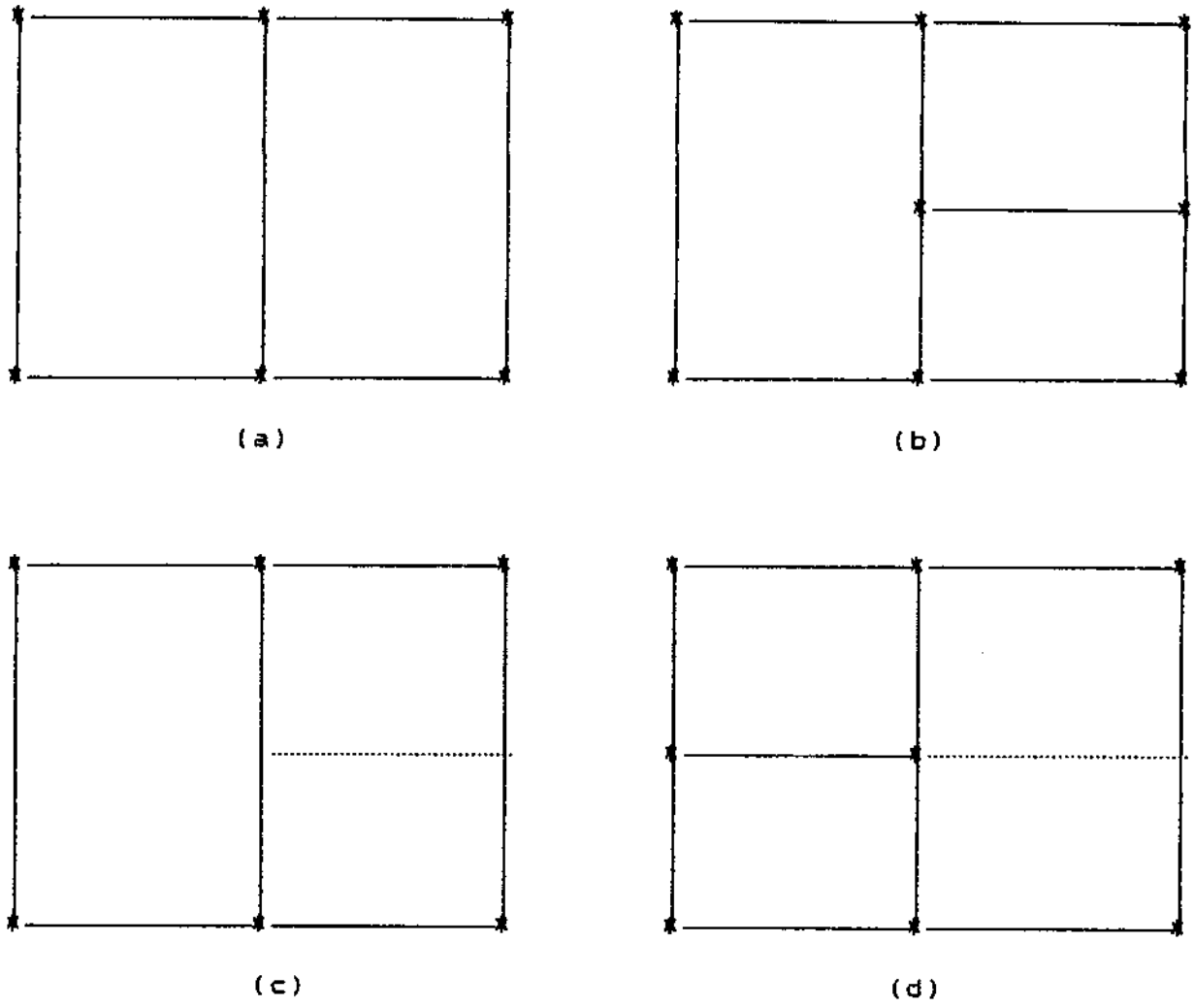


Figura 3.4 - Divisão e fusão automática de arestas.

3.1.3 - MODELO DA SUB-DIVISÃO

Após a definição das sub-regiões, e antes de se iniciar o processo de discretização em elementos finitos, é necessário especificar o número de segmentos em que os contornos das sub-regiões serão divididos. Esta tarefa é executada no nível da sub-divisão.

A implementação presente permite a representação de dois tipos de subdivisões: linear e quadrática.

No caso de uma subdivisão linear cada segmento corresponde a uma aresta neste nível e ao lado de um elemento finito, igualmente linear, no nível da malha.

Na subdivisão quadrática os segmentos representam duas arestas de igual comprimento no nível da sub-divisão, podendo representar no nível da malha tanto o lado de um elemento finito quadrático como os lados de dois elementos finitos lineares.

As divisões ao longo de uma aresta não precisam ser necessariamente iguais. O usuário pode especificar uma gradação ao longo da mesma. Para isso, basta que se forneça a razão entre os comprimentos do primeiro e do último segmentos. A diferença entre segmentos consecutivos obedece então a uma progressão aritmética, ou seja, a diferença varia linearmente ao longo da aresta.

Em contraste com os níveis superiores, os níveis da sub-divisão e malha só suportam a representação de arestas retas. Assim, quando uma aresta curva é adicionada no nível da geometria ou sub-região, o programa reproduz, automaticamente, esta aresta nos níveis inferiores como uma seqüência de arestas retas. Como não se conhece, *a priori*, o número de subdivisões associado a cada aresta da sub-região, estima-se um valor inicial em função do seu tipo geométrico.

Este valor no caso de uma aresta reta é um. Para o arco de círculo ele varia de acordo com o ângulo que as retas que passam pelo centro do círculo e seus pontos extremos fazem

entre si. No caso da curva Bezier, este número é função da razão entre os comprimentos da curva e da reta que une seus pontos extremos.

3.1.4 - MODELO DA MALHA

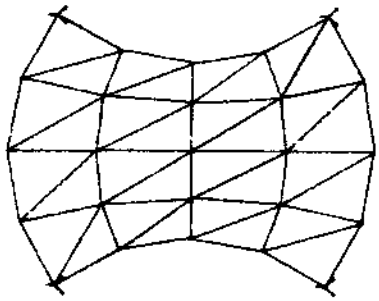
Este modelo representa a discretização do domínio de modelagem em elementos finitos. Cada face, exceto as faces definidas como furos na geometria e a face externa livre, corresponde a um elemento finito.

O fato deste nível ter o suporte de uma estrutura de dados hierárquica facilita a implementação de um ambiente baseado na geometria. Como o nível da malha está hierarquicamente subordinado ao nível da geometria, todas as informações necessárias a uma discretização numérica são consistentemente definidas de acordo com a geometria do problema.

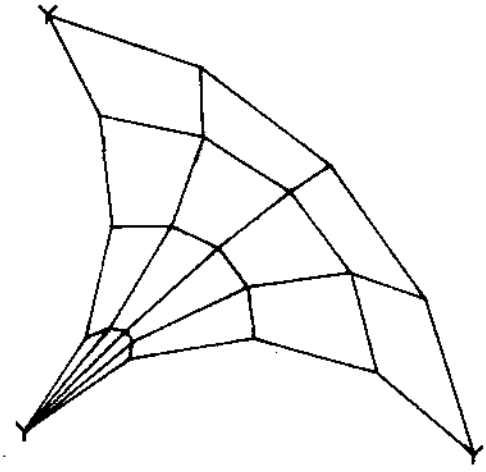
Para a geração de malhas foram implementados três algoritmos de mapeamento transfinito [HABE81] e um algoritmo de triangularização genérica [SHAW78].

Os algoritmos de mapeamentos transfinitos são: mapeamento bilinear, mapeamento bilinear com um lado degenerado para um ponto e mapeamento trilinear (Figura 3.5 a,b e c).

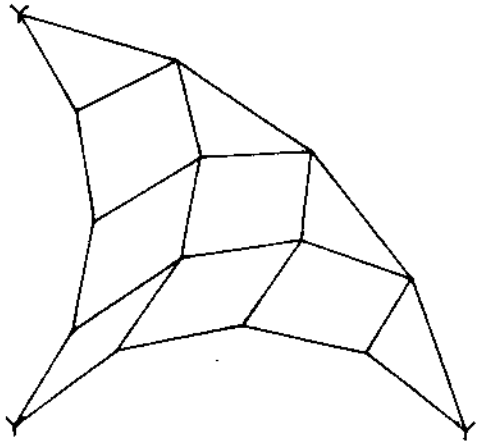
A Figura 3.5-d ilustra uma malha gerada através do algoritmo de triangularização genérica.



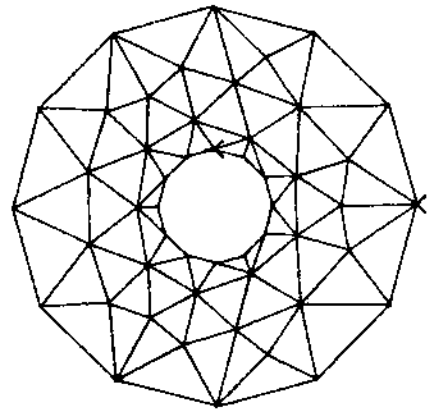
a)Mapeamento bilinear



b)Mapeamento bilinear degenerado



c)Mapeamento trilinear



d)Triangularização genérica

Figura 3.5 - Algoritmos de geração de malhas.

A principal desvantagem associada à técnica de mapeamento transfinito é a dificuldade de se definir regiões consistentes com os algoritmos. Para o mapeamento bilinear, cada região é topologicamente equivalente a um quadrilátero. No bilinear degenerado, as regiões são quadriláteros com um dos seus lados degenerados em um ponto. No mapeamento trilinear, as regiões são equivalentes a triângulos. A principal contribuição da utilização de estruturas de dados topológicas como a base para a geração de malhas está em uma maior exploração dos algoritmos de mapeamento. Um exemplo disto é a facilidade com que sub-regiões com topologias equivalentes a quadriláteros ou triângulos são formadas. Um outro exemplo é a facilidade com que arestas topológicas são agrupadas para formar um dos lados da região para mapeamento. Neste caso, o usuário necessita informar, adicionalmente, os vértices da região que formam os cantos do mapeamento.

A facilidade em se definir regiões, associada a alguns procedimentos que definem de forma automática os vértices de canto, forma um ambiemte bastante adequado para a implementação das técnicas de mapeamento.

A determinação automática dos vértices de canto, por envolver aspectos geométricos além dos topológicos, será discutida mais detalhadamente no capítulo seguinte.

3.2 - RELAÇÕES HIERÁRQUICAS

A comunicação entre os diversos níveis hierárquicos é feita através das entidades topológicas básicas (face, aresta e vértice).

O fato de só se utilizar estas entidades, dentre os diversos elementos topológicos armazenados explicitamente na FEDm, conduz a uma interpretação física mais intuitiva das relações hierárquicas entre os modelos.

As informações hierárquicas estão armazenadas na área de atributos das respectivas entidades. Desta forma, entidades de níveis distintos se comunicam entre si, indiretamente, através de seus atributos (Figura 3.6).

Existem vantagens em se tratar as relações hierárquicas separadamente das relações de adjacências da FEDm. A principal delas é que a complexidade da manipulação da referida estrutura pode ser confinada a cada nível hierárquico, pois para uma representação consistente de cada modelo suas entidades não precisam tomar conhecimento das dependências hierárquicas. Assim, os procedimentos associados à manipulação da estrutura FEDm podem ser usados em todos os níveis indistintamente.

Uma peculiaridade desta representação hierárquica é que cada nível só se comunica com os dois níveis adjacentes acima e abaixo na hierarquia. Este fato simplifica a manipulação dos ponteiros hierárquicos e facilita a modularização do programa. Por outro lado, quando se deseja obter alguma informação armazenada em um nível não adjacente, tem-se que percorrer toda a hierarquia até o referido nível para só então se obter a informação desejada.

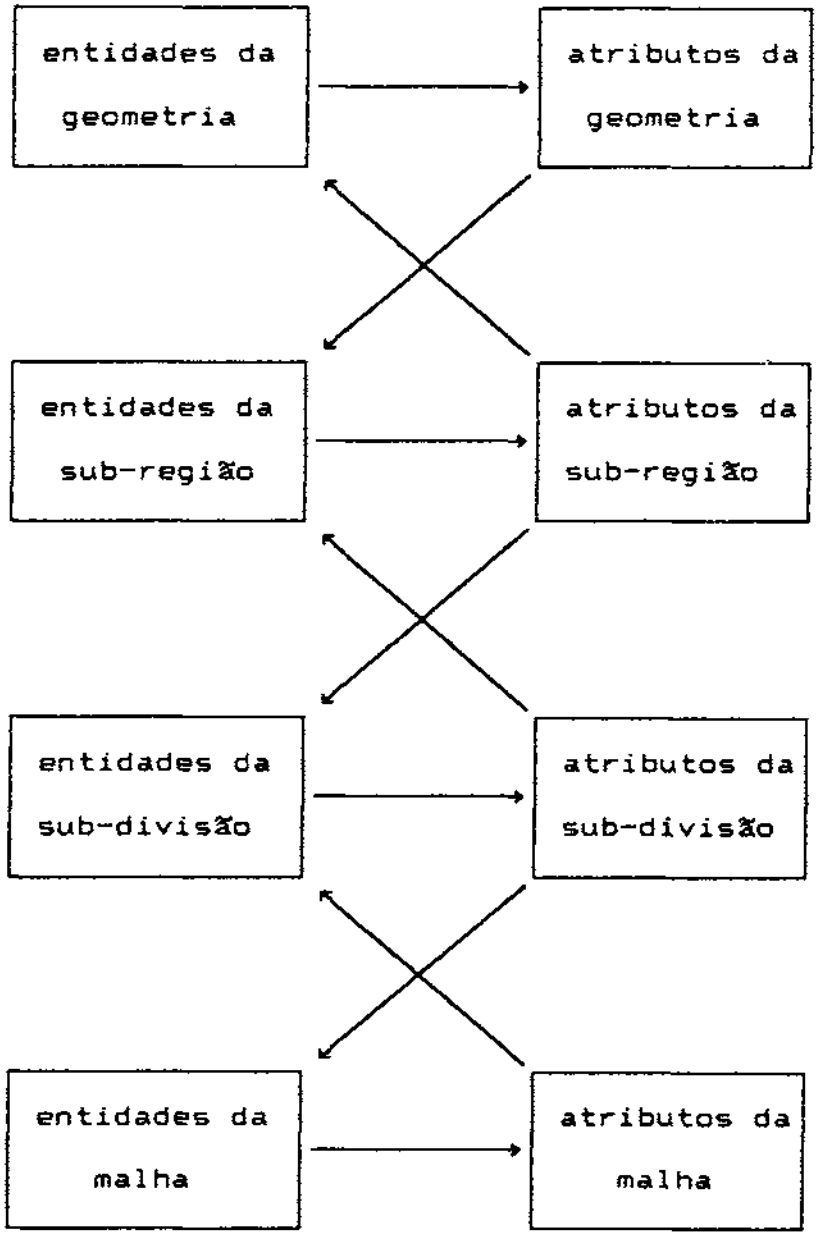


Figura 3.6 - Esquema da comunicação hierárquica.

3.2.1 - ATRIBUTOS HIERÁRQUICOS DOS ELEMENTOS TOPOLÓGICOS

Os atributos hierárquicos variam de acordo com a entidade e o nível a que pertencem.

Existem basicamente dois campos na área de atributos responsáveis pela manutenção da estrutura hierárquica. O

primeiro campo faz a relação com o nível inferior. O segundo faz relação com o nível superior. Estes campos são ponteiros para as entidades topológicas dos níveis adjacentes na hierárquia.

Existem dois conceitos associados a definição destes campos. Estes conceitos têm origem na concepção "de cima para baixo" da estrutura hierárquica, descritas na seção 3.1. São eles:

- Um elemento topológico básico de um dado nível hierárquico sempre aponta para um elemento topológico do mesmo tipo no nível imediatamente abaixo.

- Um elemento topológico básico de um dado nível hierárquico só pode apontar para um elemento de no "mínimo" o mesmo tipo topológico no nível imediatamente acima.

Neste contexto, as entidades topológicas são classificadas na seguinte ordem: vértice, aresta e face. Assim, um vértice em um determinado nível pode corresponder a um vértice, uma aresta ou uma face no primeiro nível acima; uma aresta só pode corresponder a uma aresta ou uma face; e uma face só pode corresponder a uma face.

Como os campos que apontam para o nível superior na hierarquia podem se referenciar a mais de um tipo de elemento, é necessária a manutenção de um campo que indique o tipo deste elemento. Este campo foi aqui denominado de *upptr*.

Como consequência do primeiro conceito acima, o vértice de um dado nível, com excessão do vértice da malha, sempre aponta para a sua instância no nível imediatamente abaixo.

Algumas entidades topológicas não necessitam armazenar,

explicitamente, os dois ponteiros hierárquicos, pois esta representação introduz informações redundantes à aplicação.

A determinação dos ponteiros hierárquicos é feita da seguinte forma:

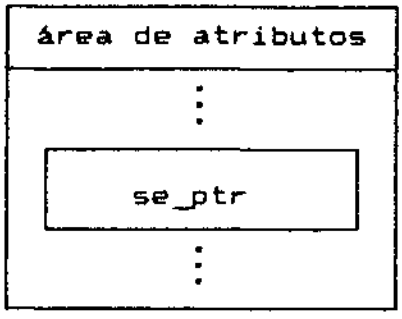
(para o nível tipo)(para a entidade tipo)_ptr

Os tipos de níveis são g (de Geometria), s (de Sub-região), v (de sub-diVisão) e m (de Malha). Os tipos de entidades são f (de Face), e (de aresta - *Edge*) e v (de Vértice).

3.2.1.1 - ATRIBUTOS DA GEOMETRIA

A aresta aponta para a sua primeira "filha" no nível da sub-região (Figura 3.7). Uma aresta da geometria possui mais de uma aresta associada a ela na sub-região, quando sua instância, *no referido nível, sofre um processo de divisão.*

A face da geometria não possui nenhum ponteiro hierárquico, e o atributo hierárquico do vértice da geometria está mostrado na Figura 3.8.



Ponteiro para a primeira aresta "filha" da sub-região.

Figura 3.7 - Atributo hierárquico da aresta da geometria.

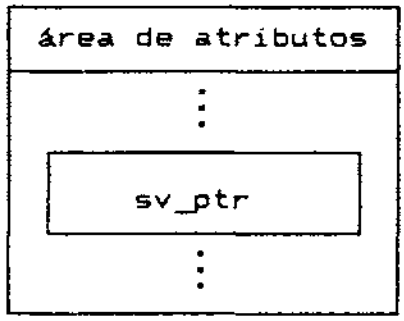


Figura 3.8 - Atributo hierárquico do vértice da geometria.

3.2.1.2 - ATRIBUTOS DA SUB-REGIÃO

A face da sub-região aponta, para cima, para a sua face "mãe" e, para baixo, para sua "filha" na sub-divisão (Figura 3.9). Só pode existir uma face "filha" pois, como somente as arestas do nível da sub-região são divididas no nível da sub-divisão, não existem novas faces no nível da sub-divisão.

A aresta aponta, no nível inferior, para sua primeira "filha", resultande do processo de sub-divisão desta aresta.

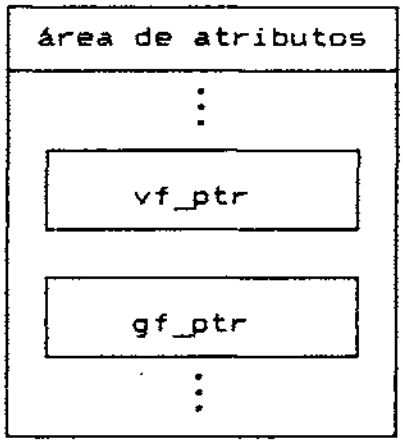


Figura 3.9 - Atributo hierárquico da face da sub-região.

Para o nível superior, esta aresta pode apontar para uma aresta da geometria que a originou ou para a face que a contém, desde que ela não tenha uma instância no nível da geometria (Figura 3.10). No primeiro caso o indicador *upptr* é igual a "aresta" e no segundo *upptr* é igual a "face".

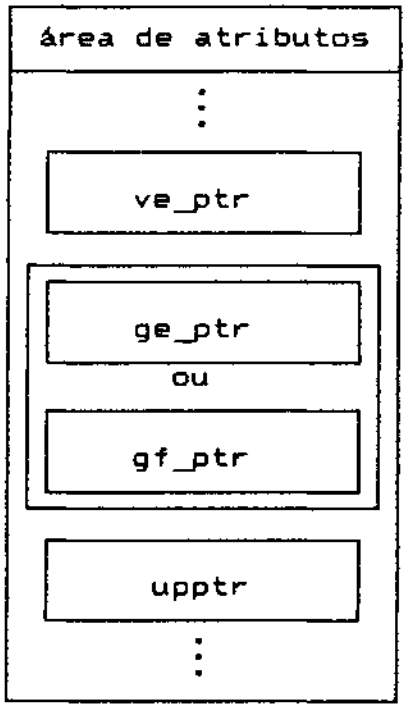
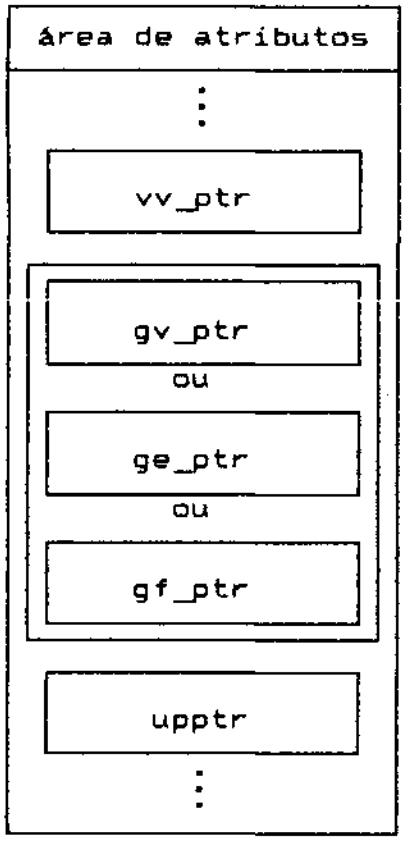


Figura 3.10 - Atributo hierárquico da aresta da sub-região.

O vértice da sub-região pode apontar, para cima, para três tipos de elementos. Se o vértice possui uma instância na geometria, ele aponta para este vértice. Se a origem do vértice está relacionada com a divisão de uma aresta que tenha uma instância na geometria, então ele aponta para a referida aresta. Em qualquer outro caso o vértice aponta para a face geométrica que o contém (Figura 3.11).



Ponteiro para o vértice "filho" na sub-divisão.

Ponteiro para o vértice "mãe" na geometria (se o campo upptr for vértice).

Ponteiro para a aresta "mãe" na geometria. (se o campo upptr for aresta).

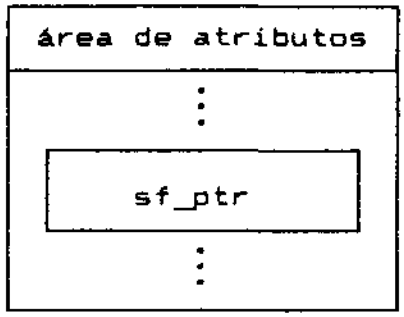
Ponteiro para a face "mãe" na geometria. (se o campo upptr for face).

Indicador do tipo topológico a que esta entidade aponta (vértice, aresta ou face).

Figura 3.11 - Atributo hierárquico do vértice da sub-região.

3.2.1.3 - ATRIBUTOS DA SUB-DIVISÃO

A face da sub-divisão só contém ponteiro para a sua face "mãe" na sub-região (Figura 3.12).



Ponteiro para a face "mãe" na sub-região.

Figura 3.12 - Atributo hierárquico da face da sub-divisão.

A aresta aponta, para cima, para a aresta que a originou, e, para baixo, para a sua instância na malha (Figura 3.13).

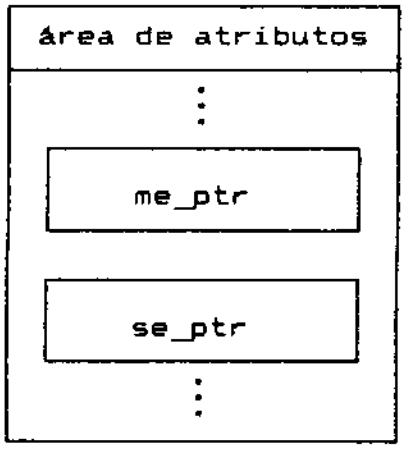


Figura 3.13 - Atributo hierárquico da aresta da sub-divisão.

O vértice pode apontar para cima para duas entidades distintas. Se ele tem uma instância no nível da sub-região, então apontará para este vértice. Se, contudo, este vértice foi criado através de um processo de subdivisão de uma aresta da sub-região, ele apontará para esta aresta (Figura 3.14).

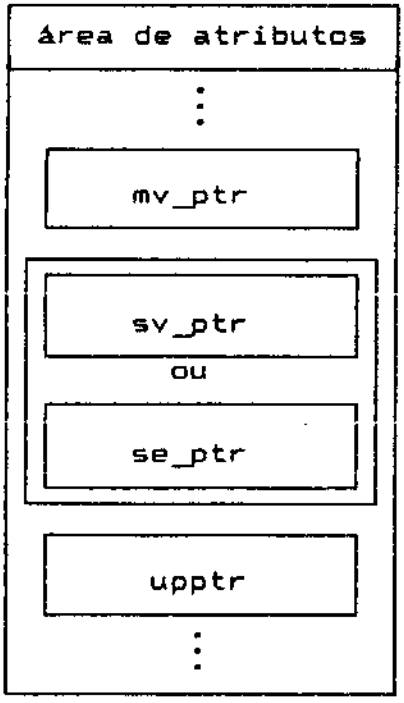
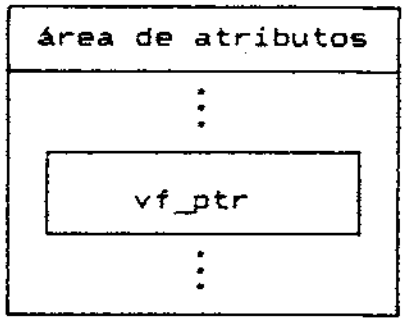


Figura 3.14 - Atributo hierárquico do vértice da sub-divisão.

3.2.1.4 - ATRIBUTOS DA MALHA

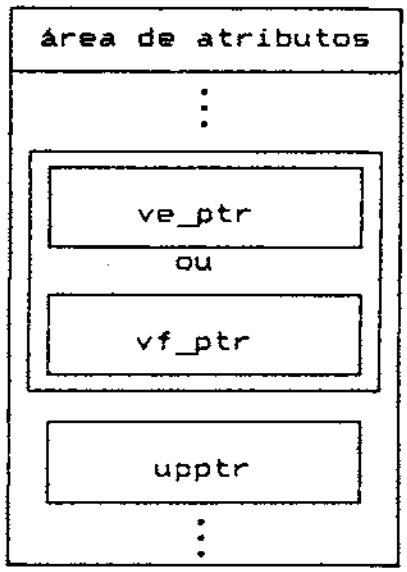
A face do nível da malha corresponde a um elemento finito. Ela aponta para cima para a face da sub-divisão que a contém (Figura 3.15).



Ponteiro para a face "mãe" na sub-divisão.

Figura 3.15 - Atributo hieráquico da face da malha.

As arestas e vértices do contorno da malha de uma sub-região apontam para suas respectivas instâncias no nível da sub-divisão. No interior da malha da sub-região estas entidades apontam para a face da sub-divisão que as contém (Figuras 3.16 e 3.17).



Ponteiro para aresta "mãe" na sub-divisão. (se o campo upptr for aresta).

Ponteiro para a face "mãe" na sub-divisão. (se o campo upptr for face).

Indicador do tipo topológico a que esta entidade aponta (aresta ou face).

Figura 3.16 - Atributo hierárquico da aresta da malha.

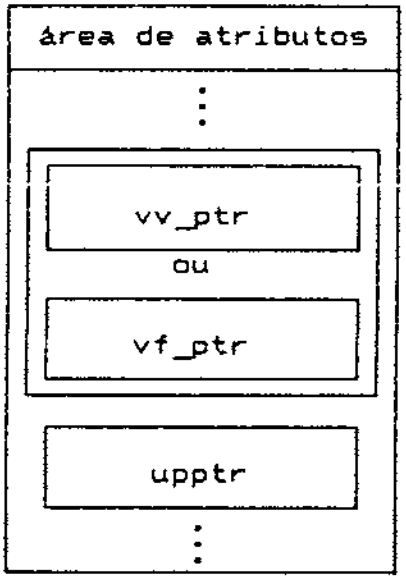


Figura 3.17 - Atributo hieráquico do vértice da malha.

3.2.1.5 - RESUMO DAS RELAÇÕES HIERÁRQUICAS

A Figura 3.18 resume as possíveis relações hierárquicas entre as entidades topológicas de todos os níveis da presente estrutura de dados. As setas para baixo indicam os ponteiros para os "filhos" no nível adjacente abaixo. As setas para cima representam os possíveis ponteiros para as entidades "mãe" do nível adjacente acima.

Nesta figura, existem três espaços para cada entidade topológica que representam os três tipos de entidades que são apontadas acima. Nos casos em que não existe a identificação do tipo de entidade que é apontada, este espaço é deixado em branco. Nos casos onde trivializa a identificação do tipo de entidade apontada, a única possibilidade é indicada entre parênteses. Neste casos, como já se sabe qual o tipo de entidade apontada, o campo *upptr* não é armazenado nas correspondentes áreas de atributos.

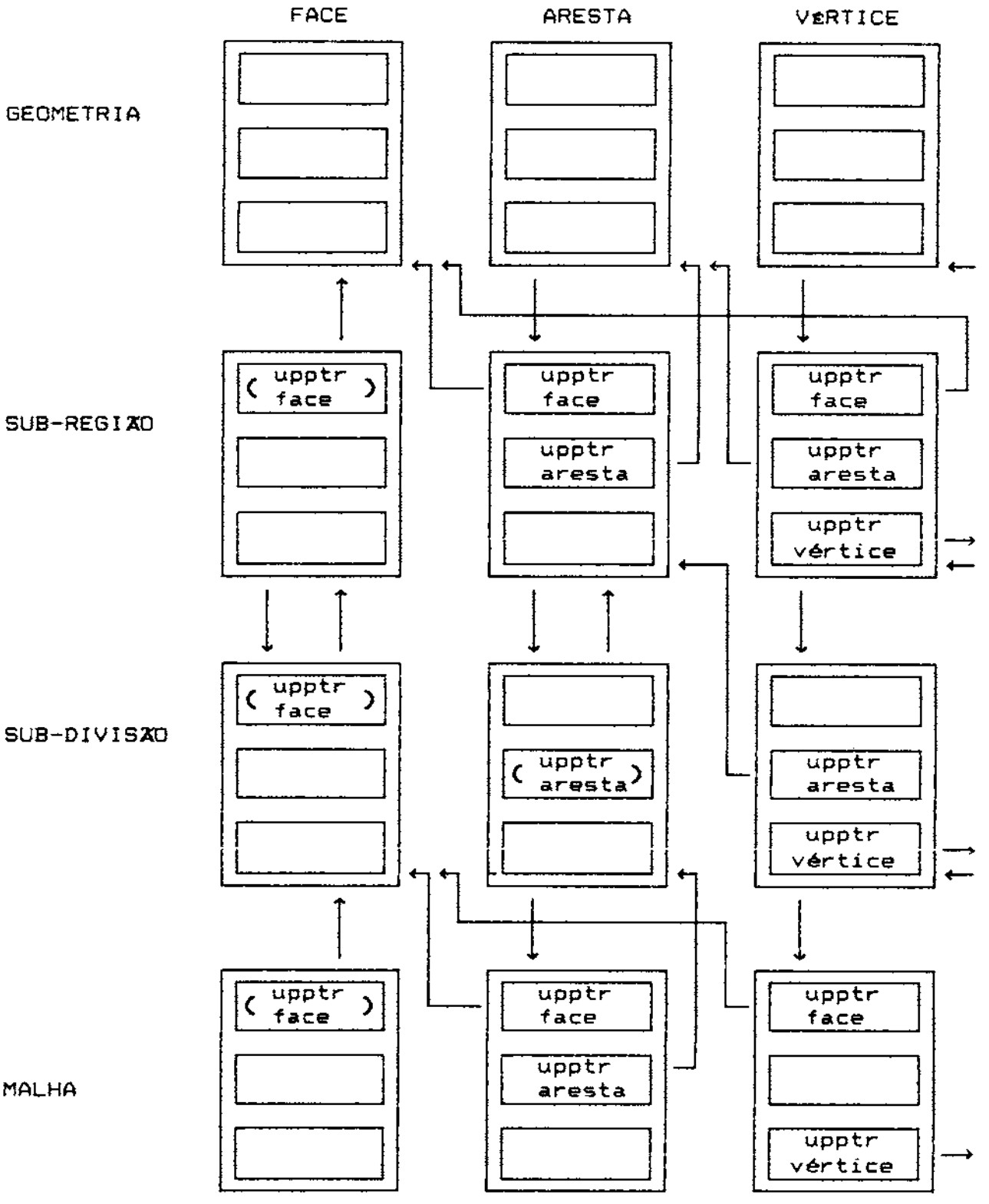


Figura 3.18 - Esquema da estrutura de dados hierárquica.

CAPÍTULO 4

INTERFACE TOPOLOGIA-GEOMETRIA

A estrutura de dados do presente programa de geração de malhas pode ser dividida em três partes. A primeira trata das relações de adjacências das entidades topológicas (Capítulo 2). A segunda é responsável pela manutenção dos ponteiros hierárquicos entre os diversos níveis (Capítulo 3). A terceira e última parte, a ser discutida neste capítulo, refere-se às informações geométricas.

As informações topológicas e geométricas identificam de maneira inequívoca um modelo. Nesta implementação, as informações topológicas são a base de representação do modelo e as geométricas são, simplesmente, atributos das entidades topológicas envolvidas. Assim, informações como as coordenadas de um ponto estão associadas a um vértice, a descrição geométrica de uma curva está associado a uma aresta e os limites mínimo e máximo de uma região bidimensional estão associados a uma face.

São três os aspectos abordados neste capítulo sobre a interface entre a topologia e geometria. O primeiro é sobre os tipos de geometria considerados para as arestas nesta implementação. O segundo aspecto abordado é o tratamento genérico dado à geometria das arestas na implementação de algoritmos que dependem desta informação geométrica. Por fim, é

indicado como critérios geométricos podem auxiliar os critérios topológicos na definição das regiões para geração de malhas por mapeamento transfinito.

4.1 - GEOMETRIA DAS ARESTAS

A geometria de um objeto a ser modelado é definida, basicamente, pelas coordenadas dos seus vértices e pela descrição geométrica das suas arestas. A atual implementação suporta a representação de três tipos geométricos de arestas: reta, círculo e curva Bezier cúbica.

Para a definição geométrica de uma aresta reta é suficiente que se especifiquem seus vértices extremos. No caso do arco de círculo são necessárias como informações adicionais a especificação do centro do círculo e a ordem dos seus vértices extremos, visto que o arco é definido do primeiro para o segundo vértice no sentido horário (Figura 4.1). Para um círculo completo, somente um vértice no círculo e o seu centro são necessários.

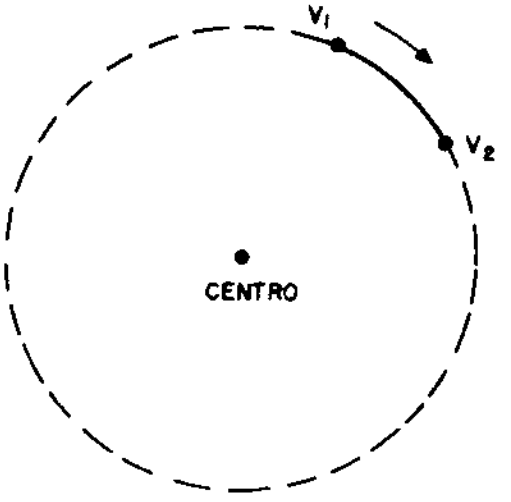


Figura 4.1 - Descrição geométrica de um círculo.

A definição geométrica da curva Bezier é dada por [BOHM84]:

$$C(t) = \sum_{i=1}^{n} CP_i \, B_i(t) \qquad (4.1)$$

onde,

t é a coordenada paramétrica ao longo da curva;

CP_i é o conjunto ordenado de pontos de controle; e

B_i são os polinômios de Bernstein.

No caso de uma curva Bezier cúbica, isto é com polinômio de Bernstein de ordem cúbica, existem quatro pontos de controle. Uma característica desta curva é que o primeiro e último pontos de controle correspondem aos pontos de sua extremidade.

Para determinação dos pontos sobre a curva é utilizado o algoritmo de de-Casteljau [BOHM84]. A figura 4.2 ilustra a obtenção das coordenadas cartesianas de um ponto situado na coordenada paramétrica t de uma curva definida pelos pontos de controle CP1, CP2, CP3 e CP4. A coordenada t varia no espaço paramétrico de 0 a 1.

O algoritmo de de-Casteljau, para o caso cúbico, se resume a três passos. No primeiro passo obtêm-se as coordenadas cartesianas dos pontos A_1 ,A_2 e A_3, que são aqueles situados na coordenada paramétrica t das retas $\overline{CP_1CP_2}$, $\overline{CP_2CP_3}$ e $\overline{CP_3CP_4}$, respectivamente. No segundo passo obtêm-se as coordenadas cartesianas dos pontos B_1 e B_2 que são os situados na coordenada paramétrica t das retas $\overline{A_1A_2}$ e $\overline{A_2A_3}$,

respectivamente. Finalmente, o terceiro passo define as coordenadas cartesianas de um ponto (C_1) situado na coordenada paramétrica t da curva Bezier. Este ponto é obtido calculando-se as coordenadas cartesianas de um ponto situado na coordenada paramétrica t da reta $\overline{B_1 B_2}$.

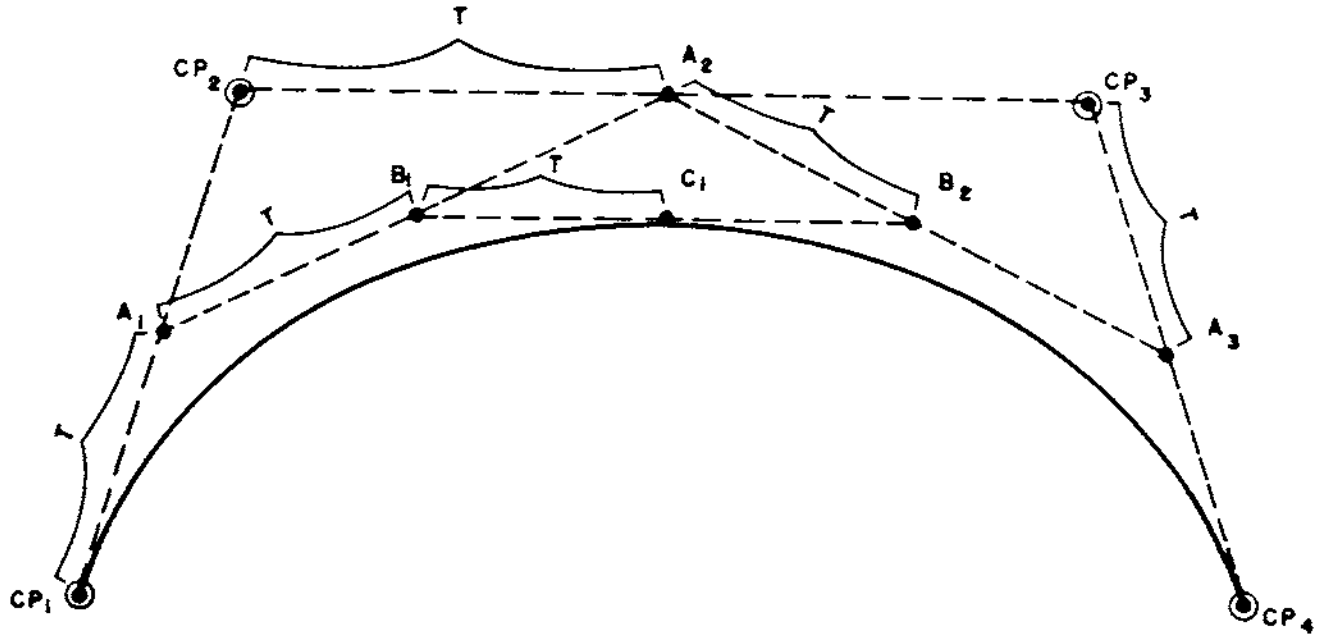


Figura 4.2 - Descrição geométrica da curva Bezier cúbica.

4.2 - FUNÇÕES GENERICAS DAS ARESTAS

Um dos grandes problemas em modelagem geométrica refere-se à implementação de novos tipos geométricos de forma encapsulada, sem alterar todo o desenvolvimento obtido. A utilização de alguns conceitos de linguagem orientada ao objeto, descritos nesta seção, minimiza o impacto, ao longo do código, da introdução de uma nova entidade geométrica.

A programação orientada ao objeto (*oriented-object programming*), tem sido largamente utilizada em programas de inteligência artificial, simulações cientificas e interfaces

gráficas. A intenção deste trabalho não é a de discutir este novo estilo de programação, mas sim apresentar alguns de seus conceitos básicos que aqui foram utilizados.

Definem-se como objetos as entidades que combinam propriedades de procedimentos e dados [STEF85].

A descrição de um ou mais objetos é denominada de classe. Esta descrição é feita, basicamente, através da definição das variáveis e dos procedimentos associados a cada objeto.

Todas as ações executadas em um ambiente deste tipo são feitas através do envio de mensagens aos objetos. Esta técnica permite a implementação de um dos princípios básicos em programação, a abstração dos dados. Com isso, o programa não precisa ter conhecimento da estrutura de dados utilizada, permitindo que esta possa ser modificada sem comprometer todo o desenvolvimento já efetuado.

As mensagens são divididas em duas partes. A primeira, chamada de seletor, indica qual procedimento a ser acionado pelo objeto. A segunda são os argumentos destes procedimentos.

O conjunto de mensagens e procedimentos associados a um objeto são denominados de protocolo e métodos, respectivamente.

Visto desta maneira, a entidade aresta pode ser considerada objeto de uma classe. As sub-classes da classe aresta são os diversos tipos geométricos (reta, círculo, Bezier, etc.) e os métodos associados a estas classes são as funções genéricas das respectivas arestas.

As funções genéricas são procedimentos que realizam tarefas específicas para cada tipo geométrico de aresta. Cada aresta contém na sua área de atributos um ponteiro para as suas

funções genéricas (Figura 4.3).

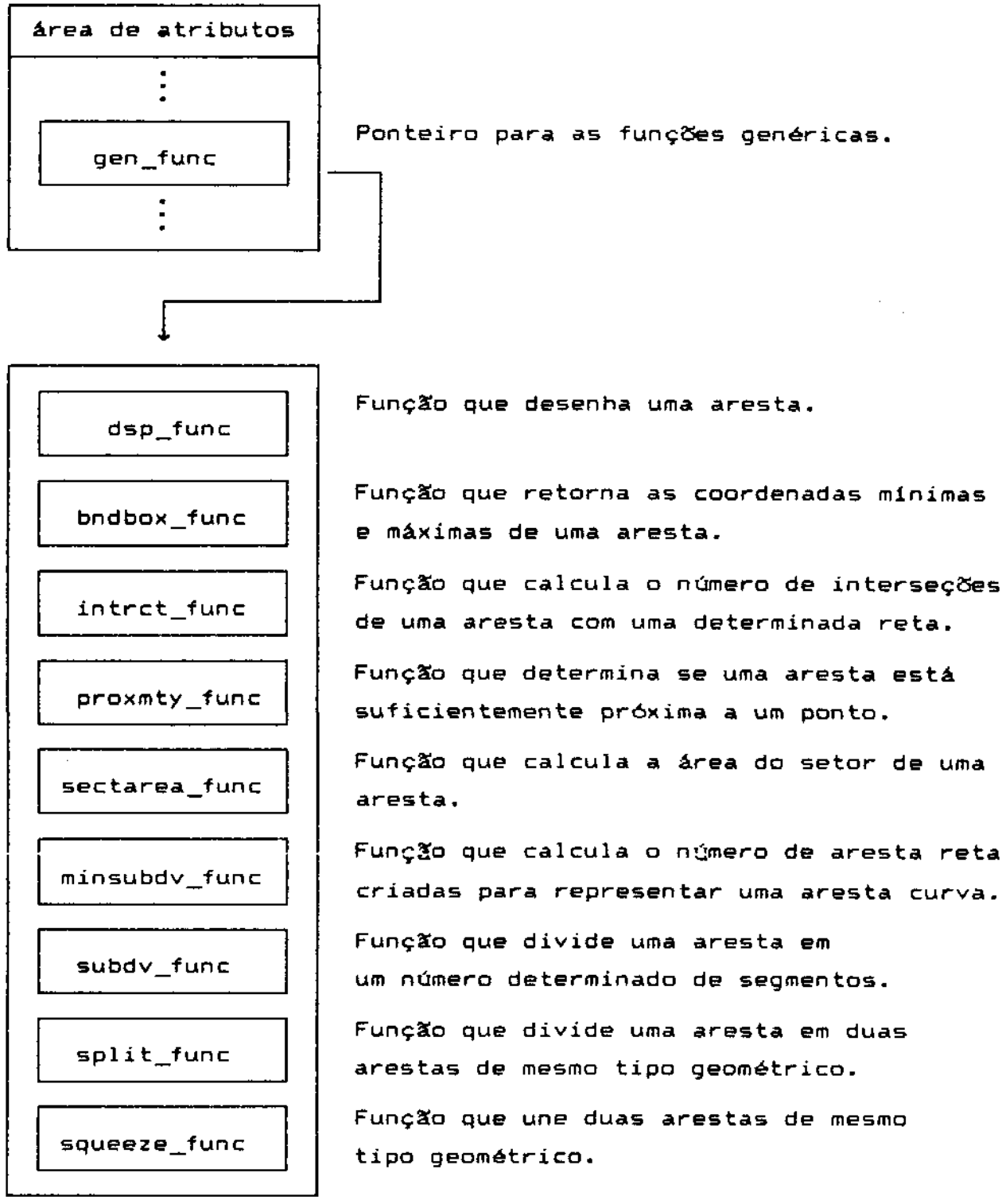


Figura 4.3 - Funções genéricas das arestas.

O acesso a estas funções é feito através de mensagens. O protocolo de mensagens contém seletores que acionam os respectivos procedimentos e passam como argumento um ponteiro para a área de atributos da respectiva aresta.

Em um ambiente deste tipo, nenhum algoritmo que dependa das características geométricas das arestas precisa ser modificado com a introdução de um novo tipo geométrico, implicando somente na elaboração de uma série de funções genéricas consistentes com o novo tipo.

Como exemplo, considere-se o procedimento responsável por desenhar todas as arestas de um dado modelo. Este procedimento percorre a lista de todas as arestas do referido modelo e aciona genericamente, através de mensagens, a função capaz de desenhar cada aresta. Como aquele procedimento independe do tipo geométrico correspondente a cada aresta, ele nunca é alterado com a introdução de um novo tipo geométrico.

A seguir, faz-se uma descrição funcional das funções genéricas desenvolvidas, bem como a indicação do seu uso mais freqüente.

4.2.1 - DSP_FUNC (*"DiSPlay FUNCtion"*)

Esta função desenha uma aresta. Como já discutida anteriormente, ela é largamente utilizada pelos algoritmos que desenham todas as arestas de um determinado modelo.

4.2.2 - BNDBOX_FUNC (*"BouNDing-BOX FUNCtion"*)

Esta função retorna as coordenadas mínimas e máximas de uma aresta e é utilizada no algoritmo que calcula as

coordenadas mínimas e máximas de uma face. Tal algoritmo percorre todas as arestas dos contornos de uma face e obtém, através da função genérica de cada aresta, as suas coordenadas mínimas e maximas, atribuindo à face a menor e a maior de cada coordenada encontrada.

4.2.3 - INTRCT_FUNC (*"INTeRseCTion FUNCtion"*)

Esta função calcula o número de interseções de uma aresta com uma semi-reta a partir de um dado ponto, sendo utilizada pelo algoritmo que faz a seleção de uma face.

Este algoritmo, primeiro, percorre a lista de faces do modelo e seleciona aquelas candidatas a terem o dado ponto no seu interior em função das suas coordenadas mínimas e máximas. A decisão se o ponto está dentro ou não de uma face é feita pelo algoritmo de *scan line* horizontal [PREP85]. Este algoritmo percorre todas as arestas dos contornos de uma face e, através função de interseção de cada aresta, acumula o número de interseções que uma reta definida de menos infinito até o dado ponto faz com as referidas arestas. Se este número for ímpar o ponto está dentro, caso contrário está fora.

4.2.4 - PROXMTY_FUNC (*"PROXiMiTY FUNCtion"*)

Esta função determina se um dado ponto, dentro de uma dada tolerância, está suficientemente próximo a uma aresta. Em caso positivo, retorna o ponto sobre a aresta mais próximo ao dado ponto.

Esta função é muito utilizada na seleção de arestas através das coordenadas de um ponto fornecidas pelo usúario. Os

algoritmos de seleção de arestas são muito caros do ponto de vista computacional, pois, envolvem diversas operações de aritmética de ponto flutuante. Para diminuir o esforço computacional, a seleção de uma aresta é dividida em dois passos. Primeiro, determina-se a que face o dado ponto pertence utilizando-se o algoritmo descrito anteriormente. Em seguida, percorrem-se todas as arestas pertencentes aos contornos desta face e, acionando-se as respectivas funções de proximidade, determina-se qual a aresta selecionada.

A função de proximidade pode ser vista como o problema inverso ao de se avaliar um ponto em uma curva dado o valor paramétrico deste ponto na curva. No caso presente, procura-se achar a coordenada paramétrica sendo dado um ponto na curva ou próximo a ela. Isto vai ser exemplificado, para o caso da curva Bezier, a seguir.

Na implementação atual desta função genérica, o problema geométrico é reduzido a achar o ponto na curva que é o mais próximo ao ponto dado. Este problema pode ser resolvido impondo-se a condição de que a linha que conecta o ponto dado Q e o ponto mais próximo P é normal à curva. Em outras palavras, P tem que satisfazer a relação de produto interno [MORT85]:

$$(\vec{P} - \vec{Q}) \cdot \vec{P}_t = 0 \qquad (4.2)$$

onde $\vec{P}_t$ é o vetor tangente à curva em P. No caso da curva Bezier, P é um ponto da curva C(t) que é dada pela expressão (4.1) e, portanto, a expressão (4.2) representa uma equação não-linear cuja incógnita é o parâmetro t. A resolução desta

equação é feita pelo método de Newton-Raphson, sendo que o valor inicial do parâmetro t nesta resolução é avaliado com base em uma linha poligonal equivalente que é obtida por uma subdivisão arbitrária da curva Bezier.

4.2.5 - SECTAREA_FUNC (*"SECTor AREA FUNCtion"*)

Esta função retorna o valor da área definida pela aresta e pela reta que une seus pontos extremos.

Esta função é utilizada no algoritmo que determina a orientação de um ciclo. Esta informação é muito importante para se determinar se o referido ciclo é interno ou externo em uma face. Na atual implementação, o ciclo externo está orientado no sentido horário e os internos no sentido oposto.

O algoritmo utilizado para a determinação da orientação de um ciclo percorre todos os vértices associados ao mesmo admitindo que estes vértices estejam ordenados no sentido horário e calcula a área do polígono definido por suas arestas. Se os vértices estiverem realmente ordenados no sentido horário a área será positiva, caso contrário será negativa.

A função *sectarea_func* tem como argumento de entrada um dos vértices-uso da aresta associado ao ciclo em questão, retornando o valor da área do setor com sinal positivo ou negativo, dependendo da orientação da geometria da aresta em relação ao referido vértice-uso.

Na maioria dos casos, é possível determinar, simplificadamente, o sinal da área somente se computando a área de um polígono definido pelos vértices de um ciclo. Porém, como a presente implementação suporta a representação de aresta

curvas, é possível a existência de ciclos formados por duas ou até mesmo uma única aresta (aresta de auto-ciclo). Nestes casos, o calcúlo da área, através do processo simplificado, resultaria na obtenção de valores nulos, tornando impossível a determinação da orientação de um ciclo.

4.2.6 - MINSUBDV_FUNC (*"MINumum SUBDiVide FUNCtion"*)

Esta função retorna o número de arestas retas, no nível da sub-divisão, que serão criadas para representar consistentemente uma aresta da sub-região.

Como os níveis da sub-divisão e da malha só suportam a representação de arestas retas, e como não se conhece, *a priori*, o número de subdivisões em que as arestas da sub-região serão divididas pelo usúario, o programa determina, com auxílio da função genérica *minsubdv_func*, este número em função do tipo e das características geométricas de cada aresta.

4.2.7 - SUBDV_FUNC (*"SUBDiVide FUNCtion"*)

Esta função divide uma aresta em um número especificado de segmentos. Ela é utilizada quando o usúario deseja alterar a discretização dos contornos das sub-regiões.

São argumentos desta função o número de segmentos em que a aresta será dividida, a razão entre os comprimentos do primeiro e último segmentos e um indicador se o tipo de divisão é quadrática ou linear. Como argumento de saída têm-se as coordenadas dos pontos da subdivisão.

4.2.8 - SPLIT_FUNC (*"SPLIT FUNCtion"*)

Esta função divide uma aresta em duas arestas de mesmo tipo geométrico.

Os argumentos de entrada desta função são a descrição geométrica da aresta e um ponto sobre a mesma, que será a fronteira das novas arestas criadas. Como argumento de saída têm-se as descrições geométricas das novas arestas.

A obtenção das descrições geométricas das arestas retas, círculos ou arcos de círculos é imediata, pois as coordenadas dos pontos extremos e do centro do círculo, quando for o caso, são facilmente obtidas a partir dos argumentos fornecidos.

No caso da curva Bezier a determinação dos pontos de controle das novas curvas não é tão intuitiva, pois estes pontos correspondem a alguns pontos intermediários do algoritmo de de-Casteljau mostrado anteriormente [FARI88].

Como exemplo, tem-se a curva Bezier definida pelos pontos de controle CP_1, CP_2, CP_3 e CP_4. Deseja-se saber quais os pontos de controle das novas curvas criadas pela divisão da curva original na posição P (Figura 4.4).

Como primeiro passo, calcula-se a coordenada paramétrica do ponto P em relação a curva original. Com este valor, são calculados os pontos intermediários do algoritmo de de-Casteljau. A primeira aresta criada tem como pontos de controle os pontos CP_1, A_1, B_1 e P e a segunda os pontos P, B_2, A_3 e CP_4.

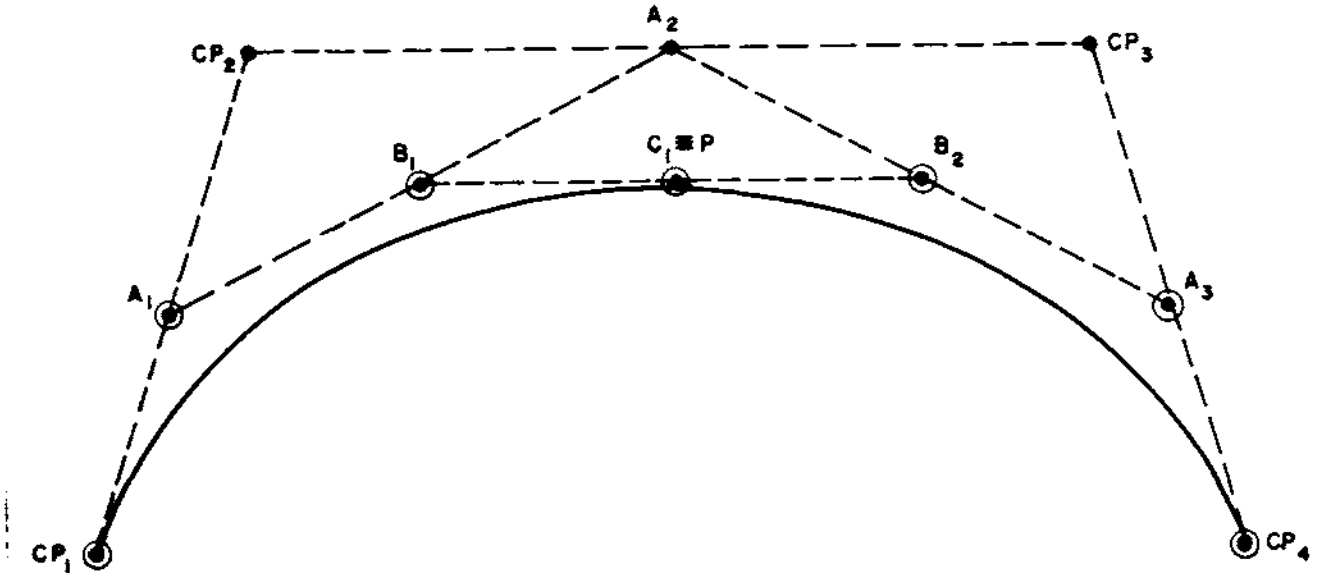


Figura 4.4 - Divisão de uma curva Bezier cúbica.

4.2.9 - SQUEEZE_FUNC (*"SQUEEZE FUNCtion"*)

Esta função une duas arestas de mesmo tipo geométrico que tenham sido obtidas através do mesmo processo de divisão.

A questão aqui é, dadas as descrições geométricas de duas arestas, obter a descrição geométrica da aresta resultante do processo de união.

Como exemplo tem-se as curvas Bezier A e B (Figura 4.5) definidas pelos pontos de controle (CPA_1,CPA_2,CPA_3,CPA_4) e (CPB_1,CPB_2,CPB_3,CPB_4), respectivamente. Deseja-se saber quais os pontos de controle (CP_1,CP_2,CP_3,CP_4) da curva obtida pela união das curvas A e B.

Os pontos de controle inicial e final são obtidos diretamente, pois correspondem aos pontos inicial e final das curvas A e B, respectivamente. Os pontos de controle intermediários são obtidos por uma inversão do algoritmo de

de-Casteljau.

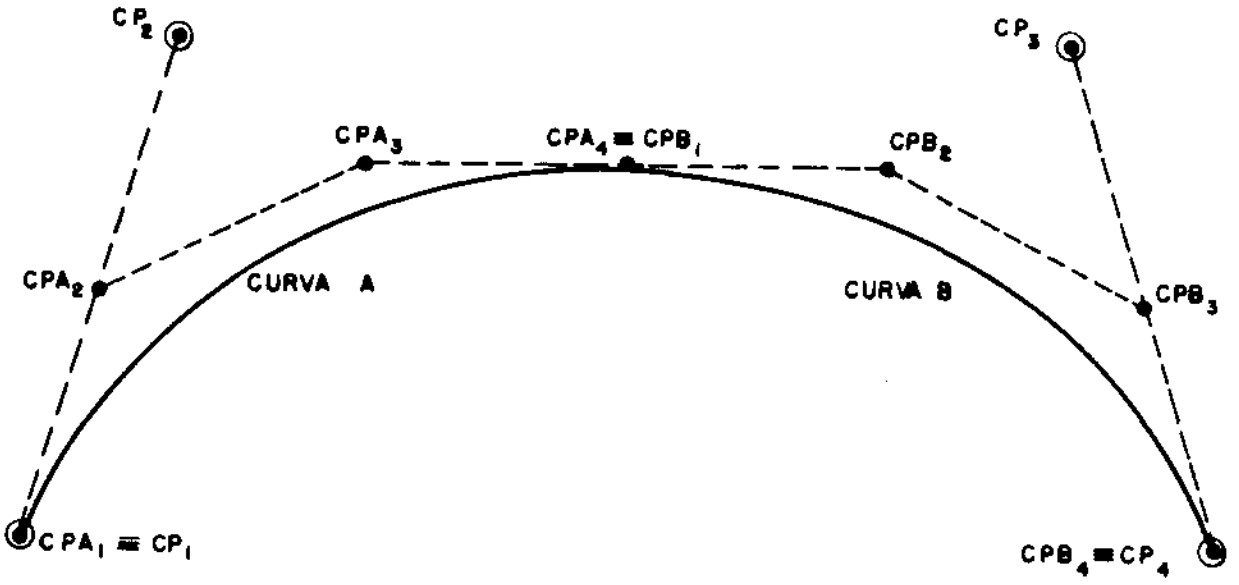


Figura 4.5 - Fusão de duas curvas Bezier cúbicas.

Tem-se como condição necessária à união de duas arestas Bezier o fato delas terem sido geradas através do mesmo processo de divisão. Esta condição permite fazer três afirmações.

a) O ponto $CPA_4 \equiv CPB_1$ está localizado na coordenada paramétrica t da reta definida pelos pontos CPA_3 e CPB_2.

b) O ponto CPA_2 está localizado na coordenada paramétrica t da reta definida pelos pontos CP_1 e CP_2.

c) O ponto CPB_3 está localizado na coordenada paramétrica (1-t) da reta definida pelos pontos CP_4 e CP_3.

Utilizando-se a primeira afirmação é possível calcular o valor da coordenada paramétrica t, pois todos os pontos são facilmente obtidos a partir dos argumentos fornecidos.

De posse do valor da referida coordenada é possível,

através das segunda e terceira afirmações, expandir os pontos CPA_2 e CPB_3 e obter os pontos de controle CP_2 e CP_3 da curva resultante.

4.3 - DETERMINAÇÃO AUTOMÁTICA DOS CANTOS DO MAPEAMENTO

Uma das maiores desvantagens associadas às técnicas de mapeamento transfinito é a necessidade de se definir regiões topológicamente consistentes com os algoritmos de geração de malhas desejados.

O nível da sub-região confere ao programa uma interface bastante flexível para se dividir a geometria do problema em regiões que sejam consistentes com os referidos algoritmos. Porém, uma determinada região pode possuir inúmeras combinações de arestas que definem regiões topologicamente consistentes.

A introdução de alguns critérios geométricos, descritos a seguir, pode definir, dentre as diversas opções, aquela que produza a malha de elementos finitos mais uniforme.

As informações geométricas utilizadas nos algoritmos que definem os cantos dos mapeamentos transfinitos são, essencialmente, os ângulos internos do polígono que define uma região. Como primeiro passo, são determinadas todas as combinações de vértices que formam regiões topologicamente consistentes com o algoritmo de mapeamento a ser utilizado. No segundo passo, é escolhida a melhor opção em função de critérios geométricos.

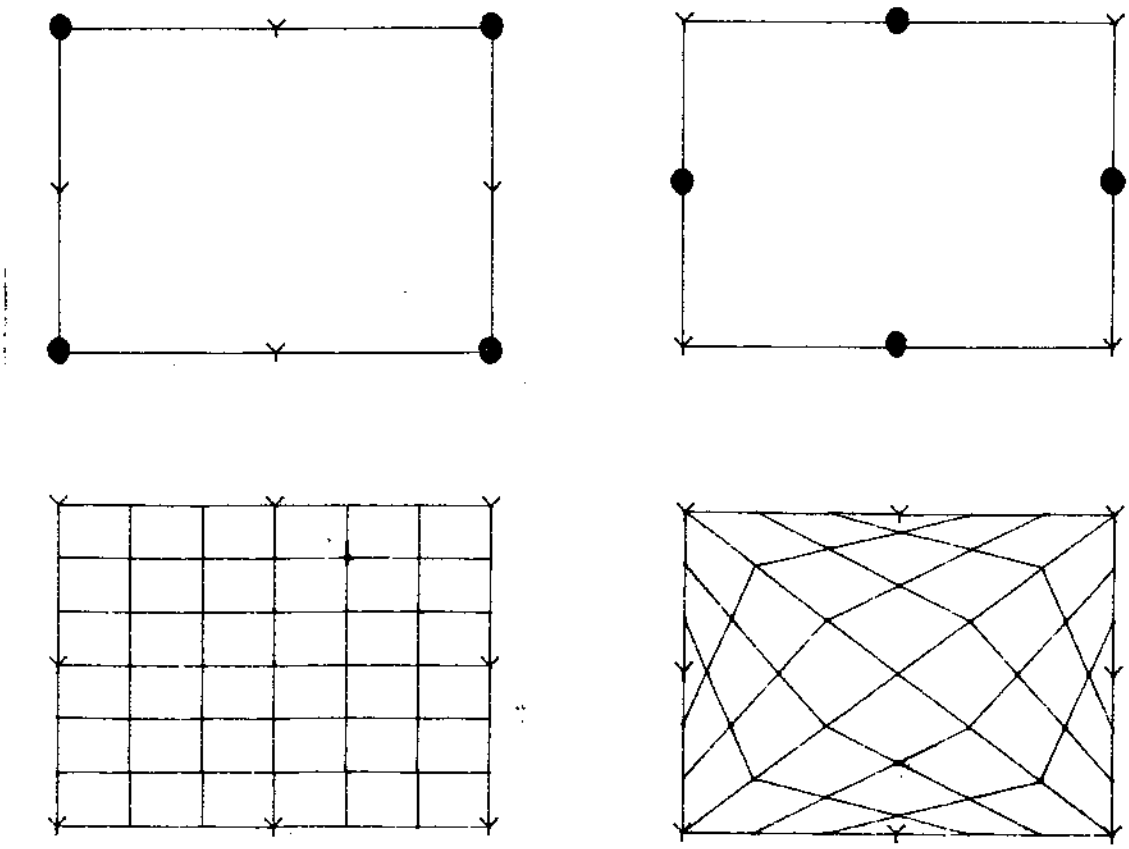
No mapeamento bilinear, as regiões a serem discretizadas são topologicamente equivalentes a quadriláteros com o mesmo número de divisões nos lados opostos. Como critério topológico opcional, os lados dos quadriláteros são definidos de maneira a terem o número de divisões o mais próximo possível da quarta parte do número total de divisões. Como critério geométrico, é definida como melhor opção aquela que tenha os ângulos internos associados aos vértices de canto do mapeamento mais próximos de Π/2 radianos.

A Figura 4.6 ilustra uma região a ser discretizada através do algoritmo bilinear. Existem, em função dos critérios topológicos, duas opções que definem regiões consistentes com o algoritmo escolhido. Na primeira opção (Figura 4.6-a), os ângulos internos associados aos vértices de canto valem Π/2 radianos, enquanto que na segunda opção (Figura 4.6-b), estes ângulos valem Π radianos. Os vértices de canto estão identificados por uma marca circular. A combinação indicada na primeira opção produziu elementos finitos mais uniformes.

No mapeamento bilinear degenerado, as regiões são topologicamente equivalentes a quadriláteros com um dos lados degenerados a um ponto ou, simplesmente, triângulos.

Neste mapeamento, os lados adjacentes ao ponto degenerado têm o mesmo número de divisões. O processo de seleção dos prováveis vértices de canto começa com a escolha do vértice que representa o ponto de colapso. Em seguida, percorre-se o polígono que define a região, em ambos os sentidos, com a finalidade de determinar os demais vértices. A procura é interrompida quando se consegue um número de divisões próximo à

terça parte do número de divisões total ou quando o ângulo interno, associado a um dos vértices, é inferior a Π/2 radianos. Esta condição evita mudanças acentuadas, para dentro da região, nos lados adjacentes ao lado colapsado. Achados os outros vértices de canto, é verificado se o número de divisões dos lados adjacentes são iguais. Em caso afirmativo, esta é uma opção válida. Este processo é repetido para todos os vértices da região e, dentre as diversas opções, é escolhida aquela que tenha o maior ângulo associado ao vértice que representa o ponto de colapso.

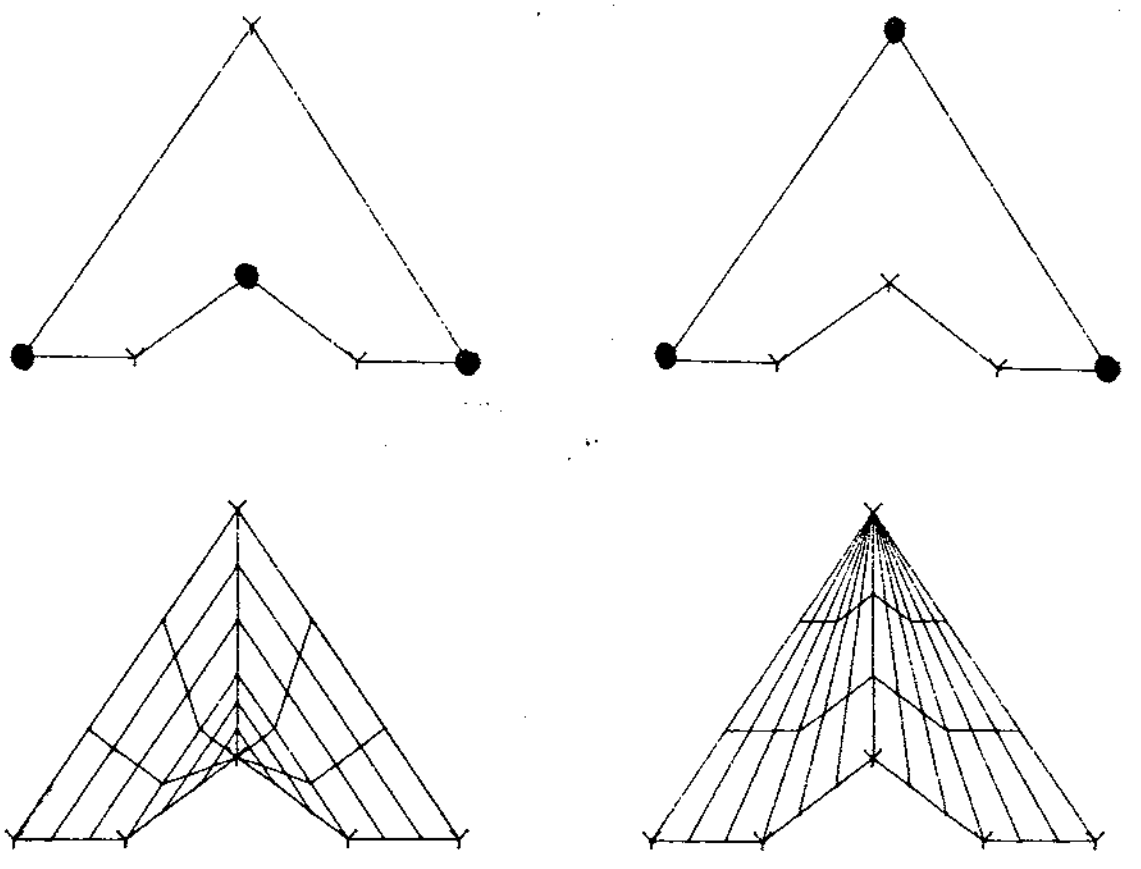


(a) (b)

Figura 4.6 - Determinação dos vértices de canto de um mapeamento bilinear

A Figura 4.7 ilustra a discretização de uma região utilizando o algoritmo bilinear degenerado.

Entre as diversas combinações de vértices, que definem regiões topologicamente equivalente a triângulos, selecionaram-se duas que melhor representam o critério de escolha em função do ângulo associado ao vértice que representa o lado degenerado. A opção pelo maior ângulo produz elementos finitos mais uniformes (Figura 4.7-a), enquanto a outra produz elementos finitos distorcidos, principalmente na região próxima ao lado degenerado (Figura 4.7-b).

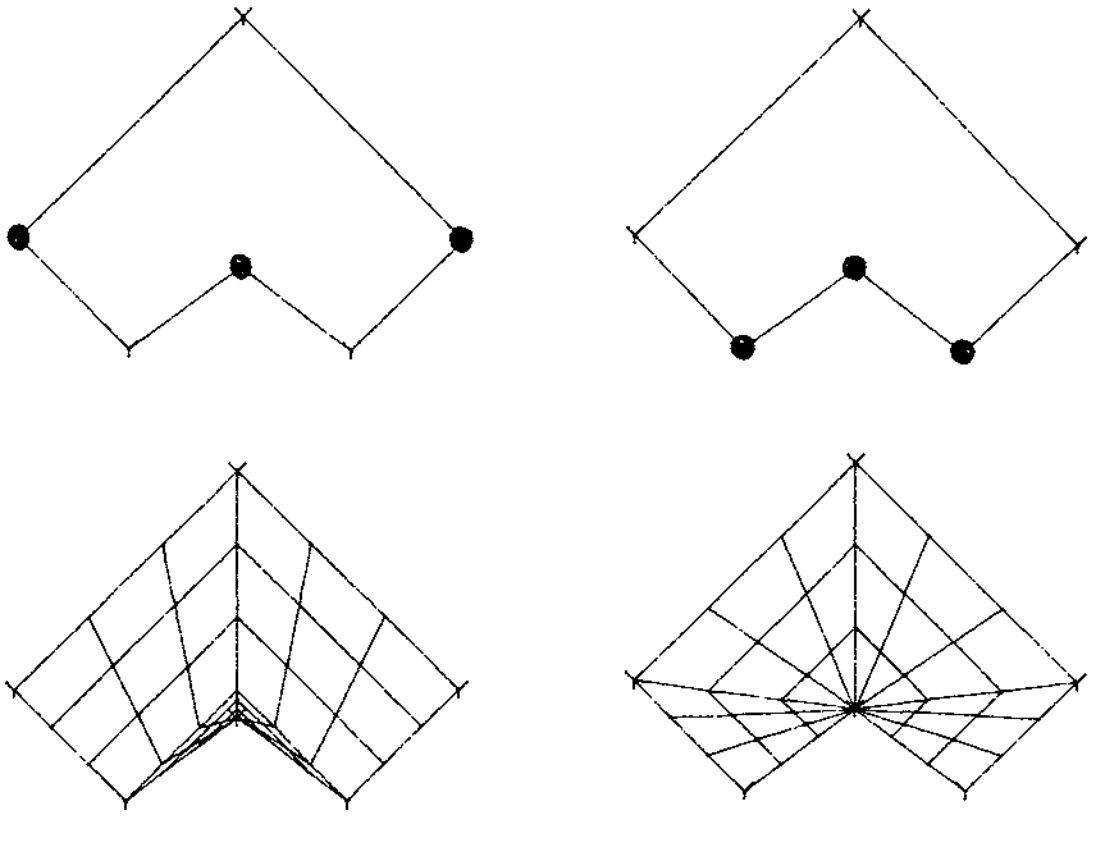


(a) (b)

Figura 4.7 - Determinação dos vértices de canto de um mapeamento bilinear degenerado (critério 1).

A Figura 4.8 ilustra o critério de escolha que evita a ocorrência de ângulos menores que Π/2 radianos ao longo dos lados adjacentes ao lado degenerado.

A região escolhida, para este exemplo, é topologicamente idêntica àquela do exemplo anterior. A diferença está na geometria do problema. Assim, se fosse usado o mesmo critério do exemplo anterior, os vértices de canto seriam os indicados na Figura 4.8-a. Com a modificação da geometria e a utilização do critério acima mencionado, a opção é a indicada na Figura 4.8-b. Comparando-se as malhas geradas, verifica-se que a segunda opção gerou elementos finitos mais uniformes.



(a) (b)

Figura 4.8 - Determinação dos vértices de canto de um mapeamento bilinear degenerado (critério 2).

No mapeamento trilinear, as regiões são topologicamente equivalentes a triângulos com o mesmo número de divisões em todos os lados. Como critério geométrico, é considerada a melhor opção aquela que tenha a menor soma dos ângulos internos. Este critério faz com que o triângulo topológico se aproxime da característica de um triângulo geométrico, onde a soma dos ângulos internos é igual a Π radianos.

A Figura 4.9 ilustra a discretização de uma região através do mapeamento trilinear. As duas possíveis opções de regiões topologicamente equivalentes a triângulos são apresentadas. A primeira opção, definida pela menor soma dos ângulos internos (Figura 4.9-a), gerou elementos finitos mais uniformes que a segunda (Figura 4.9-b).

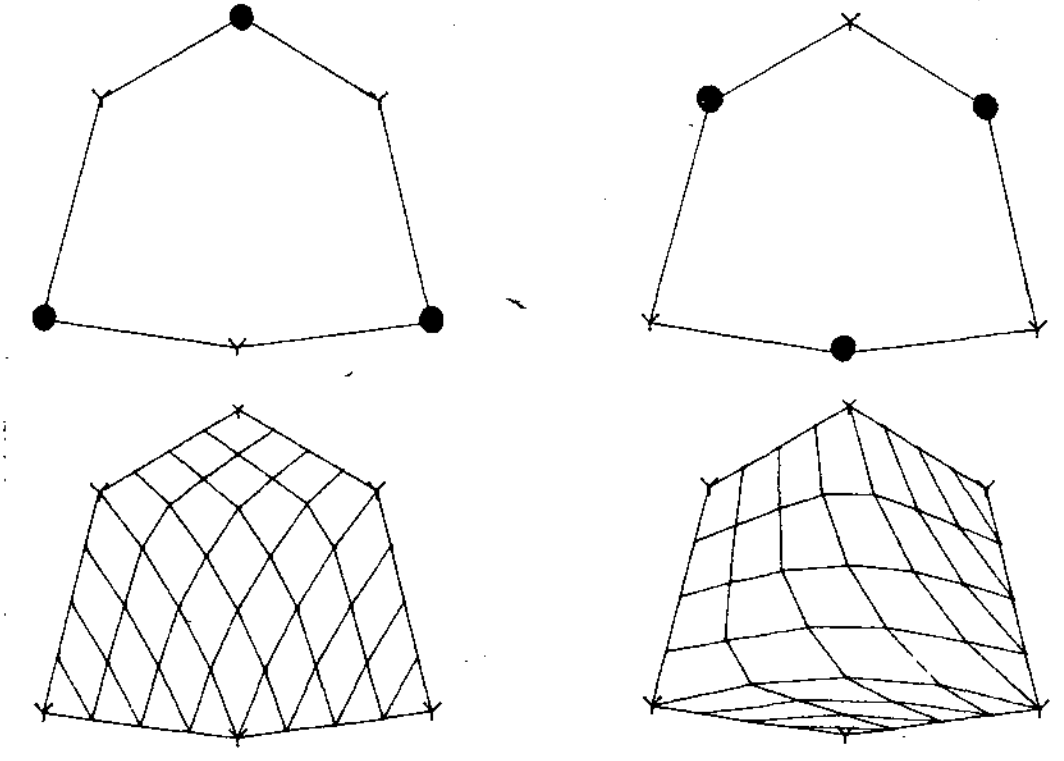


Figura 4.9 - Determinação dos vértices de canto de um mapeamento trilinear.

Os critérios aqui apresentados, de forma inétida, embora tenham se mostrado eficientes na maioria dos casos práticos, não pretendem ser definitivos ou garantir que a opção escolhida seja realmente a melhor em todos os casos, podendo ser vistos, em última instância, como uma indicação da melhor opção.

CAPÍTULO 5

EXEMPLOS

Este capítulo procura ilustrar através de exemplos os aspectos mais importantes da atual implementação do programa de geração de malhas.

A Figura 5.1 ilustra a configuração de todos os modelos após a definição da geometria de um problema. No canto superior esquerdo tem-se a representação do modelo da geometria, ao lado tem-se o modelo da sub-região, no canto inferior esquerdo o modelo da sub-divisão e no canto inferior direito o modelo da malha.

Neste instante do processo existe uma relação de um para um entre as faces dos diversos níveis hierárquicos. Este fato não ocorre com as arestas e vértices, pois os níveis inferiores (sub-divisão e malha) não suportam a representação de arestas curvas. As arestas curvas criadas nos níveis da geometria e sub-região são reproduzidas nos níveis inferiores como uma seqüência de arestas retas. Visualmente só é possivel identificar estas arestas no nível da malha, pois o nível da sub-divisão é desenhado utilizando-se a descrição geométrica das arestas do nível da sub-região (Figura 5.1).

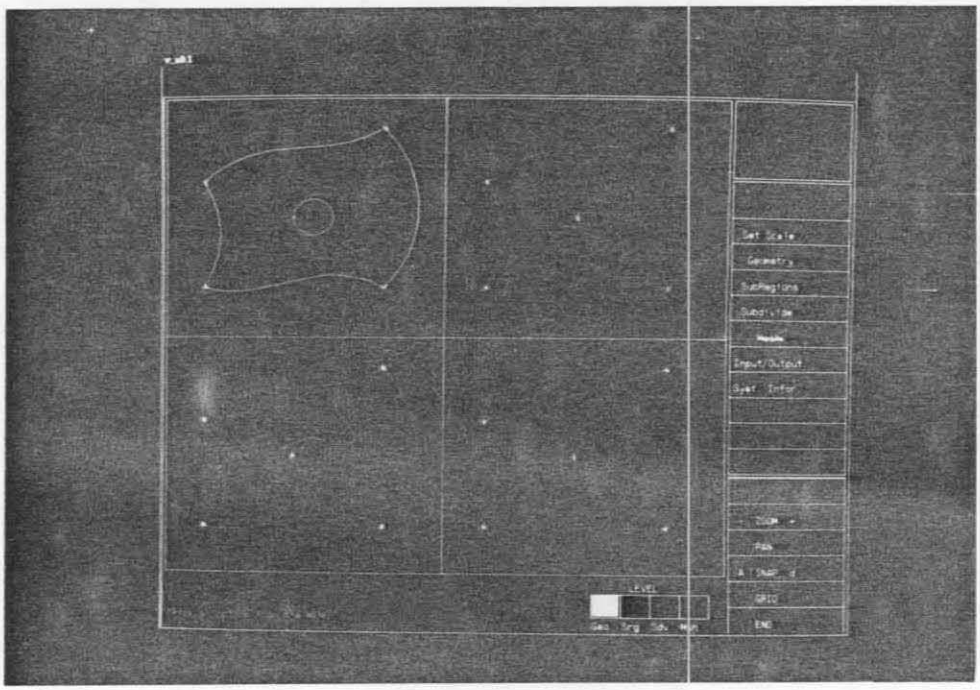


Figura 5.1 - Representação dos modelos hierárquicos.

No nível da geometria são definidos o tipo de material associado a cada região, bem como se a mesma será considerada um furo no modelo. No exemplo da Figura 5.1, o círculo no centro do modelo foi definido como um furo.

Continuando o processo de discretização da estrutura, é necessário decompor o modelo da geometria em regiões consistentes com os algoritmos de geração de malhas desejados. A decomposição do modelo em regiões é feita no nível da sub-região. A Figura 5.2 ilustra a decomposição do modelo em quatro sub-regiões. A definição das regiões é feita através da introdução de quatro arestas retas. A introdução destas arestas causou a divisão automática dos arcos de círculos nos bordos

esquerdo e direito, das curvas Bezier nos bordos inferior e superior e do círculo no centro do modelo (Figura 5.2).

Após definidas as regiões, o usuário necessita definir a dicretização dos contornos destas regiões. Esta discretização é feita no nível da sub-divisão. Neste nível o usuário informa o número de segmentos em que cada aresta será dividida, o tipo de divisão (linear ou quadrática) e a razão entre os comprimentos do primeiro e último segmentos de cada aresta. Neste exemplo foi especificada uma sub-divisão linear com a razão de um para um, entre o primeiro e último segmentos, para todas as arestas do contorno geométrico do modelo e uma razão de um para três nas arestas internas (Figura 5.2). O número de segmentos definidos para cada aresta foi escolhido em função do algoritmo de geração de malhas desejado, como é descrito a seguir.

No nível da malha são definidos o tipo de elemento finito a ser utilizado e o algoritmo de geração de malhas desejado. A seleção de algoritmos de mapeamento transfinito permite que o usuário identifique os cantos do mapeamento ou, opcionalmente, especifique a região a ser discretizada, deixando o programa determinar de forma automática os cantos do mapeamento. Como terceira opção, o usuário pode determinar a geração automática das malhas de todas as regiões do modelo. Para isso, o programa percorre a lista de todas as faces da sub-divisão e, em função de critérios topológicos e geométricos, aplica o algoritmo de geração de malhas mais adequado. Esta opção foi utilizada no exemplo da Figura 5.2. Todas as malhas deste exemplo foram geradas através do algoritmo de mapeamento bilinear com o elemento finito tipo Q4 (quadrangular de quatro nós).

Percebe-se na Figura 5.2 como as arestas, no nível da sub-região, são agrupadas para formar um único lado de uma região para mapeamento. Cada região é topologicamente equivalente a um quadrilátero, onde os lados são um dos quadrantes do círculo central, duas arestas radiais e um par de arestas externas adjacentes a um dos cantos do modelo.

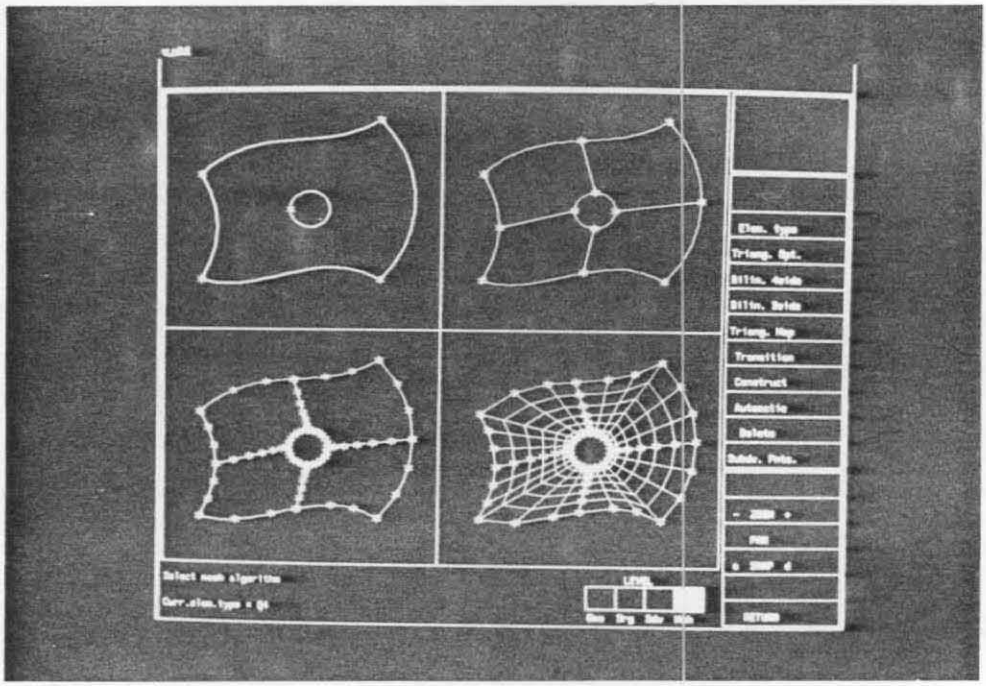


Figura 5.2 - Exemplo de geração de malhas com o algoritmo de mapeamento bi-linear.

A Figura 5.3 ilustra a eliminação de uma aresta no nível da sub-região, do exemplo da Figura 5.2. A eliminação desta aresta causa a fusão das curvas Bezier do bordo superior e dos arcos de círculo do semicírculo superior no centro do modelo. A curva Bezier resultante do processo de fusão tem a mesma

descrição geométrica da curva originalmente definida no nível da geometria. Os aspectos topológicos e geométricos da fusão de arestas foram discutidos nos capítulos 3 e 4 respectivamente.

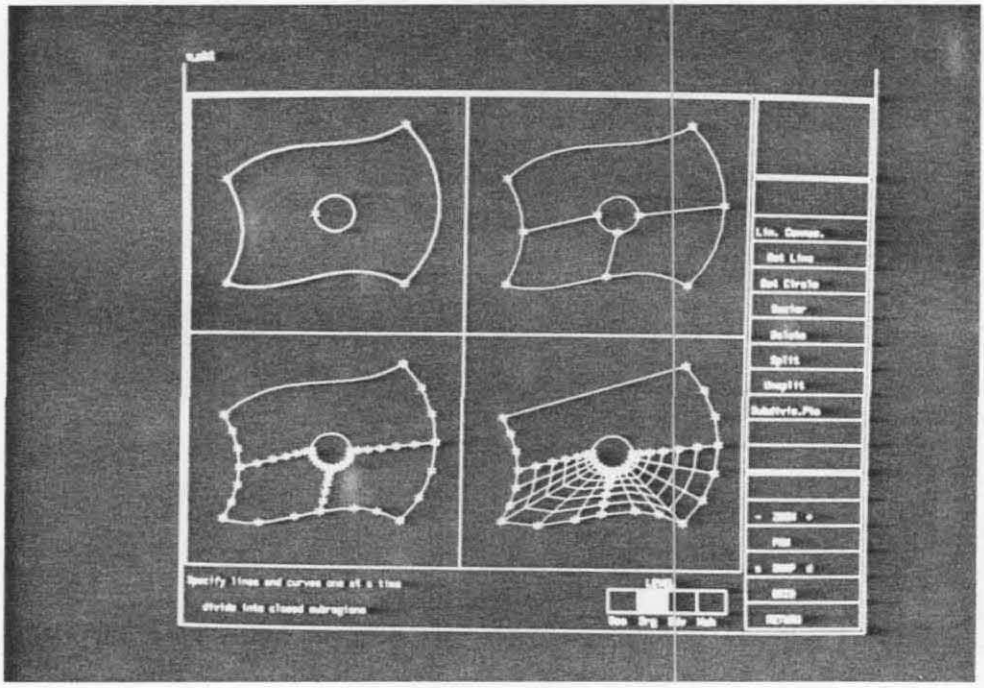


Figura 5.3 - Exemplo de eliminação de arestas do nível da sub-região.

Obedecendo a hierárquia entre os modelos, a eliminação de uma aresta do nível da sub-região causa mudanças nos níveis da sub-divisão e malha, ficando o nível da geometria inalterado. O programa elimina, automaticamente, as instâncias da referida aresta nos níveis abaixo. As aresta retas criadas nestes níveis são redefinidas de acordo com a descrição geométrica das arestas resultantes do processo de fusão no nível da sub-região. É importante observar que a informação do número de

divisões das arestas fundidas é perdida (Figura 5.3).

No nível da malha, a eliminação da aresta da sub-região causou a eliminação das malhas nas regiões adjacentes à referida aresta (Figura 5.3), ficando as demais regiões inalteradas.

A Figura 5.4 ilustra a discretização de regiões utilizando os algoritmos de mapeameamento bilinear degenerado e trilinear. As regiões para estes algoritmos são topologicamente equivalentes a triângulos. A diferença é que, para o algoritmo bilinear degenerado, o número de subdivisões nos lados adjacentes ao lado degenerado são iguais e no trilinear todos os lados têm o mesmo número de subdivisões.

As regiões que foram discretizadas utilizando os referidos algoritmos estão destacadas em vermelho na Figura 5.4. Utilizou-se na discretização dos contornos de todas as regiões uma subdivisão quadrática com a relação de um para um entre os comprimentos do primeiro e último segmentos de cada aresta.

A malha da região do lado esquerdo do modelo (Figura 5.4) foi gerada com o algoritmo bilinear degenerado e o elemento finito T6. A definição dos cantos do mapeamento foi fornecida pelo usuário. Neste caso, o primeiro canto definido foi o vértice localizado na posição mais a esquerda da região, determinando, desta forma, que o lado degenerado corresponda ao referido vértice.

Na região destacada à direita do modelo foi utilizado o algoritmo de mapeamento trilinear e o elemento finito T6. Neste caso o usuário usou a opção de determinação automática dos

cantos do mapeamento, bastando, para isso, somente identificar a região a ser discretizada.

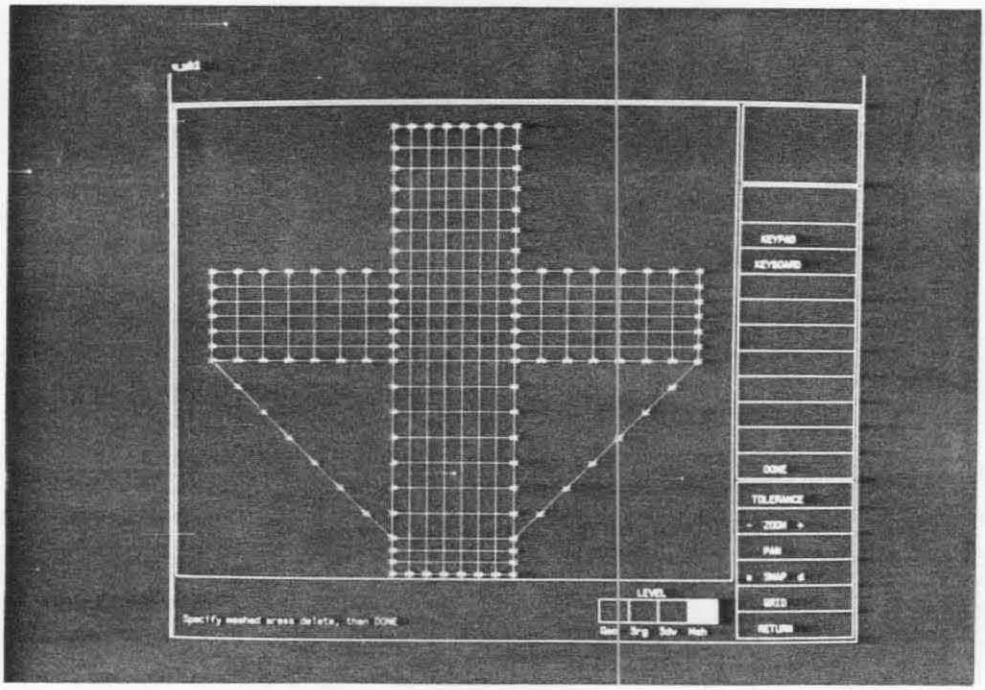


Figura 5.4 - Exemplo de geração de malhas com os algoritmos de mapeamento bilinear degenerado e trilinear.

A Figura 5.5 ilustra a discretização de uma região utilizando o algoritmo de tecelagem implementado no corrente programa. Este algoritmo é utilizado em regiões com topologias arbitrárias, sendo adequado para geração de malhas em regiões com furos.

O círculo no centro do modelo (Figura 5.5) foi definido, no nível da geometria, como um furo e os contornos das regiões foram discretizado com uma subdivisão linear e razão de um para um entre os comprimentos do primeiro e último segmentos de cada

aresta.

Este algoritmo permite que o usuário defina alguns nós no interior da região e se nós serão gerados automaticamente pelo algoritmo. No exemplo da Figura 5.5 utilizou-se a opção de geração automática e definição pelo usuário de nós interiores. Os asteriscos vermelhos no interior da região (Figura 5.5) correspondem aos nós fornecidos pelo usuário. A discretização final em elementos finitos do referido modelo é mostrada na Figura 5.6.

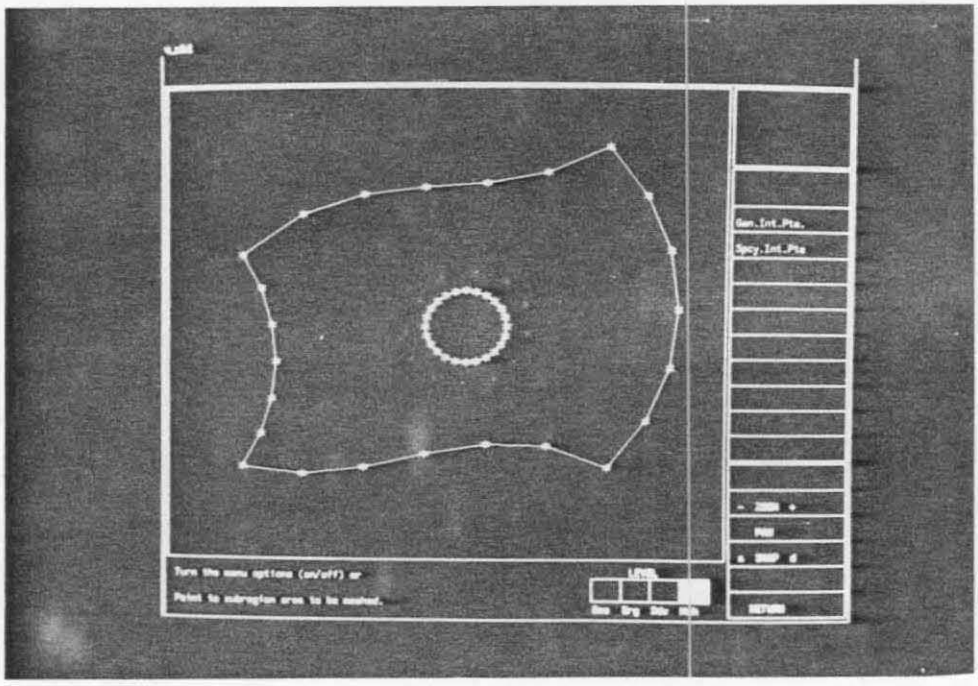


Figura 5.5 - Especificação dos pontos internos do algoritmo de tecelagem.

No nível da malha existe uma opção que permite o usuário

modificar manualmente as malhas geradas. Para ilustrar este processo, isolou-se uma região mais restrita do modelo com a finalidade de facilitar a visualização. Esta região está indicada pelo quadrado branco na Figura 5.6.

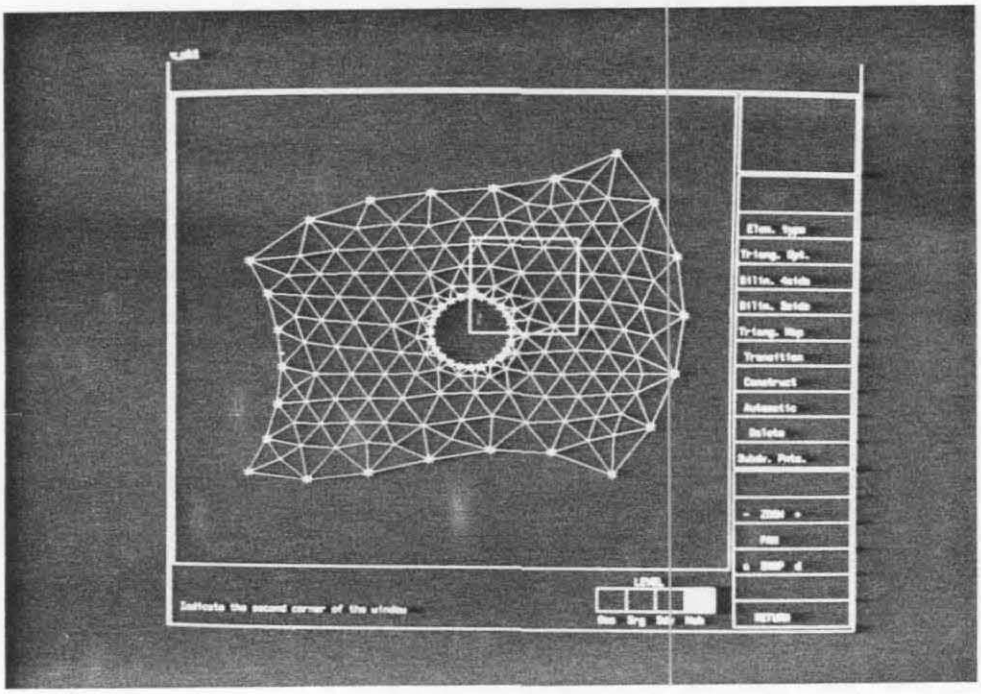


Figura 5.6 - Exemplo de geração de malhas em regiões com topologias arbitrárias.

O processo de edição de malhas de elementos finitos é feito através da eliminação de arestas e vértices e pela introdução de arestas retas. No exemplo da Figura 5.6 eliminaram-se seis arestas. Este resultado pode ser obtido pela eliminação das arestas ou pela eliminação do vértice comum a estas aresta (Figura 5.7).

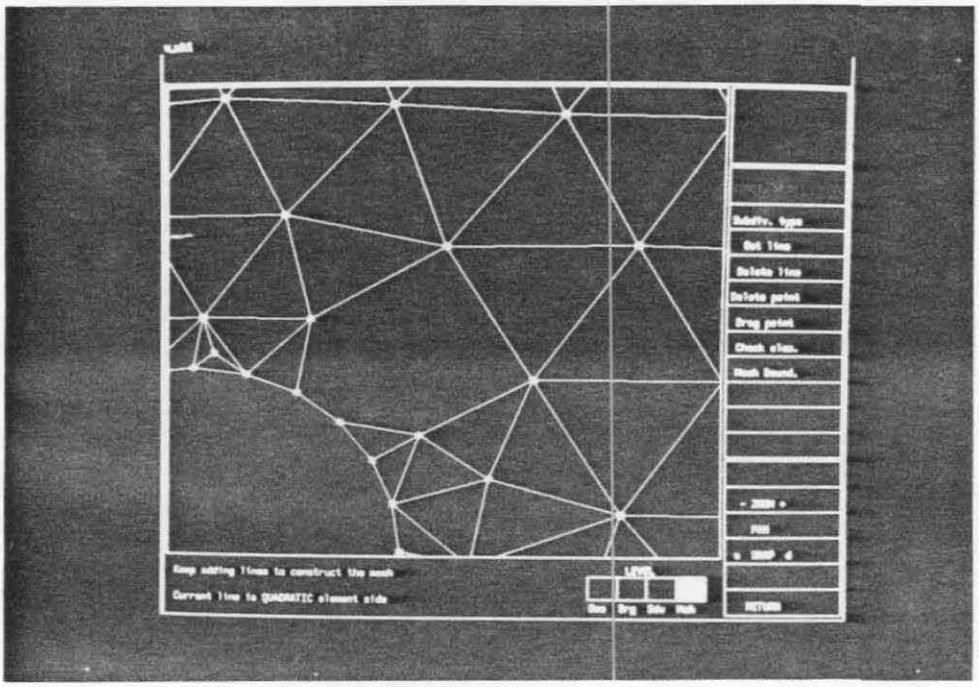


Figura 5.7 - Exemplo de eliminação de arestas e vértices no modelo da malha.

A introdução de arestas na região onde foi eliminadada localmente a malha de elementos finitos é mostrada na Figura 5.8. A alteração da malha não afeta os elementos finitos adjacentes e está consistentemente armazenada na estrutura de dados. O programa reconhece automaticamente toda vez que uma nova face é criada e atribui a esta face o tipo do elemento criado, se este for um elemento finito válido.

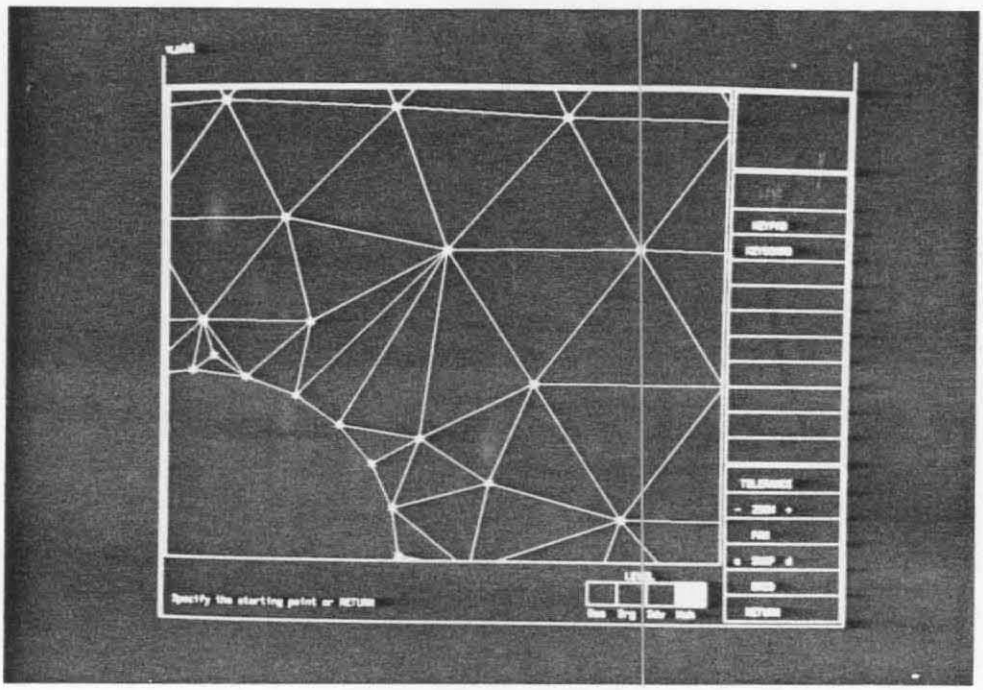


Figura 5.8 - Exemplo de edição da malha de elementos finitos.

O exemplo da Figura 5.9 ilustra toda a potencialidade do sistema em modelar estruturas de geometria complexa, decompor esta geometria em regiões arbitrárias, discretizar o contorno das regiões gerar as malhas de elementos finitos.

Neste exemplo foi utilizada a opção de geração de malhas automática e o programa conseguiu discretizar todas as regiões através dos algoritmos de mapeamento transfinito.

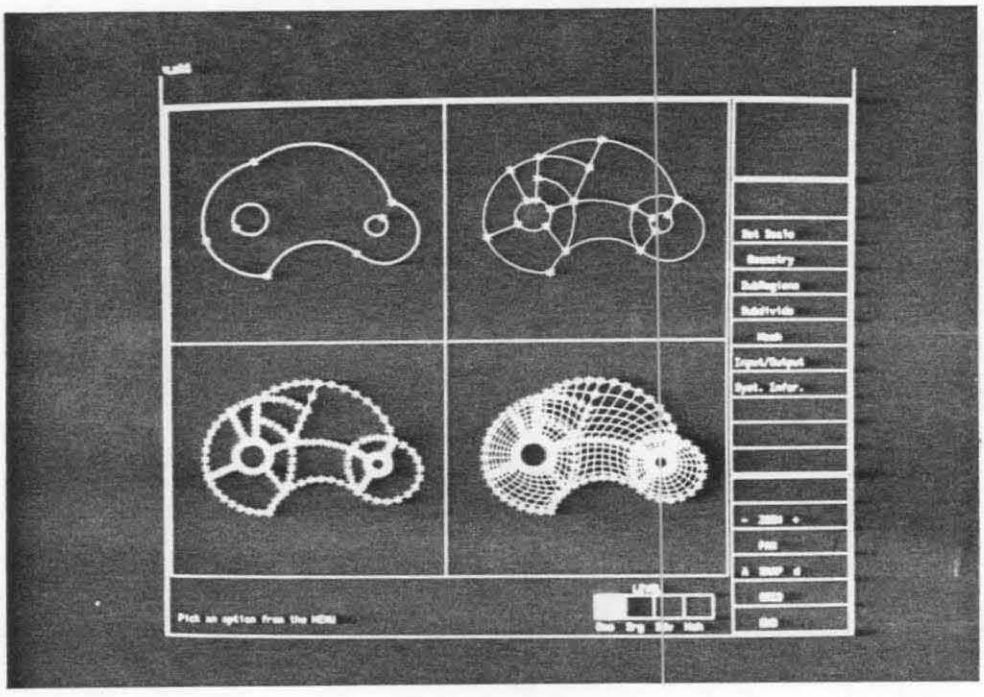


Figura 5.9 - Geração automática de malhas utilizando algoritmos de mapeamento transfinito.

CAPÍTULO 6

CONCLUSÕES

Este capítulo apresenta as principais conclusões deste trabalho, bem como algumas sugestões para futuros desenvolvimentos.

O sistema desenvolvido neste trabalho consiste em um pré-processador gráfico interativo baseado em uma estrutura de dados topológica para a discretização numérica de modelos bidimensionais a serem analisados pelo método dos elementos finitos. O referido sistema tem como característica principal uma representação do modelo baseada em sua geometria e em uma forte interação com o usuário. A definição da geometria é feita utilizando-se conceitos de modelagem geométrica e, através de uma técnica de refinamento progressivo, obtém-se a discretização da estrutura em elementos finitos. O fato de se ter uma representação do modelo baseada em informações topológicas e geométricas permite que a definição de atributos, tais como carregamentos, condições de suporte e outros atributos físicos, não se altere durante o processo de discretização do mesmo.

As conclusões deste trabalho são:

a) A estrutura de dados FEDm fornece as características necessárias a um sistema de subdivisão planar arbitrária com geometria curva.

b) A utilização de uma estrutura de dados topológica para a representação de variedade dois conferiu ao programa características de um poderoso modelador geométrico bidimensional.

c) A hierarquia dos modelos topológicos é ideal para programas de geração de malhas com a representação do modelo baseada geometria. As mudanças ocorridas nos níveis da geometria, sub-região e sub-divisão afetam localmente a malha existente.

d) A estrutura de dados topológica permite uma fácil manipulação da topologia das sub-regiões onde as malhas serão geradas. Este ambiente se mostrou bastante propício para a implementação de algoritmos de geração de malhas baseados nas técnicas de mapeamento transfinito, além de não restringir o uso de outras técnicas.

e) A utilização de alguns critérios topológicos e geométricos permite determinar, de forma automática, a melhor associação de arestas no contorno de uma sub-região de forma consistente com um determinado algoritmo de mapeamento

transfinito, possibilitando a geração de malhas com elementos mais uniformes.

f) Os procedimentos genéricos associados a arestas, introduzidos neste trabalho, simplificam os algoritmos que gerenciam a estrutura de dados e facilitam a introdução de novos tipos geométricos.

g) O tipo de ambiente conseguido neste trabalho se mostrou ideal não só para a geração de malhas, mas também para o desenvolvimento de um sistema integrado de pré-processamento, pós-processamento e análise pelos métodos dos elementos finitos e de elementos de contorno.

Como sugestões para continuação deste trabalho, têm-se:

a) A introdução no nível da geometria de algumas facilidades existentes no nível da sub-região, como a divisão e a fusão automática de arestas. Isto fornecerá ao usuário uma interação mais flexível para a definição do modelo.

b) A implementação no nível da geometria de procedimentos que definam carregamentos e condições de apoio, bem como um gerenciamento mais flexível de atributos de materiais.

c) O desenvolvimento de um sistema integrado de elementos finitos e de contorno.

APÊNDICE A

OPERADORES DE EULER

Os operadores de Euler foram, originalmente, introduzidos por Baumgart [BAUM75] na apresentação da estrutura de dados arestas aladas.

Estes operadores são procedimentos que manipulam a estrutura de dados de modelos de representação de fronteira (variedades dois), criando ou alterando instâncias das entidades topológicas e as suas relações de adjacências.

O números de operadores e a forma como eles são desenvolvidos dependem da aplicação e da estrutura de dados implementada. No presente trabalho, usaram-se, com algumas modificações, os operadores sugeridos por Weiler [WEIL86] na implementação da estrutura de dados face-aresta. Somente os operadores necessários para manipular uma subdivisão planar foram implementados.

A seguir, faz-se uma descrição dos operadores de Euler utilizados. Os argumentos destas funções são apresentados na forma de uma pseudo-linguagem similar à linguagem C.

A.1 - OPERADORES DE CONSTRUÇÃO

FED_M_M (*"Make Model"*)

Descrição funcional : cria um modelo.

Argumentos :

```
Model     newm;   /* ponteiro para o novo modelo */ (saída)
```

Este operador cria uma instância de um modelo topológico (**newm**).

FED_M_SFLV (*"Make Shell, Face, Loop and Vertex"*)

Descrição funcional : cria uma casca, face, ciclo e vértice.

Argumentos :

```
Shell     news;   /* ponteiro para a nova casca   */ (saída)
Face      newf;   /* ponteiro para a nova face    */ (saída)
Loop      newl;   /* ponteiro para o novo ciclo   */ (saída)
Vertex    newv;   /* ponteiro para o novo vértice */ (saída)
```

Teoricamente, todos os modelos planares podem ser obtidos através da manipulação de um modelo básico. A idealização de um sólido básico, topologicamente equivalente a uma esfera, constituído por uma casca, uma face, um vértice e um ciclo degenerado a este vértice, representa consistentemente este modelo. Embora este sólido básico não satisfaça a idéia intuitiva de um objeto sólido, ele é bastante útil como estado inicial da criação de um modelo através de uma sequência de operadores de Euler [MANT88].

O operador M_SFLV é responsável pela criação deste

objeto, que pode ser visto em uma última análise como a inicialização da estrutura de dados (Figura A.1).

Convém lembrar que a implementação atual é bidimensional, fazendo-se interpretar um modelo como uma subdivisão planar (projetada) da superfície do sólido básico.

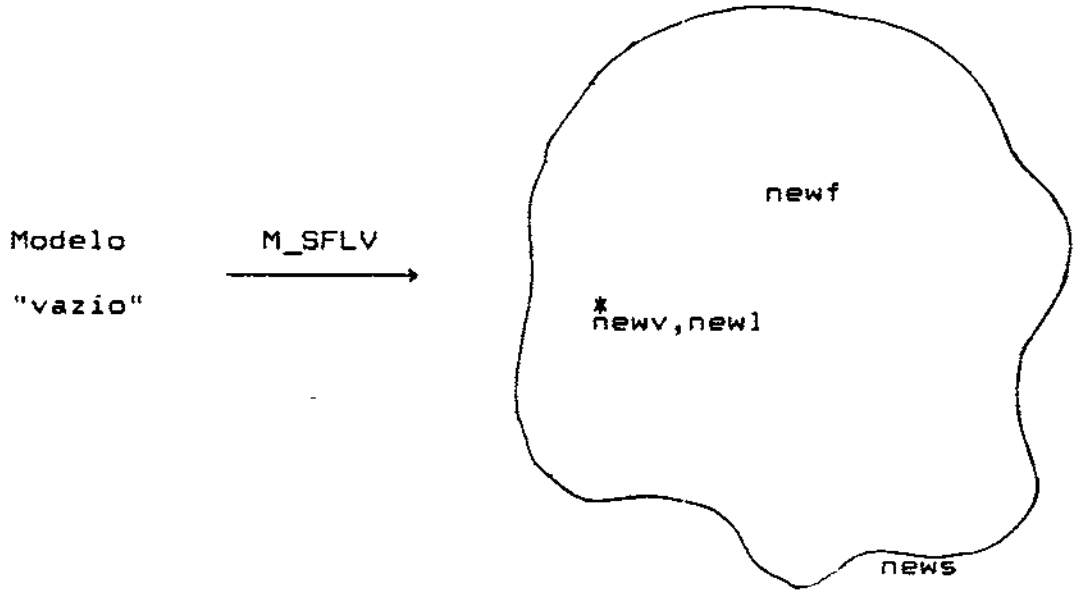


Figura A.1 - Operador M_SFLV.

FED_M_VL (*"Make Vertex and Loop"*)

Descrição funcional : cria um vértice e um ciclo.

Argumentos :

```
Face      f;      /* face de construção              */ (entrada)
Vertex    newv;   /* ponteiro para o novo vértice */ (saída)
Loop      newl;   /* ponteiro para o novo ciclo   */ (saída)
```

Este operador cria um novo vértice (**newv**) e o ciclo associado a ele (**newl**) em uma dada face (**f**) (Figura A.2).

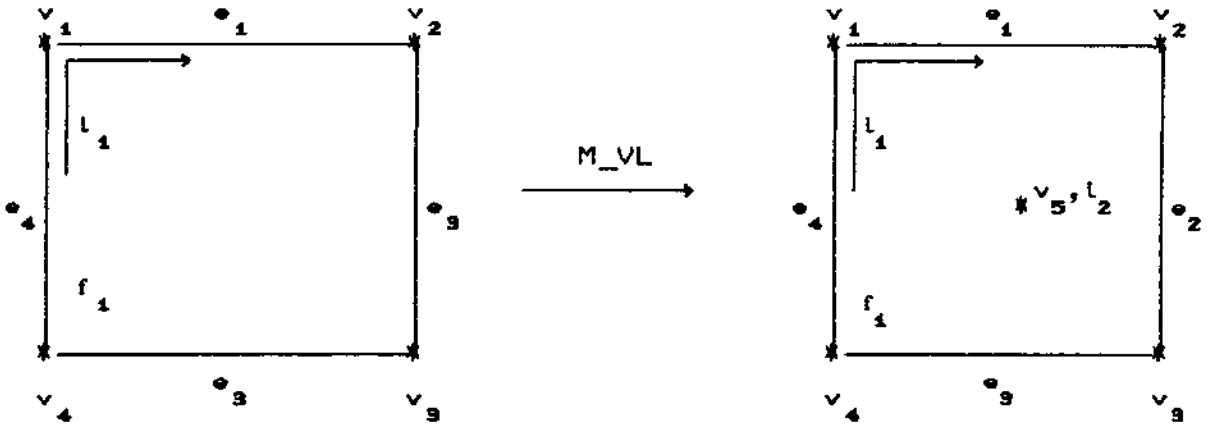


Figura A.2 - Operador M_VL.

FED_M_EV (*"Make Edge and Vertex"*)

Descrição funcional : cria uma aresta e um vértice.

Argumentos :

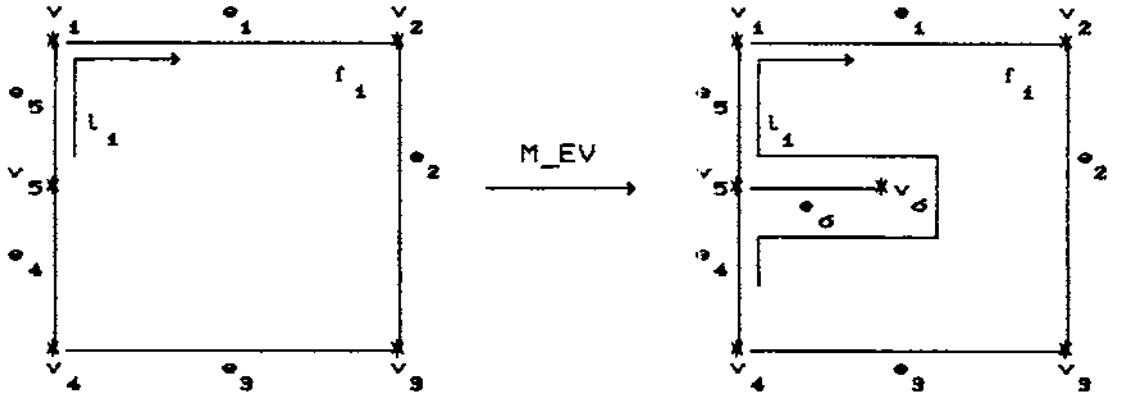
```
Vertex    v;    /* vértice existente                 */  (entrada)
Edge      e;    /* aresta-asa existente              */  (entrada)
Direction dir;  /* direção de inserção               */  (entrada)
Face      f;    /* face de construção                */  (entrada)
Edge      newe; /* ponteiro para a aresta criada     */  (saída)
Vertex    newv; /* ponterio para o vértice criado    */  (saída)
```

Este operador cria uma nova aresta (newe) que começa em um vértice existente (v) e termina no novo vértice criado (newv).

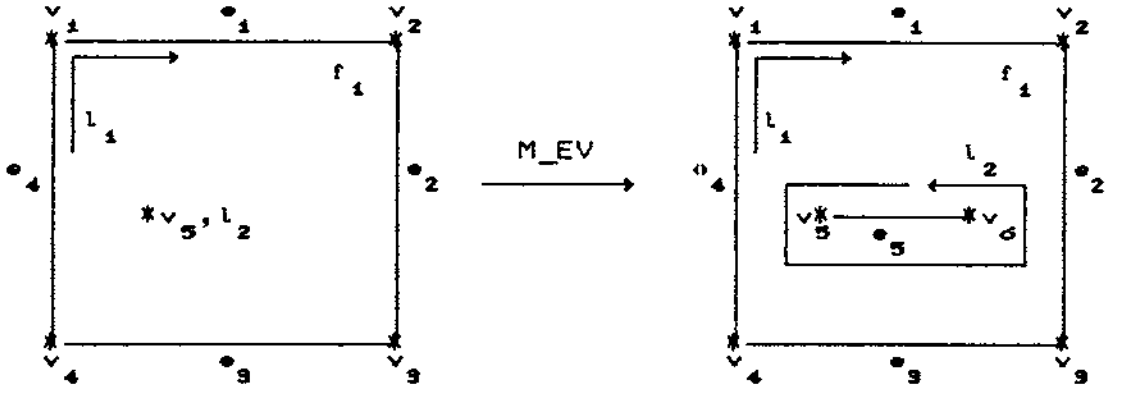
Se o vértice existente possui arestas adjacentes é fornecido como argumento a aresta-asa adjacente (e) e a direção de inserção (dir) da nova aresta (newe) (Figura A.3-a). A direção de inserção é definida pela posição da aresta e, em relação à aresta criada, em torno do vértice v, quando vista acima da superfície na vizinhança deste vértice. Esta direção

pode ser nos sentidos horário (CW) e anti-horário (CCW).

Se o vértice existente não possui arestas adjacentes, o campo aresta-asa não é especificado e a direção de inserção é não definida (*UNSP*) (Figura A.3-b).



(a)



(b)

Figura A.3 - Operador M_EV.

FED_M_E (*"Make Edge"*)

Descrição funcional : cria uma aresta .

Argumentos :

```
Vertex    vA;    /* vértice existente                 */  (entrada)
Edge      eA;    /* aresta-asa existente              */  (entrada)
Direction dirA;  /* direção de inserção               */  (entrada)
Vertex    vB;    /* vértice existente                 */  (entrada)
Edge      eB;    /* aresta-asa existente              */  (entrada)
Direction dirB;  /* direção de inserção               */  (entrada)
Face      f;     /* face de construção                */  (entrada)
Edge      newe;  /* ponteiro para a aresta criada     */  (saída)
Face      newf;  /* ponteiro para a face criada       */  (saída)
Loop      newl;  /* ponteiro para o ciclo criado      */  (saída)
```

O operador M_E cria uma aresta entre os vértices **vA** e **vB**, começando em **vA** e terminando em **vB**.

Se os argumentos **eA**, **dirA**, **eB** e **dirB** são especificados, a nova aresta será, consistentemente, adicionada de acordo com as direções de inserção fornecidas por estes argumentos. Neste caso, os vértices **vA** e **vB** e as arestas **eA** e **eB** devem fazer parte do contorno da face **f**. Senão ocorrerá uma condição de erro e a aresta não será adicionada.

No caso dos vértices especificados não possuirem arestas adjacentes, as arestas-asa e as direções associadas a elas não são definidas. Porém a condição destes vértices pertencerem a face **f** ainda é necessária para que a aresta possa ser adicionada ao modelo.

A adição de uma aresta pode ocasionar duas situações

distintas, tratadas pelas sub-classes deste operador descritas abaixo :

MEFL (*"Make Edge, Face and Loop"*)

Esta situação ocorre quando a adição da aresta isola uma parte da face existente provocando a criação de uma nova face (**newf**) e um ciclo associada a mesma (**newl**)(Figura A.4).

Se **vA** e **vB** são os mesmos vértices, é criada uma aresta de auto-ciclo. Neste caso, a referida aresta é orientada no sentido horário quando vista logo acima da superfície na vizinhança do vértice existente.

A aresta de auto-ciclo é sempre criada de maneira a sua primeira aresta-uso ficar associada ao ciclo interno, que será o novo ciclo criado (**newl**).

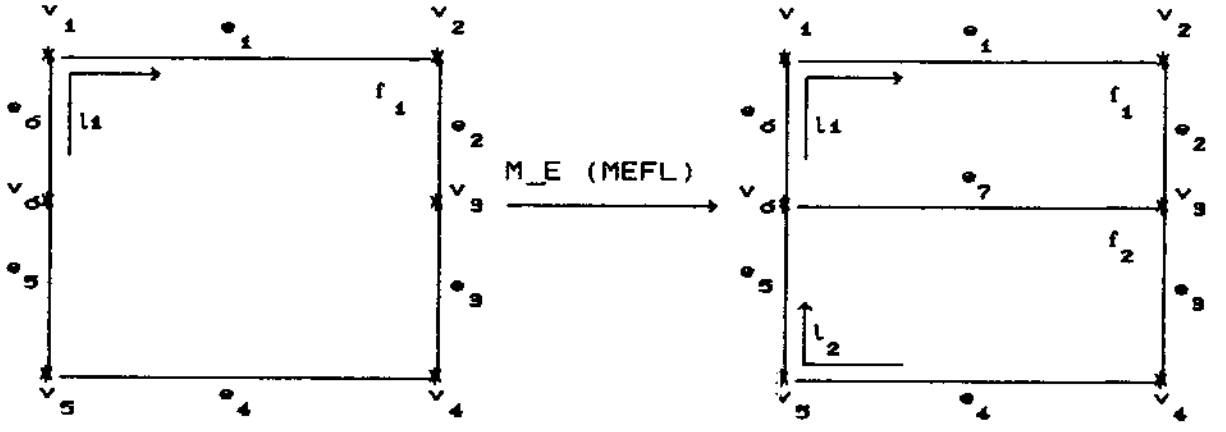


Figura A.4 - Operador M_E (sub-classe MEFL).

MEKL (*"Make Edge, Kill Loop"*)

Este sub-operador é usado quando a adição da nova aresta não isola parte da face existente. Isto ocorre quando os vértices **vA** e **vB** fazem parte de diferentes ciclos da mesma

face (f). A adição da nova aresta determina a eliminação de um destes ciclos, ficando como ciclo remanescente aquele associado ao vértice vA (Figura A.5).

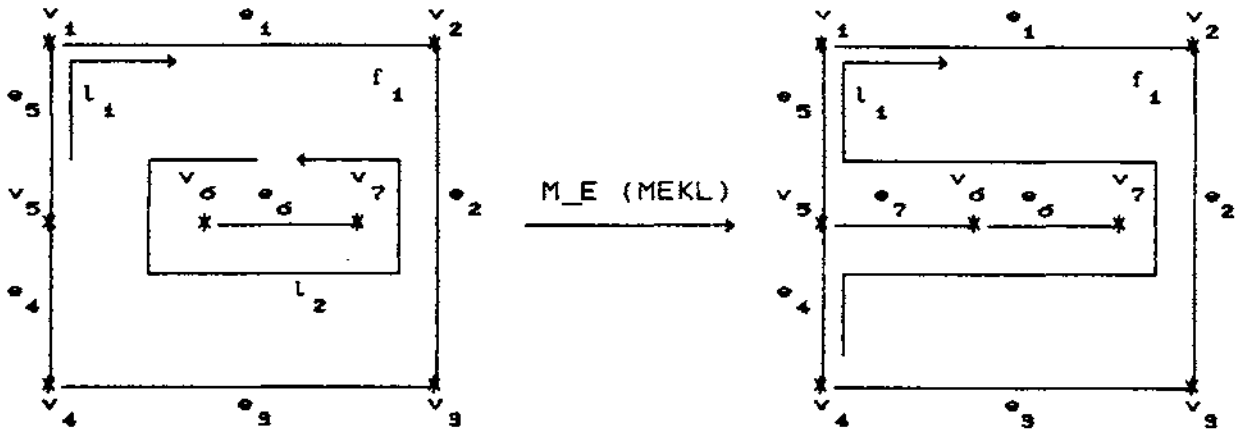


Figura A.5 - Operador M_E (sub-classe MEKL).

FED_E_SPLIT (*"Edge SPLIT"*)

Descrição funcional : divide uma aresta.

Argumentos :

```
Edge      e;      /* aresta a ser dividida          */(entrada)
Vertex    v;      /* vértice existente associado a
                     nova aresta                    */(entrada)
Edge      newe;   /* ponteiro para a aresta criada  */(saída)
Vertex    newv;   /* ponteiro para o vértice criado */(saída)
```

Este operador divide uma dada aresta (e) em duas arestas (e e newe) conectadas pelo novo vértice criado (newv) (Figura A.6).

O argumento v é opcional e, quando especificado, indica qual o vértice da aresta original (e) será associado à nova aresta (newe).

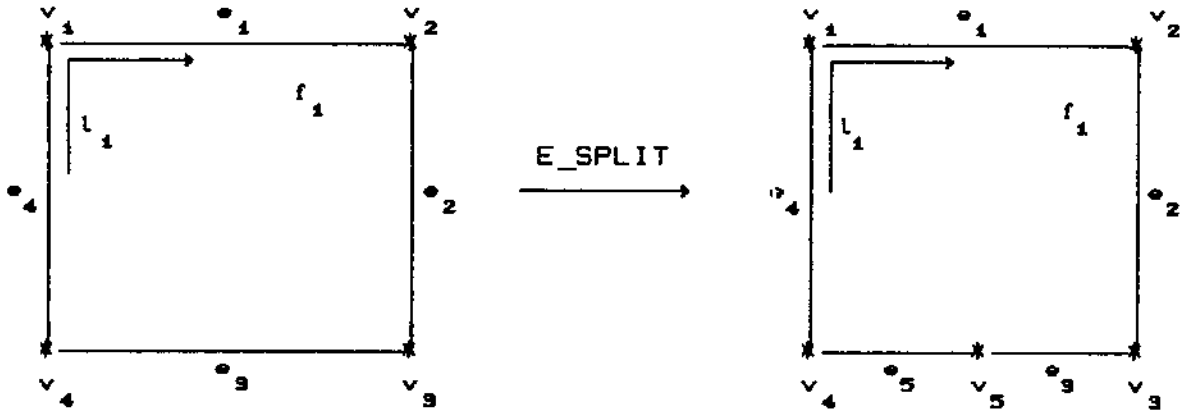


Figura A.6 - Operador E_SPLIT.

A.2 OPERADORES DE ELIMINAÇÃO

FED_K_E (*"Kill Edge"*)

Descrição funcional : elimina uma aresta.

Argumentos :

```
Edge      e;      /* aresta a ser eliminada        */ (entrada)
Face      f_surv; /* face remanescente             */ (entrada)
Loop      l;      /* ponteiro para o ciclo criado */ (saída)
```

Este operador elimina uma dada aresta (e). A eliminação de uma aresta envolve duas situações distintas. A cada situação corresponde um procedimento específico que são as sub-classes deste operador:

KEML (*"Kill Edge, Make loop"*)

Este procedimento é utilizado quando a aresta a ser eliminada pertence, exclusivamente, a uma face e a sua

eliminação ocasiona a divisão de um dos ciclos desta face, fazendo com que seja criado um novo ciclo (**newl**) (Figura A.7).

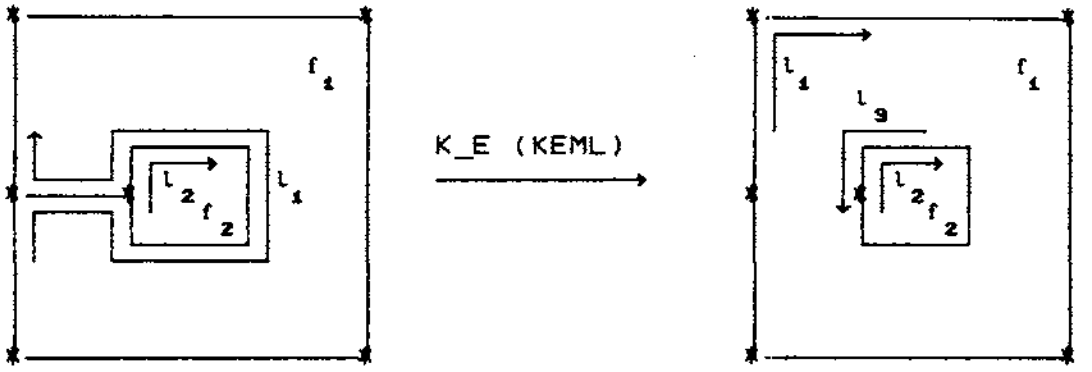


Figura A.7 - Operador K_E (sub-classe KEML).

KEFL (*"Kill Edge, Face and Loop"*)

Este sub-operador é usado quando a aresta pertence a faces distintas. Neste caso uma face é eliminada e a remanescente absorve os ciclos daquela eliminada (Figura A.8).

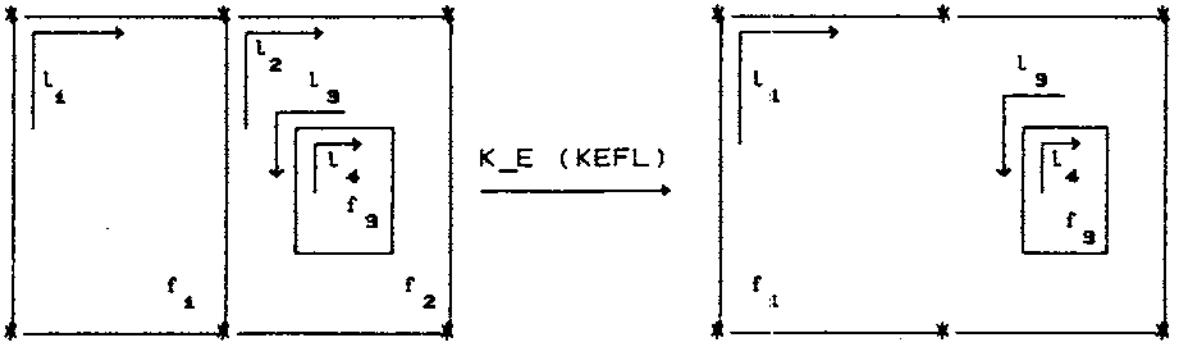


Figura A.8 - Operador K_E (sub-classe KEFL).

FED_K_V (*"Kill Vertex"*)

Descrição funcional : elimina um vértice.

Argumentos :

```
Vertex     v;        /* vértice a ser eliminado */ (entrada)
```

Este operador elimina um dado vértice (v) e todas as arestas adjacentes a ele.

Existem diversas situações que podem ocorrer com a eliminação de um vértice e suas arestas adjacentes. Cada situação é tratada por uma das sub-classes deste operador descritas abaixo :

KVFLE (*"Kill Vertex, Face, Loop and Edge"*)

Esta situação ocorre quando o vértice a ser eliminado tem uma ou mais arestas adjacentes que delimitam faces diferentes (Figura A.9). Neste caso, a eliminação do vértice determina a eliminação das aresta adjacentes, das faces adjacentes a estas arestas e dos ciclos associados às arestas eliminadas.

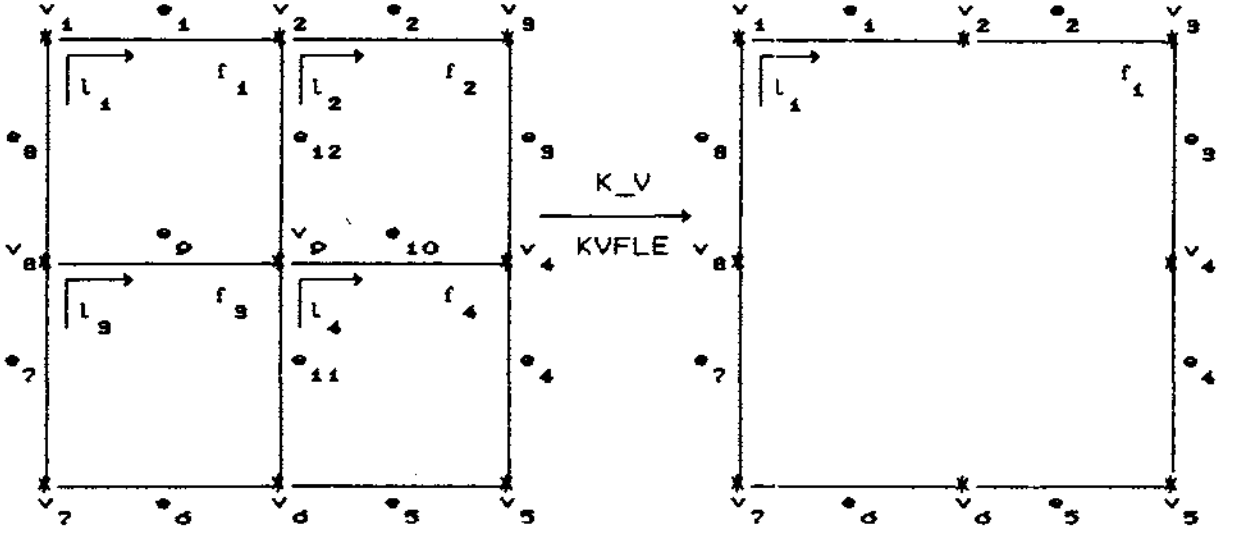


Figura A.9 - Operador K_V (sub-classe KVFLE).

KVE (*"Kill Vertex and Edge"*)

Este procedimento é utilizado quando o vértice a ser eliminado contém arestas adjacentes que pertencem exclusivamente a uma face.

Nesta implementação, primeiro elimina-se as arestas adjacentes com o operador K_E e depois, em função da situação gerada, elimina-se o vértice com um dos sub-operadores descritos abaixo :

KVL (*"Kill Vertex and Loop"*)

Este sub-operador é utilizado quando o vértice a ser eliminado não contém arestas adjacentes e define um ciclo em uma face que contém outros ciclos (Figura A.10).

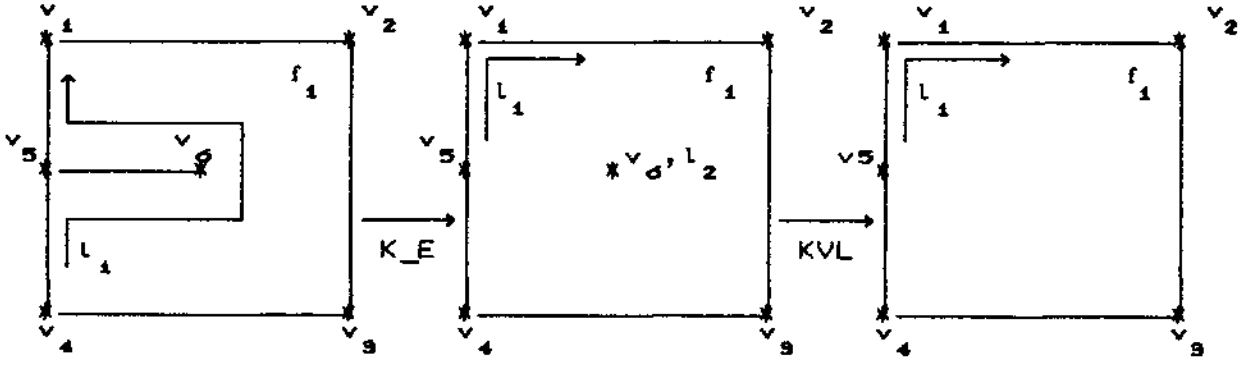


Figura A.10 - Operador K_V (sub-classe KVE-KVL).

KVSFL (*"Kill Vertex, Shell, Face and Loop"*)

Este sub-operador é utilizado quando o vértice a ser eliminado não contém arestas adjacentes e define um ciclo em uma face que não contém outros ciclos (Figura A.11). Neste caso, se este vértice é o único no modelo, este procedimento resulta em um modelo "vazio".

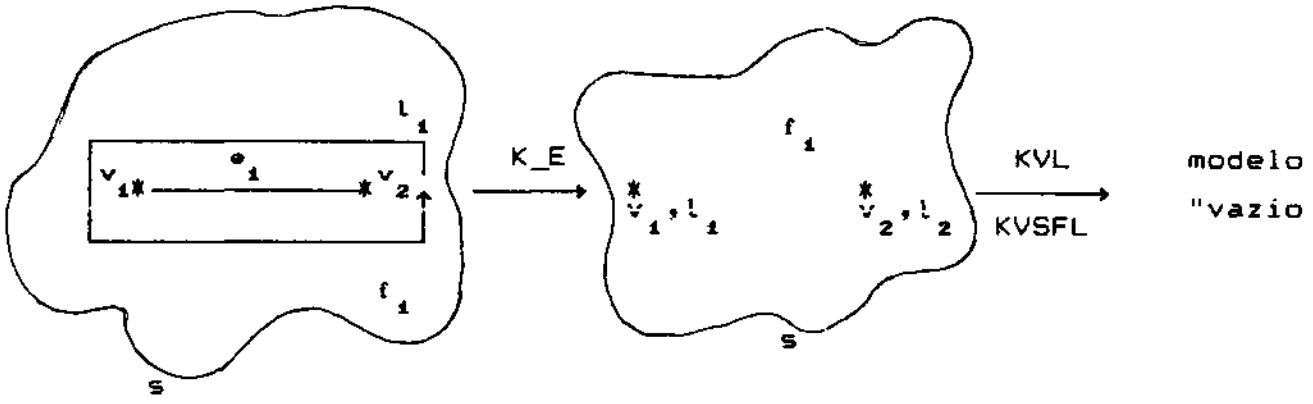


Figura A.11 - Operador K_V (sub-classe KVE-KVSFL).

FED_E_SQUEEZE (*"Edge SQUEEZE"*)

Descrição funcional : elimina uma aresta e funde os vértices extremos.

Argumentos :

```
Edge      e;      /* aresta a ser eliminada */ (entrada)
Vertex    v;      /* vértice remanescente   */ (entrada)
Vertex    v_surv; /* vértice remanescente   */ (saída)
```

Este operador une os vértices de uma dada aresta (e), eliminando esta aresta e um dos seus vértices extremos. O argumento **v** é opcional e indica qual o vértice desta aresta que deverá permanecer. O argumento **v_surv** sempre retorna o vértice remanescente.

Este operador é dividido em duas sub-classes em função do tipo da aresta a ser eliminada.

ESQUEEZEKEV (*"Edge SQUEEZE, Kill Edge and Vertex"*)

Este operador é utilizado quando a aresta (e) a ser eliminada não é uma aresta de auto-ciclo. Desta forma existem quatro situações possíveis com a eliminação desta aresta.

a) Ambos os vértices desta aresta estão desconectados dos demais vértices. Neste caso, um dos vértices é eliminado e o ciclo degenerado a um vértice é associado ao vértice remanescente (**v_surv**).

b) Se somente o vértice não remanescente está desconectado dos demais vértices, ele é simplesmente eliminado.

c) Se somente o vértice remanescente (**v_surv**) está desconectado dos demais vértices, as relações de adjacência do vértice não remanescente são transferidas para o referido vértice.

d) Se ambos os vértices estão conectados aos demais vértices, as relações de adjacências de ambos são fundidas no vértice remanescente.

ESQUEEZEKE (*"Edge SQUEEZE, Kill Edge"*)

Esta situação ocorre quando a aresta a ser eliminada é uma aresta de auto-ciclo. Neste procedimento, somente a aresta é eliminada.

APÊNDICE B

OPERADORES DE ALTO NÍVEL

Este apêndice faz uma descrição sucinta dos operadores de alto nível implementados no presente programa de geração de malhas. O objetivo destes operadores é promover uma interface entre o usúario e os operadores de Euler (Apêndice A), além de serem responsáveis pela criação e modificação das relações hierárquicas entre os diversos níveis (geometria, sub-região, sub-divisão e malha).

Os argumentos destas funções são apresentados na forma de uma pseudo-linguagem similar à linguagem C.

FED_A_M (*"Add Models"*)

Descrição funcional : adiciona modelos.

Este operador inicializa os modelos da geometria, sub-região, sub-divisão e malha.

FED_A_GxE (*"Add Geometric Edge"*)

Descrição funcional : adiciona uma aresta do tipo x no modelo da geometria.

Argumentos :

```
float     tol;     /* tolerância                    */
float     coords[];/* descrição geométrica da aresta */
```

Este operador adiciona uma aresta no nível da geometria. Cada tipo geométrico (reta, círculo e Bezier) é adicionado através dos respectivos operadores relacionados abaixo :

FED_A_GLE (*"Add Geometric Line Edge"*)

FED_A_GCE (*"Add Geometric Circular Edge"*)

FED_A_GBE (*"Add Geometric Bezier Edge"*)

O argumento **coords** define as coordenadas dos pontos que fazem a descrição geométrica da aresta. No caso da reta, tem-se as coordenada dos vértices extremos. No círculo, tem-se as coordenadas do centro do círculo e dos seus vértices extremos, que no caso do círculo completo são as mesmas. No caso da curva Bezier, tem-se as coordenadas dos pontos de controle da curva.

O argumento **tol** é utilizado para testar se os vértices extremos da aresta a ser adicionada estão suficientemente próximos a um vértice da geometria. Se isto ocorrer, as coordenadas dos vértices da aresta serão corrigidas de maneira a corresponderem exatamente às coordenadas dos referidos vértices. Caso contrário, os vértices da aresta são testados contra as arestas da geometria e, se existir pelo menos uma aresta suficientemente próxima a um destes vértices, é gerada

uma condição de erro, não sendo executada nenhuma ação.

Após os testes de proximidade o operador auxiliar **FED_A_GENEDGE** (*"Add GENeric EDGE"*) é chamado para traduzir a informação geométrica em informação topológica e adicionar a topologia da aresta na estrutura de dados. Depois, são definidos os atributos desta aresta em função do seu tipo geométrico.

É verificado, ainda, se a adição da nova aresta cria uma nova face. Se isto ocorrer, e existindo ciclos desconexos associados à face de construção que sejam envolvidos pela nova face, estes ciclos são então associados à referida face.

Por fim, é chamado o operador **FED_A_HxSE** (*"Add Hierarchical Subregion Edge"*) para adicionar a aresta correspondente no nível da sub-região e atribuir os ponteiros hierárquicos entre os novos vértices, aresta e faces (se criada) dos níveis da geometria e sub-região.

FED_A_SxE (*"Add Subregion Edge"*)

Descrição funcional : adiciona uma aresta do tipo x no modelo da sub-região.

Argumentos :

```
float     tol;      /* tolerância                          */
float     coords[]; /* descrição geométrica da aresta */
```

Este operador adiciona uma aresta no nível da sub-região. Em função do tipo geométrico, é utilizado um dos seguintes operadores:

FED_A_SLE (*"Add Subregion Line Edge"*)

FED_A_SCE (*"Add Subregion Circular Edge"*)

FED_A_SBE (*"Add Subregion Bezier Edge"*)

Estes operadores são semelhantes aos operadores do nível da geometria descritos anteriormente. Uma das grandes diferenças é que no teste de proximidade, se existir uma aresta suficientemente próxima a um dos vértices da aresta a ser adicionada, a aresta existente é, automaticamente, dividida em duas de maneira a acomodar a introdução deste novo elemento.

Caso os vértices da nova aresta não estejam próximos a nenhum outro vértice ou aresta, este operador testa se a localização destes vértices está dentro da fronteira geométrica do modelo. Um ponto é considerado dentro da fronteira geométrica do modelo, se a face a qual pertence não é a face externa livre ou uma face definida como furo no nível da geometria.

Este operador define, ainda, os ponteiros hierárquicos das novas entidades criadas entre os níveis da geometria e sub-região. Em seguida, o operador **FED_A_HSDE** (*"Add Hierarchical SubDivision Edge"*) é chamado para adicionar os vértices, arestas e faces (se criadas) nos níveis da sub-divisão e malha e para definir os ponteiros hierárquicos destas entidades entre os diversos níveis.

FED_A_ME (*"Add Mesh Edge"*)

Descrição funcional : adiciona uma aresta reta no modelo da malha.

Argumentos :

```
float     tol;      /* tolerância                          */
float     coords[]; /* descrição geométrica da aresta      */
int       quad;     /* indicador de elementos finitos
                       quadráticos                         */
```

Este operador adiciona uma aresta reta no nível da malha. O argumento **coords** especifica as coordenadas dos vértices da referida aresta.

Se o argumento **quad** é verdadeiro, indicando que a aresta a ser adicionada corresponde ao lado de um elemento finito quadrático, e existindo um vértice no nível da malha suficientemente próximo a um ponto no meio desta aresta, é gerada uma condição de erro e nenhuma ação é executada.

Este procedimento prossegue de forma semelhante ao operador **FED_A_GLE** com os testes de proximidade, adição do elemento topológico na estrutura de dados, definição dos atributos do novo elemento, teste se novas faces foram criadas e atribuição dos ponteiros hierárquicos.

Por fim, se esta aresta é o lado de um elemento finito quadrático ela é automaticamente dividida em duas arestas iguais.

Se, ainda, a adição desta aresta define uma face que é consistente com os tipos de elementos finitos implementados, é definido como atributo desta face o tipo correspondente: T3/T6 (triangular de três e seis nós respectivamente) ou Q4/Q8

(quadrangular de quatro e oito nós respectivamente).

FED_A_MF (*"Add Mesh Face"*)

Descrição funcional : adiciona uma face no modelo da malha.

Argumentos :

```
float     tol;       /* tolerância                    */
float     num_verts; /* numero de vértices da face    */
float     coords[];  /* coordenadas dos vértices      */
Face      sbv_face;  /* face "mãe" na sub-divisão     */
```

Este procedimento começa com a chamada do operador auxiliar **FED_A_GENLOOP** (*"Add GENeric LOOP"*), que traduz a informação geométrica em informação topológica e adiciona a topologia da nova face na estrutura de dados. Depois são definidos os atributos da face, entre eles o tipo de elemento finito que esta face representa. Este tipo é definido pelo argumento **num_verts**, que indica o número de vértices que compoêm esta face. Este número pode ser 3, 4, 6 ou 8 para elementos finitos do tipo T3, Q4, T6 e Q8, respectivamente.

Por fim, este operador define os ponteiros hierárquicos das novas entidades criadas (vértices, arestas e faces).

FED_A_HxSE (*"Add Hierarchical Subregion Edge"*)

Descrição funcional : adiciona hierarquicamente uma aresta do tipo geométrico x no nível da sub-região.

Argumentos :

```
float     tol;        /* tolerância                              */
float     coords[];   /* descrição geométrica da aresta          */
Vertex    geo_verts;  /* vértices geométricos correspondentes    */
Edge      geo_newe;   /* aresta geométrica correspondente        */
Face      const_face; /* face de construção na geometria         */
Face      geo_newf;   /* nova face criada na geometria           */
```

Este procedimento é chamado pelo operador **FED_A_GxE** para adicionar no nível da sub-região uma instância da aresta adicionada na geometria. Em função do tipo geométrico da aresta, é utilizado um dos operadores relacionados abaixo :

FED_A_HSLE (*"Add Hierarchical Subregion Line Edge"*)

FED_A_HSCE (*"Add Hierarchical Subregion Circular Edge"*)

FED_A_HSBE (*"Add Hierarchical Subregion Bezier Edge"*)

Estes operadores são semelhantes aos operadores **FED_A_SxE** descritos anteriormente. Porém, aqui, as entidades topológicas do nível da geometria que fazem a ligação hierárquica com o modelo da sub-região são fornecidas como argumentos.

FED_A_HSDE (*"Add Hierarchical SubDivision Edge"*)

Descrição funcional : adiciona hierarquicamente arestas retas nos níveis da sub-divisão e malha.

Argumentos :

```
float     tol;            /* tolerância                          */
float     coords[];       /* coordenadas dos novos vértices      */
float     const_coords[];/* coordenadas de um ponto na face
                             de construção                       */
Vertex    subr_verts;     /* vértices da sub-região
                             correspondente                      */
Edge      subr_newe;      /* aresta da sub-região
                             correspondente                      */
Face      subr_constf;    /* face de construção na sub-região */
Face      subr_newf;      /* nova face criada na sub-região      */
```

Este operador, chamado pelos operadores **FED_A_SxE** e **FED_A_HSxE**, é responsável pela criação das arestas nos níveis da sub-divisão e malha e pela manutenção dos ponteiros hierárquicos entre as entidades dos níveis da sub-região, sub-divisão e malha.

Uma característica importante dos níveis da sub-divisão e malha é que eles só suportam a representação de arestas retas. Este operador é, então, responsável por converter as arestas curvas, criadas nos níveis superiores, em uma sequência de arestas retas, de maneira a manter a consistência entre os modelos.

FED_D_GE (*"Delete Geometric Edge"*)

Descrição funcional : elimina uma aresta no nível da geometria.

Argumentos :

```
float     tol;      /* tolerância */
float     coords[];/* coordenadas de um ponto próximo à aresta */
```

Este operador elimina uma aresta na geometria. Para a eliminação da aresta da estrutura de dados é utilizado o operador de Euler **FED_K_E** (*"Kill Edge"*). Se, após a eliminação da aresta, pelo menos um dos seus vértices ficar desconectado das demais arestas, este vértice é também eliminado.

Este procedimento chama ainda os operadores **FED_D_HSE** (*"Delete Hierarchical Subregion Edge"*) e **FED_D_HSDE** (*"Delete Hierarchical SubDivision Edge"*) que eliminam as arestas correspondentes no níveis da sub-região, sub-divisão e malha.

Por fim, se existirem malhas de elementos finitos nas faces adjacentes às arestas eliminadas, estas malhas serão igualmente eliminadas.

FED_D_SE (*"Delete Subregion Edge"*)

Descrição funcional : elimina uma aresta no nível da sub-região.

Argumentos :

```
float     tol;      /* tolerância */
float     coords[];/* coordenadas de um ponto próximo a aresta */
```

Este operador elimina uma aresta no nível da sub-região. Como primeiro passo, este procedimento testa se a aresta a ser eliminada tem uma instância no nível da geometria. Este teste é feito consultando-se o ponteiro hierárquico desta aresta que

faz a relação com o nível superior. Se a referida aresta apontar para uma aresta da geometria, é gerada uma condição de erro, não sendo executada nenhuma ação.

Após a eliminação da aresta, é consultado o campo *split_flag* dos seus respectivos vértices. Se um destes campos é verdadeiro, significa que este vértice foi criado por um processo de divisão de aresta. Neste caso, é verificado se a aresta eliminada é uma das arestas geradas pela divisão da aresta original. Em caso positivo, o campo *split_flag* do referido vértice é definido como falso. As arestas originárias de um processo de divisão estão armazenadas no registro do respectivo vértice. Caso a aresta eliminada não seja uma destas arestas e só existam duas arestas adjacentes a este vértice, estas arestas são unidas e o vértice é eliminado.

A eliminação da aresta causa, ainda, a eliminação das arestas correspondentes nos níveis da sub-divisão e malha, e, caso existam malhas de elementos finitos nas faces adjacentes à aresta eliminada, estas malhas são igualmente eliminadas.

FED_D_HSE (*"Delete Hierarchical Subregion Edge"*)
Descrição funcional : elimina hierarquicamente arestas no nível da sub-região.
Argumentos :

```
Edge      geom_e;          /* aresta "mãe" na geometria   */
Vertex    geom_v0,geom_v1; /* vértices da aresta "mãe" na
                              geometria                   */
```

Este operador é chamado pelo operador **FED_D_GE** após a eliminação da aresta da geometria. Ele percorre a lista de

arestas da sub-região e elimina todas as arestas que se enquadram em uma das três situações:

a) A aresta aponta para cima para a dada aresta da geometria (geo_e).

b) Se um dos seus vértices aponta para a dada aresta da geometria (geo_e) e o outro não aponta para outra aresta geométrica.

c) Se um dos seus vértices aponta para um dos dados vértices geométricos (geom_v0 ou geom_v1) e o outro não aponta para uma aresta da geometria.

A eliminação da aresta causa, ainda, nos níveis da sub-divisão e malha, a eliminação das arestas correspondentes. Caso existam malhas de elementos finitos nas faces adjacentes às arestas eliminadas, elas são, igualmente, eliminadas.

FED_D_HSDE (*"Delete Hierarchical SubDivision Edge"*)

Descrição funcional : elimina hierarquicamente arestas nos níveis da sub-divisão e malha.

Argumentos :

```
Edge      subr_e;  /* aresta "mãe" na sub-região      */
Face      subr_f;  /* face remanescente na sub-região */
```

Este operador é chamado pelos operadores **FED_D_SE** ou **FED_D_HSE** descritos anteriormente. Após a eliminação da aresta da sub-região, este operador elimina todas as arestas correspondentes nos níveis da sub-divisão e da malha.

Por fim, são definidos os ponteiros hierárquicos entre as faces remanescentes nos níveis da sub-região, sub-divisão e malha.

FED_D_MSHAREA (*"Delete MeSH AREA"*)

Descrição funcional : elimina arestas e vértices no nível da malha .

Argumentos :

```
Face      subdv_face; /* face "mãe" na sub-divisão */
```

Este operador elimina todas as arestas e vértices em uma área no nível da malha que corresponda àquela definida pela dada face da sub-divisão (**subdv_face**).

A eliminação das arestas é feita percorrendo todas as arestas do nível da malha e eliminado aquelas que apontam para a dada face da sub-divisão. Os vértices que ficam desconectados das demais arestas e vértices são automaticamente eliminados.

Este operador poderia se tornar mais eficiente se ao invés percorrer a lista de todas as aresta, fossem utilizadas as informações de adjacência disponíveis, com a finalidade de se identificar um grupo mais restrito de arestas a serem testadas.

FED_S_SE (*"Split Subregion Edge"*)

Descrição funcional : divide uma aresta da sub-região.

Argumentos :

```
float     tol;   /* tolerância */
float coords[];  /* coordenadas do ponto de localização */
```

Este operador divide uma aresta do nível da sub-região, suficientemente próxima ao ponto de localização, em duas arestas do mesmo tipo geométrico da aresta original. As coordenadas do novo vértice, criado pela divisão da aresta, correspondem ao ponto sobre a aresta que se encontra mais

próximo ao ponto de localização. Este vértice tem, ainda, seu campo *split_flag* definido como verdadeiro, indicando que este vértice foi criado através do processo de divisão de uma aresta.

As arestas correspondentes nos níveis da sub-divisão e da malha são igualmente divididas e os ponteiros hierárquicos entre os níveis da sub-região, sub-divisão e malha são redefinidos.

FED_SQZ_SE ("*SQueeZe Subregion Edge*")
Descrição funcional : funde arestas da sub-região.
Argumentos :

```
float      tol;    /* tolerância */
float coords[];  /* coordenadas do ponto de localização */
```

Este operador une um grupo de arestas do nível sub-região em uma única aresta do mesmo tipo geométrico. Somente as arestas consecutivas deste grupo geradas pelo mesmo processo de divisão são unidas. Outra condição para que cada par destas arestas sejam unidas é que o vértice que elas compartilham só tenha como arestas adjacentes as referidas arestas.

As arestas correspondentes no nível da sub-divisão e malha são, igualmente, unidas e seus ponteiros hierárquicos apontam para a nova aresta resultante do processo no nível superior.

FED_SD_SDE (*"SubDivide SubDivision Edge"*)

Descrição funcional : divide aresta no nível da sub-divisão.

Argumentos :

```
Edge    subr_e; /* aresta "mãe" na sub-região                    */
int     n_seg;  /* número de segmentos                           */
int     quad;   /* indicador de elementos finitos
                   quadráticos                                   */
float   ratio;  /* razão entre os comprimentos da primeira
                   e última arestas                              */
```

Este operador re-subdivide um grupo de arestas do nível da sub-divisão que apontam para uma dada aresta do nível da sub-região (**subr_e**).

Este procedimento começa com a união de todas as arestas dos níveis da sub-divisão e da malha, que correspondam à aresta da sub-região, em apenas um só segmento. Depois, em função do número de segmentos (**n_seg**) e da razão entre os comprimentos da primeira e última arestas (**ratio**), são geradas as coordenadas dos vértices das novas arestas nos níveis inferiores. Se, ainda, o argumento **quad** é verdadeiro, indicando que as arestas geradas no nível da malha correspondem aos lados de elementos finitos quadráticos, as referidas arestas são divididas em dois segmentos de igual comprimento.

Por fim, este operador define os ponteiros hieráquicos entre a aresta da sub-região e as novas arestas dos níveis da sub-divisão e malha.

REFERENCIAS BIBLIOGRAFICAS

[BAUM75] Baumgart,B.G.,"A Polyhedron Representation for Computer Vision," AFIPS Proc., 44, 589-596 (1975).

[BOHM84] Böhm, W., Farin, G. and Kahmann, J., "A Survay of Curve and Surface Methods in CAGD," Comput. Aided Geom. Design", 1, 1-60 (1984).

[CARV90] Carvalho, P.C.P., Gattass., M. and Martha, L.F.," A Software Tool Which Allows Interactive Creation of Planar Subdivisions and Aplications to Educational Programs," International Conference on Computer Aided Training in Science and Technology, Barcelona, July, 201-207 (1990).

[CAVE85] Cavendish, J.C., Field, D.A. and Frey, W.H., "An Aproach to Automatic Three-Dimensional Finite Element Mesh Generation," Int. J. Num. Meth. Engng., 21, 329-347 (1985).

[CHEW89] Chew, L.P., "Constrained Delaunay Triangulations," Algorithmica, 4, 97-108 (1989).

[FARI88] Farin, G. , Curves and Surfaces for Computer Aided Geometric Design, Academic Press, London, 1988.

[FOLE90] Foley, J.D., Vam Dam, A., Feiner, S.K., & Hughes,J.F., Computer Graphics Principles and Practice, Addison-Wesley, Reading, MA, 1990.

[HABE81] Haber, R., Shephard, M.S., Greenberg, D.P., Abel, J.F. and Gallagher, R.H., "A General Two-dimensional, Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings," Int J. Num. Meth. Engng., 17, 1015-1044 (1981).

[HARR88] Harrington, S., Computer Graphics - A Programming Aproach, McGraw-Hill Book Company, NY, 1988.

[HOPG86] Hopgood, F.R.A., Duce, D.A., Gallop, J.R. & Sutcliffe, D.C., Introduction to the Graphical Kernel System (GKS), Academic Press, London, 1988.

[KELA87] Kela, A., Saxena, M. and Peruchio, R., "A Hierarchical Structure for Automatic Meshing and Adaptive FEM Analysis", Engng. Comput., 4, 104-112 (1987).

[MANT82] Mäntylä, M. and Sulonen, R., "GWB: A Solid Modeler with Euler Operators," IEEE Comput. Graph. And Appl., 2, 17-31 (1982).

[MANT88] Mäntylä, M., An Introduction to Solid Modeling, Computer Science Press, Rockville, Maryland, 1988.

[MART87] Martha, L.F., "A Two-Dimensional Mesh Generator Based on a Topological Data Structure," Notas Pessoais, 1987.

[MART89] Martha, L.F., "Topological and Geometrical Modeling Approach to Numerical Discretization and Arbitrary Fracture Simulation in Three-Dimensions", Ph.D. Dissertation, Cornell University, Ithaca, NY, 1989.

[MORT85] Mortenson, M.E., Geometric Modeling, John Wiley & Sons, New York, NY, 1985.

[NEWM81] Newman, W.M. & Spoull, R.F., Principles of Interactive Computer Graphics, McGraw-Hill Book Company, Auckland, 1981.

[PREP85] Preparata, F.P. and Ian Shamos, M., Computational Geometry - An introduction, Springer-Verlag, New York, NY, 1985.

[SCHI87] Schildt, H., C Avançado - Guia do Usuário, McGraw-Hill, São Paulo, 1987.

[SCHI87] Schildt, H., Turbo C - Guia do Usuário, McGraw-Hill, São Paulo, 1989.

[SCHR88] Schroeder, W.J. and Shepard, M.S., "Geometry-Based Fully Automatic Mesh Generation and the Delaunay Triangulation,"Int. J. Num. Meth. Engng., 26, 2503-2515 (1988).

[SHAW78] Shaw, R.D. and Pitchen, R.G., "Modifications to the Suhara-Fukuda Method of Network Generation," Int. J. Num. Meth. Engng., 12, 93-99 (1978).

[STEF85] Stefik, M. & Bobrow, D.G., " Object-Oriented Programming: Themes and Variations," The AI Magazine, 40-62, (1985).

[TECG89] Grupo de Tecnologia em Computação Gráfica, GKS/PUC, Manual de Utilização e Referência, Versão 3.0, Pontifícia Universidade Católica do Rio de Janeiro, 1989.

[WAWR87a] Wawrzynek, P.A. and Ingraffea,A.R., " An Edge-Based Data Structure for Two-Dimensional Finite Element Analysis," Engng. with Comput., 3, 13-20 (1987).

[WAWR87b] Wawrzynek, P.A. and Ingraffea, A.R., " Interactive Finite Element Analysis of Fracture Processes: An Integrated Approach," Theor. Appl. Fract. Mech., 8, 137-150 (1987).

[WEIL85] Weiler, K., "Edge-Based Data Structure for Solid Modeling in Curved-Surface Environments," IEEE Comput. Graph. Appl., 5, 21-40 (1985).

[WEIL86] Weiler, K., "Topological Structure for Geometric Modeling," Ph.D. Dissertation, Rensselaer Polytechnical Institute, Troy, NY, 1986.

[WILS85] Wilson,P.R., "Euler Formulas and Geometric Modeling," IEEE Comput. Graph. and appl., 24-36, August 1985.

[YERR84] Yerry, M.A. and Shepard, M.S., "Automatic Three-Dimensional Mesh Generation by Modified Octree Technique," Int. J. Num. Meth. Engng., 20, 1965-1990 (1984).

[YERR85] Yerry, M.A. and Shepard, M.S., "Automatic Mesh Generation for Three-Dimensional Solids," Comput. and Struct., 20, 31-39 (1985).

"Geração de Malhas de Elementos Finitos Bidimensionais Baseada em uma Estrutura de Dados Topológica". Dissertação de Mestrado apresentada por JORGE ALBERTO PRADO DE CAMPOS em 21 de Março de 1991 ao Departamento de Engenharia Civil da PUC/Rio e aprovada pela Comissão Julgadora, formada pelos seguintes professores:

Prof. Luiz Fernando Campos Ramos Martha (Orientador)
Departamento de Engenharia Civil - PUC/Rio

Prof. Marcelo Gattass
Departamento de Engenharia Civil - PUC/Rio

Prof. Ney Augusto Dumont
Departamento de Engenharia Civil - PUC/Rio

Prof. Ronaldo Marinho Persiano
Prog. de Engenharia de Sistemas e Computação - COPPE/UFRJ

Vista e permitida a impressão

Rio de Janeiro, 06 / 05 / 91

Prof. Francisco Eduardo M. Saboya
Coordenador dos Programas de Pós-Graduação
do Centro Técnico Científico