Pontifícia Universidade Católica
do Rio de Janeiro

Marcos Chataignier de Arruda

Operações booleanas com sólidos compostos representados por fronteira

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Civil da PUC-Rio.

Orientador: Luiz Fernando C. R. Martha

Rio de Janeiro, janeiro de 2005

Pontifícia Universidade Católica
do Rio de Janeiro

Marcos Chataignier de Arruda

Operações booleanas com sólidos compostos representados por fronteira

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Civil da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Luiz Fernando C. R. Martha
Presidente/Orientador
Departamento de Engenharia Civil – PUC-Rio

Prof. Waldemar Celes Filho
Departamento de Informática – PUC-Rio

Prof. Luiz Cristovão G. Coelho
TecGraf/PUC-Rio

Prof. Paulo Cezar P. Carvalho
IMPA

Prof. Marcelo de Andrade Dreux
Departamento de Engenharia Mecânica - PUC-Rio

Prof. José Eugênio Leal
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 28 de janeiro de 2005

Marcos Chataignier de Arruda

Graduou-se em Engenharia Civil, ênfase em Estruturas, pela PUC-Rio – Pontifícia Universidade Católica do Rio de Janeiro em 2002. Desenvolveu seu trabalho de pesquisa com ênfase em computação gráfica aplicada.

Ficha Catalográfica

Arruda, Marcos Chataignier de

Operações booleanas com sólidos compostos repre sentados por fronteira / Marcos Chataignier de Arruda ; orientador: Luiz Fernando C. R. Martha. – Rio de Janeiro : PUC-Rio, Departamento de Engenharia Civil, 2005.

v., 267 f. : il. ; 29,7 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil.

Inclui referências bibliográficas

1. Engenharia Civil – Teses. 2. Operações booleanas. 3. Modelagem geométrica. 4. Topologia de Adjacência. 5. Estruturas de dados topológicos. 6. Sólidos non-manifold. 7. Representação por fronteira. I. Martha, Luiz Fernando C. R. II. Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil. III. Título.

CDD:624

Agradecimentos

Aos meus pais Lucia e Miguel, às minhas irmãs Isabela e Tatiana e a todos os meus familiares, pelo apoio, paciência e incentivo demonstrados ao longo do desenvolvimento deste trabalho.

A minha namorada Christina, pelo apoio, paciência e por todos os fins-de-semana que passou ao meu lado dedicando seu tempo, sua atenção, seu carinho e suas palavras de incentivo e encorajamento para que este trabalho pudesse ser concluído da melhor forma possível e dentro do prazo.

Ao professor Luiz Fernando Martha, orientador deste trabalho, pelo incentivo, ensinamento, orientação e, principalmente, pela amizade e dedicação demonstradas ao longo do desenvolvimento deste trabalho.

Ao amigo William, que poderia ser considerado um verdadeiro co-orientador deste trabalho, por toda a orientação, auxílio e paciência e pelas modificações e inovações promovidas no código-fonte do modelador MG para tornar possível a implementação do algoritmo proposto neste trabalho.

A amiga Christiana, por todo o auxílio prestado a mim durante o mestrado, mas principalmente pela sua amizade inestimável.

Ao meu cunhado Pedro, pelo auxílio prestado de forma extremamente eficiente na elaboração de várias figuras deste trabalho.

A todos os amigos do TecGraf e da PUC pela amizade consolidada durante os quatro anos durante os quais estive vinculado a estas instituições, e pelo apoio e colaboração no desenvolvimento deste trabalho.

A Ana Roxo e a todos os funcionários e professores do Departamento de Engenharia Civil da PUC.

Ao Tecgraf pelo apoio financeiro e tecnológico durante o curso de mestrado.

A CAPES e ao CNPq pelo apoio financeiro durante o curso de mestrado.

Resumo

Arruda, Marcos Chataignier de; Martha, Luiz Fernando C. R. **Operações booleanas com sólidos compostos representados por fronteira.** Rio de Janeiro, 2005. 267p. Dissertação de Mestrado - Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro.

Num modelador de sólidos, uma das ferramentas mais poderosas para a criação de objetos tridimensionais de qualquer nível de complexidade geométrica é a aplicação das operações booleanas. Elas são formas intuitivas e populares de combinar sólidos, baseadas nas operações aplicadas a conjuntos. Os tipos principais de operações booleanas comumente aplicadas a sólidos são: união, interseção e diferença. Havendo interesse prático, para garantir que os objetos resultantes possuam a mesma dimensão dos objetos originais, sem partes soltas ou pendentes, o processo de regularização é aplicado. Regularizar significa restringir o resultado de tal forma que apenas volumes preenchíveis possam existir. Na prática, a regularização é realizada classificando-se os elementos topológicos e eliminando-se estruturas de dimensão inferior. A proposta deste trabalho é o desenvolvimento de um algoritmo genérico que permita a aplicação do conjunto de operações booleanas em um ambiente de modelagem geométrica aplicada à análise por elementos finitos e que agregue as seguintes funcionalidades: trabalhar com um número indefinido de entidades topológicas (conceito de Grupo), trabalhar com objetos de dimensões diferentes, trabalhar com objetos non-manifold, trabalhar com objetos não necessariamente poliedrais ou planos e garantir a eficiência, robustez e aplicabilidade em qualquer ambiente de modelagem baseado em representação B-Rep. Neste contexto, apresenta-se a implementação do algoritmo num modelador geométrico pré-existente, denominado MG, seguindo o conceito de programação orientada a objetos e mantendo a interface com o usuário simples e eficiente.

Palavras-chave

Operações Booleanas, Modelagem Geométrica, Topologia de Adjacência, Estruturas de Dados Topológicas, Sólidos *Non-Manifold*, Representação por Fronteira.

Abstract

Arruda, Marcos Chataignier de; Martha, Luiz Fernando C. R. (Advisor). **Boolean operations with compound solids represented by boundary**. Rio de Janeiro, 2005. 267p. MSc. Dissertation – Civil Engineering Department, Pontifícia Universidade Católica do Rio de Janeiro.

In a solid modeler, one of the most powerful tools to create three-dimensional objects with any level of geometric complexity is the application of the Boolean set operations. They are intuitive and popular ways to combine solids, based on the operations applied to sets. The main types of Boolean operations commonly applied to solids are: union, intersection and difference. If there is practical interest, in order to assure that the resulting objects have the same dimension of the original objects, without loose or dangling parts, the regularization process is applied. To regularize means to restrict the result in a way that only filling volumes are allowed. In practice, the regularization is performed classifying the topological elements and removing the lower dimensional structures. The objective of this work is the development of a generic algorithm that allows the application of the Boolean set operations in a geometric modeling environment applied to finite element analysis, which aggregates the following functionalities: working with an undefined number of topological entities (Group concept), working with objects of different dimensions, working with non-manifold objects, working with objects not necessarily plane or polyhedrical and assuring the efficiency, robustness and applicability in any modeling environment based on B-Rep representation. In this context, the implementation of the algorithm in a pre-existing geometric modeler named MG is presented, using the concept of object oriented programming and keeping the user interface simple and efficient.

.

Key-Words

Boolean Set Operations, Geometric Modeling, Adjacency Topology, Topological Data Structures, *Non-Manifold* Solids, Boundary Representation.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

MEF	Método dos Elementos Finitos
MG	*Mesh Generator*
CGC	*Complete Geometric Complex*
POO	Programação Orientada a Objetos
ESAM	*Extensible System Attributes Management*
NURBS	*Non-Uniform Rational B-Splines*
CSG	*Constructive Solid Geometry*
B-REP	*Boundary Representation*
RED	*Radial Edge*

1
Introdução

No contexto da modelagem geométrica para elementos finitos, a malha de elementos finitos é definida a partir da descrição geométrica do domínio do problema que está sendo estudado. Sendo assim, a criação de um modelo geométrico que possa reproduzir os objetos reais de engenharia que precisam ser modelados torna-se uma etapa crucial no conjunto dos procedimentos necessários para se realizar uma análise completa utilizando-se o método dos elementos finitos (MEF).

Como muitos modelos em engenharia envolvem a criação de objetos extremamente complexos em sua forma e geometria, é necessária a utilização de modeladores que possam reproduzir, de forma completa, simples e eficiente, a geometria dos objetos em estudo. Além disso, é preciso que tais modeladores possuam recursos suficientes para manter uma interface amigável com o usuário e para que o mesmo possa reproduzir digitalmente o modelo geométrico que deseja analisar da forma mais simples e rápida possível.

As metodologias de modelagem existentes são diversas, e na literatura podem-se encontrar várias bibliotecas e aplicativos gráficos que adotam enfoques e representações de dados de diferentes tipos [1,2,3,4,5]. Segundo Lira [6], na simulação 3D de problemas de engenharia usando o MEF, qualquer metodologia de modelagem deve tratar dois diferentes aspectos: modelagem geométrica com detecção automática de múltiplas regiões fechadas e interseções de superfícies e suporte para geração automática de malhas de elementos finitos.

Este tipo de metodologia foi implementado num modelador geométrico tridimensional de elementos finitos denominado MG [7]. O MG é um pré-processador com suporte para geração de malhas em superfícies e malhas sólidas [8]. Atualmente, o MG incorpora a capacidade de realização de interseções entre superfícies paramétricas, entre superfícies paramétricas e curvas paramétricas e a detecção automática de múltiplas regiões fechadas a partir de retalhos de superfícies com geometria qualquer. As Figuras 1.1, 1.2 e 1.3 ilustram modelos gerados no MG que utilizam tais facilidades.

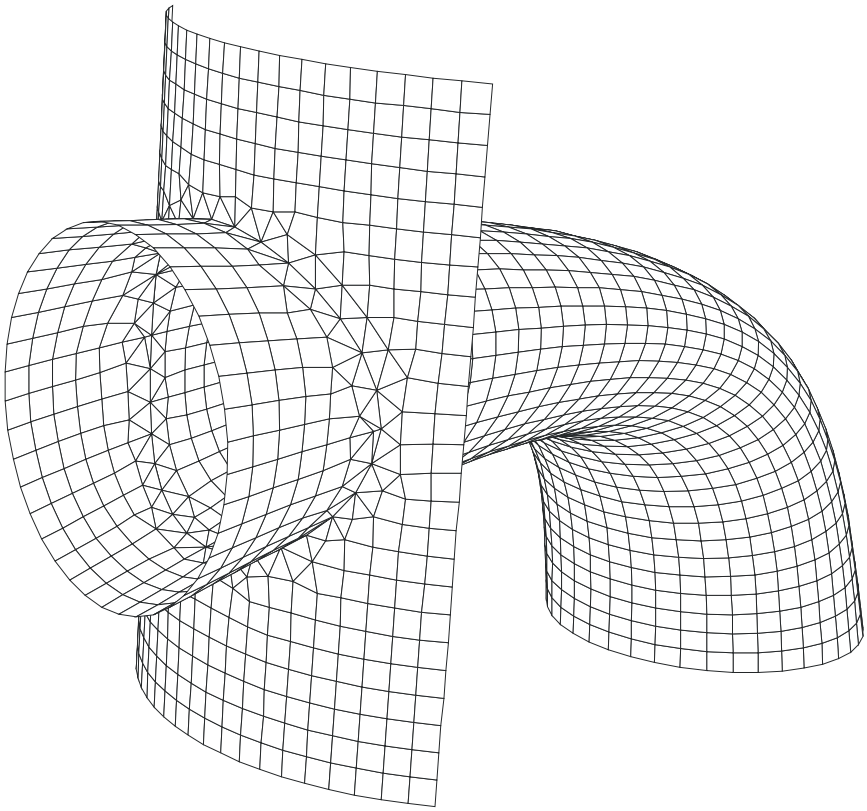


Figura 1.1 – Interseção entre malhas de superfícies [8].

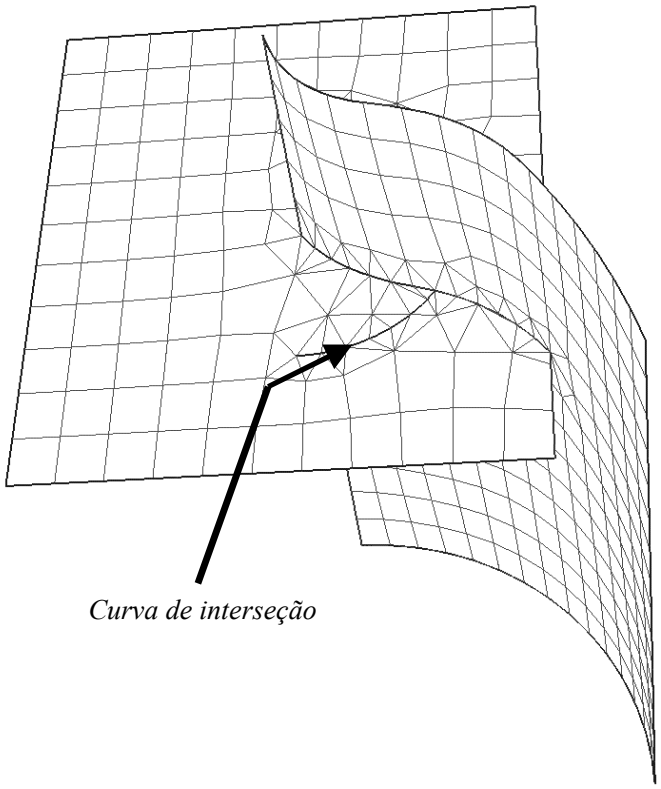


Figura 1.2 – Caso patológico de interseção de superfícies [6].

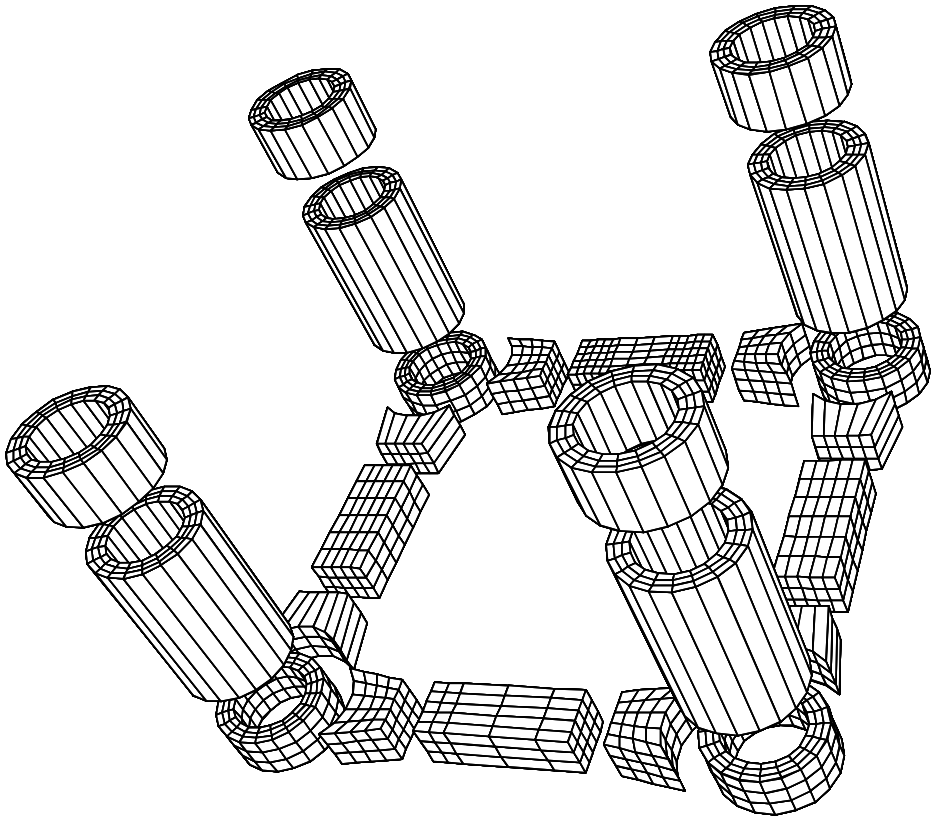


Figura 1.3 – Modelo explodido: detecção automática de regiões [6].

A estratégia de modelagem do MG pode ser classificada como uma Representação de Fronteira (*Boundary Representation* – B-Rep) [9,10], pois os volumes sólidos são representados explicitamente pelas curvas e superfícies na sua borda.

O paradigma de modelagem do MG, na sua versão original, era baseado na criação e manipulação das curvas e superfícies que compõem o modelo. Para tanto, o MG apresenta uma interface com o usuário amigável e eficiente. Este paradigma tem como principal vantagem o abrangente poder de representação, pois permite que praticamente qualquer tipo de geometria sólida seja representada, incluindo regiões múltiplas e degenerações de sólidos, tais como cascas representadas pela sua superfície média. O principal problema deste paradigma é que a modelagem depende muito da intervenção do usuário para a determinação de todas as curvas do modelo, o que poderia ser mais facilmente gerado com interseções automáticas.

A organização interna do MG segue o conceito de Programação Orientada a Objetos, com uma estruturação interna dos dados de forma simples e organizada. Com o trabalho de Lira [6], o MG incorporou o uso de diversas bibliotecas auxiliares com implementações computacionais já comprovadamente

eficientes e robustas, resultando em um modelador cada vez mais poderoso no contexto da modelagem B-Rep de sólidos e na geração automática de malhas bi e tridimensionais.

Entretanto, para que a aplicabilidade do MG no campo da engenharia se tornasse ainda maior, de modo que a criação de modelos complexos pudesse ser realizada sem exigir do usuário tempo e um número de passos de modelagem excessivos, era necessário que o MG passasse a incorporar um outro paradigma de modelagem, já presente em outros modeladores: as operações booleanas. Neste paradigma, utilizam-se primitivas sólidas simples para obterem-se modelos com geometrias mais complicadas, desta forma minimizando as intervenções do usuário no processo de modelagem. O paradigma de modelagem através de operações booleanas é típico de modeladores cuja representação não é baseada na representação explícita da fronteira, mas sim na representação direta do conjunto de pontos que formam o sólido. Este tipo de representação é chamado de CSG (*Constructive Solid Geometry*) [11]. Esta representação é dita *implícita* pois a fronteira não é representada explicitamente, ao passo que a representação B-Rep é chamada de explícita.

Deve-se ressaltar que as operações booleanas apresentam-se apenas como um paradigma de modelagem, não sendo exclusivas de representações CSG. No contexto deste trabalho, o que se buscou foi incorporar este paradigma de modelagem a um modelador com representação B-Rep com capacidade de representar múltiplas regiões. Para manter o caráter genérico das outras ferramentas implementadas no MG, sem que restrições tenham que ser impostas ao domínio dos problemas já tratados, era necessário que estas operações booleanas fossem implementadas de forma que quaisquer objetos, em qualquer número e com qualquer dimensão, pudessem ser combinados com outros objetos com as mesmas características, gerando um resultado válido e passível de ser representado topológica e geometricamente pela representação de fronteira do MG.

É fácil notar que para que isto fosse possível, as ferramentas já mencionadas relativas à interseção de superfícies paramétricas e detecção automática de regiões já deveriam estar implementadas no MG, do contrário a generalização requerida não seria possível de ser atingida. Tais ferramentas aliadas a algoritmos geométricos presentes nas bibliotecas utilizadas pelo MG permitiram que o conjunto das operações booleanas pudesse ser implementado de forma razoavelmente simples e extremamente eficiente.

1.1. Motivações e Trabalhos Anteriores

Desde que a primeira versão do MG [8] foi implementada, sempre se buscou manter como meta a simulação de problemas tridimensionais reais de engenharia pelo MEF, através da modelagem geométrica e da geração de malhas de elementos finitos. A versão original já possuía uma interface amigável e eficiente, sendo bastante satisfatória na representação dos modelos geométricos desejados, contudo possuía certas limitações que ao longo do tempo tiveram de ser eliminadas.

Uma destas limitações diz respeito à maneira como foi implementada a sua estrutura interna de dados. Não havia um formalismo desejável para atender a todas as capacidades de modelagem necessárias, de modo que muitas vezes a intervenção do usuário se fazia necessária para que a consistência geométrica do modelo fosse mantida. Na Figura 1.4 pode-se vislumbrar um problema real de engenharia em que a detecção automática de regiões se faz bastante necessária, visto que a determinação explícita dos retalhos de superfície que formam um volume fechado é uma tarefa não-trivial.

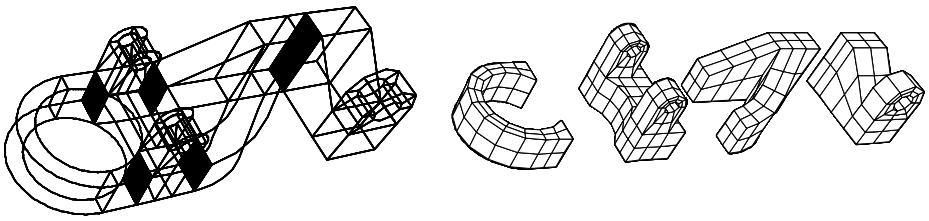


Figura 1.4 – Dificuldade para detecção de multi-regiões [6].

Entre Coelho [8] e Lira [6], pode-se perceber a evolução do modelador MG desde a sua versão original, valendo ressaltar a reestruturação interna do código seguindo o conceito de POO, o que proporcionou ao MG um nível de organização bastante significativo, e a utilização de uma metodologia de representação topológica denominada CGC (*Complete Geometric Complex*) [12], responsável pela criação e manutenção de subdivisões espaciais. Atualmente, o MG pode ser classificado como um modelador de sólidos *non-manifold* [12,13,14,15,16,17,18], pois seu poder de representação não está limitado a sólidos cujas fronteiras são variedades de dimensão 2 (2-*manifolds*). Na representação 2-*manifold*, ou simplesmente *manifold* [17], cada porção conexa da fronteira de um sólido (face) é homeomorfa a uma esfera, onde cada ponto

tem uma vizinhança mapeável para um disco bidimensional. A representação é dita *non-manifold* pois é capaz de tratar sólidos com múltiplas regiões (interfaces internas) e porções degeneradas (cascas e arames) de forma consistente. Nestes tipos de sólidos as fronteiras não são homeomorfas à esfera.

Tendo em vista a continuidade do trabalho de Coelho e Lira, surgiu a necessidade de se implementar dentro do MG uma ferramenta de modelagem ainda não presente neste modelador, de forma a tornar o MG um aplicativo mais abrangente em termos de modelagem geométrica tridimensional pelo MEF.

As operações booleanas surgem então como um elemento adicional imprescindível para o conjunto das ferramentas de modelagem, de forma a permitir ao usuário do MG um leque amplo de opções para gerar modelos reais de engenharia.

A possibilidade de se optar pela criação explícita de curvas e retalhos de superfície ou pela combinação de operações booleanas aplicadas a primitivas geométricas simples para produzir um modelo qualquer faz com que o MG passe a ser um aplicativo mais flexível e eficiente para o usuário, o qual passa a ter nas mãos a tarefa de decidir por qual caminho se enveredar, podendo usar critérios pessoais relativos à complexidade do modelo, tempo necessário para a criação do mesmo e número de passos de modelagem aliado ao grau de dificuldade para gerar o modelo, inerentes a cada forma de modelagem.

1.2. Objetivos e Principais Contribuições

Este trabalho se propõe a apresentar um algoritmo genérico para a realização de operações booleanas entre entidades geométricas de qualquer dimensão.

Este algoritmo foi implementado no MG. Para isso, diversas facilidades pré-existentes neste modelador ou em bibliotecas de que ele se utiliza são usadas como elementos-chave para que tal implementação pudesse ser realizada de forma simples, eficiente e robusta. Buscando-se manter a organização interna do código, novas classes foram criadas para que o conjunto das operações booleanas se tornasse uma ferramenta totalmente independente da forma particular de representação de dados do MG. As classes e métodos utilizados pelo algoritmo podem ser facilmente generalizados através dos conceitos básicos de modelagem baseada em representações B-Rep.

Outra preocupação foi a manutenção de uma interface simples que permitisse ao usuário a utilização das operações booleanas da forma mais eficiente possível. Uma nova classe, responsável exclusivamente pela amarração entre as operações booleanas e a interface do programa, foi criada especialmente com este objetivo.

As operações booleanas são realizadas usando conceitos apresentados por Hoffmann [9] e Mäntylä [10], sendo que neste trabalho estes conceitos foram adaptados para que estas operações não precisassem se limitar a sólidos poliedrais *2-manifold* e para permitir que os sólidos pudessem se interceptar antes da aplicação de uma destas operações. A representação *non-manifold* do MG permite que as operações booleanas não regularizadas sejam tratadas. Em modeladores que só são capazes de representar sólidos *manifold*, as possíveis situações *non-manifold* resultantes das operações booleanas têm que ser eliminadas, em um processo que se denomina regularização [9,11].

Algumas facilidades necessárias para que o conjunto das operações booleanas pudesse ser implementado no MG de forma eficaz dizem respeito a algoritmos geométricos de grande utilidade prática, como os de classificação do tipo ponto/sólido e curva/sólido, além de algoritmos de verificação do fechamento de regiões (volumes) e identificação de cada uma destas regiões dado um conjunto de retalhos de superfície. Estes algoritmos encontravam-se disponíveis numa biblioteca de classes baseada na representação CGC, descrita por Cavalcanti [19]. Uma classe auxiliar foi criada no MG apenas para disponibilizar tais métodos para serem utilizados na realização das operações booleanas, fazendo assim a interface entre a CGC e as operações booleanas no MG. Tal classe tem papel crucial na adaptação destes algoritmos geométricos para o contexto do MG, de forma a permitir o uso de retalhos de superfícies curvas.

Podem-se resumir as principais contribuições deste trabalho da seguinte forma:

- Desenvolvimento de um algoritmo genérico para a realização de operações booleanas regularizadas ou não em ambientes de modelagem baseados em representações B-Rep.
- Extensão do algoritmo proposto por Mäntylä [10] de forma a eliminar as restrições inerentes ao universo de representação dos objetos com os quais se deseja trabalhar (superfícies não-poliedrais e sólidos *non-manifold*).

- Apresentação do conceito de Grupo para definir um conjunto de entidades topológicas com qualquer dimensão pertencentes a um mesmo domínio.
- Extensão dos algoritmos geométricos presentes na biblioteca CGC para detecção de ponto em região, ponto sobre superfície ou ponto sobre curva para que curvas e superfícies modeladas com NURBS possam ser consideradas.
- Implementação do algoritmo proposto no modelador geométrico tridimensional MG, aumentando assim a sua capacidade de simular com eficiência e rapidez problemas complexos de engenharia.
- Tratamento de casos particulares e patológicos do algoritmo de operações booleanas, permitindo a sua utilização de forma confiável e robusta em uma grande quantidade de problemas.

1.3. Organização da dissertação

Esta dissertação encontra-se dividida em 6 capítulos. Este capítulo buscou dar uma visão geral do trabalho desenvolvido e da organização da dissertação.

O Capítulo 2 dá uma visão global do modelador MG e da estrutura interna de dados do mesmo, mostrando a organização sistemática do código baseada no conceito de POO.

O Capítulo 3 descreve o algoritmo proposto neste trabalho, enfatizando todas as generalidades já mencionadas, como a inexistência de restrições quanto ao número de entidades a serem combinadas, a geometria destas entidades, a dimensão das mesmas, e o tratamento de casos patológicos.

O Capítulo 4 mostra como o algoritmo proposto está implementado no MG, apresentando as novas classes criadas, a adaptação dos métodos apresentados no algoritmo à estrutura de dados do MG, o esquema de interface com o usuário e a extensão dos algoritmos geométricos da classe CGC.

O Capítulo 5 ilustra diversos exemplos, detalhando as etapas do processo de modelagem dos objetos criados através das operações booleanas. Neste capítulo também são feitos paralelos entre modelos gerados através da manipulação explícita de curvas e superfícies e através da aplicação de um conjunto de operações booleanas, em termos de tempo de modelagem, dificuldade e número de passos necessários.

O Capítulo 6 apresenta as considerações finais deste trabalho, resumindo sua importância e aplicabilidade e apresentando sugestões para trabalhos futuros.

O Apêndice trata de conceitos básicos referentes à modelagem geométrica de sólidos, apresentando o contexto histórico no qual está inserida, a evolução dos sistemas de modelagem, as formas de representações de sólidos, passando pela topologia como elemento-chave de uma sistema de modelagem e pela importância e aplicação das estruturas de dados topológicas.

2
O Modelador Geométrico MG

Apresenta-se aqui um modelador de elementos finitos existente, o MG. Trata-se de um pré-processador desenvolvido para a geração de malhas sólidas de elementos finitos. É feita uma breve análise do ambiente de modelagem e da estrutura de classes do modelador, que é todo implementado segundo o conceito de programação orientada a objetos (POO), frisando-se as estruturas de dados topológicas utilizadas e a existência das ferramentas de interseção e detecção automática de regiões comentadas anteriormente, que serão de extrema importância na implementação do algoritmo de operações booleanas neste modelador, como será descrito nos próximos capítulos.

2.1.
Características gerais do modelador

No modelador MG, a base para a geração das superfícies (cascas) e dos volumes é a construção de curvas. A manipulação das entidades gráficas (vértices e curvas) é feita diretamente na área de desenho (*canvas*) e estas operações são orientadas por um plano de interface que faz a transformação do espaço bidimensional para o espaço tridimensional (Figura 2.1). A interface por manipulação direta permite maior liberdade na criação da geometria do objeto, contudo existe também a possibilidade de se entrar com os valores exatos das coordenadas ou dos parâmetros das transformações através de itens de menu.

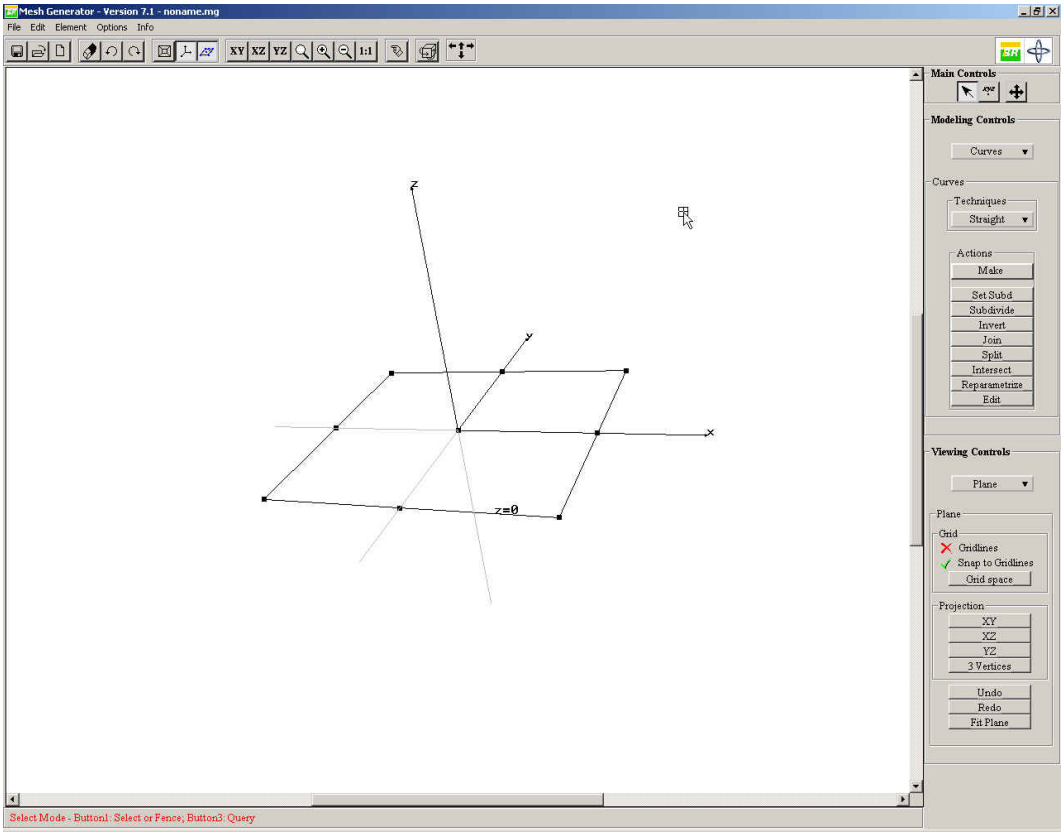


Figura 2.1 – Interface do MG e plano de interface.

Este trabalho traz uma inovação para o usuário do MG, que passa a usufruir de mais uma opção de interface para a criação de objetos. Independentemente da maneira como esta nova ferramenta está internamente implementada no modelador, buscou-se criar uma interface para as operações booleanas que se assemelhe às interfaces dos sistemas de modelagem CSG, permitindo que o usuário possa gerar sólidos unicamente através da combinação de outros sólidos pré-existentes, sem a necessidade da manipulação direta das curvas e superfícies do modelo.

O MG cria automaticamente um arquivo para armazenar as diversas etapas de edição do modelo, ou seja, ou arquivo de *backup* das informações do modelo, ou arquivo de *check point*. Quando o usuário chama a função de *undo* do programa, este arquivo é usado para fazer as alterações necessárias à estrutura de dados. Arquivos de *check point* não possuem utilidade somente no caso de o usuário desejar desfazer um passo de modelagem, mas também no caso de uma falha do sistema (execução de uma operação ilegal, invasão de memória) ou interrupção de energia. Através de parâmetros adicionais na linha de comando utilizada para a execução do programa, pode-se recuperar o estado imediatamente anterior à interrupção.

Nas simulações por elementos finitos modeladas com o MG, várias técnicas para geração de malhas não-estruturadas de elementos finitos foram e vêm sendo desenvolvidas, com algoritmos baseados em triangulação de Delaunay [20], em decomposição espacial recursiva [21] e em métodos de avanço de fronteira [22,23,24]. O modelador MG utiliza um algoritmo para geração de malhas sólidas de tetraedros em domínios arbitrários, apresentado por Cavalcante Neto [25], que também combina técnicas de avanço de fronteira com decomposição espacial recursiva. Algoritmos para geração de malhas estruturadas também estão presentes no MG, tornando-o capaz de gerar malhas com os mais diversos tipos de elementos finitos conhecidos na literatura.

É desejável que um modelador seja capaz de especificar atributos para os mais diversos tipos de problemas de engenharia, ou seja, que os atributos sejam configuráveis com relação ao tipo de análise desejada. Lira [6] incorporou ao MG o sistema de gerenciamento e configuração de atributos ESAM (*Extensible System Attributes Management*). Isto possibilitou o uso deste modelador nos mais diversos problemas de engenharia usando elementos finitos. Em seu trabalho, Lira também reorganizou internamente o código do programa MG de forma a torná-lo totalmente orientado a objetos e implementou as ferramentas de interseção de superfícies paramétricas e detecção automática de multi-regiões que serão analisadas mais detalhadamente em seções posteriores.

2.2. Modos de interface do MG

O MG incorpora os seguintes modos de interface:

- Seleção
- Mudança de Projeções
- Edição do Plano de Interface
- Transformações por Manipulação Direta
- Criação:
    1. de Curvas
    2. de Vértices
    3. de Malhas
    4. de Volumes
    5. de Sólidos
    6. de Grupos
- Visualização:

1. de Curvas
2. de Vértices

- de Malhas
- de Volumes
- de Sólidos

O modo de interação corrente determina qual o conjunto de operações disponíveis para o usuário. Alguns destes modos de interação serão brevemente discutidos a seguir.

2.2.1. Seleção

O modo de seleção é o modo *default* do programa. No MG as entidades gráficas são tridimensionais projetadas na tela. O programa seleciona a entidade que estiver à frente na projeção.

As entidades podem usufruir de dois estados: *selecionado* e *não selecionado*. O MG permite a seleção de múltiplas entidades. A seleção de uma ou mais entidades provoca a alteração do estado de seleção de todas as outras entidades para *não selecionado*. A seleção de várias entidades também pode ser realizada por meio de *fence*, ou seja, pela definição de uma janela de forma que todas as entidades integralmente contidas dentro dos limites da janela sejam selecionadas.

A *tolerância* é um aspecto importante na seleção de entidades. Dois pontos estão *próximos* quando a distância entre os dois pontos for menor que a tolerância, que pode ser definida pelo usuário. No modo de seleção, trabalha-se o tempo todo no espaço projetado da tela, então a proximidade também é calculada neste espaço projetado.

É importante frisar também a questão da *prioridade* de seleção. O programa adota a ordem de prioridade vértice, região, curva, malha 2D e malha 3D. Isto significa que se, por exemplo, o usuário apontar uma posição onde estejam presentes um vértice e uma curva dentro da tolerância de seleção, o MG seleciona o vértice.

Vértices de curvas distintas que são colocados próximos sofrem *atração*. Quando um vértice é posicionado no espaço, é feita uma pesquisa global, testando-se a proximidade de acordo com a tolerância corrente. Se algum vértice pré-existente do modelo estiver dentro do limite da tolerância, o sistema passa a

utilizar a referência deste vértice, alterando as coordenadas inicialmente fornecidas pelo usuário (Figura 2.2).

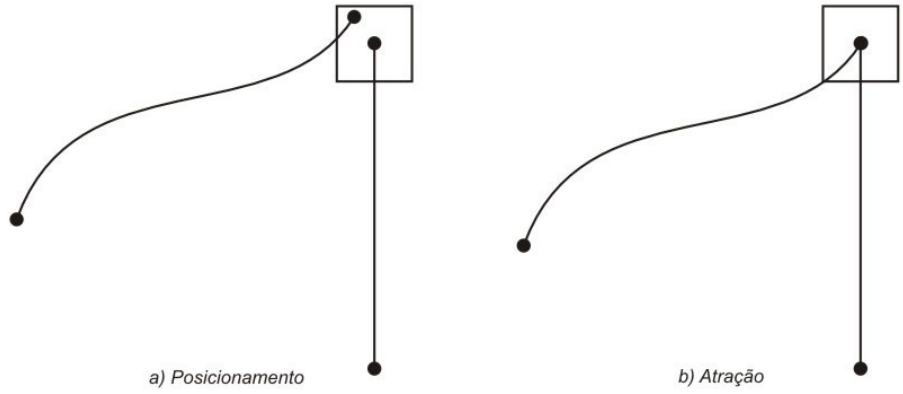


Figura 2.2 – Atração de vértices: a) posicionamento; b) atração [7].

2.2.2. Mudança de Projeções

O MG utiliza uma biblioteca que usa o modelo de câmera, que possui apenas o ponto de referência (*vrp*), a posição da câmera (*eye*) e o vetor que indica a direção vertical de projeção (*vup*). O *vrp* é colocado automaticamente pelo MG no centro do modelo e o usuário não tem acesso a este parâmetro. A posição inicial da câmera encontra-se a uma distância razoável do modelo e pode ser alterada pelo usuário. A Figura 2.3 mostra a posição inicial dos parâmetros de visualização, a esfera imaginária sobre a qual são feitas as movimentações e a vista esquemática da imagem que seria gerada na tela do computador.

Podem-se fazer translações da câmera, rotações da mesma na superfície da esfera ou em torno do seu eixo radial, alteração da distância da câmera ao ponto de referência, escala no modelo (aumentando ou diminuindo a projeção mas sem alterar nenhum parâmetro de visualização), dentre outras funções de visualização.

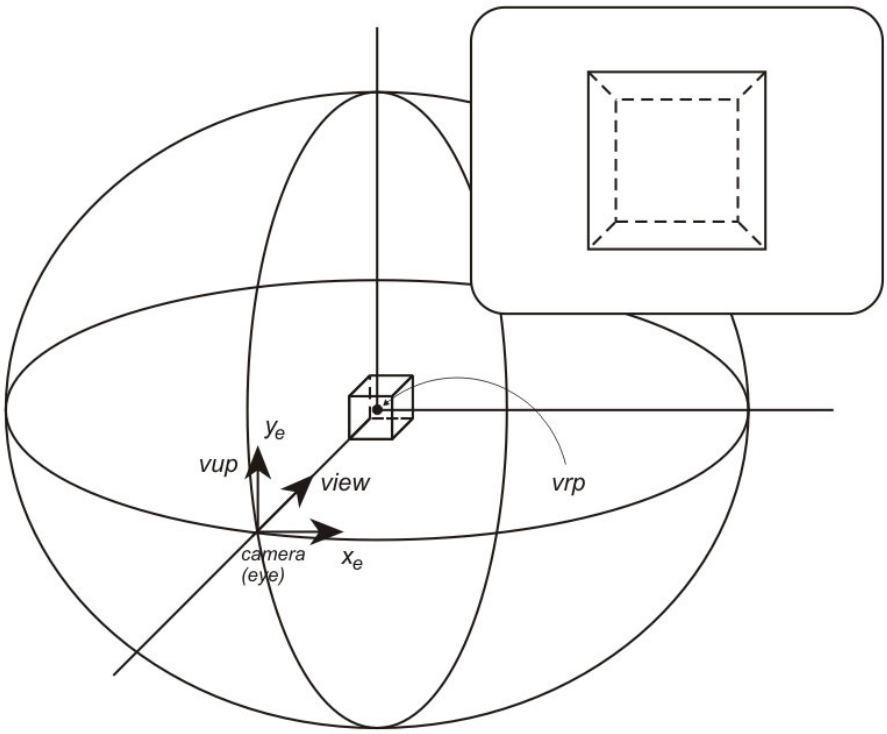


Figura 2.3 – Parâmetros de visualização e posição inicial da câmera (visão esquemática do objeto em destaque) [7].

2.2.3. Edição do plano de interface

O elemento gráfico de interface mais importante do MG é o plano de interface. Escalas, rotações e translações deste plano são as operações disponíveis por manipulação direta na área de desenho.

O plano de interface pode ser colocado nas posições padrões $z = 0$, $y = 0$ e $x = 0$, se for desejado. O usuário também possui controle sobre o desenho do *grid*, atração para os pontos do *grid* e espaçamento do *grid*. Pode-se adaptar o plano de interface aos limites do modelo, fazendo-se uma escala centrada na origem corrente do plano. O usuário também pode apontar três vértices consecutivos do modelo para que o plano de interface seja posicionado no plano geométrico definido por estes três vértices, possuindo origem no baricentro destas três posições.

2.2.4.
Transformações por manipulação direta

Este modo de interface visa a realização de transformações (translações e rotações) de entidades gráficas selecionadas por manipulação direta. Ao se selecionar este modo, eixos auxiliares são desenhados com origem no centro das entidades selecionadas para facilitar a execução das transformações pelo usuário (Figura 2.4). O centro de rotação é sempre a origem dos eixos. Podem-se fazer transformações apenas com o sistema de eixos para colocá-lo em uma posição diferente da inicial ou com todas as entidades selecionadas. É possível se alterar os valores escalares das translações e rotações através de campos editáveis na interface.

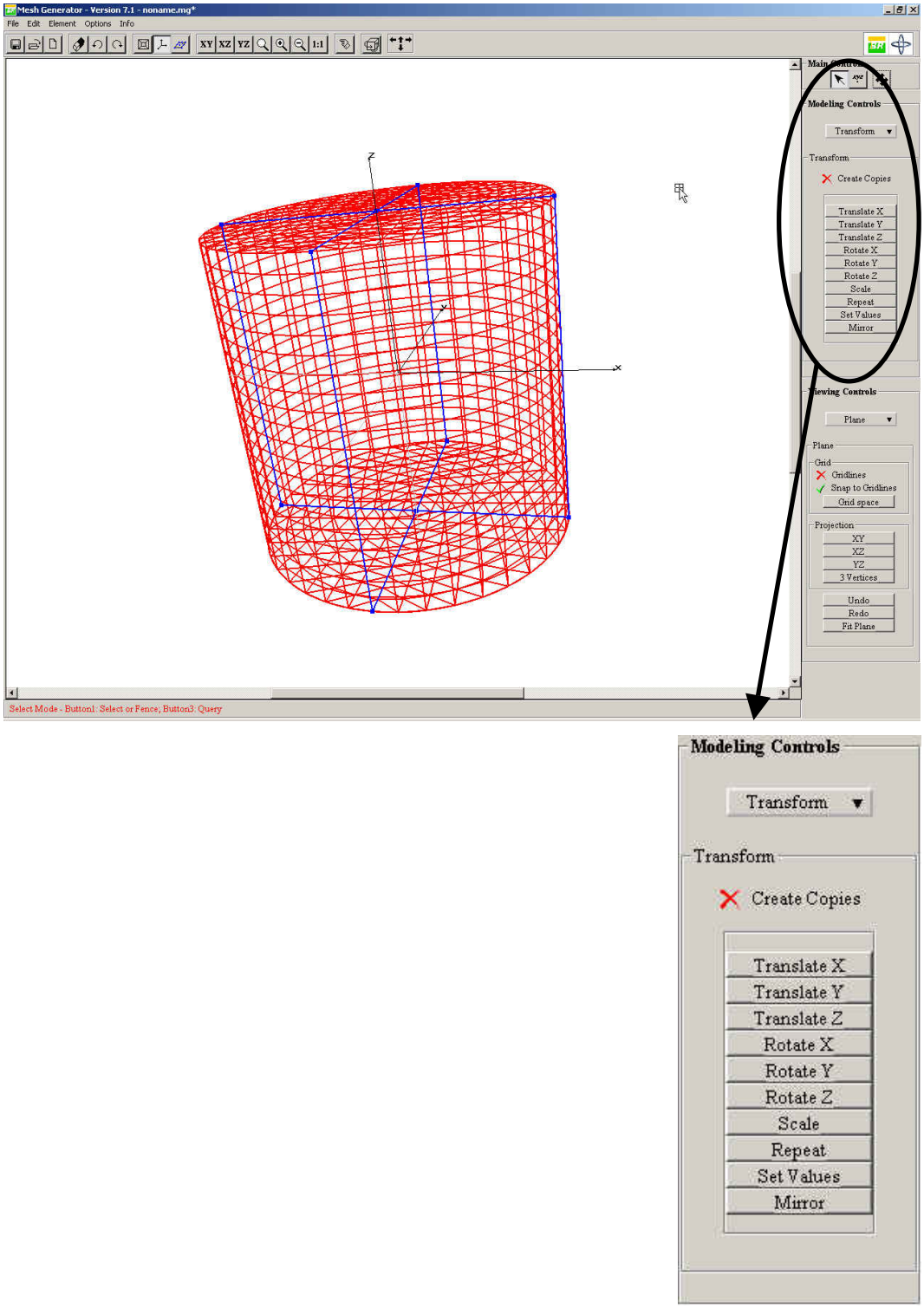


Figura 2.4 – Modo de transformação por manipulação direta.

2.2.5. Criação

A criação de vértices e curvas depende da especificação das coordenadas de vértices, que dependem da posição do plano de interface e do estado de habilitação de alguns elementos de interface que controlam a atração de pontos.

Uma das etapas mais importantes na criação de um vértice geométrico é a transformação das coordenadas bidimensionais (espaço da tela) para coordenadas tridimensionais (espaço tridimensional do objeto). Este processo é feito internamente pelo programa em duas etapas: cálculo da interseção da linha que parte do ponto que o usuário definiu na janela, possuindo a direção de projeção, com o plano de interface; e obtenção do ponto no espaço do modelo pela determinação da terceira coordenada a partir da equação implícita do plano e resolução de um sistema linear de equações com esse ponto [7].

Se o *snap* do plano de interface estiver ativo, é feita a atração para o ponto mais próximo do *grid*, respeitando-se o espaçamento corrente. Pode-se escolher também a opção de *snap* para os vértices já existentes no modelo. Esta opção faz com que seja feito um teste de *pick* nos vértices do modelo antes da transformação para o espaço do objeto. Se algum vértice estiver próximo (de acordo com a tolerância corrente) ao cursor, é feita a atração do cursor para a posição deste vértice, e suas coordenadas são utilizadas no posicionamento.

A criação de curvas, superfícies, malhas bidimensionais e tridimensionais, volumes, sólidos e grupos será analisada mais detalhadamente na próxima seção, que descreve a forma de representação interna destas entidades no modelador.

2.3. Modelagem geométrica no MG

Na seção anterior foi possível observar as principais características do MG de um ponto-de-vista de usuário. As funcionalidades básicas e a interface com o usuário foram apresentadas, buscando-se familiarizar o leitor com o ambiente de modelagem em que este trabalho está inserido.

Nesta seção, é apresentada inicialmente uma análise mais detalhada das entidades geométricas que podem ser criadas no MG, como curvas, superfícies, malhas 2D e malhas 3D. Algumas definições e a maneira como estas entidades são criadas no MG são mostradas, para que se possa ter uma noção do domínio representacional deste modelador.

Em seguida, uma visão mais interna do modelador será exposta. A maneira como as informações topológicas e geométricas são descritas e tratadas e a estrutura de dados utilizada para armazenar estas informações segundo uma representação B-Rep serão brevemente estudadas. A organização de classes do MG também será mostrada, enfatizando-se a importância do conceito de orientação a objetos no paradigma de modelagem utilizado.

Por fim, são apresentadas duas ferramentas essenciais num sistema de modelagem e que foram introduzidas no MG por Lira [6]: a interseção de superfícies paramétricas e a detecção automática de regiões. No escopo deste trabalho, estas duas ferramentas são imprescindíveis para a implementação correta e eficiente das operações booleanas, como será visto posteriormente.

2.3.1. Representação paramétrica de superfícies

Segundo Hoffmann [9], os dois métodos mais comuns para se representar superfícies são as formas implícitas e as equações paramétricas. Uma superfície é descrita pela sua forma implícita pela equação $f(x,y,z) = 0$, onde x, y e z formam o sistema de eixos no espaço Euclidiano. A equação $S(u,v) = (x(u,v),y(u,v),z(u,v))$, onde u e v formam o sistema de eixos no espaço paramétrico da superfície, define uma representação paramétrica da superfície (Figura 2.5).

O MG adota a forma paramétrica para tirar proveito de algumas de suas potencialidades e por esta forma ser a mais adequada na representação de objetos formados por seções transversais. A descrição paramétrica de superfícies com geometria arbitrária é utilizada na geração de malhas triangulares nestas superfícies através de algoritmo proposto por Miranda [26]. A superfície 3D é mapeada para uma superfície 2D (representação paramétrica da superfície), realizando-se então a triangulação bidimensional, corrigindo-se as distâncias e os ângulos distorcidos na transformação da malha do espaço 3D para o espaço paramétrico. Em seguida, a malha resultante é reconduzida ao espaço 3D. O algoritmo de interseção originalmente implementado no MG também tirava proveito da representação paramétrica das superfícies, sendo baseado na interseção entre malhas sobre as superfícies paramétricas[6,8].

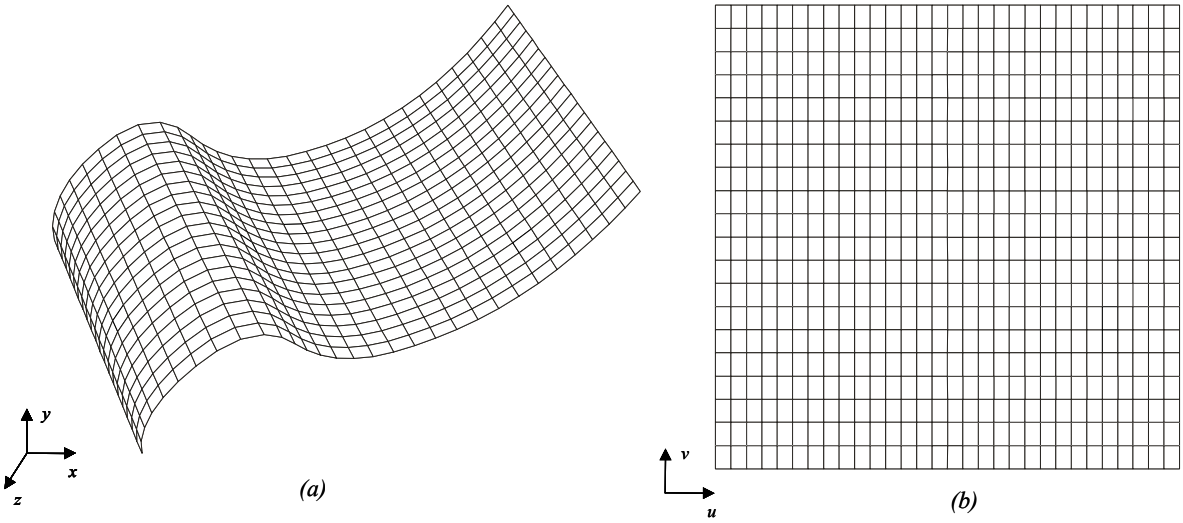


Figura 2.5 – Representações de uma superfície: a) espaço Euclidiano; b) espaço paramétrico [6].

2.3.2. Descrição geométrica de curvas e superfícies usando NURBS

A partir do trabalho de Lira [6], o MG passou a incorporar as representações de curvas e superfícies conhecidas como NURBS (*Non Uniform Rational B-Splines*) [27,28]. Segundo Lira, algumas vantagens deste tipo de representação são:

- NURBS provê uma base matemática única para representação das formas analíticas, tais como seções cônicas e superfícies quadráticas, bem como superfícies com formas quaisquer (por exemplo, cascos de navios ou carenagem de carros);
- a modelagem usando NURBS é intuitiva: quase todas as ferramentas e algoritmos geométricos possuem interpretação de fácil compreensão;
- as superfícies e curvas NURBS são invariantes quando submetidas a transformações geométricas afins (translação, rotação, projeções, etc.);
- as superfícies e curvas NURBS são generalizações de superfícies e curvas *B-Splines* e *Béziers*.

Um exemplo de superfície NURBS pode ser visto na Figura 2.6. A definição formal e as propriedades de curvas e superfícies NURBS podem ser encontradas em [27] e [28]. Para os objetivos deste trabalho, não é necessária uma descrição matemática completa deste tipo de geometria.

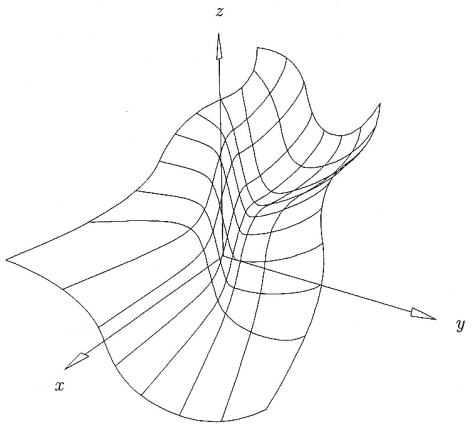


Figura 2.6 – Exemplo de superfície NURBS [6].

Diversas bibliotecas computacionais implementam as representações de curvas e superfícies do tipo NURBS, como por exemplo Nlib [29], *SOLID*S++ [30] e NURBS++ [31]. Grande parte destas bibliotecas é utilizada em empresas ou em programas comerciais de modelagem, o que garante uma certa confiabilidade aos seus usuários. No trabalho de Lira [6], a biblioteca NURBS++ foi incorporada ao MG. Esta biblioteca é implementada em linguagem C++, seguindo o conceito de programação orientada a objetos. Isto representa uma grande vantagem, pois permite a expansão da estrutura de classes. A disponibilização do código fonte também torna possível a implementação de novas funcionalidades na biblioteca. Algumas destas funcionalidades foram implementadas por Lira em seu trabalho.

A biblioteca NURBS++ possui duas classes básicas, uma contendo informações necessárias para a definição de uma curva NURBS e outra contendo informações para a definição de uma superfície NURBS. As Figuras 2.7 e 2.8 resumem a descrição destas duas classes.

Um objeto da classe que representa uma curva NURBS armazena o grau de interpolação da curva, o vetor de *knots* e os seus pontos de controle (ver [27] e [28] para uma descrição completa destes elementos). Existem algumas maneiras diferentes de se criar objetos desta classe. Uma delas é através de um construtor (método de uma classe que cria um objeto) padrão, fornecendo-se as informações descritas acima. Outras maneiras permitem a criação de diversos tipos de curvas conhecidas, como arcos, retas e *splines*, a partir de informações básicas e mais intuitivas em modelagem, como por exemplo e centro, o ponto inicial e o ponto final de um arco.

Um objeto da classe que representa uma superfície NURBS armazena os graus de interpolação da superfície nas duas direções paramétricas *u* e *v*, o vetor de *knots* em ambas as direções e os seus pontos de controle, armazenados de forma matricial. Os construtores desta classe são semelhantes aos descritos para a classe de curvas NURBS, ou seja, pode-se utilizar um construtor padrão ou outros que instanciam parâmetros de curvas mais conhecidas, como *sweep*, *Gordon* e revolução.

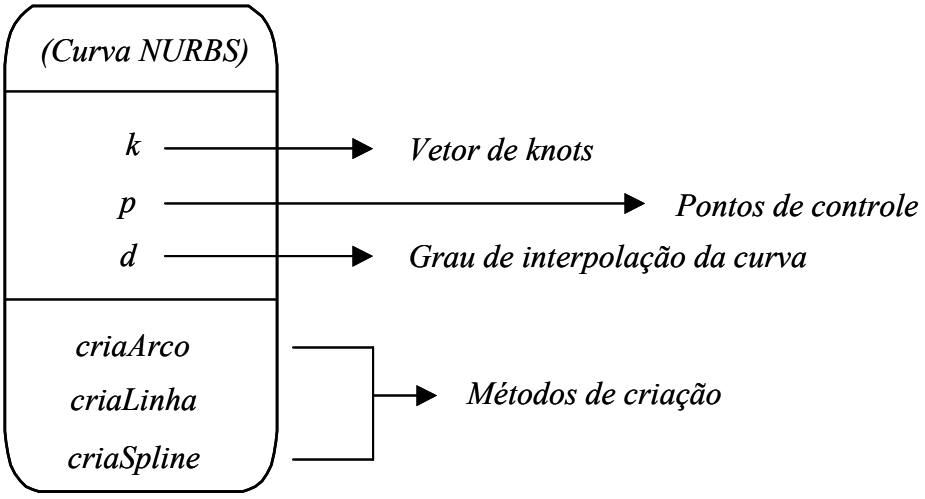


Figura 2.7 – Representação de curvas na biblioteca NURBS++ [6].

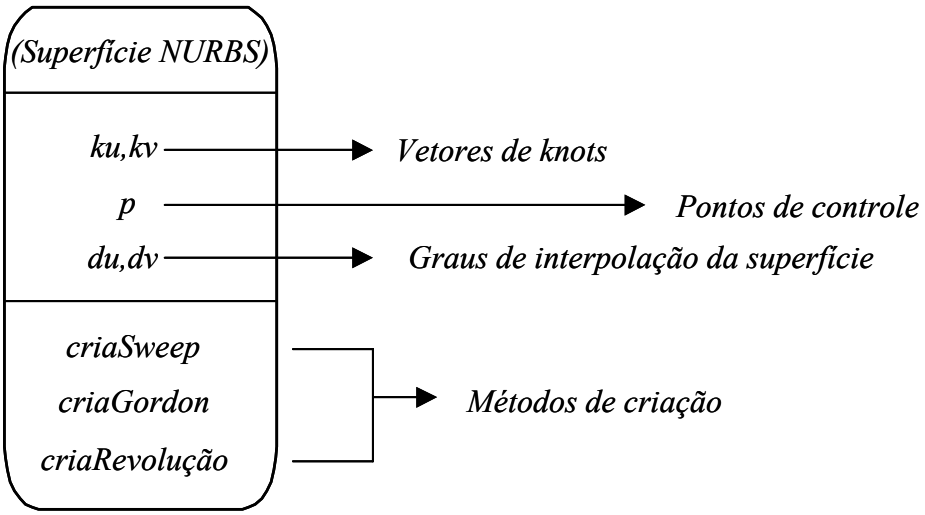


Figura 2.8 – Representação de superfícies na biblioteca NURBS++ [6].

O uso de NURBS no MG foi motivado sobretudo pela sua capacidade de representar o espaço paramétrico das superfícies. Em seu trabalho, Lira [6] incorporou a biblioteca NURBS++ no MG através da criação de uma hierarquia

de classes para representar os tipos de curvas e superfícies utilizados pelo modelador. Foram criadas classes que gerenciam a interface entre o modelador MG e a biblioteca NURBS++, que possuem referências para o objeto NURBS correspondente. Através deste, pode-se acessar os dados das curvas ou superfícies armazenados nas classes correspondentes na biblioteca NURBS++. As classes derivadas destas representam diretamente os tipos de curvas e superfícies implementados por Lira no MG. Essas são as classes que podem ser instanciadas pelo usuário através da interface do MG. As Figuras 2.9 e 2.10 ilustram a organização destas classes.

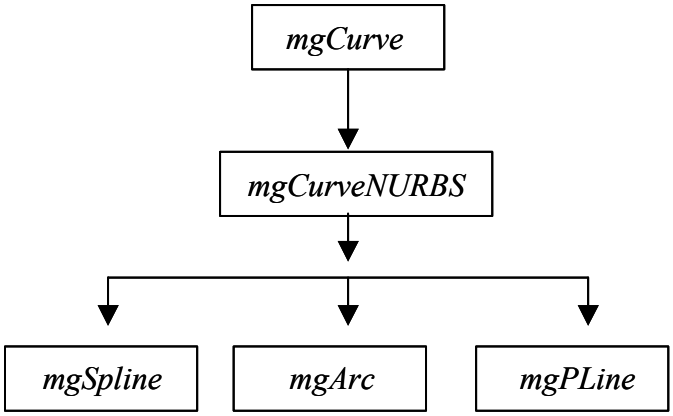


Figura 2.9 – Organização das classes de curvas no modelador MG [6].

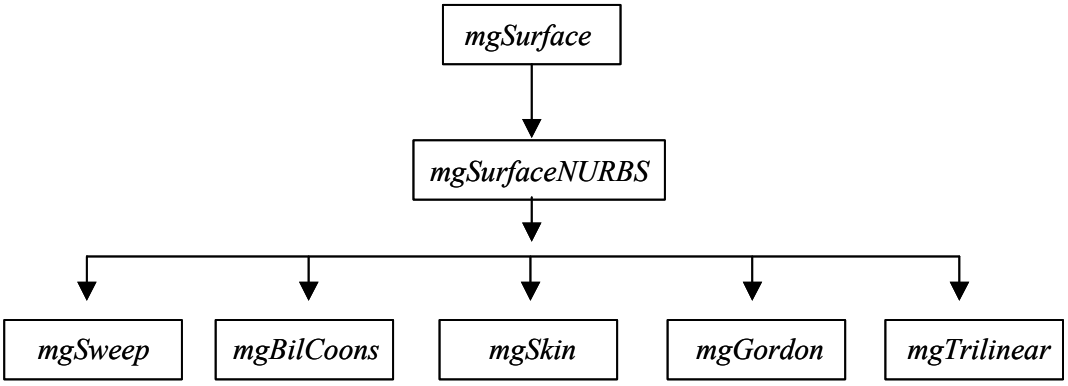


Figura 2.10 – Organização das classes de superfícies no modelador MG [6].

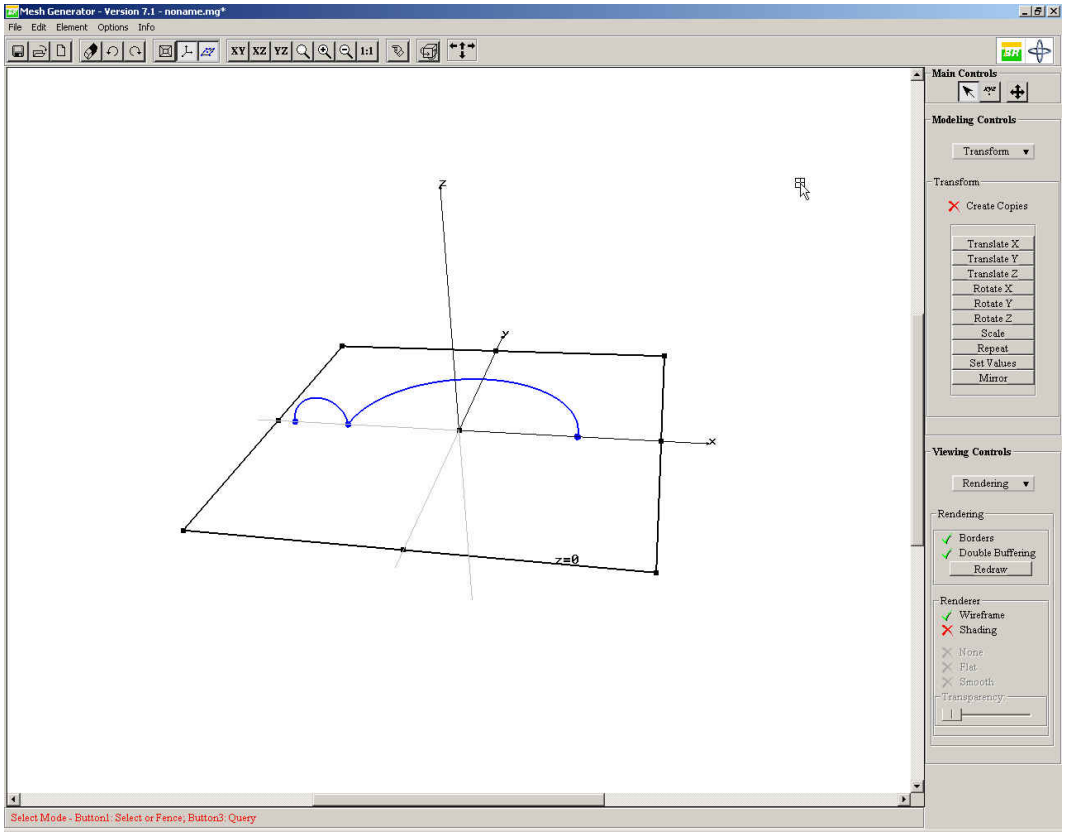
2.3.3. Geração de curvas e superfícies no MG

Para que uma nova curva possa ser criada no ambiente de modelagem do MG, o usuário deve escolher a *forma* da curva que deseja inserir, dentre três opções possíveis: linha poligonal, arco ou *B-Spline*. Estando o programa no modo de interface de criação de curvas, basta ao usuário posicionar o cursor na

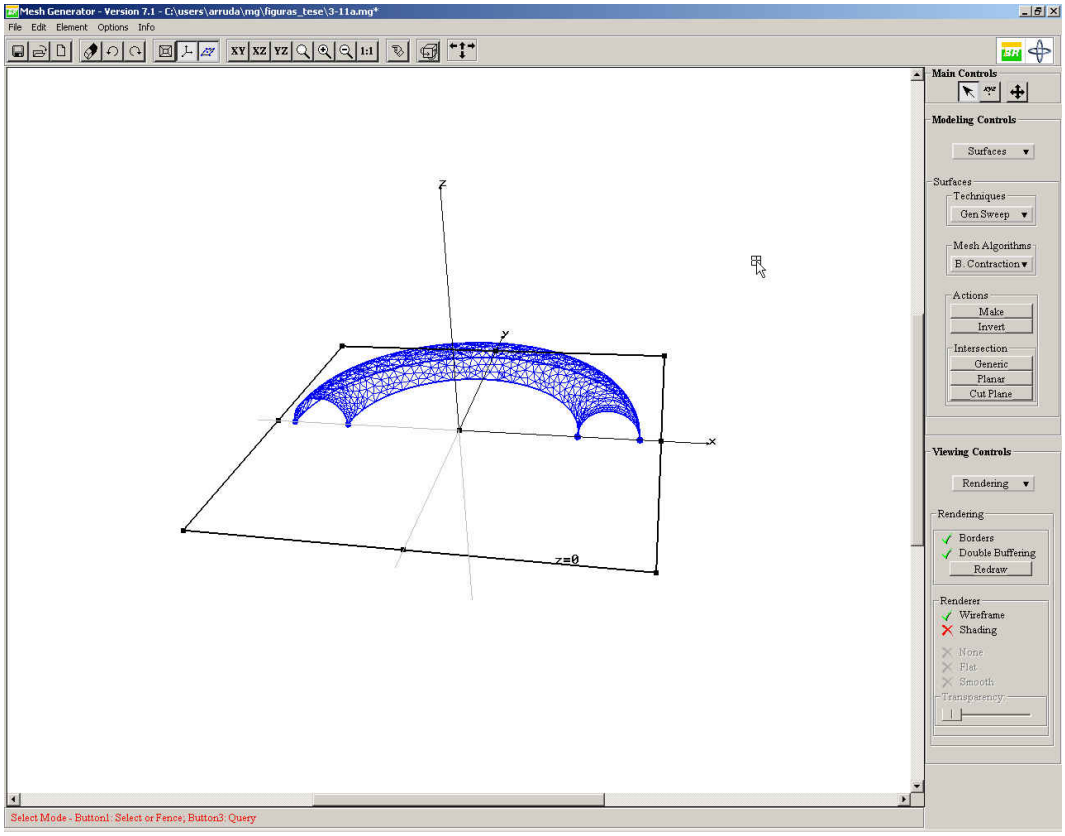
localização desejada dentro da área de desenho da interface para fornecer os pontos necessários para a descrição geométrica da curva. As questões referentes à *atração* e *tolerância*, já discutidas, são aplicáveis neste modo de interface.

Ao final da edição, os pontos são passados para a biblioteca NURBS++ que gera a curva NURBS correspondente. A estrutura de dados do MG armazena o ponteiro para esta curva juntamente com a representação de uma poligonal equivalente [32] adaptativa associada a ela. Esta poligonal é utilizada, por exemplo, em eventos de seleção e desenho destas curvas.

A criação de superfícies é feita utilizando-se as curvas existentes. O usuário escolhe um método de construção de superfícies, pré-seleciona as curvas que definem completamente a superfície desejada e cria a mesma pressionando um botão na interface. Mais uma vez, o objeto NURBS correspondente é criado na biblioteca NURBS++ e uma referência para este objeto é armazenada no MG juntamente com uma discretização da superfície (malha de elementos finitos), utilizada em eventos de seleção e desenho das superfícies. A Figura 2.11 mostra a criação de um toro usando a técnica de *sweep*.



(a)



(b)

Figura 2.11 – Criação de um toro através da técnica de *sweep*: a) curvas bases; b) superfície gerada.

A seguir são apresentadas resumidamente algumas formulações e características dos tipos de superfícies existentes no MG [6]. Nos livros de Piegl e Tiller [28] e Farin [33] encontram-se informações mais detalhadas sobre as descrições matemáticas destas superfícies.

Superfícies *bilineares* são completamente definidas por um circuito de quatro pontos. Uma representação NURBS dessa superfície é obtida pela interpolação linear entre os segmentos formados por estes pontos.

Superfícies *skin*, também conhecidas como *loft*, são aquelas obtidas pela interpolação de um conjunto de curvas ao longo de uma direção. Essas curvas representam as seções transversais da superfície gerada e não são necessariamente planas. Na direção dos bordos livres destas superfícies, uma interpolação linear é feita pela biblioteca NURBS++, gerando uma superfície linear ao longo dessa direção. As curvas geradoras fornecidas são respeitadas na construção da superfície. Assim, a superfície criada tem estas curvas como suas seções transversais (Figura 2.12). Este tipo de superfície é importante, pois serve como base para a geração de outros tipos de superfície, como será visto adiante.

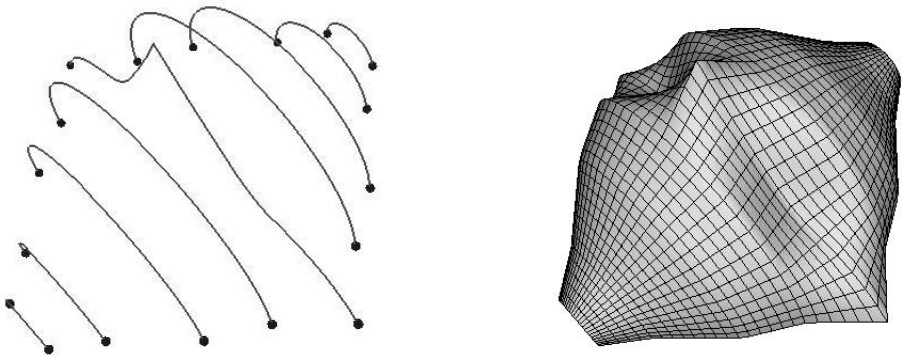


Figura 2.12 – Superfície gerada por *skin*.

Superfícies de *coons* são extensões das superfícies bilineares. Este tipo de representação usa os pontos definidos ao longo dos segmentos do contorno como pontos de controle da superfície. Não existe limitação quanto ao grau de interpolação das curvas do contorno que definem as superfície de *coons*. No MG, a construção deste tipo de superfície é realizada a partir da definição das quatro curvas de bordo que limitam a superfície. Essas curvas podem ser de qualquer tipo, não necessariamente retas, como pode ser visto na Figura 2.13.

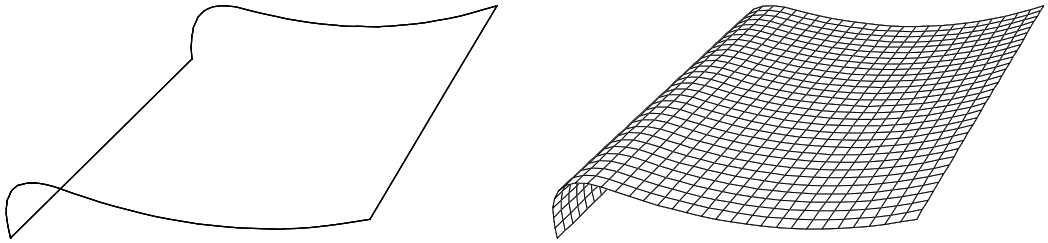


Figura 2.13 – Superfície *coons* [6].

Superfícies geradas por *sweeps* são obtidas pelo arrasto de uma seção curva ao longo de uma trajetória, também curva. No MG, estas superfícies podem ser criadas de três formas diferentes: através de um *sweep translacional*, indicando-se a curva base para o *sweep* e a direção de propagação; através de um *sweep rotacional*, rotacionando-se a curva base em torno de uma direção; ou através de um *sweep genérico*, onde a curva base é arrastada ao longo de uma outra curva que define a trajetória do *sweep* (Figura 2.14).

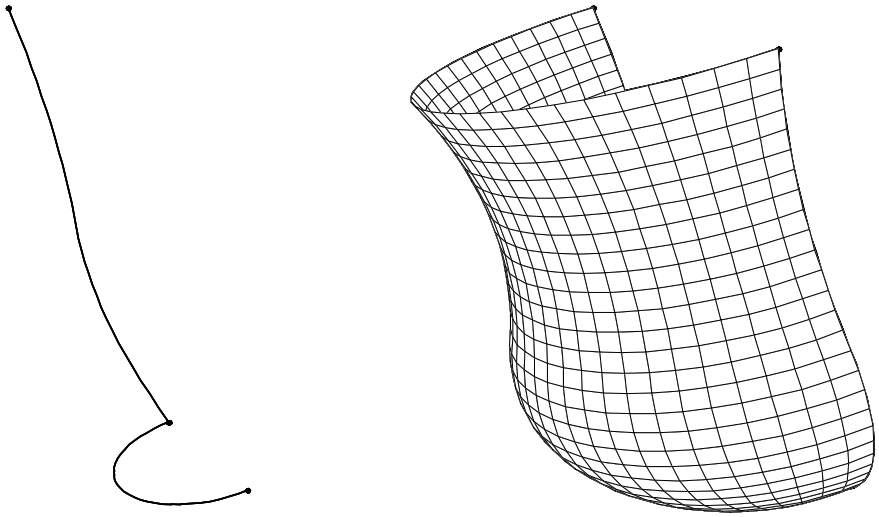


Figura 2.14 – Superfície gerada por sweep genérico [6].

Superfícies de *Gordon* são aquelas geradas por uma rede bidirecional de curvas espaciais (Figura 2.15). No MG, a construção deste tipo de superfície é realizada a partir de uma seqüência de curvas espaciais definidas pelo usuário. Deve-se indicar quais são as curvas geradoras em cada uma das direções paramétricas da superfície. Elas funcionam como seções transversais, em ambas as direções, da superfície gerada. Este tipo de superfície é uma generalização das superfícies de *coons*. Enquanto as superfícies de *coons* são geradas exatamente por uma rede de duas curvas em cada direção, as superfícies de *Gordon* não possuem tal limitação.

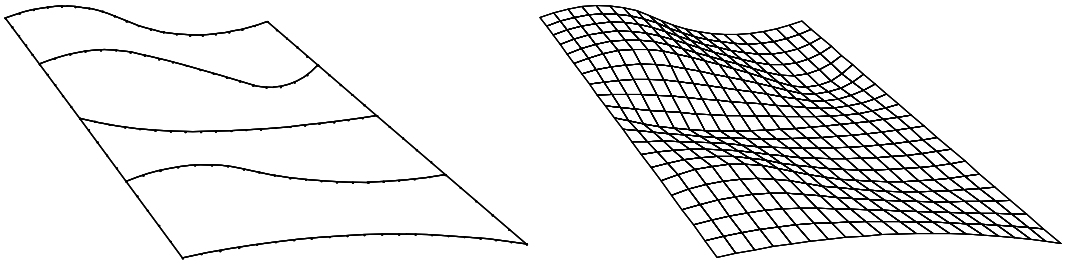
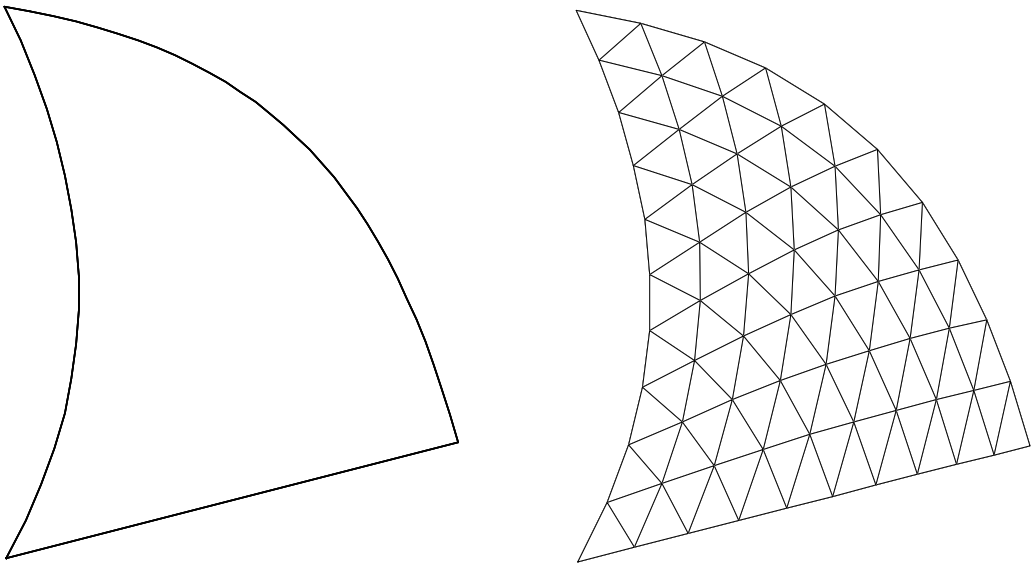


Figura 2.15 – Superfície de *Gordon* [6].

Superfícies *triangulares* são definidas por um circuito de três curvas. A representação dessas superfícies utiliza os pontos localizados ao longo dos segmentos do contorno como seus pontos de controle (Figura 2.16). Em seu trabalho, Lira [6] buscou solucionar um problema que dificultava a representação deste tipo de superfície no ambiente de modelagem do MG. Como não se conhece uma formulação matemática NURBS para representar superfícies triangulares, Lira expandiu a biblioteca NURBS++ de forma que superfícies triangulares pudessem ser aproximadas por superfícies cúbicas do tipo *Bézier* [34]. Contudo, esta implementação apresenta restrições, pois nem todas as curvas geométricas usadas na definição de uma superfície triangular podem ser representadas por curvas *Bézier*. A construção deste tipo de superfície no MG é realizada a partir da definição das três curvas de bordo que limitam a superfície.



(a) *(b)*

Figura 2.16 – Superfície triangular [6].

2.3.4.
Geração de sólidos no MG

No MG, as técnicas utilizadas para modelagem de sólidos confundem-se com o próprio mapeamento associado a cada sólido (malha 3D de elementos finitos). A geração do sólido é feita pela geração da malha que o representa [26].

Quatro técnicas são utilizadas no MG na geração de sólidos. Três delas utilizam técnicas de *sweep*, arrastando-se uma seção transversal bidimensional (retalho de superfície) ao longo de uma trajetória no espaço. A outra descreve uma metodologia para geração de malhas sólidas, não-estruturadas, em domínios arbitrários. Recomenda-se a leitura dos trabalhos de Miranda [26,35] para um maior detalhamento destas técnicas.

Vale ressaltar que no MG, dois sólidos adjacentes devem ter pelo menos um retalho de superfície comum a ambos, localizado na interface entre eles. Essa necessidade exige que retalhos de superfícies sejam criados automaticamente pelo MG para delimitar o contorno dos sólidos gerados e garantindo a consistência topológica nesta etapa da modelagem.

O *sweep translacional*, também chamado de *extrusão*, é uma técnica em que o arrasto da seção transversal do sólido é feito ao longo de uma linha reta no espaço tridimensional (Figura 2.17). Os elementos finitos gerados podem ser pentaédricos ou hexaédricos.

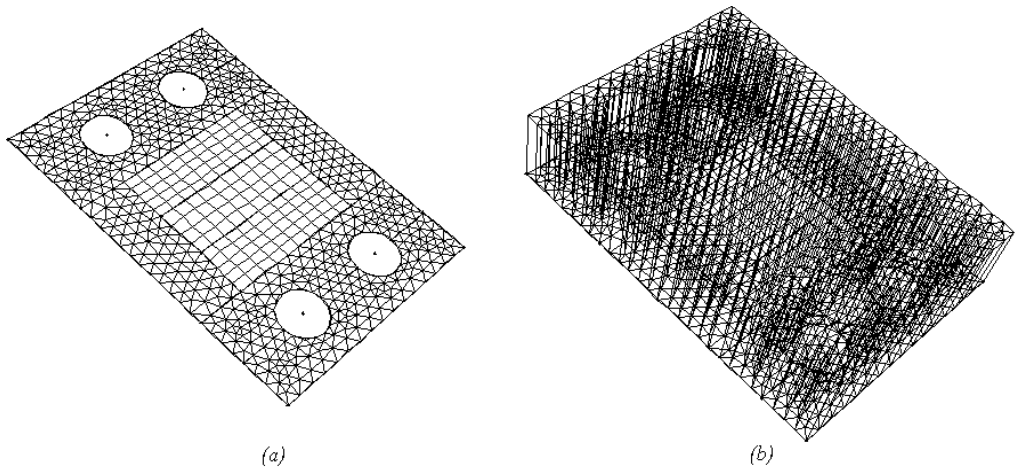


Figura 2.17 – Mapeamento de sólido por extrusão [6].

O *sweep curvo* é uma técnica semelhante à anterior, com a diferença que o arrasto da seção transversal é feito ao longo de uma curva qualquer no espaço tridimensional, que define a trajetória do *sweep* (Figura 2.18). A malha associada

à superfície pode ser formada por elementos triangulares ou quadrilaterais. Os elementos finitos sólidos podem ser pentaédricos ou hexaédricos.

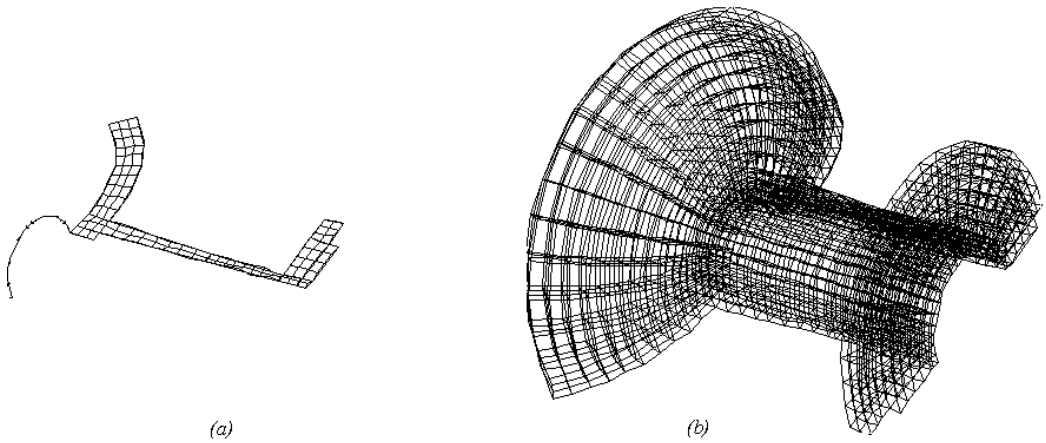


Figura 2.18 – Mapeamento de sólido por sweep curvo [6].

O *mapeamento transfinito tridimensional* consiste em uma técnica para a construção automática de malhas sólidas a partir das seções transversais de um modelo a ser discretizado em elementos finitos (Figura 2.19). Essas seções transversais devem estar ligadas por uma curva que permita identificar os pontos bases de cada seção transversal (pontos onde as curvas e seções transversais se interceptam) e o número de seções intermediárias entre duas seções, informações necessárias para o algoritmo de geração deste tipo de malha sólida. Mais uma vez, os elementos sólidos gerados podem ser pentaédricos ou hexaédricos.

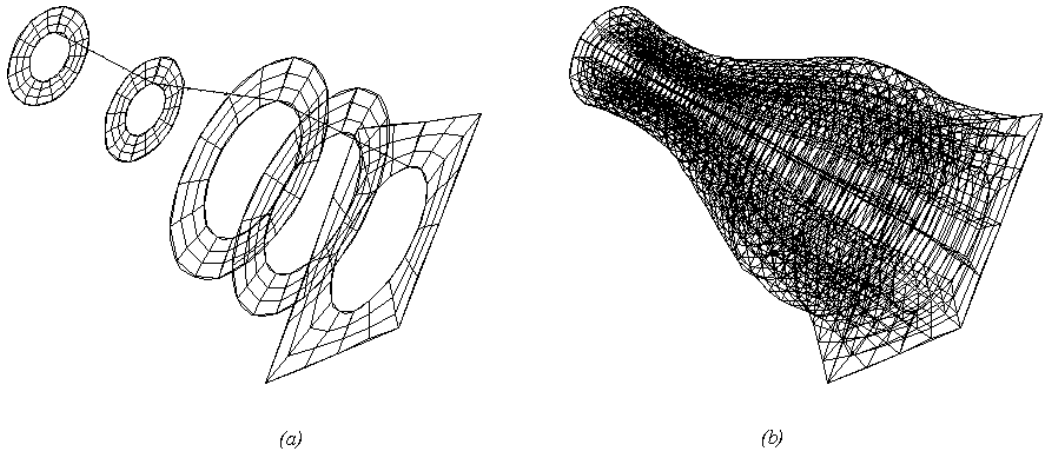


Figura 2.19 – Sólido gerado por mapeamento transfinito tridimensional [6].

A geração de *malhas sólidas em domínios arbitrários* utiliza um algoritmo para geração de malhas volumétricas não-estruturadas de tetraedros para

domínios arbitrários [25,36]. As superfícies que delimitam os sólidos gerados constituem a entrada de dados para o algoritmo responsável pela geração de tais sólidos. As malhas associadas a cada superfície são convertidas em uma representação única de elementos finitos e é verificado o fechamento de uma região com estas superfícies. Se uma região fechada for detectada, é gerada a malha tetraédrica na região correspondente (Figura 2.20). A próxima seção faz um detalhamento do algoritmo de geração de malhas de tetraedros.

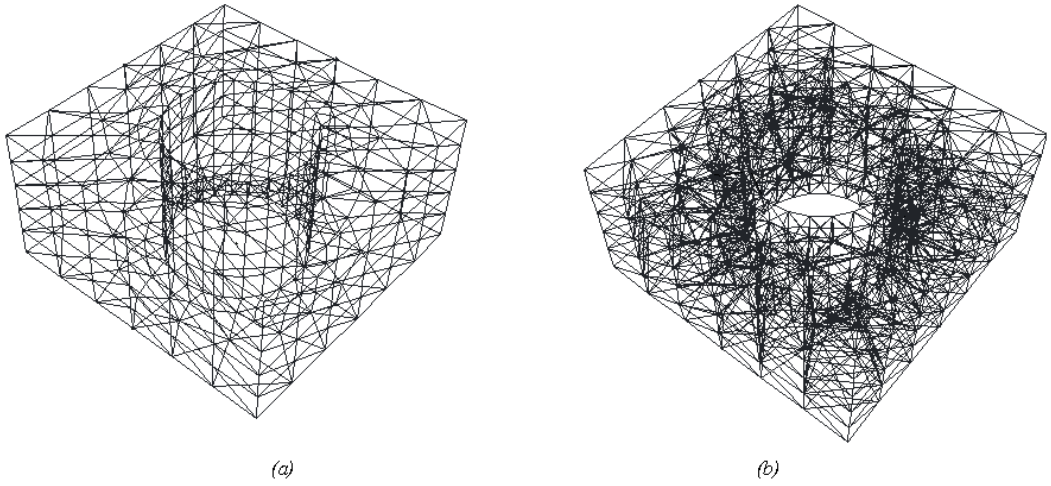


Figura 2.20 – Geração de sólido com domínio arbitrário [6].

2.3.5. Geração de Malhas Não-Estruturadas no MG

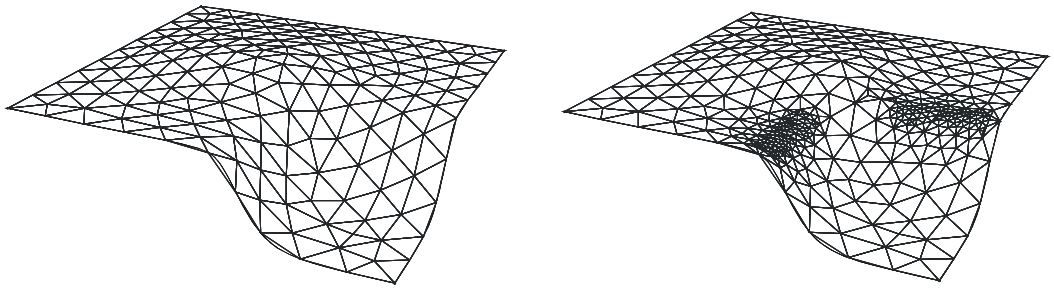
Os algoritmos utilizados no MG para geração de malhas não-estruturadas volumétricas e de superfícies combinam técnicas de avanço de fronteira [37,38] com decomposição espacial recursiva [39,40]. Estes algoritmos usam um procedimento que procura melhorar localmente a qualidade dos elementos gerados. Além disso, esses algoritmos apresentam boa transição entre as malhas de regiões com diferentes tamanhos de elementos.

Tais algoritmos devem gerar malhas que sejam conformes com as discretizações existentes no domínio, para que se possam gerar localmente novas malhas em processos adaptativos. Além disso, eles devem tratar casos onde as superfícies possuem curvaturas acentuadas, refinando a malha localmente nas regiões onde ocorrem estas curvaturas e evitando assim que elementos vizinhos formem locais pontiagudos.

2.3.5.1.
Geração de malhas em superfícies

O algoritmo implementado no MG para geração de malhas não-estruturadas em superfícies foi desenvolvido por Miranda [26]. Utilizando o espaço paramétrico da superfície, a superfície 3D é mapeada para uma superfície 2D e então a triangulação é realizada com correções nas distorções geométricas. Em seguida, a malha resultante dessa triangulação é reconduzida para o espaço 3D. A Figura 2.21 ilustra uma malha em superfície 3D.

Os dados de entrada para geração de malhas de superfícies utilizando este algoritmo são uma lista de nós definidos por suas coordenadas paramétricas e uma lista com o número de arestas em cada *loop* do modelo.



(a) *(b)*

Figura 2.21 – Exemplo de malha em uma superfície 3D: a) sem considerar distorções; b) considerando distorções.

2.3.5.2.
Geração de malhas volumétricas

No MG, um algoritmo desenvolvido por Cavalcante Neto [25,36] é utilizado para gerar malhas volumétricas não-estruturadas através de uma técnica de avanço de fronteira combinada com uma técnica de decomposição espacial recursiva, como no caso das malhas superficiais.

O algoritmo permite a representação de geometrias quaisquer, incluindo furos, cavidades e trincas, e está dividido em três passos. O primeiro é a geração de uma *octree*, utilizada para desenvolver diretrizes locais usadas para definir o tamanho dos elementos a serem gerados a partir do tamanho dos elementos triangulares na malha de contorno fornecida como dado de entrada. O segundo passo é o avanço de fronteira, que por sua vez é dividido em três etapas: geração de elementos baseada na geometria, geração de elementos

baseada na topologia e *backtracking*, usado para eliminar algumas faces dos elementos que estão impedindo o algoritmo de completar a malha.

O último passo consiste numa melhoria local da malha gerada pelo procedimento de avanço de fronteira. Ele é dividido em duas etapas: a primeira é uma técnica de suavização convencional pela realocação de nós baseado na média das coordenadas nodais, com testes de validação. A segunda etapa consiste no procedimento de *backtracking*, que remove faces de elementos de forma ruim para criar uma região onde elementos com melhor forma possam ser gerados.

2.3.6. Criação de grupos

A criação de grupos é uma ferramenta nova no MG, implementada durante o desenvolvimento deste trabalho. Além de outras funcionalidades, a existência de grupos no MG é essencial para a implementação das operações booleanas neste modelador, para que se possa atingir o nível de generalização e abrangência desejados.

De uma forma geral, um *grupo* é um conjunto de entidades topológicas formando um sub-domínio do modelo em estudo. Grupos podem conter vértices, arestas, retalhos de superfícies 2D (*patches 2D*) e regiões (*patches 3D*). Como não há nenhuma restrição quanto à seleção das entidades que farão parte de um grupo, um mesmo grupo pode conter várias entidades com dimensões diferentes, inclusive entidades pendentes ou totalmente “soltas” no espaço, como faces que não pertencem ao contorno de nenhuma região, arestas que não pertencem ao contorno de nenhuma face ou vértices sem arestas incidentes. A partir de uma entidade topológica pertencente a um grupo podem-se pesquisar as suas relações de adjacência para se obter todos os elementos adjacentes a esta entidade.

No MG, os grupos são representados por meio de uma classe *Group*. Os objetos desta classe contêm uma lista encadeada que guarda as entidades topológicas pertencentes àquele grupo. Cada grupo possui uma identificação (*label*) e não há restrições quanto à presença de uma mesma entidade topológica em dois ou mais grupos distintos. Uma outra classe, *GroupItf*, é responsável pelo gerenciamento da interface. Ela faz a comunicação entre a o modelador e o usuário na criação e manipulação de grupos. Esta classe é responsável pela criação de uma árvore na interface do programa para exibir de

forma organizada os grupos e suas entidades. Mais detalhes sobre estas classes e seus métodos serão apresentados no capítulo que descreve a implementação das operações booleanas no MG.

A criação ou remoção de um grupo no MG é feita através de sub-menus na interface do programa. No momento da criação, o usuário escolhe um *label* para o grupo. A seleção das entidades que farão parte de um grupo é realizada da forma convencional, no modo de seleção. Pode-se selecionar uma entidade de cada vez e acrescentá-la ao grupo escolhido, ou adicionar todas as entidades que farão parte daquele grupo de uma só vez. As Figuras 2.22 e 2.23 ilustram algumas destas funcionalidades.

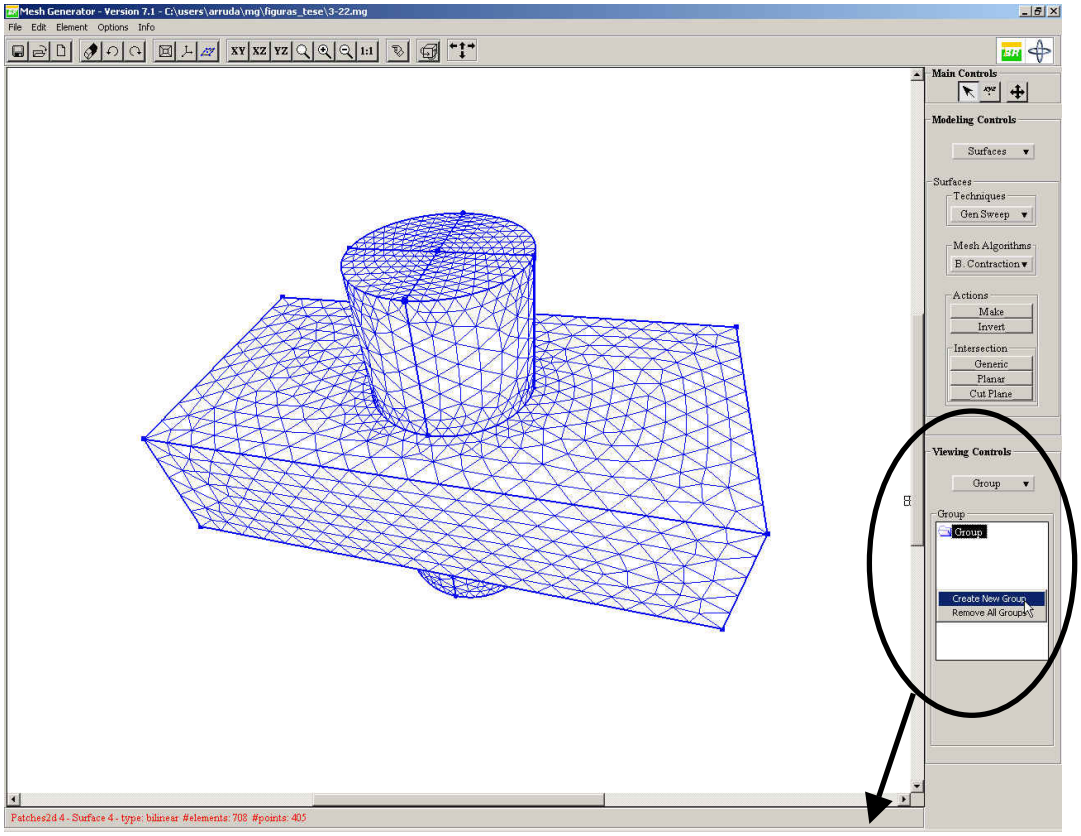
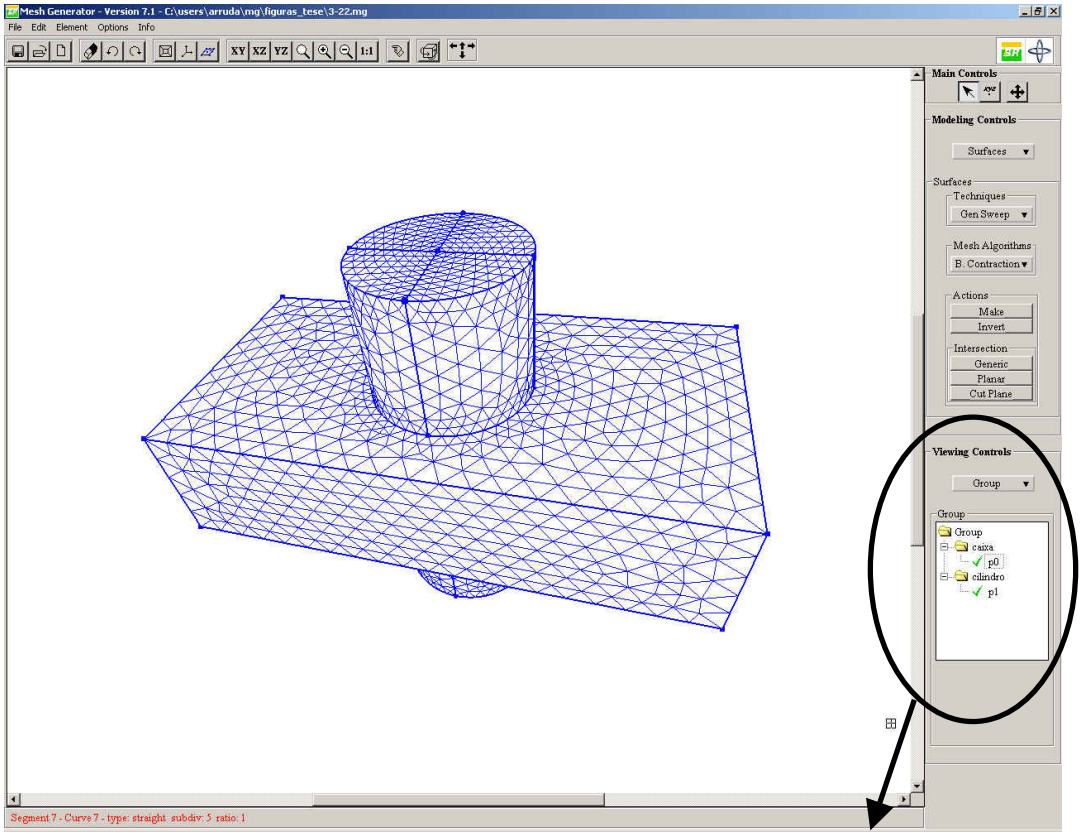


Figura 2.22 – Criação de um novo grupo no MG.



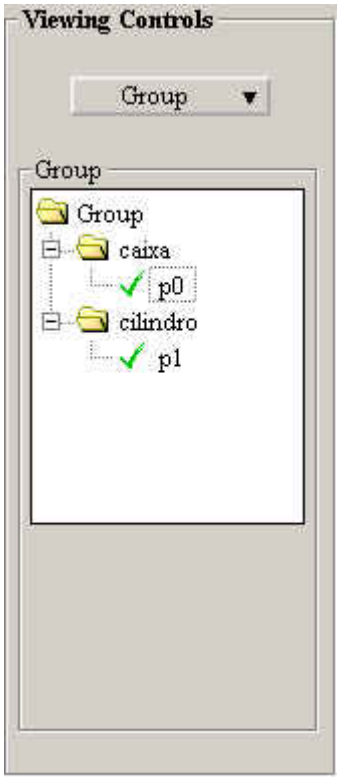


Figura 2.23 – Árvore de grupos com suas entidades topológicas.

2.3.7. Atributos

O ESAM, sistema de configuração e gerenciamento de atributos utilizado pelo MG, utiliza uma abordagem conhecida como modelagem baseada em geometria, onde os atributos são associados às entidades geométricas do modelo e a discretização dessa geometria (malha de elementos finitos) herda, automaticamente, estes atributos [6]. Esta abordagem oferece várias vantagens, como um melhor suporte para a automação do processo de modelagem, incluindo geração automática de malha e análise adaptativa, e a simplificação da geração do modelo para a simulação.

Uma descrição completa do sistema ESAM pode ser vista em [41,42], inclusive com as especificações necessárias à sua utilização.

2.3.8. Estrutura de dados híbrida

A partir do trabalho de Lira [6], o MG passou a adotar um enfoque híbrido de modelagem, baseado na combinação da representação CGC (*Complete Geometric Complex*) com a estrutura de dados do modelador, onde a representação CGC não é mantida durante todo o processo de modelagem. Um modelo CGC passou a poder ser criado em qualquer instante, quando solicitado pelo usuário para realizar a detecção automática de regiões. A criação do modelo CGC apenas nesta etapa de modelagem é devida à dificuldade de se alcançar um grau razoável de eficiência na interface com o usuário se a consistência entre geometria e topologia for forçada após cada tarefa realizada pelo usuário.

Cavalcanti [16,19] propôs um enfoque *non-manifold* para modelagem de multi-regiões. Uma metodologia geral para a criação e manipulação de uma subdivisão espacial em células com forma e geometria arbitrária foi desenvolvida. A subdivisão espacial pode ser criada através da inserção, uma a uma, de retalhos de superfícies planas, permitindo a inserção de novos retalhos em tempo real. O objeto resultante desta decomposição é classificado como um CGC. A representação CGC utilizada no MG é implementada como uma biblioteca de classes C++. Nesta representação, foi adotada a estrutura de dados *Radial Edge* (RED), proposta por Weiler [17,18] e já discutida neste trabalho. As classes da biblioteca CGC provêm operadores de alto nível que manipulam uma subdivisão espacial. Eles recebem como dados de entrada

informações geométricas dos retalhos das superfícies que são inseridas na subdivisão espacial. Estas informações geométricas são automaticamente transferidas para as informações topológicas requeridas pelos operadores *manifold* e *non-manifold* de Weiler.

Lira [6] estendeu a implementação original da CGC, que considerava apenas retalhos de superfícies planares. A partir do seu trabalho, superfícies com geometrias curvas usando NURBS passaram a ser consideradas.

O MG possui então duas representações separadas do mesmo modelo. Uma representação é armazenada na estrutura de dados do modelador. A outra é uma conversão temporária da estrutura de dados do modelador em uma representação CGC. Nessa conversão, a topologia do modelo é determinada tal que diferentes regiões possam ser reconhecidas. A CGC pode ser vista como um “costurador topológico” que gera informações de adjacência de um modelo que é consistente com a sua geometria [6]. As entidades topológicas identificadas pela representação CGC são passadas de volta para a representação MG, incluindo as regiões detectadas automaticamente.

Pode-se resumir a metodologia da modelagem híbrida adotada no MG da seguinte forma [6]:

1. O usuário gera retalhos de superfícies simples que podem se interceptar;
2. O modelador MG determina as interseções usando o algoritmo para interseção de superfícies, gerando novas curvas, segmentos de curvas e retalhos de superfícies;
3. O usuário seleciona os retalhos de superfícies que irão ser usados no modelo final, removendo partes indesejáveis;
4. O módulo CGC identifica as regiões fechadas e retorna essas informações para a estrutura de dados do MG;
5. O modelador MG calcula a malha final combinando as malhas dos retalhos e sólidos individuais.

A informação básica passada da estrutura de dados do modelador para a estrutura de dados da CGC consiste em um conjunto de retalhos de superfícies definidos pelo usuário através da interface gráfica do MG ou resultantes da interseção de superfícies. Na representação CGC, os retalhos de superfícies interceptam-se apenas em suas fronteiras.

Na conversão de volta da estrutura de dados da CGC para a estrutura de dados do MG, percorre-se todas as regiões e faces geradas pelo módulo CGC e transforma-as em entidades da representação MG.

Vale ressaltar que, no passo 3 da metodologia de modelagem descrita, a remoção de partes indesejáveis do modelo era uma tarefa que deveria ser realizada explicitamente pelo usuário. No entanto, após a inclusão das operações booleanas no MG como ferramenta adicional, foco deste trabalho, esta remoção pode ser feita implicitamente através da aplicação de uma ou mais operações booleanas sobre as entidades presentes no modelo. Isto pode ser útil, por exemplo, quando as entidades que se deseja remover são difíceis de serem selecionadas pelo usuário na área de desenho da interface (*canvas*).

2.3.9. A estrutura de dados do modelador MG

A estrutura de dados do MG distingue entidades topológicas de geométricas. Optou-se por não utilizar diretamente a *Radial Edge* como estrutura de dados do MG para não haver a necessidade de manter a consistência entre geometria e topologia em todos os passos de modelagem. Apesar disso, a estrutura de dados do MG consegue representar um modelo *non-manifold* com multi-regiões mantendo as relações de adjacência necessárias. A Figura 2.24 ilustra a organização de classes do modelador MG.

A classe *Entity* é a classe base, da qual se derivam duas outras classes, *Geometry* e *Topology*, responsáveis, respectivamente, pela representação geométrica e topológica do modelo.

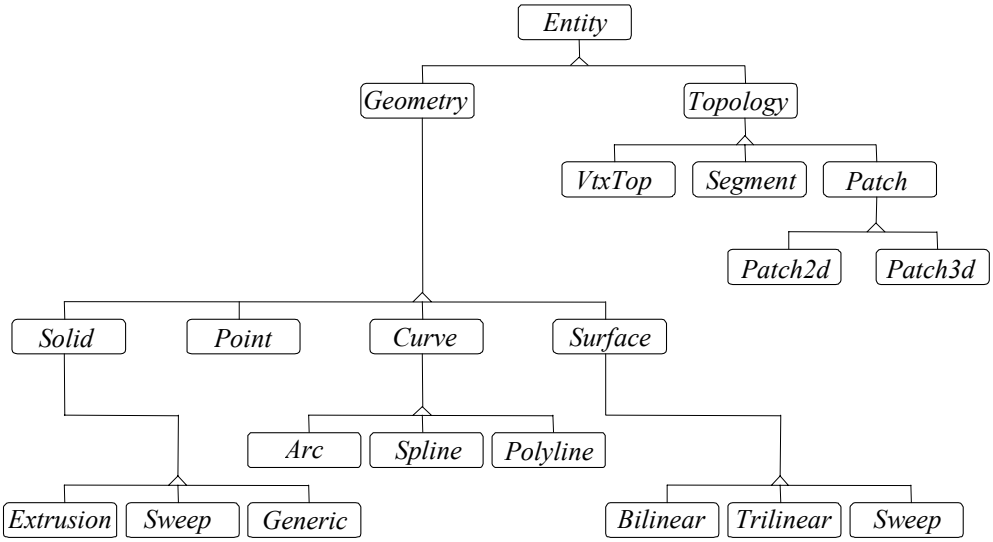


Figura 2.24 – Organização de classes do modelador MG [6].

Cada classe representando uma entidade topológica possui uma classe correspondente que representa a entidade geométrica equivalente. Ou seja, todas as entidades possuem duas instâncias, uma topológica e outra geométrica. A seguir são expostas estas entidades, as classes que as representam e algumas características destas classes:

- *Vértices* / *Pontos*: são representados por uma classe topológica *Vtxtop* e uma classe geométrica *Point*. Um objeto da classe *Vtxtop* armazena um ponteiro para o objeto da classe *Point* correspondente, uma lista de *usos* associados a este vértice (o conceito de *uso* utilizado no MG é diferente daquele introduzido pela *Radial Edge* e será definido mais tarde) e uma lista de segmentos (objetos da classe *Segment*, definida a seguir) incidentes neste vértice. Um objeto da classe *Point* armazena as coordenadas espaciais do ponto e um ponteiro para o objeto da classe *Vtxtop* correspondente. As Figuras 2.25 e 2.26 mostram as relações topológicas e geométricas dos objetos destas duas classes.

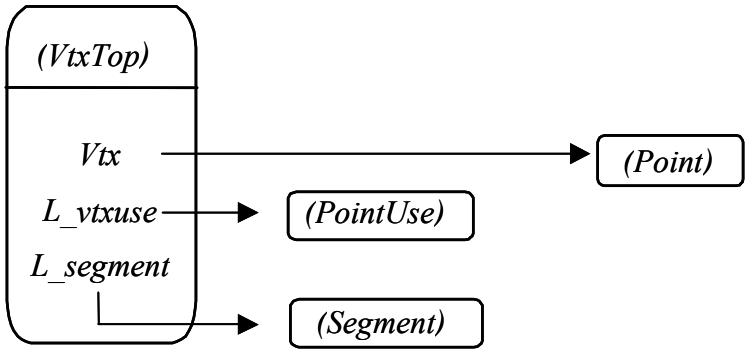


Figura 2.25 – Relações dos objetos da classe *Vtxtop* [6].

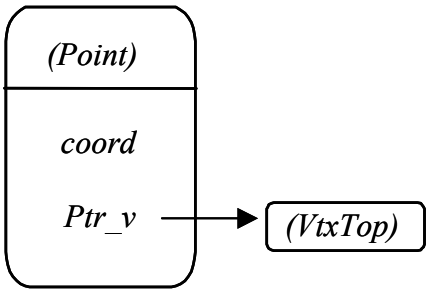


Figura 2.26 – Relações dos objetos da classe *Point* [6].

- *Segmentos* / *Curvas*: são representados por uma classe topológica *Segment* e uma classe geométrica *Curve*. Um objeto da classe *Segment* armazena uma lista de *usos* deste segmento (vale a ressalva feita anteriormente quanto ao conceito de *uso* adotado), uma lista de retalhos de superfície (objetos da classe *Patch2d*) de cujos contornos este segmento faz parte, um ponteiro para o objeto Curve correspondente, ponteiros para os objetos da classe *Vtxtop* que representam os vértices inicial e final deste segmento e dois parâmetros (números reais) que representam os limites extremos do segmento no espaço paramétrico da curva geométrica que o contém. Um objeto da classe *Curve*, por sua vez, armazena uma referência para o objeto NURBS que define a geometria da curva e uma lista de segmentos (objetos da classe *Segment*) que estão ao longo desta curva. As Figuras 2.27 e 2.28 mostram as relações topológicas e geométricas dos objetos destas duas classes.

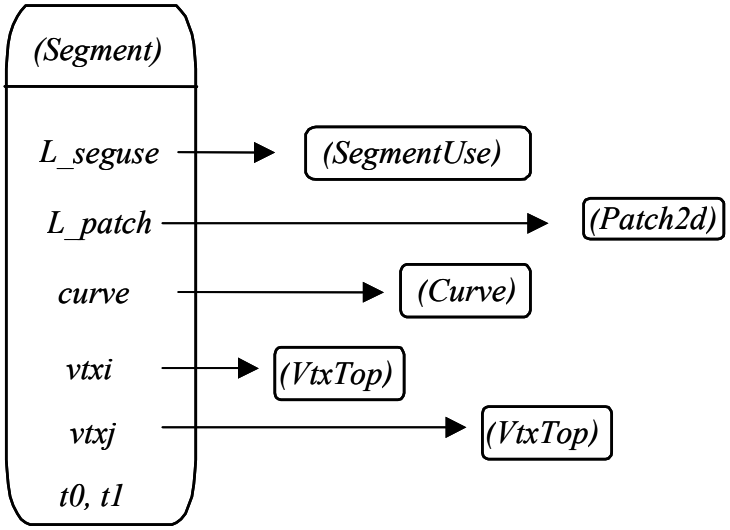


Figura 2.27 – Relações dos objetos da classe *Segment* [6].

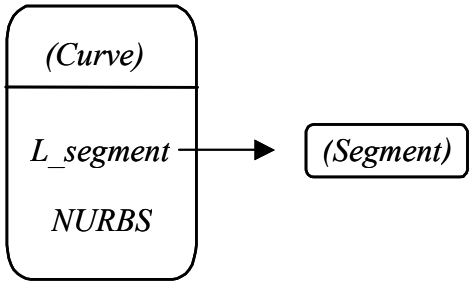


Figura 2.28 – Relações dos objetos da classe Curve [6].

- *Retalhos de superfície* / *Superfícies*: são representados por uma classe topológica *Patch2d* e por uma classe geométrica *Surface*. Um objeto da classe *Patch2d* armazena uma lista de segmentos da sua fronteira (objetos da classe *Segment*), uma lista de regiões sólidas adjacentes (objetos da classe *Patch3d*), um ponteiro para o objeto da classe *Surface* que representa a superfície geométrica que contém este retalho e uma malha de elementos finitos de superfície que eventualmente é gerada sobre ela. Um objeto da classe *Surface* armazena uma lista de retalhos de superfície existentes sobre a superfície geométrica (objetos da classe *Patch2d*), uma lista de curvas geradoras desta superfície (objetos da classe *Curve*), referências sobre seus métodos de criação (*bilinear*, *trilinear* ou *sweep*) e uma referência para o objeto NURBS que define a geometria da superfície. As Figuras 2.29 e 2.30 mostram as relações topológicas e geométricas dos objetos destas duas classes.

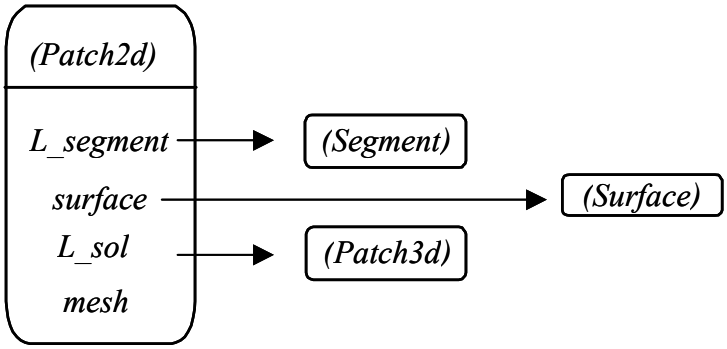


Figura 2.29 – Relações dos objetos da classe *Patch2d* [6].

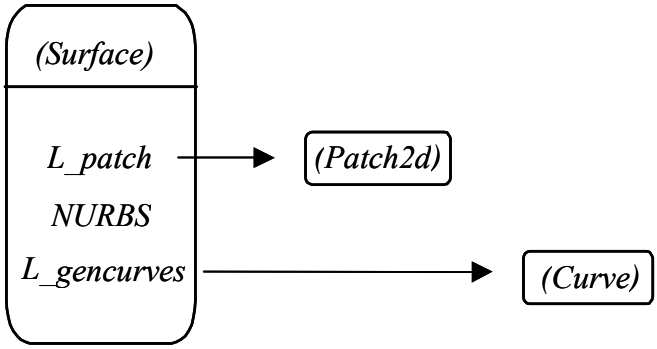


Figura 2.30 – Relações dos objetos da classe *Surface* [6].

- *Regiões* / *Sólidos*: são representados por uma classe topológica *Patch3d* e por uma classe geométrica *Solid*. Um objeto da classe *Patch3d* armazena uma lista de retalhos de superfície que formam o seu contorno (objetos da classe *Patch2d*), um ponteiro para o sólido geométrico correspondente (objeto da classe *Solid*) e uma malha de elementos finitos que eventualmente pode ser gerada sobre ele. Um objeto da classe *Solid* armazena uma lista de regiões ao longo deste sólido (objetos da classe *Patch3d*), listas de curvas e superfícies geradoras deste sólido (objetos das classes *Curve* e *Surface*, respectivamente) e informações sobre o seu método de criação. As Figuras 2.31 e 2.32 mostram as relações topológicas e geométricas dos objetos destas duas classes.

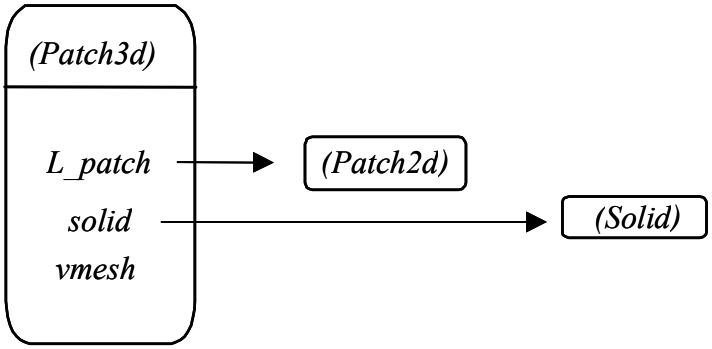


Figura 2.31 – Relações dos objetos da classe *Patch3d* [6].

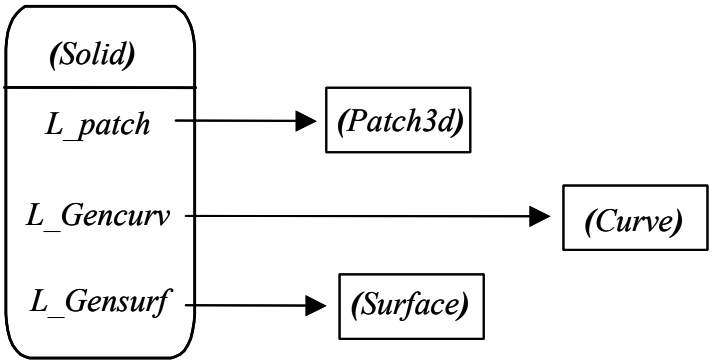


Figura 2.32 – Relações dos objetos da classe *Solid* [6].

A metodologia utilizada no MG na geração de malhas adaptativas é baseada no refinamento de cada entidade topológica no seu espaço paramétrico. Por exemplo, o refinamento dos segmentos do contorno de um retalho de superfície é necessário para a geração de malhas neste retalho. Neste contexto entra o conceito de *uso* na estrutura de dados do MG. O *uso* representa a informação geométrica de uma determinada entidade no espaço paramétrico de uma superfície. As entidades *usos* armazenam as coordenadas paramétricas [6]. Por exemplo, um vértice que está sobre duas superfícies adjacentes tem dois *usos*. A Figura 2.33 ilustra este exemplo. No caso de segmentos, cada *uso* de um segmento possui as coordenadas paramétricas dos seus pontos extremos. As classes que representam estes dois tipo de uso são descritas a seguir:

- *VertexUse*: um objeto desta classe possui uma referência para o seu objeto *VtxTop* correspondente e uma outra referência para o objeto *Surface* correspondente. Além disso, possui duas variáveis que armazenam as coordenadas paramétricas do ponto na superfície. Vale lembrar que um objeto da classe *VtxTop* (vértice topológico) possui uma lista de usos associados a ele (objetos da classe *PointUse*).

- *SegmentUse*: um objeto desta classe possui uma referência para o seu objeto *Segment* correspondente e uma outra referência para o objeto *Surface* correspondente. Além disso, possui um vetor com as coordenadas paramétricas do segmento na superfície correspondente. Vale lembrar que um objeto da classe *Segment* possui uma lista de *usos* associados a ele (objetos da classe *SegmentUse*).

As Figuras 2.34 e 2.35 mostram as relações dos objetos destas duas classes.

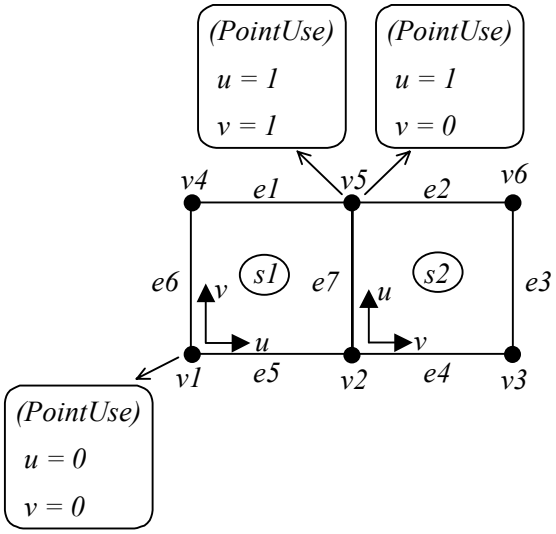


Figura 2.33 – Usos de vértices em duas superfícies adjacentes [6].

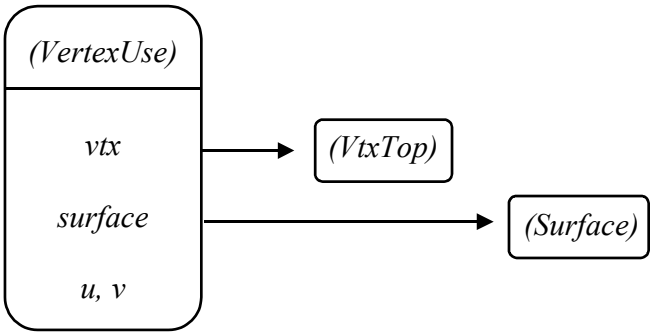


Figura 2.34 – Relações dos objetos da classe *VertexUse* [6].

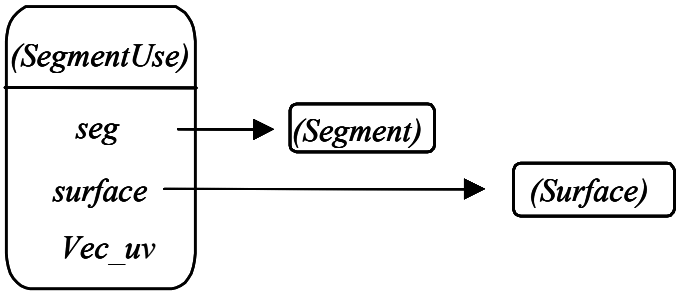


Figura 2.35 – Relações dos objetos da classe *SegmentUse* [6].

2.4. Interseção de superfícies e detecção automática de regiões

Num modelador geométrico que busca promover a simulação 3D de problemas reais de engenharia usando o MEF, a interseção de superfícies e a detecção automática de múltiplas regiões fechadas tornam-se duas ferramentas essenciais. Um número significativo de modelos das mais diversas áreas de engenharia, como estrutural, naval ou mecânica deixariam de poder ser representados, ou teriam de ser representados somente de forma aproximada caso o sistema de modelagem não comportasse tais facilidades.

Um aspecto importante também está relacionado com a definição das malhas sobre cada uma das superfícies do modelo. Dessa forma, o problema de interseção de superfícies também envolve o problema de reconstrução de malhas.

Em sua versão original, o MG já incorporava um algoritmo, proposto por Coelho [8], que resolvia o problema de interseção entre malhas de superfícies. As buscas requeridas para a determinação das curvas de interseção e a reconstrução das malhas eram suportadas por uma estrutura de dados topológica cujas principais características eram a simplicidade e o armazenamento das entidades topológicas em estruturas espaciais *B-trees* [43] e *R*-trees* [44], em vez de listas encadeadas. O uso destas estruturas espaciais aumentava a eficiência do algoritmo.

O algoritmo era baseado em um esquema para interseção de superfícies paramétricas onde as malhas de superfícies existentes eram usadas como suporte para a definição das curvas de interseção. A estrutura de dados auxiliar, definida no espaço paramétrico de cada superfície, permitia a construção das curvas de *trimming* (curvas resultantes da interseção entre superfícies e que possuem representação nos espaços paramétricos destas e no espaço tridimensional) sem a necessidade de se realizar buscas globais.

O algoritmo proposto por Coelho para determinar a interseção entre as malhas das superfícies *A* e *B* tem três passos básicos [6]:

1. Determinação dos pontos de interseção:
   1a) Cálculo e armazenamento das interseções das arestas em A contra as faces em B;
   1b) Cálculo e armazenamento das interseções das arestas em B contras as faces em A.
2. Determinação das curvas de *trimming*:
   2a) Conexão dos pontos de interseção em linhas poligonais representando as curvas de *trimming*;
   2b) Interpolação das curvas paramétricas passando pelos pontos da linha poligonal;
   2c) Determinação de novos pontos com espaçamento adequado nessas curvas;
   2d) Projeção destes novos pontos em cada superfície.
3. Reconstrução da topologia:
   3a) Determinação das regiões de *trimming* removendo vértices e arestas baseadas na linha poligonal;
   3b) Inserção de novas arestas sobre as curvas de *trimming* usando os novos pontos definidos no passo 2c;
   3c) Triangulação das regiões de *trimming* em ambas as superfícies;
   3d) Suavização das malhas.

Este algoritmo pode ser aplicado em uma grande quantidade de situações envolvendo problemas de engenharia, como pode ser visto na Figura 2.36. No entanto, existem alguns casos patológicos que não eram tratados por esta implementação original do algoritmo, tais como a interseção entre superfícies onde uma delas já foi interceptada anteriormente e a nova curva de *trimming* intercepta uma já existente.

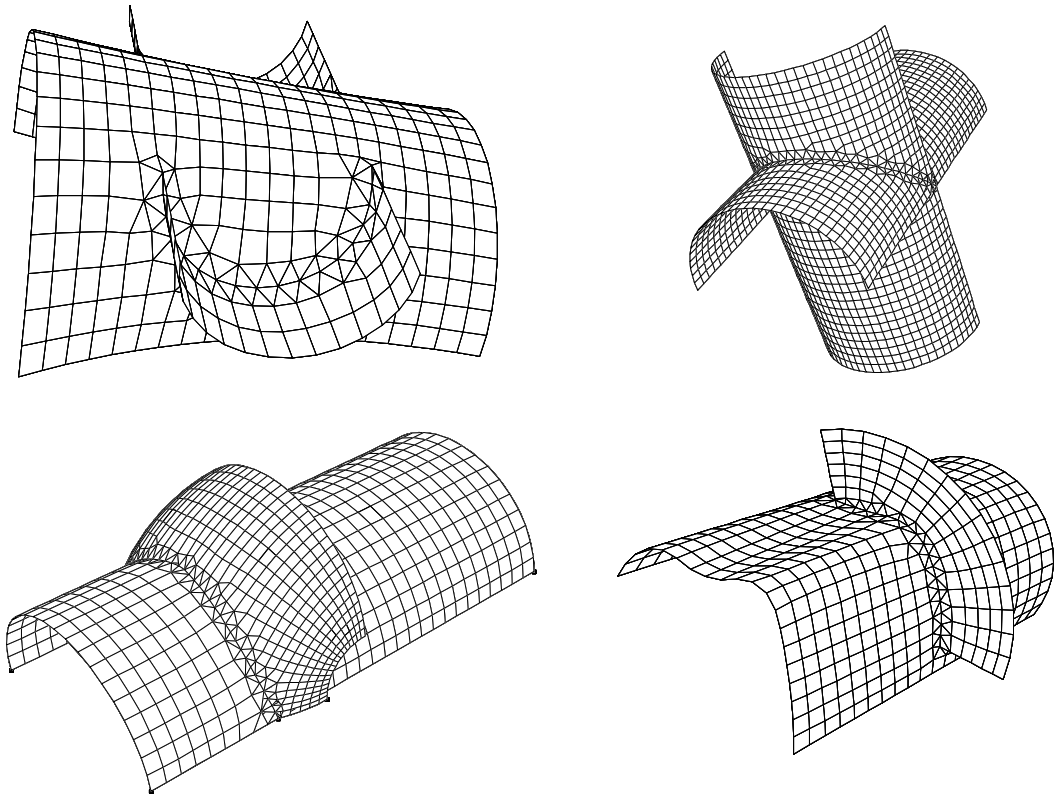


Figura 2.36 – Exemplos de interseções de superfícies [6].

Baseado nestes aspectos, Lira [6] propôs em seu trabalho alterações na implementação original do algoritmo com o objetivo de fornecer mais robustez e confiabilidade ao mesmo. Deve-se observar que no trabalho de Lira as idéias centrais do algoritmo permaneceram inalteradas, estando as principais modificações no algoritmo relacionadas com a sua implementação.

Os casos especiais tratados no trabalho de Lira referem-se às seguintes situações: *interseção aresta*/*aresta*, quando uma aresta de uma superfície intercepta uma aresta da outra superfície; *interseção aresta*/*vértice*, quando a aresta de uma superfície intercepta um vértice da outra; *curvas interceptando curvas já existentes*, quando a curva de *trimming* intercepta uma outra já existente; *interseção de superfícies não-retangulares*, como por exemplo superfícies triangulares; e *reconstrução das malhas em superfícies incidentes às curvas de interseção*. As Figuras 2.37, 2.38, 2.39 e 2.40 ilustram exemplos desses casos patológicos. Para uma descrição completa do algoritmo de interseção originalmente implementado no MG e do tratamento dos casos patológicos, inclusive com detalhes sobre as estruturas de dados utilizadas, recomenda-se a leitura dos trabalhos de Coelho [8] e Lira [6].

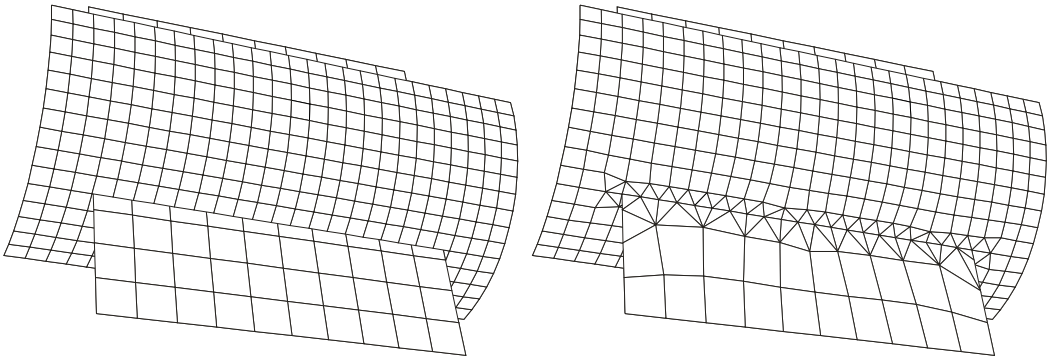


Figura 2.37 – Exemplo de interseção de malhas dos casos especiais aresta / aresta e aresta / vértice [8] e [6].

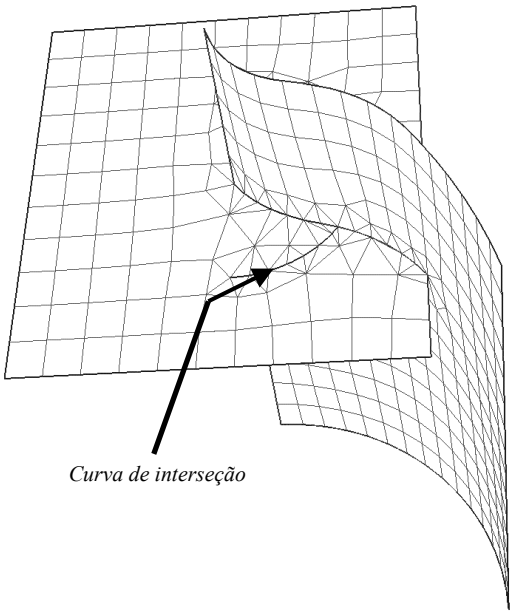


Figura 2.38 – Exemplo de interseção onde uma curva de *trimming* intercepta outra [6].

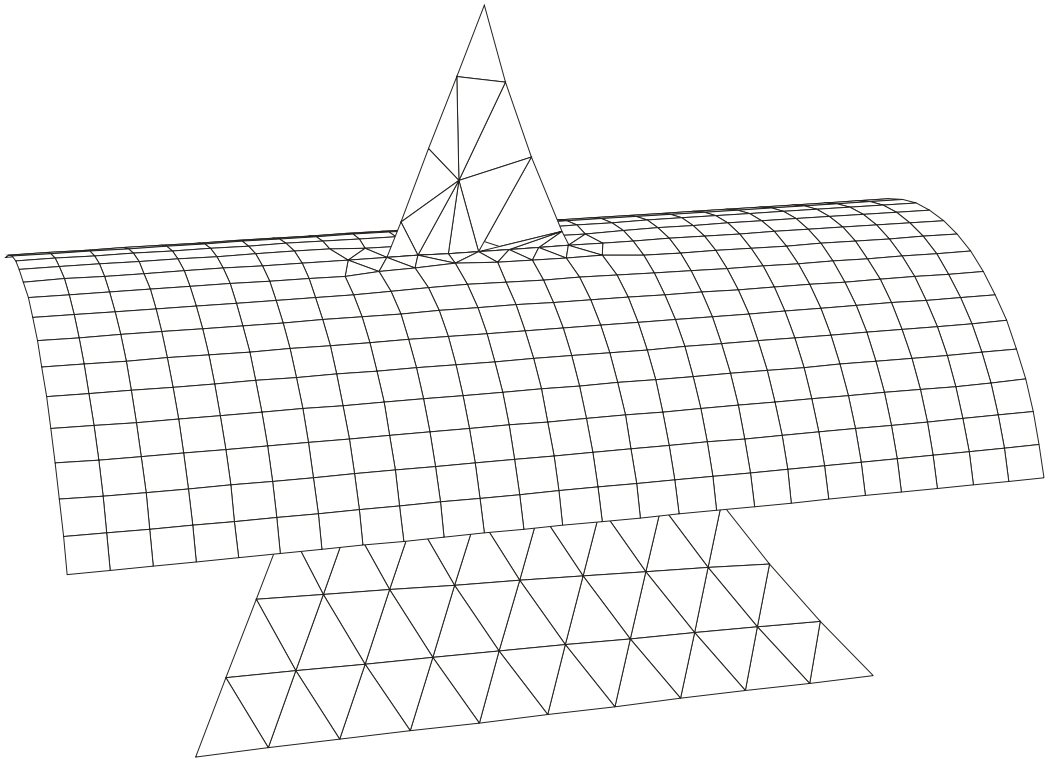


Figura 2.39 – Interseção de superfícies não-retangulares [6].

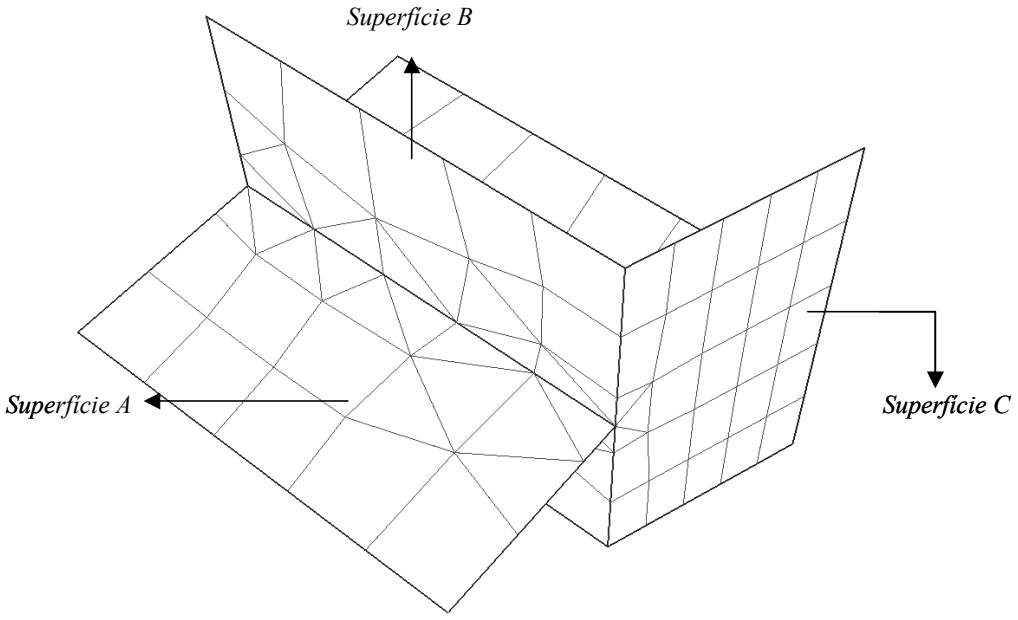


Figura 2.40 – Reconstrução da malha em superfícies incidentes às curvas de interseção [6].

Recentemente, o algoritmo de interseção do MG foi modificado por Lira de forma que todas as interseções entre retalhos de superfícies sejam calculadas através da biblioteca CGC. Os parâmetros de entrada do algoritmo são dois retalhos de superfícies (*patches2d*) e suas respectivas malhas superficiais. Os passos do novo algoritmo são os seguintes:

1. Refinamento das malhas, forçando que todos os elementos sejam triangulares. Isto garante que todos os elementos sejam faces planares. É importante observar que cada elemento possui uma referência para o retalho de superfície (*patch2d*) ao qual ele pertence. Isto permite que se recupere posteriormente a superfície associada a um determinado elemento.

2. Insere-se cada um dos elementos na RED (usando-se a biblioteca CGC). Cada um desses elementos é considerado como uma face planar. À medida em que os elementos são inseridos, a CGC vai computando automaticamente as interseções entre essas faces [19]. Ao final desta etapa, tem-se um modelo planar totalmente costurado topologicamente e consistente. Vale notar que novos vértices, arestas e faces podem ser criados nas regiões próximas às interseções. Dois importantes detalhes devem ser observados:
   - Novas arestas só serão criadas sobre a poligonal que representa a linha de interseção. Com isso, é possível detectar quais são os pontos de interseção obtidos;
   - Devido à representação topológica do modelo planar, é possível identificar os componentes conexos da interseção. Ou seja, pode-se verificar a existência de uma ou mais poligonais contendo os pontos de interseção e se essas poligonais são abertas ou fechadas.

3. Com as respostas obtidas no item 2, é possível computar todas as poligonais que representam as curvas de interseção entre as malhas planares utilizadas. Esses pontos de interseção obtidos servem como uma excelente aproximação para o cálculo de pontos de interseção entre as superfícies.

4. Usando um algoritmo de minimização pelo método de *Newton-Raphson*, calculam-se os pontos de interseção das superfícies a partir dos pontos obtidos no item 3. Isto é feito para cada ponto de interseção. Como os pontos obtidos no passo 3 estão próximos das interseções das superfícies (inclusive, no caso de superfícies planares eles são exatamente os pontos de interseção), o algoritmo converge rapidamente e encontra a resposta desejada. Alguns casos especiais

são tratados, principalmente em pontos próximos ao contorno das superfícies.

5. Os pontos de interseção obtidos no item 4 são interpolados, usando-se curvas NURBS, gerando as curvas de interseção.

6. Com essas curvas (segmentos), faz-se a compatibilidade topológica no MG, gerando novos *patches2d* (se necessário) a partir dessas curvas de interseção obtidas.

7. As malhas de elementos finitos associadas aos *patches2d* (antigos ou novos) são re-geradas usando-se o algoritmo para geração de malhas de superfícies desenvolvido por Miranda [26].

Um caso especial que ainda não havia sido tratado até a conclusão deste trabalho é aquele em que duas superfícies se sobrepõem (*overlapping*). Por se tratar de uma situação bastante peculiar no caso de superfícies curvas, não havia sido dada ênfase para o tratamento de patologias deste tipo. Contudo, quando se trabalha com superfícies planares, é mais comum surgirem situações onde isto ocorre. Apesar do algoritmo para operações booleanas proposto neste trabalho considerar casos especiais como este, a geração de modelos no MG para validar o algoritmo proposto nestes casos torna-se difícil, já que o algoritmo de interseção de superfícies, cuja chamada é requisitada antes da aplicação das operações booleanas, na maioria das vezes falha quando as superfícies se sobrepõem.

A interseção entre superfícies e curvas foi recentemente implementada no MG. Na verdade, este procedimento já era realizado como uma etapa da interseção de superfícies, mas não estava disponível para a sua utilização direta. Isto quer dizer que não havia suporte para se calcular explicitamente a interseção entre uma curva qualquer e uma superfície. A partir de sua última versão, o MG passou a contar com mais esta ferramenta, também crucial para a implementação correta do algoritmo de operações booleanas.

Um último tipo de interseção também está implementado no MG: a interseção entre duas curvas quaisquer. Obviamente, um algoritmo para resolver este problema é mais simples do que aquele apresentado para a interseção de superfícies. Na verdade, como a interseção de superfícies no MG é feita utilizando-se as malhas dos retalhos de superfície como suporte para a

determinação das curvas de interseção, este procedimento também já era realizado como uma etapa da interseção de superfícies.

O procedimento para a detecção de múltiplas regiões fechadas já foi descrito quando se apresentou a estrutura de dados híbrida do modelador MG. A representação CGC, que adota a estrutura de dados *Radial Edge* proposta por Weiler [17], é utilizada como uma forma de representação temporária da topologia do modelo quando o reconhecimento de regiões é requisitado pelo usuário. Vale ressaltar que as interseções de superfícies são realizadas antes do reconhecimento de regiões, para que na representação CGC os retalhos de superfície se interceptem apenas nas suas fronteiras. No final do processo, cada região na CGC gera um sólido na representação MG. A Figura 1.3 mostra o modelo explodido de parte de uma plataforma usada na exploração de petróleo em águas profundas, onde as regiões foram detectadas automaticamente pela técnica descrita.

A ênfase dada neste trabalho a estas duas ferramentas de modelagem é devida à importância que elas exercem na implementação do algoritmo de operações booleanas no MG. Como será visto nos próximos capítulos, para que o algoritmo possa ser aplicado corretamente sobre um conjunto de entidades topológicas pertencentes a um modelo, é necessário que todas as interseções entre superfícies, entre superfícies e curvas e entre curvas já tenham sido computadas. Esta condição forma a base para um critério de classificação das entidades topológicas de um *grupo* perante as entidades topológicas de outro *grupo*. Além disso, o reconhecimento automático de multi-regiões se faz necessário para que um conjunto conexo e fechado de retalhos de superfície possa ser tratado como um sólido, o que também é crucial para a classificação das entidades topológicas. Além disso, após a aplicação das operações booleanas sobre as entidades selecionadas, o modelador deve ser capaz de reconhecer eventuais regiões formadas com as entidades remanescentes.

3
Algoritmo para Operações Booleanas

Este capítulo traz o foco principal deste trabalho, que é a apresentação de um algoritmo genérico para a realização das operações booleanas em um sistema de modelagem geométrica baseado em representação de fronteira (B-Rep). O algoritmo pode ser qualificado como genérico no sentido de não impor restrições quanto ao domínio dos objetos a serem combinados por meio das operações booleanas, não impor restrições quanto aos resultados obtidos após a aplicação destas operações, não estar amarrado a um modelador específico, não restringir a geometria ou a quantidade dos objetos a serem tratados numa única operação e tratar casos patológicos como superposição de superfícies (*overlapping*).

Consideradas como uma ferramenta importante para qualquer sistema de modelagem, as operações booleanas são uma maneira simples e eficiente de se determinar objetos sólidos complexos. Em sistemas de modelagem baseados em representações de fronteira (B-Rep), as operações booleanas também podem ser um dos maiores desafios em termos de implementação. Neste capítulo os conceitos já apresentados nos capítulos anteriores são utilizados para descrever um algoritmo relativamente simples para realizar operações booleanas em sistemas de modelagem com representação B-Rep e domínio representacional *non-manifold*. Discutem-se as condições de aplicabilidade do algoritmo, as informações que devem ser armazenadas antes e durante a sua aplicação, os algoritmos geométricos auxiliares necessários em alguns passos do algoritmo, e cada operação booleana separadamente. Em seguida, a questão da regularização é tratada, analisando-se a importância da existência de tal facilidade.

3.1.
Domínio representacional *non-manifold*

O algoritmo que será proposto pode ser aplicado em situações topológicas *non-manifold*. Isto significa que ele é aplicável num ambiente que englobe todas as formas de representação já comentadas: por arames, por superfícies e

modelagem de sólidos. Objetos sólidos com estruturas internas ou pendentes, multigrafos, mais de duas faces possuindo uma aresta em comum, dois sólidos se tocando em um único vértice ou com uma face em comum, arames emanando de superfícies, vértices, arames ou faces soltos no espaço (cada elemento representando uma casca) e outras situações são permitidas.

As condições topológicas *non-manifold* descritas acima podem aparecer como dados de entrada do algoritmo ou como resultado da aplicação do mesmo sobre objetos quaisquer que podem ou não ser *manifold*. As Figuras A.8 e A.36 mostram a aplicação das operações booleanas de união e interseção entre dois sólidos *manifold* levando a resultados *non-manifold*.

A vantagem de se utilizar um domínio *non-manifold* para se aplicar as operações booleanas é que estas passam a ser um conjunto *fechado* de operações. Diz-se que uma operação é *fechada* num domínio quando a aplicação desta operação sobre uma entidade do domínio gera um resultado igualmente representável naquele domínio. Desta forma, evita-se a imposição de restrições ou do uso de aproximações quanto aos resultados obtidos. A Figura A.9 mostra possíveis aproximações para uma situação *non-manifold* de forma que o resultado da aplicação da operação booleana possa ser representado num ambiente de modelagem *manifold*.

Mesmo a operação de *regularização* do resultado pode não ser suficiente para se garantir que o mesmo seja representável num ambiente de modelagem *manifold*. O processo de regularização consiste em eliminar do resultado da aplicação da operação booleana todas as entidades de dimensão inferior que ficaram pendentes ou soltas, ou seja, considerar apenas os volumes preenchíveis. Diz-se que a regularização de um conjunto de pontos é definida como o *fecho do interior* deste conjunto de pontos (Apêndice). A Figura 3.1 ilustra um exemplo em que a aplicação da regularização ao resultado da diferença dos dois sólidos não altera as condições *non-manifold* do resultado.

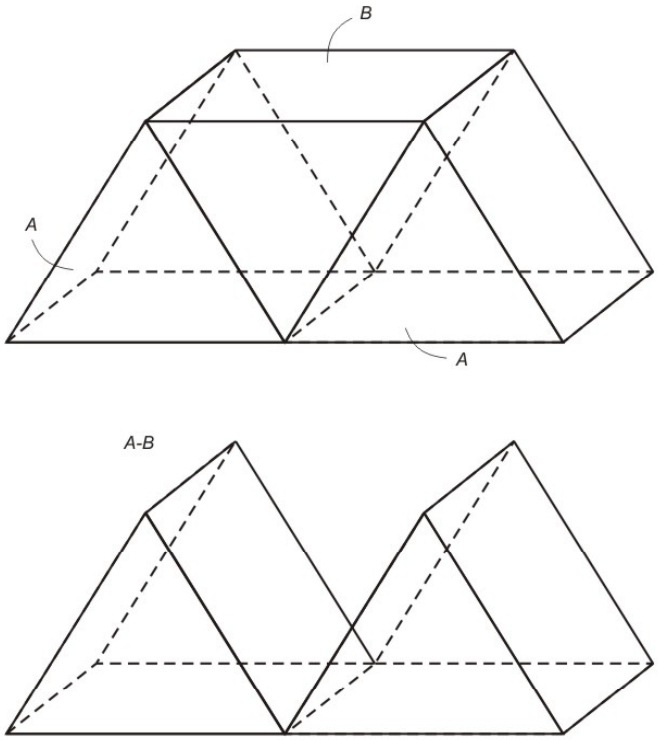


Figura 3.1 – Operação de diferença levando a um resultado *non-manifold*.

3.2. Conceito de fronteira

É necessário se fazer uma ressalva quanto ao significado do termo *fronteira* que será utilizado nas seções seguintes. Este é um termo que será utilizado somente quando se estiver tratando de faces ou de regiões (não é necessário se definir a fronteira de uma aresta ou de um vértice).

Operações booleanas usualmente estão relacionadas a conjuntos de pontos (*point sets*). Isto quer dizer que, quando dois objetos ocupam um lugar no espaço, as operações booleanas aplicadas a estes objetos costumam fornecer um conjunto de pontos que ocupam um lugar no espaço que é um subconjunto do lugar ocupado por estes dois objetos. Desta forma, considera-se que uma região fechada contém todos os seus pontos interiores, assim como uma face contém todos os seus pontos que obedecem à descrição geométrica da superfície que a contém. Pode-se dizer que as operações booleanas costumam ser aplicadas considerando-se o conceito de lugar geométrico ocupado por dois objetos quaisquer.

Contudo, quando se fala em operações booleanas em domínios *non-manifold*, surge uma inconsistência nesta definição. Situações topológicas *non-manifold* como vértices, arestas ou faces soltos no interior de regiões, arestas ou

faces pendentes de regiões internamente, arestas pendentes de faces interiormente ou arestas soltas no interior de faces representam redundâncias quando se pensa em conjuntos de pontos. Isto quer dizer que as entidades topológicas pendentes ou soltas descritas acima ocupam um lugar no espaço que já era ocupado pelas entidades topologicamente superiores que as contêm. Logo, elas deveriam ser desconsideradas na aplicação de uma operação booleana.

Todavia, em modelagem geométrica aplicada à engenharia existem situações em que tais entidades pendentes ou soltas podem ser importantes na caracterização do domínio do problema a ser estudado. Em problemas reais de engenharia usando o MEF, por exemplo, estas entidades podem representar restrições na geração das malhas de elementos finitos.

Tendo isto em vista, para que não se tenha que restringir o domínio dos modelos sobre os quais o algoritmo de operações booleanas poderá ser aplicado, e ao mesmo tempo buscando-se não se desviar do ponto-de-vista de lugar geométrico comumente atribuído às operações booleanas, as entidades topológicas soltas ou pendentes internamente a outras entidades serão consideradas, mas o conceito de fronteira de uma face ou de uma região adotado será o mesmo aplicado aos domínios *manifold*, como será visto a seguir.

A *fronteira* de uma entidade topológica, neste contexto, corresponde a todas as entidades topológicas hierarquicamente inferiores a ela (de dimensões menores que ela) que definem o seu contorno e que podem ser obtidas por relações de adjacência através da sua estrutura de dados. A fronteira de uma entidade pode conter várias porções desconexas, mas cada porção deve formar um conjunto conexo e fechado de entidades de mais baixa dimensão. Por exemplo, as arestas e vértices do *loop externo* de uma face e de um *loop interno* desta mesma face constituído por um ciclo fechado e alternado de arestas e vértices fazem parte da sua fronteira. Contudo, uma aresta solta no interior de uma face correspondendo a um *loop interno* formado somente por esta aresta não faz parte da fronteira desta face. Da mesma forma, uma casca formada por um conjunto conexo e fechado de faces no interior de uma região pertence à fronteira desta região. Contudo, uma face pendente internamente ou externamente a uma região não faz parte da fronteira desta região. Vale lembrar que esta definição de fronteira não corresponde necessariamente àquela adotada em outros trabalhos. Na estrutura de dados *Radial Edge*, por exemplo, uma face pendente de uma região internamente ou uma aresta pendente de uma

face internamente fazem parte da fronteira destas entidades, bem como faces soltas no interior de regiões e arestas soltas no interior de faces [17,18]. Ao se percorrer a lista de *usos* de uma aresta pendente de uma face internamente ou solta no interior da face, verifica-se que esta aresta possui dois *usos* associados a esta face. O mesmo ocorre com uma face pendente de uma região internamente ou solta no interior da região, que possui dois *usos* associados a esta região. Logo, deve-se atentar para definição adotada neste trabalho para o termo *fronteira*.

3.3. Grupos como parâmetros de entrada

No contexto deste trabalho, um *grupo* é definido como um conjunto de entidades topológicas formando um sub-domínio do modelo em estudo. Grupos podem conter vértices, arestas, retalhos de superfícies (faces) e regiões. Um grupo pode conter várias entidades com dimensões diferentes, inclusive entidades pendentes ou totalmente "soltas" no espaço, como faces que não pertencem ao contorno de nenhuma região, arestas que não pertencem ao contorno de nenhuma face ou vértices sem arestas incidentes. Algumas relações de adjacência devem estar diretamente disponíveis. Para cada entidade topológica, deve-se ter acesso às entidades adjacentes de níveis hierárquicos imediatamente superior e imediatamente inferior ao dela. Ou seja, deve-se ter acesso às arestas incidentes num vértice, aos vértices e faces adjacentes a uma aresta e às arestas e regiões adjacentes a uma face. Obviamente, este constitui um conjunto suficiente de relações de adjacência, visto que a partir destas relações pode-se obter quaisquer outras relações de adjacência desejadas.

As operações booleanas são sempre aplicadas entre dois grupos de cada vez. Os dois grupos devem estar sob um mesmo *framework* topológico. Cada grupo pode possuir um número qualquer de entidades topológicas. As entidades topológicas de cada grupo têm que ser pré-processadas para atender às condições de aplicabilidade do algoritmo que serão descritas na próxima seção.

3.4. Condições de aplicabilidade do algoritmo

Algumas restrições devem ser impostas em relação às entidades topológicas que servirão de entrada para o algoritmo, e quanto às suas relações

de adjacência. Para que isto seja atendido, um pré-processamento das entidades topológicas dos dois grupos é necessária.

De uma forma geral, o pré-processamento faz com que vértices, arestas e faces dos dois grupos efetivamente não se interceptem, mas apenas se toquem, ainda que não necessariamente ao longo das fronteiras destes elementos (por exemplo, a extremidade de uma aresta pode tocar o interior de uma face). Ou seja, qualquer tipo de interseção entre pontos, curvas e superfícies deve ser calculado antes da chamada do algoritmo, dividindo-se as entidades necessárias em duas ou mais entidades de mesma dimensão e criando-se os elementos topológicos (vértices, arestas e faces) que irão satisfazer às restrições a seguir.

A primeira restrição é quanto à forma de representação das informações do modelo. O algoritmo que será apresentado se destina a sistemas de modelagem com representação B-Rep, ou seja, aqueles em que os volumes sólidos são representados explicitamente pelas superfícies na sua borda. O algoritmo se baseia nas informações topológicas do modelo, que devem conter especificações sobre vértices, arestas e faces, indicando suas incidências e adjacências.

Outra restrição diz respeito às interseções entre os elementos presentes. Algumas *propriedades de não-interseção* já discutidas neste trabalho quando foram apresentados os conceitos de topologia em representações *non-manifold* são válidas aqui, mas de forma menos restritiva:

- Regiões *pertencentes a um mesmo grupo* não podem se interceptar a não ser ao longo de suas fronteiras. No entanto, regiões *pertencentes a grupos distintos* podem possuir volumes comuns.
- Uma face (retalho de superfície) só pode interceptar outra face ao longo da fronteira *de pelo menos uma das faces*.
- Arestas não podem se interceptar a não ser em seus pontos extremos.
- Arestas não podem interceptar faces a não ser nos pontos extremos *das arestas*.
- Vértices devem ser distintos espacialmente.

Para que a superposição interna de faces (sem interseção das respectivas arestas) possa ser considerada obedecendo às propriedades expostas acima, considera-se que quando uma face F_2 se sobrepõe a uma face F_1, na verdade a face F_1 possui duas fronteiras desconexas (face com um *loop* interno) e a face F_2 possui uma única fronteira que é exatamente o *loop* interno da face F_1. Isto significa que a face F_1 não contém a face F_2, ela é uma face com um buraco no

seu interior onde se “encaixa” a face F_2. Este caso pode ser visualizado na Figura 3.2.

A última restrição é que para que regiões possam ser tratadas como volumes fechados, elas já devem ser passadas para o algoritmo como regiões. Isto significa que o algoritmo não inclui uma etapa de reconhecimento de regiões, estas devem ser passadas para o mesmo contendo a informação de que constituem um volume fechado, ou seja, o reconhecimento de regiões é parte do pré-processamento. Isto é necessário pois pode haver interesse em se tratar um conjunto conexo e fechado de faces apenas como uma casca. A Figura 3.3 mostra um exemplo de aplicação da operação booleana de interseção entre dois objetos, considerando um deles como um sólido ou apenas como uma casca.

Não há restrições quanto à geometria do modelo analisado. A descrição geométrica das arestas e faces pode ser qualquer, pois as outras restrições impostas garantem um tratamento essencialmente topológico do problema.

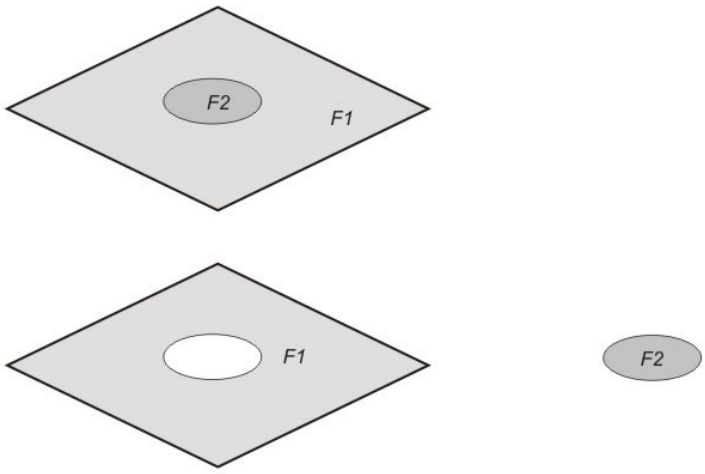


Figura 3.2 – Superposição de faces: na verdade, a face F_1 possui um *loop* interno formando uma cavidade onde se encaixa a face F_2.

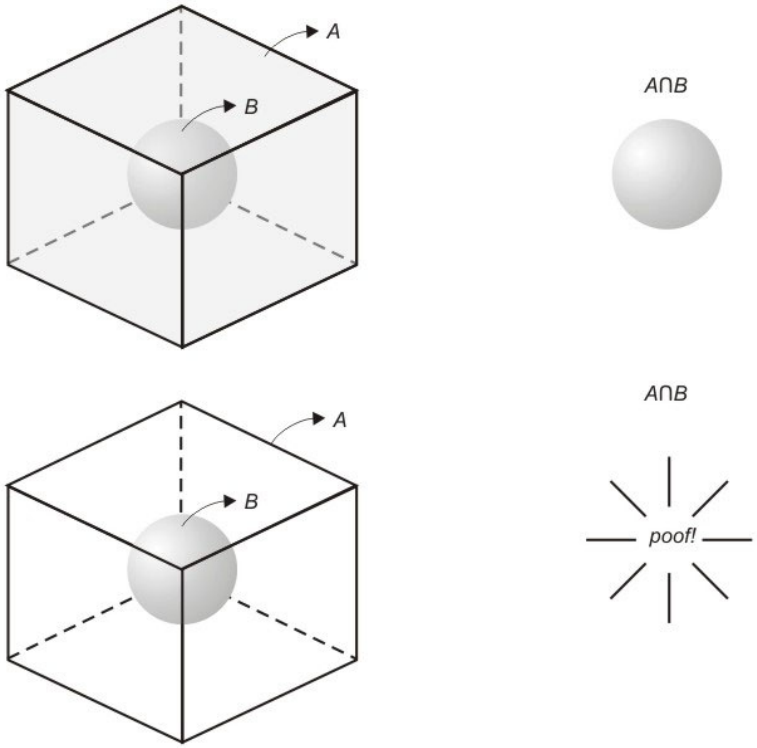


Figura 3.3 – Interseção entre dois objetos considerando um deles como uma região ou somente como uma casca.

3.5. Descrição do algoritmo

O algoritmo que será proposto nesta seção tem como base o algoritmo proposto por Mäntylä [10] para operações booleanas em representações B-Rep. Na verdade, apenas a idéia central do algoritmo é a mesma, já que o algoritmo completo proposto por Mäntylä aplica-se apenas a sólidos poliedrais (constituídos somente por faces planas) *manifold*. Além disso, o algoritmo de Mäntylä trabalha apenas com dois sólidos de cada vez, enquanto o algoritmo aqui proposto trabalha com o conceito de *grupo* exposto acima.

3.5.1. Parâmetros de entrada

As informações que devem ser passadas para o algoritmo são as seguintes:

- Dois grupos de entidades topológicas, cada grupo podendo conter um número qualquer de regiões, retalhos de superfície (faces), arestas e vértices. Os grupos serão denominados *A* e *B*.
- Tipo de operação booleana que se quer realizar com as entidades pertencentes aos grupos: união ($\cup$), interseção ($\cap$) ou diferença (-).
- *Flag* para indicar se a regularização é desejada ou não.

3.5.2. Tratamento dos parâmetros de entrada

O passo inicial do algoritmo consiste na organização dos parâmetros de entrada por meio do armazenamento em estruturas de dados adequadas (vetores ou listas) dos elementos topológicos diretamente disponíveis em cada grupo e dos elementos hierarquicamente inferiores deriváveis destes a partir das relações de adjacência. A partir daí, faz-se a classificação de todos estes elementos segundo critérios geométricos de posicionamento relativo no espaço tridimensional. Este procedimento pode ser dividido em cinco etapas:

1) Armazenamento das entidades topológicas de cada grupo

Esta etapa consiste em percorrer as listas de entidades de cada grupo e armazenar, em listas distintas, cada entidade e todas as entidades topologicamente inferiores à mesma que pertençam à sua fronteira ou que estejam pendentes ou soltas no seu interior (no caso de faces e regiões). Ou seja, quando uma aresta é armazenada, os seus vértices extremos também devem ser armazenados. Quando uma face é armazenada, todas as arestas e vértices do seu contorno, incluindo eventuais *loops* internos (que podem ser constituídos por seqüências alternadas de arestas e vértices fechando um ciclo no interior da face, arestas pendentes (*strut edges*) (Apêndice) ou ainda arestas ou vértices soltos no interior da face) também devem ser armazenados. Quando uma região é armazenada, todas as faces, arestas e vértices pertencentes ao seu contorno devem ser armazenados. A princípio, todas as estruturas internas a uma região também devem ser armazenadas, mas deve-se verificar se a estrutura de dados do modelador onde o algoritmo está sendo implementado considera as estruturas internas a uma região como parte integrante da fronteira da região. Caso não considere, estas estruturas devem ser explicitamente

selecionadas na criação dos grupos (este é o caso do modelador MG, onde este algoritmo foi implementado).

A Figura 3.4a ilustra um exemplo deste procedimento. Arestas ou faces externamente pendentes de sólidos e arestas externamente pendentes de faces não fazem parte da fronteira destas entidades, logo só serão armazenadas se pertencerem explicitamente a algum grupo (Figura 3.4b). São criadas então listas de vértices, arestas, faces e regiões para cada grupo.

É importante ressaltar que uma mesma entidade só deve ser armazenada uma única vez. Por exemplo, se um grupo contém explicitamente uma região e uma face que pertence à fronteira desta região, tal face será selecionada duas vezes, mas só deverá ser armazenada uma vez na lista de faces daquele grupo. Ou seja, deve-se verificar se uma entidade já pertence a uma lista antes de armazená-la.

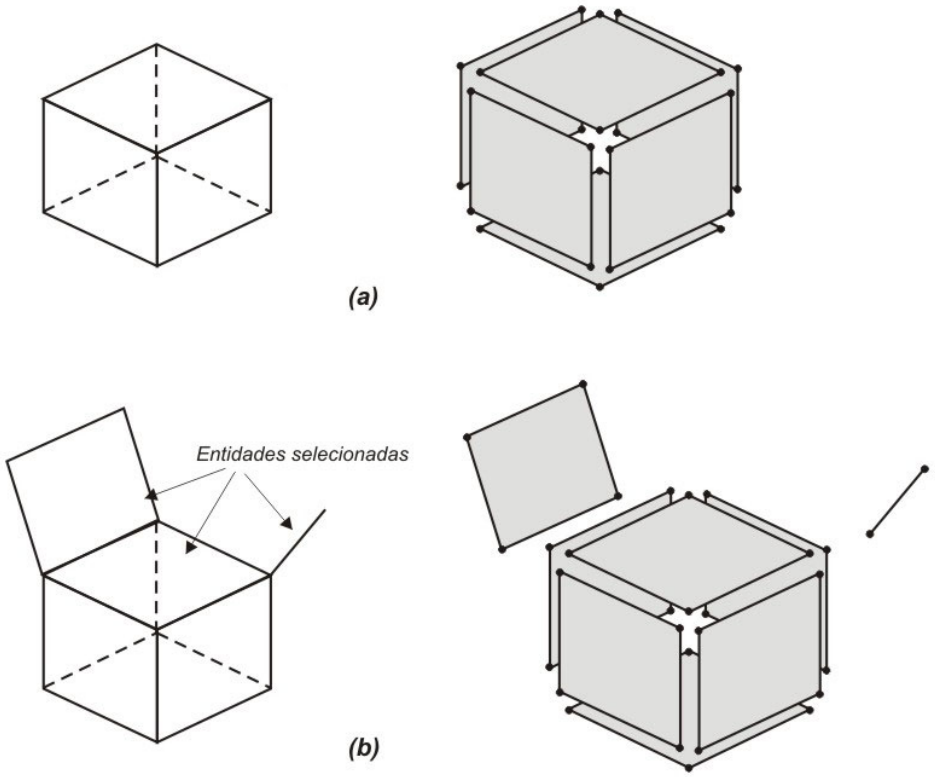


Figura 3.4 – Armazenamento das entidades topológicas explicitamente selecionadas e daquelas obtidas a partir destas por relações de adjacência: a) cubo selecionado; b) cubo, aresta pendente e face pendente selecionados.

2) Determinação das entidades comuns aos grupos

Nesta etapa, as listas de vértices, arestas, faces e regiões de cada grupo são percorridas a fim de se encontrar todas as entidades topológicas comuns aos dois grupos. Estas devem ser armazenadas numa outra lista, que será chamada de *lcommon*.

3) Classificação dos vértices

A classificação de vértices é um dos pontos cruciais do algoritmo. Nesta etapa, todos os vértices de ambos os grupos devem ser classificados. Isto inclui os vértices que possuem uma ou mais arestas incidentes, vértices que constituem por si só *loops* no interior de faces (vértices soltos no interior de faces), vértices que constituem por si só cascas no interior de regiões ou vértices soltos no espaço, também representando cascas individuais. Todos estes vértices já estão devidamente armazenados nas listas de vértices de cada grupo mencionadas na etapa número 1.

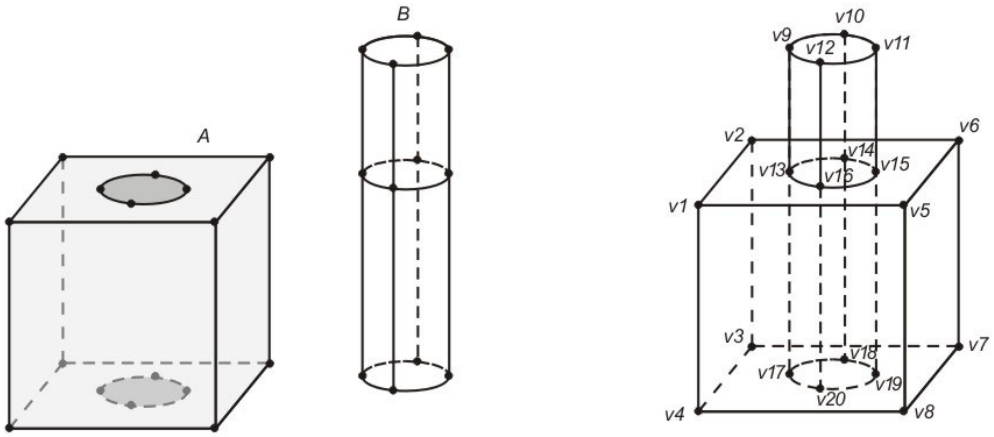
É criado um registro (estrutura de dados) para cada vértice, que contém a referência para o vértice e uma informação sobre a localização do vértice no espaço em relação às entidades do outro grupo. O critério de classificação dos vértices depende de algoritmos geométricos de *detecção de ponto em volume* e *detecção de ponto em superfície*. Estes algoritmos devem estar disponíveis para que os vértices possam ser corretamente classificados. Estes algoritmos devem ser capazes de detectar, respectivamente, se um ponto é interior ou exterior a um volume cuja fronteira é constituída por superfícies com geometria qualquer e se um ponto localiza-se ou não sobre uma superfície com geometria qualquer. Os vértices são classificados da seguinte forma:

- Se um vértice pertence à lista *lcommon*, ele é classificado como INTERS.
- Se um vértice de um grupo localiza-se no interior de uma região qualquer do outro grupo, ele é classificado como *AinB* ou *BinA*, sendo a primeira letra correspondente ao grupo ao qual o vértice pertence, e a segunda letra correspondente ao grupo que contém a região dentro da qual o vértice se localiza.
- Se um vértice de um grupo localiza-se sobre uma face qualquer do outro grupo, ele é classificado como *INTERS* (na verdade, este vértice já deveria pertencer à lista *lcommon*, pois ele constituiria um *loop*

interno da face sobre a qual ele se localiza, mas este caso foi tratado separadamente devido ao fato do modelador MG não dar suporte a vértices soltos no interior de faces).

- Todos os outros vértices são classificados como *AoutB* ou *BoutA*.

Alguns exemplos de classificação de vértices são ilustrados na Figura 3.5. Após a classificação dos vértices, as estruturas que contêm as referências para os vértices e as informações de classificação são armazenadas numa nova lista, comum a todos os vértices de ambos os grupos. Esta lista será chamada de *lvtx*.



Vértice	Classificação
v1	AoutB
v2	AoutB
v3	AoutB
v4	AoutB
v5	AoutB
v6	AoutB
v7	AoutB
v8	AoutB
v9	BoutA
v10	BoutA
v11	BoutA
v12	BoutA
v13	INTERS
v14	INTERS
v15	INTERS
v16	INTERS
v17	INTERS
v18	INTERS
v19	INTERS
v20	INTERS

Figura 3.5 – Exemplo de classificação de vértices.

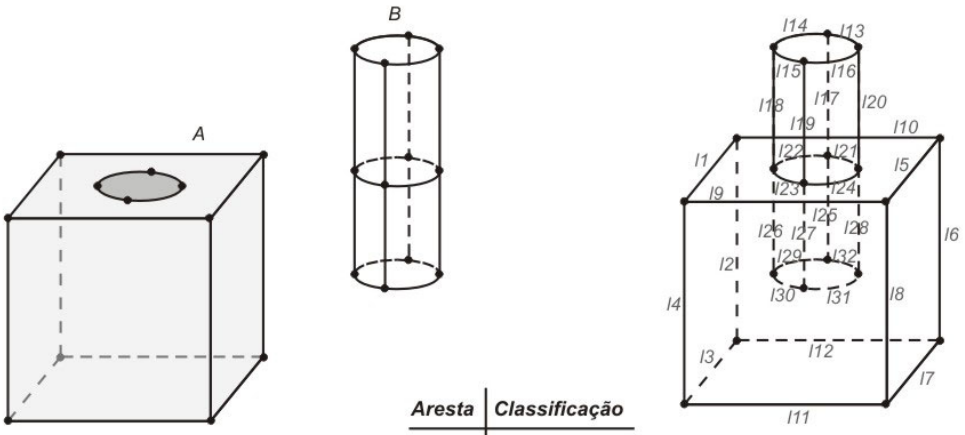
4) Classificação das arestas

Nesta etapa, todas as arestas de ambos os grupos devem ser classificadas. Isto inclui as arestas pertencentes a *loops* internos ou externos de faces, arestas pendentes (*strut edges*) e arestas soltas que constituem por si só cascas no interior ou no exterior de regiões. Todas estas arestas já estão devidamente armazenadas nas listas de arestas de cada grupo mencionadas na etapa número 1.

É criado um registro para cada aresta, que contém uma referência para a aresta e uma informação sobre a localização da aresta no espaço em relação às entidades do outro grupo. Uma aresta é classificada através da classificação dos vértices que a delimitam. O critério de classificação é o seguinte:

- Se uma aresta pertence à lista *lcommon*, ela é classificada como *INTERS*.
- Se um dos vértices de uma aresta de um grupo localiza-se no interior de uma região qualquer do outro grupo, esta aresta é classificada como *AinB* ou *BinA*.
- Se um dos vértices de uma aresta de um grupo localiza-se no exterior de todas as regiões do outro grupo e não se localiza sobre nenhuma superfície do outro grupo, a aresta é classificada como *AoutB* ou *BoutA*.
- Se ambos os vértices de uma aresta de um grupo pertencem à lista *lcommon* ou localizam-se sobre faces do outro grupo, um ponto qualquer no interior da aresta deve ser testado contra as regiões do outro grupo para se determinar a classificação da aresta. Se este ponto for interior, a aresta é classificada como *AinB* ou *BinA*. Se for exterior, a aresta é classificada como *AoutB* ou *BoutA*.

Alguns exemplos de classificação de arestas são mostrados na Figura 3.6. Após a classificação das arestas, os registros que contêm as referências para as arestas e as informações de classificação são armazenados numa nova lista, comum a todas as arestas de ambos os grupos. Esta lista será chamada de *ledg*.



Aresta	Classificação
I1	AoutB
I2	AoutB
I3	AoutB
I4	AoutB
I5	AoutB
I6	AoutB
I7	AoutB
I8	AoutB
I9	AoutB
I10	AoutB
I11	AoutB
I12	AoutB
I13	BoutA
I14	BoutA
I15	BoutA
I16	BoutA
I17	BoutA
I18	BoutA
I19	BoutA
I20	BoutA
I21	INTERS
I22	INTERS
I23	INTERS
I24	INTERS
I25	BinA
I26	BinA
I27	BinA
I28	BinA
I29	BinA
I30	BinA
I31	BinA
I32	BinA

Figura 3.6 – Exemplo de classificação de arestas.

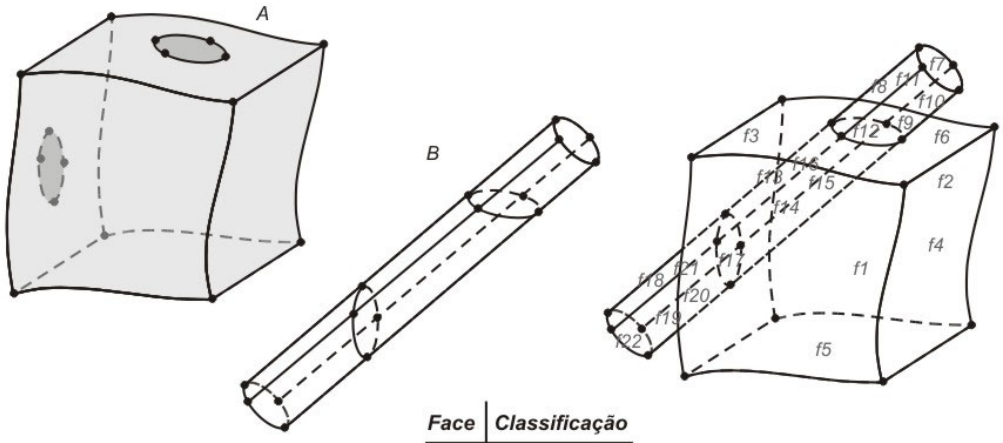
5) Classificação das faces

Nesta etapa, todas as faces de ambos os grupos devem ser classificadas. Isto inclui as faces que compõem a fronteira de regiões, as faces pendentes internamente ou externamente a alguma região e as faces soltas internamente ou externamente às regiões existentes, constituindo cascas individuais. Todas estas faces já estão devidamente armazenadas nas listas de faces de cada grupo mencionadas na etapa número 1.

É criado um registro para cada face, que contém uma referência para a face e uma informação sobre a localização da face no espaço em relação às entidades do outro grupo. A classificação das faces é feita da seguinte forma:

- Se a face pertencer à lista *lcommon*, ela é classificada como *INTERS*.
- Se ela não for comum aos grupos, a lista de vértices do *loop externo* desta face é percorrida, verificando-se a classificação de cada vértice na estrutura que contém a sua referência na lista *lvtx*. Se um vértice qualquer for classificado como *AinB*, *BinA*, *AoutB* ou *BoutA* então a face é automaticamente classificada da mesma forma.
- Se todos os vértices do *loop externo* da face estiverem sobre uma ou mais faces do outro grupo, um ponto qualquer no interior da face deve ser testado contra as regiões do outro grupo para se determinar a classificação da face. Se este ponto for interior, a face é classificada como *AinB* ou *BinA*. Se for exterior, a face é classificada como *AoutB* ou *BoutA*.

Alguns exemplos de classificação de faces são mostrados na Figura 3.7. Após a classificação das faces, os registros que contêm as referências para as faces e as informações de classificação são armazenados numa nova lista, comum a todas as faces de ambos os grupos. Esta lista será chamada de *lface*.



Face	Classificação
f1	AoutB
f2	AoutB
f3	AoutB
f4	AoutB
f5	AoutB
f6	AoutB
f7	BoutA
f8	BoutA
f9	BoutA
f10	BoutA
f11	BoutA
f12	AinB
f13	BinA
f14	BinA
f15	BinA
f16	BinA
f17	AinB
f18	BoutA
f19	BoutA
f20	BoutA
f21	BoutA
f22	BoutA

Figura 3.7 – Exemplo de classificação de faces.

3.5.3. Aplicação das Operações Booleanas sobre os Grupos

A base para a determinação das entidades topológicas resultantes da aplicação de uma operação booleana sobre os grupos é a classificação das entidades da maneira como foi descrita na seção anterior. De acordo com a operação booleana que for selecionada, vértices, arestas e faces de cada grupo serão removidos ou permanecerão como parte integrante do resultado dependendo da sua localização no espaço tridimensional em relação às entidades do outro grupo.

É importante ressaltar que o algoritmo proposto por Mäntylä [10] trata apenas dos casos em que duas *regiões* são combinadas por meio das operações booleanas. Desta forma, o algoritmo preocupa-se somente com as *faces* que farão parte do resultado ou que serão removidas. Assim, quando uma

face é removida, automaticamente todas as arestas da sua fronteira são também removidas, a não ser aquelas que pertencem à fronteira de uma outra face que não foi removida. Da mesma forma, quando uma face é removida, todos os vértices da sua fronteira também são automaticamente removidos, a não ser aqueles que pertencem à fronteira de uma ou mais faces que não foram removidas. No presente algoritmo, este raciocínio não pode ser utilizado, pois o domínio do problema inclui entidades de qualquer dimensão, e estas entidades podem nem fazer parte da fronteira de outras hierarquicamente superiores. Cada entidade deve ser analisada separadamente, para se determinar se a mesma irá ou não fazer parte do resultado da operação booleana. Alguns casos são ilustrados na Figura 3.8.

Para que a ordem de remoção das entidades não participantes do resultado da operação booleana seja compatível com os níveis de hierarquia presentes nas estruturas de dados topológicas dos modeladores em geral, primeiramente são removidas as faces indesejadas, em seguida as arestas e por último os vértices.

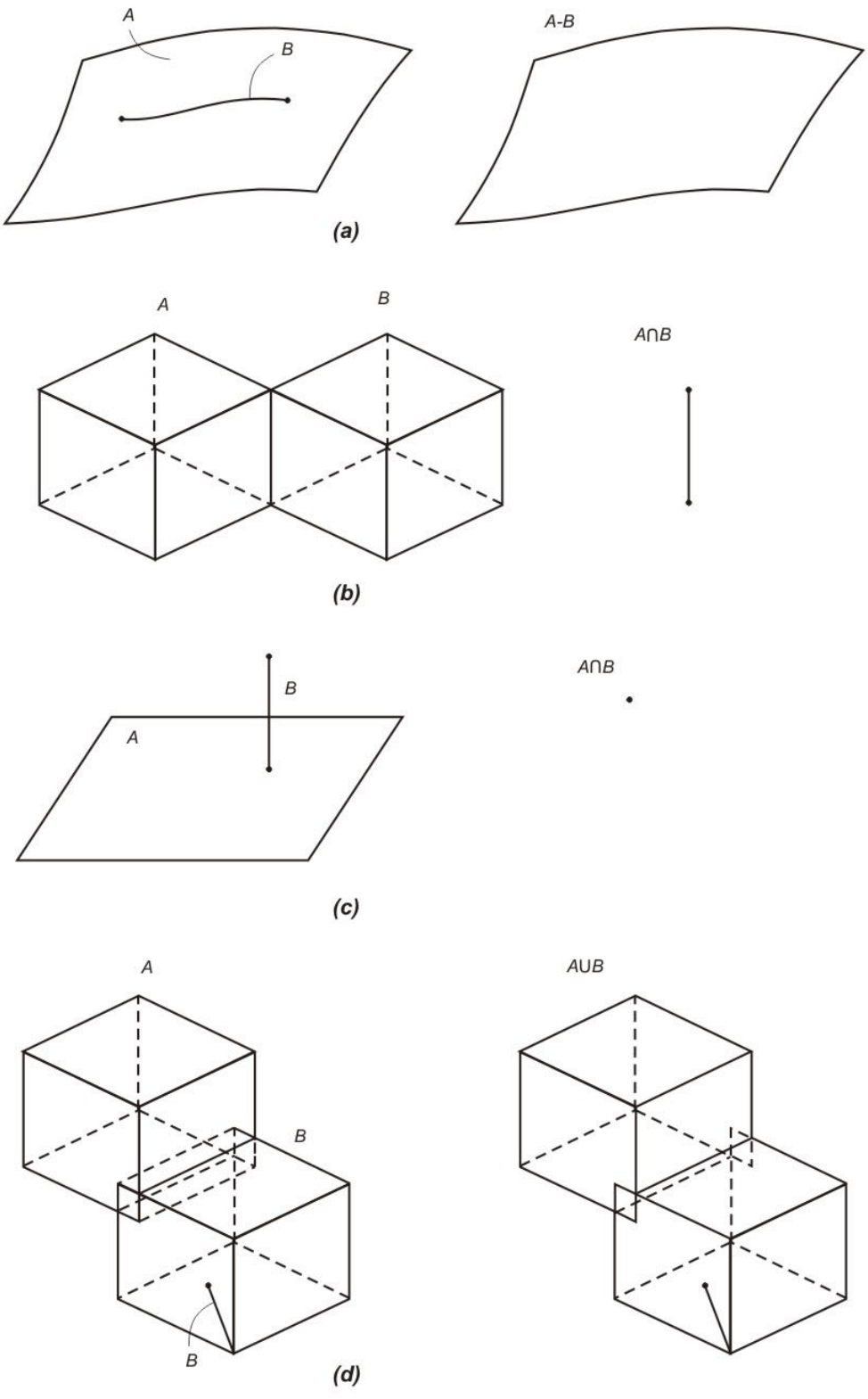


Figura 3.8 – Operações booleanas entre grupos de entidades.

3.5.3.1. Operação Booleana de União

A *união* entre dois grupos *A* e *B* representa o conjunto de pontos no espaço tridimensional que pertencem ao grupo *A ou* ao grupo *B* (pontos que pertencem somente a *A*, pontos que pertencem somente a *B* e pontos que pertencem a *A* e a *B* simultaneamente).

Quando se trabalha apenas com sólidos em domínios *manifold*, diz-se que a união entre dois sólidos resulta em um sólido que ocupa todo o volume ocupado pelos sólidos operantes. No caso de domínios *non-manifold*, esta definição deve ser estendida, pois mesmo no caso em que apenas dois sólidos são considerados, se estes contiverem estruturas internas (representando por exemplo restrições no domínio para quando uma malha de elementos finitos for gerada para estes sólidos) estas devem ser consideradas na obtenção do resultado. Além disso, o algoritmo que está sendo proposto não requer que os grupos necessariamente contenham *regiões* (volumes fechados), podendo ser constituídos somente por faces, arestas e vértices.

A união entre dois grupos se divide em três partes:

1) Detecção das faces a serem removidas

Percorre-se a lista *lface*, e para cada face é feita a seguinte análise:

- Se a face for do tipo *AinB* ela é removida se fizer parte da fronteira de alguma região de *A*. Caso contrário, é mantida.
- Se a face for do tipo *BinA* ela é removida se fizer parte da fronteira de alguma região de *B*. Caso contrário, é mantida.
- Se a face for do tipo *AoutB* ela é mantida.
- Se a face for do tipo *BoutA* ela é mantida.
- Se a face for do tipo *INTERS*, caso ela pertença à fronteira de mais de uma região de qualquer um dos dois grupos, ela é removida. Caso contrário, se ela não pertencer à fronteira de nenhuma região de pelo menos um dos dois grupos, ela é mantida. Se ela pertencer à fronteira de exatamente uma região de cada grupo, devem-se analisar os vetores normais a esta face em relação às duas regiões. Se estes vetores normais possuírem orientações opostas, a face é removida. Se possuírem a mesma orientação, ela é mantida [10].

2) Detecção das arestas a serem removidas

Percorre-se a lista *ledg*, e para cada aresta é feita a seguinte análise:

- Se a aresta for do tipo *AinB*, ela é removida se fizer parte da fronteira de alguma região de *A*, ou se for uma aresta pendente ou solta no interior de alguma face de *A* que foi removida. Caso contrário, é mantida.
- Se a aresta for do tipo *BinA*, ela é removida se fizer parte da fronteira de alguma região de *B* ou se for uma aresta pendente ou solta no interior de alguma face de *B* que foi removida. Caso contrário, é mantida.
- Se a aresta for do tipo *AoutB* ou *BoutA*, ela é mantida.
- Se a aresta for do tipo *INTERS*, ela é mantida.

3) Detecção dos vértices a serem removidos

Percorre-se a lista *lvtx*, e para cada vértice é feita a seguinte análise:

- Se o vértice for do tipo *AinB*, ele é removido se fizer parte da fronteira de alguma região de *A* ou se for um vértice solto no interior de alguma face de *A* que foi removida.
- Se o vértice for do tipo *BinA*, ele é removido se fizer parte da fronteira de alguma região de *B* ou se for um vértice solto no interior de alguma face de *B* que foi removida.
- Se o vértice for do tipo *AoutB* ou *BoutA*, ele é mantido.
- Se o vértice for do tipo *INTERS*, ele é mantido.

Alguns exemplos da união entre grupos de entidades são mostrados na Figura 3.9.

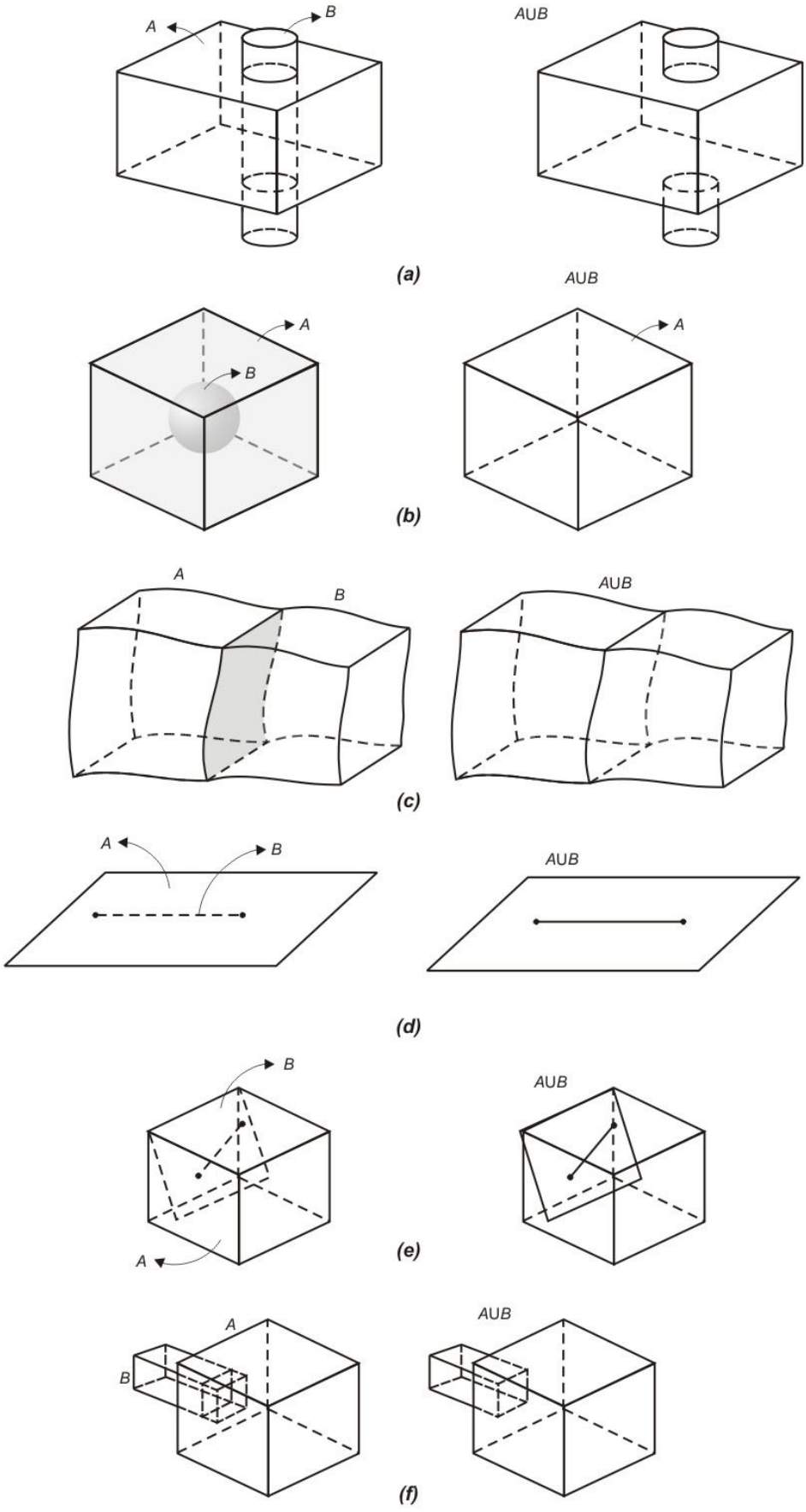


Figura 3.9 – Operação booleana de união.

3.5.3.2. Operação Booleana de Interseção

A *interseção* entre dois grupos *A* e *B* representa o conjunto de pontos no espaço tridimensional que pertencem ao grupo *A* e ao grupo *B* (pontos que pertencem a *A* e *B* simultaneamente).

Quando se trabalha apenas com sólidos em domínios *manifold*, diz-se que a interseção entre dois sólidos resulta em um sólido que ocupa o volume dos sólidos operantes que é comum a todos eles. No caso de domínios *non-manifold*, novamente esta definição deve ser estendida, seja devido à presença de estruturas internas às eventuais regiões presentes ou pelo fato de não haver necessidade de haver regiões nos grupos. A interseção entre dois grupos também se divide em três partes:

3) Detecção das faces a serem removidas

Percorre-se a lista *lface*, e para cada face é feita a seguinte análise:

- Se a face for do tipo *AoutB* ou *BoutA* ela é removida.
- Se a face for do tipo *AinB*, ela é removida se fizer parte da fronteira de mais de uma região do grupo *A*. Caso contrário, é mantida.
- Se a face for do tipo *BinA*, ela é removida se fizer parte da fronteira de mais de uma região do grupo *B*. Caso contrário, é mantida.
- Se a face for do tipo *INTERS*, ela é removida se fizer parte da fronteira de mais de uma região do grupo *A* e da fronteira de mais de uma região de grupo *B*.

2) Detecção das arestas a serem removidas

Percorre-se a lista *ledg*, e para cada aresta é feita a seguinte análise:

- Se a aresta for do tipo *AoutB* ou *BoutA*, ela é removida.
- Se a aresta for do tipo *AinB*, *BinA* ou *INTERS* ela é mantida.

3) Detecção dos vértices a serem removidos

Percorre-se a lista *lvtx*, e para cada vértice é feita a seguinte análise:

- Se o vértice for do tipo *AoutB* ou *BoutA*, ele é removido.

- Se o vértice for do tipo *AinB*, *BinA* ou *INTERS*, ele é mantido.

Alguns exemplos da interseção entre grupos de entidades são mostrados na Figura 3.10.

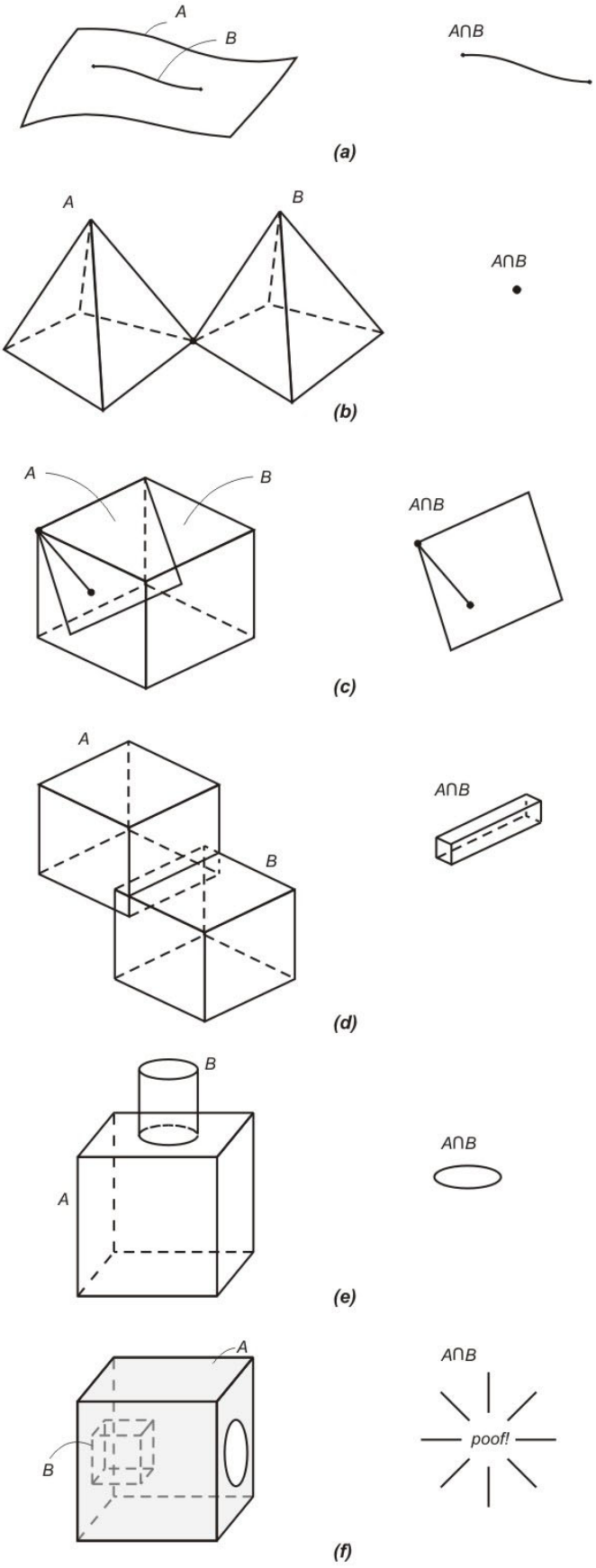


Figura 3.10 – Operação booleana de interseção.

3.5.3.3. Operação Booleana de Diferença

A *diferença* entre dois grupos *A* e *B* representa o conjunto de pontos no espaço tridimensional que pertencem somente ao grupo *A* (no caso da operação ser *A* – *B*) ou somente ao grupo *B* (no caso da operação ser *B* – *A*).

Quando se trabalha apenas com sólidos em domínios *manifold*, diz-se que a diferença entre dois sólidos resulta em um sólido que ocupa o volume de um dos sólidos operantes que os outros não ocupam. No caso de domínios *non-manifold*, esta definição também deve ser estendida, pelos mesmos motivos expostos anteriormente.

A diferença entre dois grupos também se divide em três partes (para simplificar o raciocínio, será considerada a diferença *A* – *B*):

3) Detecção das faces a serem removidas

Percorre-se a lista *lface*, e para cada face é feita a seguinte análise:

- Se a face for do tipo *AinB* ou *BoutA* ela é removida.
- Se a face for do tipo *AoutB* ela é mantida.
- Se a face for do tipo *BinA* ela é mantida temporariamente, a não ser que faça parte da fronteira de mais de uma região do grupo *B* ou não faça parte da fronteira de nenhuma região do grupo *B*. Nestes casos, ela é removida.
- Se a face for do tipo *INTERS*, caso ela pertença à fronteira de mais de uma região do grupo *B*, ela é removida. Se ela pertencer à fronteira de mais de uma região do grupo *A*, ela é mantida. Se ela pertencer à fronteira de uma região de *A* e de nenhuma região de *B* ela é mantida se for exterior a todas as regiões de *B*. Caso contrário, é removida. Se ela pertencer à fronteira de uma região de *B* e de nenhuma região de *A* ela é removida se for exterior a todas as regiões de *A*. Caso contrário, é mantida. Se ela não pertencer à fronteira de nenhuma região dos dois grupos, ela é removida. Se ela pertencer à fronteira de exatamente uma região de cada grupo, devem-se analisar os vetores normais a esta face em relação às duas regiões. Se estes vetores normais possuírem orientações opostas, a face é mantida. Se possuírem a mesma orientação, ela é removida [10].

Após a remoção das faces devidas, deve-se fazer o reconhecimento de regiões com as faces remanescentes. As faces do tipo *BinA* que não fizerem parte das fronteiras das regiões formadas, devem então ser removidas.

2) Detecção das arestas a serem removidas

Percorre-se a lista *ledg*, e para cada aresta é feita a seguinte análise:

- Se a aresta for do tipo *AinB* ou *BoutA* ela é removida.
- Se a aresta for do tipo *AoutB* ela é mantida.
- Se a aresta for do tipo *BinA* ou *INTERS*, caso ela não pertença à fronteira de nenhuma face, ela é removida. Se ela pertencer à fronteira de uma ou mais faces, e todas estas faces tiverem sido removidas, ela também será removida. Caso contrário, ela é mantida.

3) Detecção dos vértices a serem removidos

Percorre-se a lista *lvtx*, e para cada vértice é feita a seguinte análise:

- Se o vértice for do tipo *AinB* ou *BoutA* ele é removido.
- Se o vértice for do tipo *AoutB* ele é mantido.
- Se o vértice for do tipo *BinA* ou *INTERS*, caso ele não possua nenhuma aresta incidente, ele é removido. Se ele possuir uma ou mais arestas incidentes, e todas elas tiverem sido removidas, ele também será removido. Caso contrário, ele é mantido.

Alguns exemplos da diferença entre grupos de entidades são mostrados na Figura 3.11.

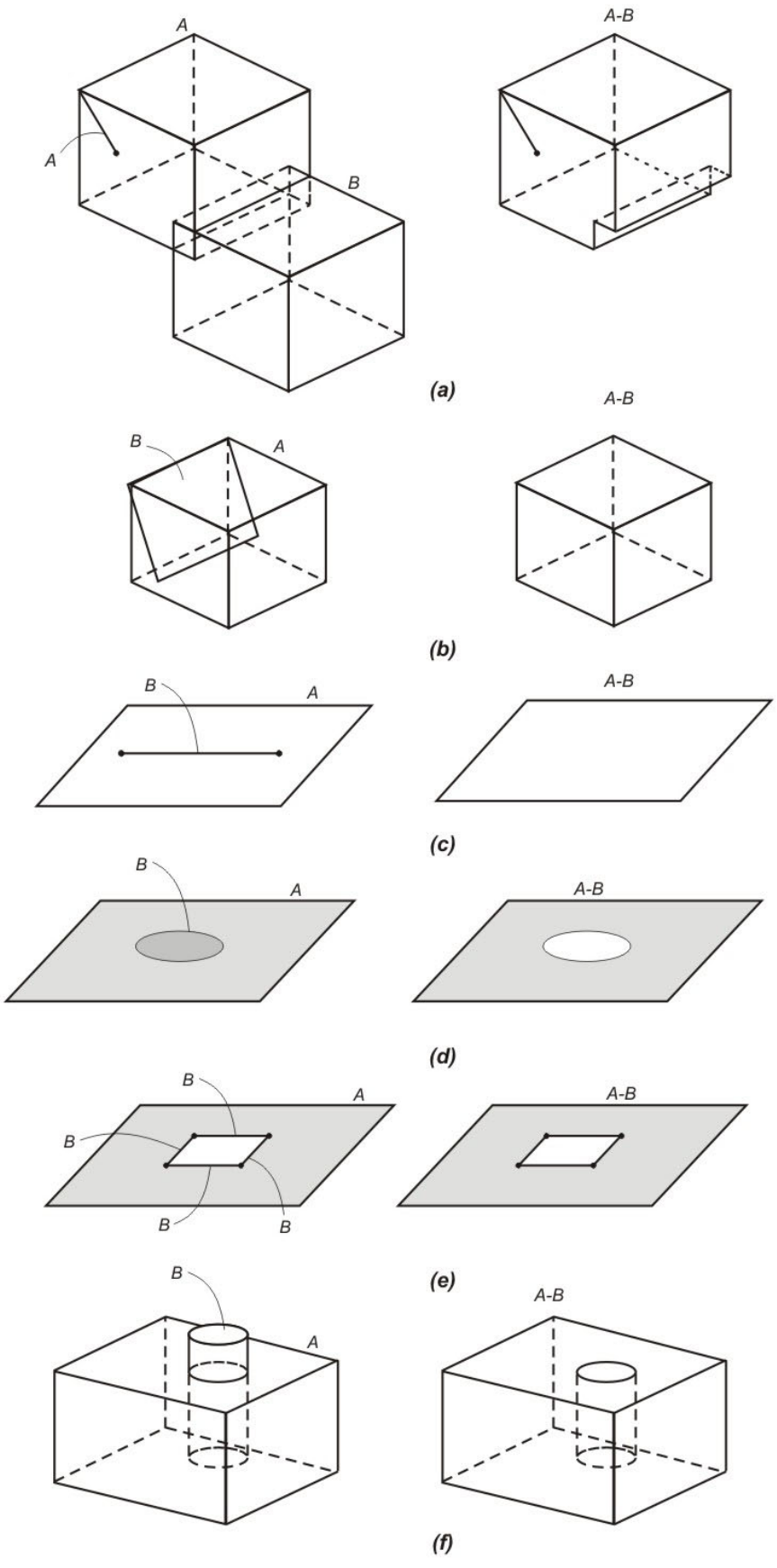


Figura 3.11 – Operação booleana de diferença.

3.5.3.4. Regularização do resultado

Como foi dito, em muitas situações, a combinação de dois ou mais sólidos a partir da aplicação de uma operação booleana pode ocasionar resultados que possuam faces ou arestas pendentes, ou ainda vértices, faces ou arestas soltos no espaço (internamente ou externamente a alguma região). É o caso de alguns exemplos vistos nas Figuras 3.9, 3.10 e 4.11. Contudo, em muitas aplicações deseja-se eliminar estas entidades de dimensões inferiores, de forma que o resultado final corresponda somente a volumes preenchíveis, ou seja, deseja-se *regularizar* o resultado [11]. Como já foi exposto, *regularizar* um conjunto *A* de pontos no espaço tridimensional significa considerar o *fecho* do *interior* de *A* ($R(A) = F(I(A))$). Pode-se dizer que a regularização do resultado de uma operação booleana é obtida considerando-se todos os pontos interiores (pontos que possuem alguma vizinhança contendo somente outros pontos pertencentes ao resultado da operação) e envolvendo-os por uma casca. Pode-se perceber então que no processo de regularização o conceito de lugar geométrico volta a ser o foco principal na determinação das entidades participantes do resultado.

Operações booleanas regularizadas são importantes em aplicações que desejam evitar um subconjunto dos resultados *non-manifold* que podem ser obtidos pela aplicação das operações booleanas em objetos *manifold*. São também importantes em processos onde apenas volumes fechados são relevantes, como por exemplo em processos industriais de montagem de peças. Nestes processos, entidades de dimensões inferiores como faces ou arestas não são fisicamente representáveis, por possuírem espessura nula.

O algoritmo de regularização é bem simples, pois a idéia central é remover todas as entidades de dimensão inferior que não pertencem à fronteira das regiões resultantes da aplicação das operações booleanas. Para isto, deve-se primeiramente fazer o reconhecimento das regiões formadas com as entidades resultantes destas operações. As faces que deverão ser removidas são aquelas que não pertencem à fronteira das regiões formadas. As arestas e vértices que não fizerem parte da fronteira das faces remanescentes deverão ser igualmente removidos. O procedimento de regularização pode então ser descrito da seguinte forma:

- Com as entidades resultantes da aplicação de uma operação booleana, faz-se o reconhecimento de regiões. Isto significa ter acesso a listas de faces pertencentes à fronteira de cada região formada.

- As faces que não fizerem parte de nenhuma destas listas devem ser removidas, pois representam faces soltas ou pendentes interiormente ou exteriormente às regiões formadas.
- As arestas que não fizerem parte da fronteira de alguma das faces remanescentes devem ser removidas.
- Os vértices que não fizerem parte da fronteira de alguma das faces remanescentes devem ser removidos.

O processo de regularização, conseqüentemente, elimina as estruturas internas eventualmente remanescentes nos sólidos resultantes, o que significa que neste processo restrições ao domínio representadas por estruturas de mais baixa dimensão são desconsideradas, já que apenas *volumes* são importantes. A Figura 3.12 mostra exemplos de regularização dos resultados de operações booleanas.

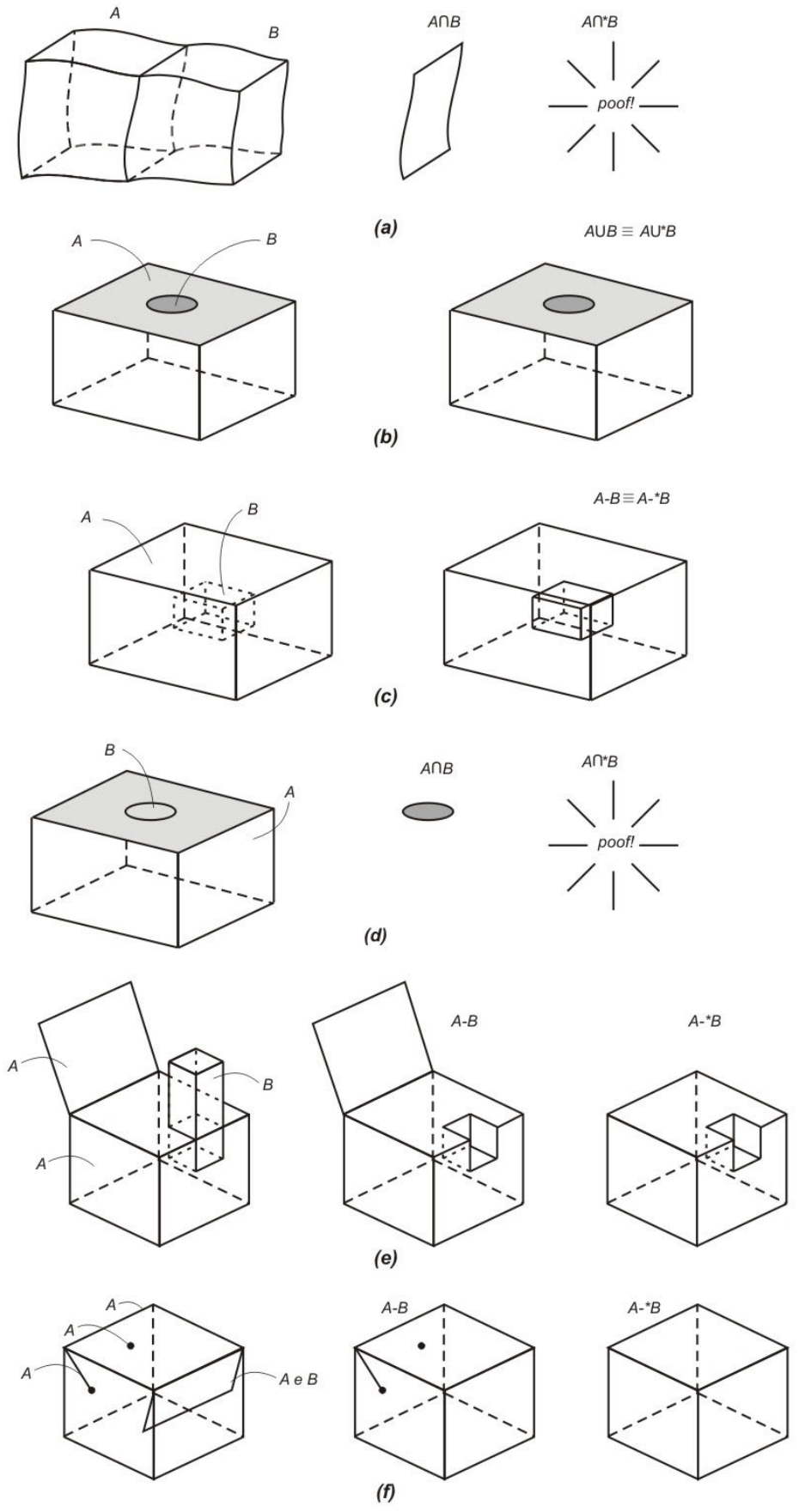


Figura 3.12 – Exemplos de regularização do resultado.

3.6. Considerações sobre o algoritmo

De uma forma geral, o algoritmo proposto pode ser caracterizado como um algoritmo simples, já que em sua essência utiliza entidades topológicas e as suas relações de adjacência para descrever os procedimentos necessários para a execução das operações booleanas regularizadas ou não em ambientes de modelagem com representação B-Rep. Os algoritmos geométricos utilizados são apenas o de detecção de ponto em volume e detecção de ponto em superfície, que são algoritmos que, apesar de não serem triviais, podem ser encontrados em diversos textos sobre modelagem geométrica [45,46], ou mesmo já implementados em bibliotecas ou aplicativos para modelagem geométrica [19].

Vale ressaltar que as ferramentas de interseção entre superfícies e curvas e de reconhecimento automático de multi-regiões são essenciais para que o algoritmo possa ser implementado num modelador qualquer obedecendo às restrições impostas nas suas condições de aplicabilidade. Para que os parâmetros de entrada estejam devidamente representados, é necessário que todas as interseções entre superfícies, entre superfícies e curvas e entre curvas tenham sido previamente calculadas. Além disso, é necessário que as eventuais regiões existentes já tenham sido reconhecidas. Ao final do algoritmo, é necessário que as novas regiões sejam reconhecidas, sobretudo se o processo de regularização é requerido, para que as entidades de dimensão inferior não pertencentes às fronteiras das regiões formadas sejam removidas.

É importante notar que estruturas internas às faces ou regiões dos objetos operantes são consideradas de maneira formal, como restrições impostas ao domínio do problema ou como se os objetos fossem heterogêneos (constituídos por diversos materiais). Assim, *loops* formados apenas por um arame, por um conjunto aberto de arestas ou apenas por um vértice e cascas formadas por faces soltas, arestas soltas, conjuntos abertos de arestas ou faces, ou ainda por vértices soltos, que são estruturas passíveis de estarem presentes em modelos *non-manifold*, recebem o tratamento devido.

Um problema que pode surgir na implementação do algoritmo é o uso de estruturas de dados que considerem estruturas internas a uma face ou a uma região como parte da fronteira destas entidades. Este é o caso da *Radial Edge* proposta por Weiler [17]. Para que o algoritmo possa ser aplicado de forma correta, é necessário que as estruturas internas pendentes ou soltas possam ser identificadas, para que se possa efetuar o tratamento devido. Uma solução é

armazenar estas estruturas em listas auxiliares para que se possa diferenciá-las das entidades topológicas que efetivamente fazem parte da fronteira da face ou região, segundo o conceito de fronteira apresentado neste capítulo.

4
As Operações Booleanas no MG

Este capítulo visa mostrar a adaptação do algoritmo apresentado para as operações booleanas em domínios *non-manifold* ao ambiente de modelagem do modelador MG. Descrevem-se aqui todos os passos realizados para que o MG passasse a incorporar mais esta facilidade de modelagem, tornando-se um sistema mais completo e eficiente na geração de modelos para análise pelo MEF. São apresentadas as novas classes criadas, seus métodos e sua integração com o resto do código do programa.

Ressalvam-se os seguintes pontos:

- Importância do uso da POO na inserção de uma nova ferramenta de modelagem.
- Pré-processamento do modelo através do uso dos algoritmos de interseção de curvas e superfícies e de detecção de regiões.
- Utilização de facilidades pré-existentes no MG que tornaram a implementação destas operações uma tarefa bem menos árdua.
- Modificações realizadas em algumas funções presentes no código fonte do modelador e da biblioteca CGC para permitir a correta implementação das operações booleanas.
- Criação de uma nova classe responsável pela interface entre a biblioteca CGC e as operações booleanas no MG.
- Detalhes de implementação e artifícios usados para tratamento de casos particulares.
- Limitações do modelador.
- Tempo de execução do algoritmo e robustez e eficiência do mesmo.

4.1.
Modelagem geométrica usando POO

Um sistema de modelagem que se propõe a representar modelos tridimensionais complexos deve priorizar a organização do seu código fonte. Um programador não familiarizado com o ambiente de modelagem que este sistema representa deve ser capaz de compreender seu código fonte com facilidade.

Além disso, o código fonte deve estar implementado de modo que a inserção de uma nova ferramenta de modelagem seja uma tarefa praticamente isolada do resto das facilidades pré-existentes. Esta *modularização* do código fonte permite que se promova grandes modificações em partes do programa sem haver a necessidade de se reimplementar outras partes que não guardam uma relação estreita com aquelas que foram modificadas.

Neste contexto, o uso da POO se faz de grande utilidade, pois permite uma organização de classes na estrutura do modelador com base em graus de hierarquia e níveis de abstração. Um programa desenvolvido numa linguagem orientada a objetos torna-se mais simples de ser interpretado, estendido e modificado. Além disso, o uso de aplicativos que geram informações compactas, organizadas e didáticas a partir do código fonte de um programa, como por exemplo o Doxygen [47], passa a ser extremamente útil quando se trata de um programa implementado usando POO.

No desenvolvimento deste trabalho, a importância da modularização do código fonte tornou-se evidente. As operações booleanas puderam ser implementadas no MG sem que praticamente nenhuma parte do código fonte já existente precisasse ser modificada. As poucas modificações que foram feitas não representaram nenhuma mudança significativa no ambiente de modelagem, e a estrutura de dados do MG permaneceu intacta. Em linhas gerais, o que se fez foi criar novas classes que pudessem representar os novos conceitos de modelagem que estavam sendo introduzidos, como os grupos e as operações booleanas, e classes que gerenciassem a interface entre estas e o modelador MG ou a biblioteca CGC. Como no MG cada entidade topológica é representada por uma classe, como foi visto no Capítulo 2, a manipulação destas entidades pelas operações booleanas pôde ser feita sem muita dificuldade.

4.2. Descrição das novas classes criadas no MG

Para que o algoritmo de operações booleanas pudesse ser implementado no ambiente de modelagem do MG, novas classes foram criadas no intuito de permitir que os objetos das classes que representam as informações topológicas do modelo pudessem ser manipulados da forma como foi descrito no último capítulo. Todas as classes criadas derivam diretamente de uma mesma classe base já existente, *Application*, como pode ser visualizado na Figura 4.1.

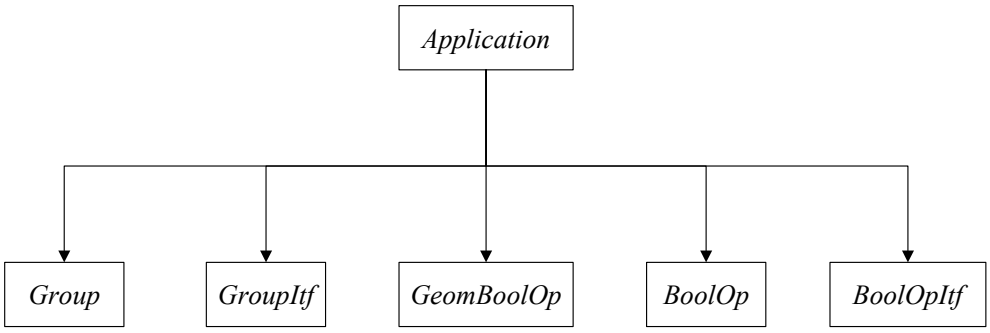


Figura 4.1 – Novas classes criadas no MG.

4.2.1. A classe *Group*

O primeiro novo conceito que precisava ser introduzido era o de *grupo*. No MG, a classe que representa grupos de entidades topológicas não foi criada exclusivamente para que as operações booleanas pudessem ser introduzidas. Outras funcionalidades presentes no modelador também puderam tirar proveito deste novo conceito.

Um objeto da classe *Group* possui simplesmente um identificador do grupo e uma lista de entidades topológicas associadas àquele grupo (Figura 4.2). Os métodos desta classe são responsáveis por operações simples de consulta e manipulação das entidades topológicas pertencentes a um grupo, como a inserção e a remoção de uma entidade, verificação da presença de uma determinada entidade num grupo, consulta ao identificador do grupo e ao número de entidades pertencentes ao mesmo.

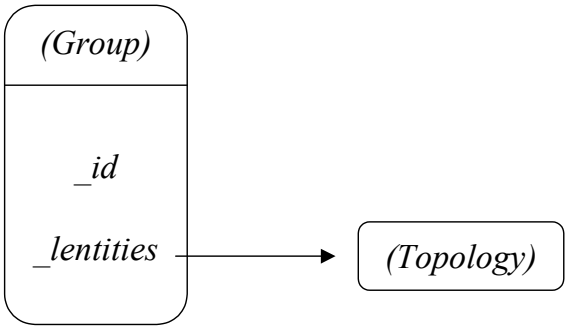


Figura 4.2 – Relações dos objetos da classe *Group*.

4.2.2. A classe *GroupItf*

Para que os grupos pudessem ser manipulados pelo usuário através da interface do MG de forma simples, amigável e eficiente, foi criada uma classe

que gerencia as informações dos grupos e a maneira como elas são representadas e manipuladas na interface do MG. Além disso, a classe *GroupItf* promove a comunicação entre a classe *Group* e a estrutura de dados do modelador.

Um objeto da classe *GroupItf* possui um ponteiro para um elemento de interface que é representado por uma árvore [48]. Esta árvore contém um nó raiz denominado *Groups*, cujos filhos são nós representando os grupos associados ao modelo que está sendo representado. As folhas desta árvore são os nós derivados diretamente dos grupos e representam as entidades topológicas pertencentes a cada grupo (Figura 4.3). Um objeto desta classe possui também uma lista de grupos existentes no modelo e uma outra lista cujos nós são registros contendo uma referência para um grupo e uma referência para uma entidade topológica (Figura 4.4). Este tipo de registro foi criado para auxiliar no gerenciamento das informações de cada grupo contido na lista de grupos.

A classe *GroupItf* possui também dois campos (variáveis) *estáticos*. Um campo *estático* pertencente a uma classe é aquele que é compartilhado por todos os objetos desta classe. Ou seja, existe uma única área de memória que armazena as informações daquele campo para todos os objetos daquela classe. No caso da classe *GroupItf*, um destes campos, denominado *curr_nod*, é um número inteiro representando o nó da árvore que está selecionado na interface (a seleção de um nó é feita pelo usuário através do *mouse*). Apenas um nó pode ser selecionado de cada vez. O outro campo estático da classe *GroupItf* representa um ponteiro para um objeto da própria classe *GroupItf* que sempre apontará para o último objeto desta classe que foi criado. Isto é feito da seguinte forma: na implementação de um *construtor* desta classe (*construtores* são funções automaticamente chamadas quando um objeto de uma classe é criado) faz-se com que este campo estático aponte para o próprio objeto que está sendo criado. Este artifício é utilizado em linguagens orientadas a objetos como uma forma de permitir que funções de *callback* de elementos de interface sejam definidas como métodos privados de alguma classe (estes métodos também devem ser estáticos) [49,50].

O construtor da classe *GroupItf* também registra as *callbacks* de interface, relativas à criação, seleção e remoção de grupos, inclusão de entidades selecionadas na área de *canvas*, remoção de entidades e rotulação (*labeling*) dos grupos criados. As Figuras 2.22 e 2.23 mostram algumas destas funcionalidades na interface do MG.

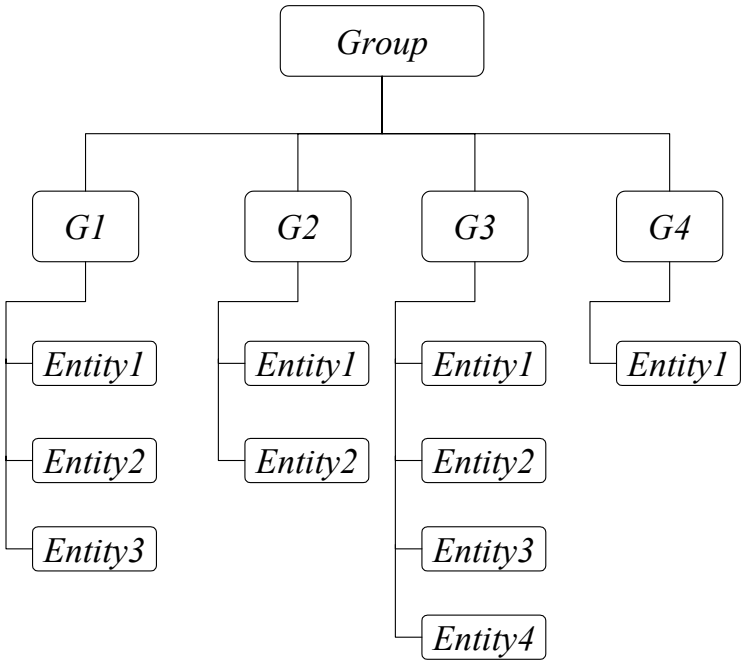


Figura 4.3 – Exemplo de organização da árvore da classe *GroupItf*.

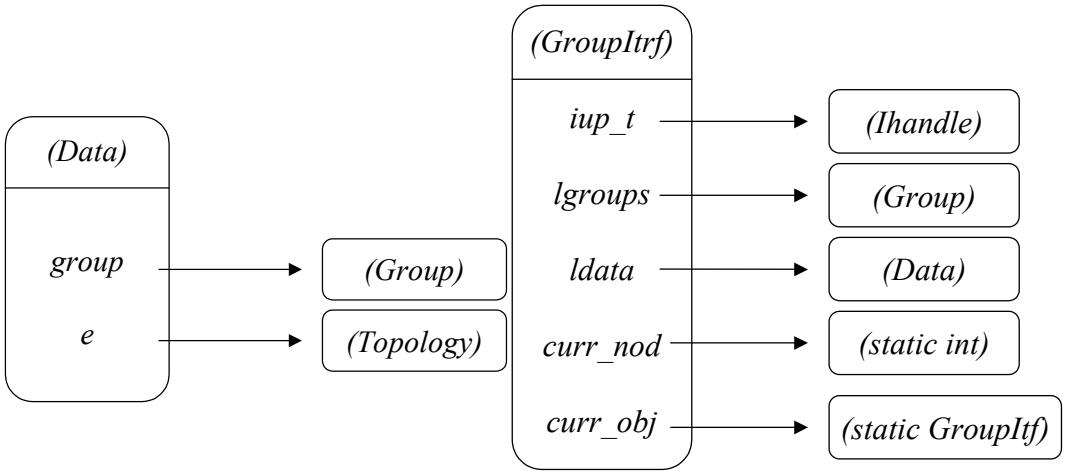


Figura 4.4 – Relações dos objetos da classe *GroupItf*.

4.2.3. A classe *GeomBoolOp*

A classe *GeomBoolOp* está diretamente relacionada com a implementação das operações booleanas no MG. Ela é responsável pelos algoritmos geométricos necessários para que as operações booleanas possam ser realizadas de forma correta.

Um objeto desta classe possui somente um ponteiro para uma lista de listas de referências para objetos da classe *Topology* (Figura 4.5). Os métodos públicos da classe *GeomBoolOp* são:

- Detecção de ponto em região.
- Detecção de ponto sobre retalho de superfície (face).
- Detecção de ponto sobre aresta.
- Detecção de ponto sobre vértice.
- Determinação das coordenadas tridimensionais de um ponto sobre um retalho de superfície a partir das suas coordenadas paramétricas sobre o retalho de superfície.
- Determinação das coordenadas tridimensionais de um ponto sobre uma aresta a partir da sua coordenada paramétrica sobre a aresta.
- Detecção de eventuais regiões formadas a partir de um conjunto de retalhos de superfícies.
- Consulta ao número de regiões formadas a partir de um conjunto de retalhos de superfícies.
- Determinação dos subconjuntos de superfícies que compõem a fronteira de cada região formada a partir de um conjunto de retalhos de superfícies.

Os métodos de detecção de ponto em região e de detecção de eventuais regiões formadas a partir de um conjunto de retalhos de superfície são os únicos que necessitam de uma comunicação entre o modelador MG e a biblioteca CGC, pois estes métodos já se encontram disponíveis nesta biblioteca. Na verdade, na versão original da CGC, apenas retalhos de superfícies poligonais eram considerados. Contudo, durante o seu trabalho, Lira [6] estendeu esta biblioteca de forma que passasse a considerar retalhos de superfície com geometria qualquer (baseada na representação NURBS).

O método de detecção de ponto sobre vértice é implementado de forma bem simples, usando a descrição geométrica do vértice topológico (ponto no espaço tridimensional associado a este vértice) e analisando-se a distância entre os dois pontos.

Os métodos de detecção de ponto sobre aresta e sobre retalho de superfície (face) também são implementados de forma bem simples, pois as classes que representam arestas e retalhos de superfície já disponibilizam

métodos que comparam a distância entre um ponto qualquer e o ponto mais próximo a ele localizado sobre a aresta ou retalho de superfície.

Os métodos de detecção de ponto em região e de detecção de ponto sobre retalho de superfície, aresta ou vértice recebem como parâmetro adicional o valor de uma tolerância, que é utilizada na comparação das distâncias entre pontos. Isto significa que este valor pode ser determinado no momento em que um destes métodos for requisitado, baseando-se em critérios relacionados ao modelo ou ao tipo de operação que está sendo realizada. Métodos baseados na geometria do modelo estão sempre sujeitos a eventuais erros numéricos provocados pelo uso de tolerâncias. A determinação de uma tolerância satisfatória para cada aplicação, modelo ou operação de interface é uma área de estudo extremamente importante em modelagem geométrica. Trabalhos relacionados ao uso de tolerâncias adaptativas aos modelos e ao uso de tolerâncias mínimas suportadas por cada máquina onde as aplicações são executadas podem ser encontrados em [51].

Os métodos de determinação das coordenadas tridimensionais de um ponto sobre uma aresta ou retalho de superfície a partir das coordenadas paramétricas do ponto sobre estas entidades também são muito simples, pois os objetos das classes que representam arestas e retalhos de superfície no MG já incorporam métodos para realizar esta tarefa.

Os objetos da classe *GeomBoolOp* possuem um campo *privado* (campo acessível somente através de métodos públicos disponibilizados pela classe) que é uma lista encadeada de listas encadeadas de entidades topológicas. Esta lista guarda as listas de retalhos de superfície que formam a fronteira de cada região detectada pelo método de detecção de regiões a partir de um conjunto de retalhos de superfície. Logo, o método de consulta ao número de regiões formadas apenas retorna o número de elementos desta lista e o método de determinação dos subconjuntos de superfícies que formam a fronteira de cada região formada apenas retorna cada uma das listas de retalhos de superfície contida nesta lista.

Vale ressaltar que quando a CGC detecta quais são os retalhos de superfície pertencentes à fronteira de cada região formada e retorna esta informação para o MG, este não *cria* efetivamente as regiões (*patches3d*) no modelo. Este é o motivo pelo qual o campo privado citado no parágrafo anterior não foi implementado como uma lista de regiões, mas como uma lista de listas de retalhos de superfície que formam a fronteira de cada região. Este método não tem como objetivo alterar as características do modelo, mas apenas

determinar quais são os retalhos de superfície que contribuem para a formação de regiões, a partir de um conjunto de retalhos de superfície presentes no modelo. Isto porque o método consiste apenas em uma etapa intermediária na aplicação das operações booleanas em um grupo de entidades topológicas.

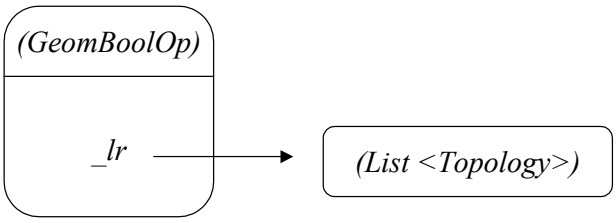


Figura 4.5 – Relações dos objetos da classe *GeomBoolOp*.

4.2.4. A classe *BoolOp*

A classe *BoolOp* é diretamente responsável pelas operações booleanas no MG. Esta é a classe cujos campos e métodos representam as variáveis e procedimentos descritos no algoritmo proposto no capítulo anterior, adaptados ao ambiente de modelagem do MG.

Na implementação desta classe, foi criado um registro auxiliar do tipo enumerável denominado *Type* representando as possíveis posições relativas no espaço de uma entidade topológica de um grupo perante as entidades topológicas do outro grupo: *AoutB*, *BoutA*, *AinB*, *BinA* e *INTERS*.

Além disso, três classes auxiliares também foram criadas: *BoolVtx*, *BoolSegment* e *BoolPatch2d*. Cada uma destas classes contém dois campos privados: um deles é um ponteiro para um objeto do tipo vértice, aresta (segmento) ou face (retalho de superfície ou *patch2d*), respectivamente. O outro é um campo do tipo *Type*, que guarda a informação sobre o grupo que contém aquele elemento topológico e a sua posição relativa (exterior, interior ou compartilhado pelos dois grupos) em relação às entidades do outro grupo.

Um objeto da classe *BoolOp* possui os seguintes campos (Figura 4.6):

- Dois ponteiros para objetos da classe *Group*, que representam os grupos sobre os quais as operações booleanas serão aplicadas (*A* e *B*).
- Duas listas encadeadas de ponteiros para vértices, que armazenam referências para os vértices de cada grupo (*lvtxA* e *lvtxB*).

- Duas listas encadeadas de ponteiros para arestas, que armazenam referências para as arestas de cada grupo (*ledgA* e *ledgB*).
- Duas listas encadeadas de ponteiros para faces, que armazenam referências para as faces de cada grupo (*lfacA* e *lfacB*).
- Duas listas encadeadas de ponteiros para regiões, que armazenam referências para as regiões de cada grupo (*lregA* e *lregB*).
- Uma lista encadeada de objetos da classe *BoolVtx*, contendo referências para todos os vértices dos dois grupos e as suas posições relativas no espaço em relação às entidades do outro grupo (*lallvtx*).
- Uma lista encadeada de objetos da classe *BoolSegment*, contendo referências para todas as arestas dos dois grupos e as suas posições relativas no espaço em relação às entidades do outro grupo (*lallseg*).
- Uma lista encadeada de objetos da classe *BoolPatch2d*, contendo referências para todas as faces dos dois grupos e as suas posições relativas no espaço em relação às entidades do outro grupo (*lallp2d*).
- Duas listas encadeadas de entidades topológicas genéricas (objetos da classe *Topology*), que armazenam referências para as entidades topológicas comuns aos dois grupos e para as entidades topológicas que deverão ser removidas do modelo após a aplicação das operações booleanas sobre os grupos (*lcommon* e *ldelent*).

Alguns detalhes de implementação relativos aos métodos privados que manipulam estas informações serão apresentados posteriormente. Os métodos públicos disponibilizados pela classe *BoolOp* são:

- Método *addgroups*, que recebe como parâmetros referências para os dois grupos e armazena estas referências em *A* e *B*. Este método se encarrega da chamada de métodos privados responsáveis pelo armazenamento das informações provenientes dos grupos nas respectivas listas, e pela classificação das entidades topológicas de um grupo em relação às entidades do outro grupo.
- Métodos *boolUnion*, *boolInters* e *boolDiff*, que são responsáveis pela aplicação das operações booleanas de união, interseção e diferença sobre os grupos *A* e *B*.
- Método *getDelList*, que retorna a lista de entidades topológicas a serem removidas após a aplicação das operações booleanas.

A classe *BoolOp* também disponibiliza um construtor que recebe referências para dois grupos como parâmetros. Este construtor chama o método *addgroups* descrito acima.

Ressalta-se que o método *BoolDiff* executa a operação booleana de diferença entre dois grupos de entidades na ordem em que os grupos foram passados como parâmetros para a função *addgroups*, ou seja, este método faz a operação *A – B*.

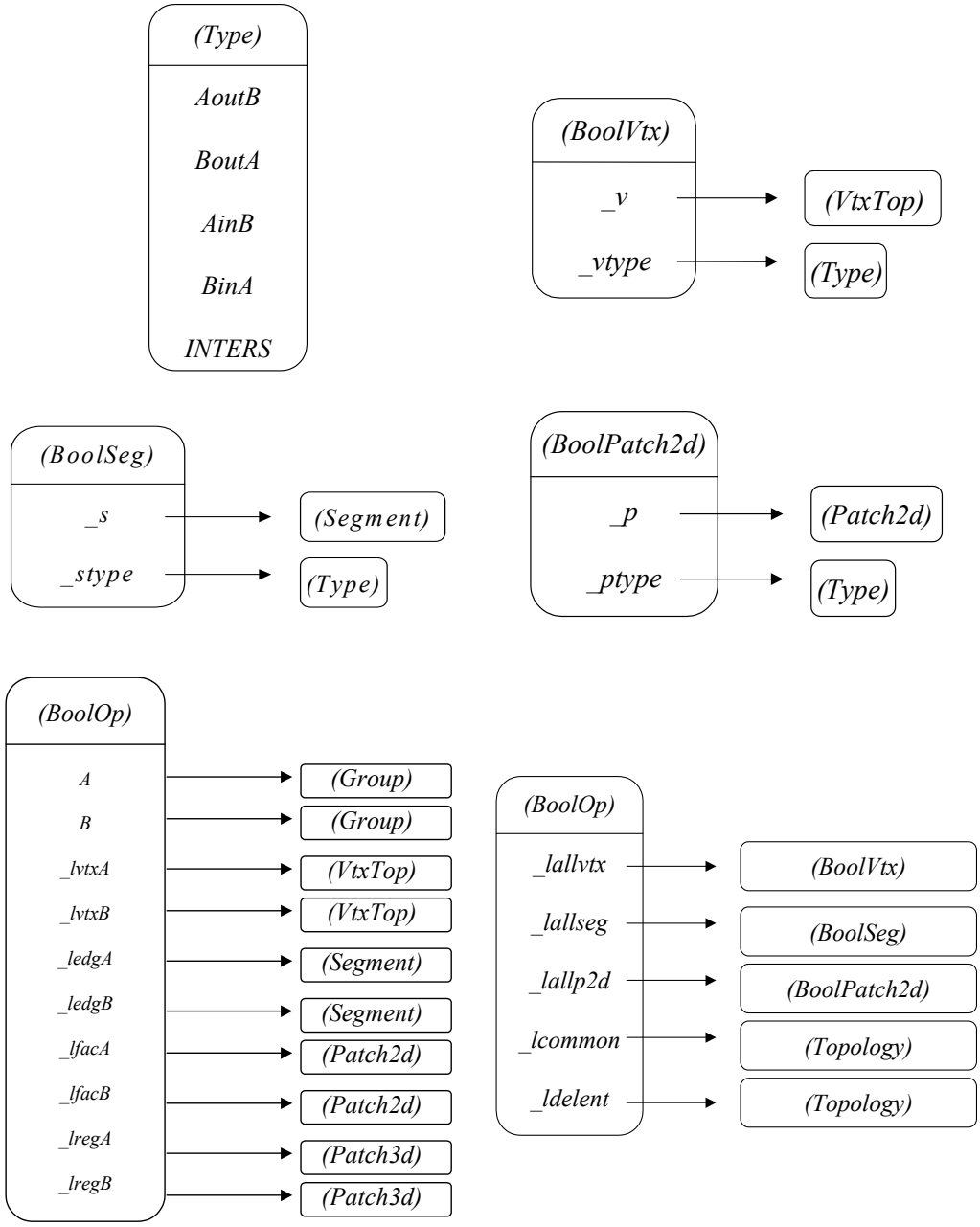


Figura 4.6 – Relações dos objetos da classe *BoolOp*.

4.2.5.
A classe *BoolOpItf*

Analogamente à classe *GroupItf*, a classe *BoolOpItf* gerencia as informações relacionadas às operações booleanas e a maneira como elas são representadas e manipuladas na interface do MG, buscando manter a simplicidade e a eficiência na interação com o usuário. Além disso, a classe *BoolOpItf* promove a comunicação entre a classe *BoolOp* e a estrutura de dados do modelador.

Um objeto da classe *BoolOpItf* possui ponteiros para uma série de elementos de interface: um *label* para orientar o usuário durante a seleção ou a remoção de grupos, uma lista com os identificadores dos grupos existentes, uma lista com os identificadores dos grupos já selecionados pelo usuário e três botões, um para adicionar um grupo à lista de entidades selecionadas, um para remover um grupo desta lista e outro para inverter a ordem de seleção dos grupos. Todos estes elementos de interface fazem parte de um diálogo definido dentro da *callback* de um botão presente na interface do MG responsável pela manipulação dos grupos que serão combinados por meio das operações booleanas (esta *callback* também é um método da classe *BoolOpItf*). Além disso, um objeto desta classe armazena o número de grupos já selecionados, um *flag* para indicar se a regularização é requerida ou não, uma lista encadeada de ponteiros para os grupos selecionados pelo usuário, um ponteiro para um objeto da classe *BoolOp* e um campo estático representado por um ponteiro para um objeto da própria classe *BoolOpItf*, que estará sempre apontando para o último objeto desta classe que foi criado durante a execução do programa (Figura 4.7).

Quatro botões presentes na interface do MG para permitir ao usuário a manipulação dos grupos que serão combinados e das operações booleanas que serão executadas possuem *callbacks* que são métodos privados da classe *BoolOpItf*. Ressalta-se novamente que isto é possível graças ao artifício de utilização de um campo estático da classe que aponte sempre para o último objeto criado. Desta forma, pode-se manipular o objeto dentro de uma *callback* de interface.

O primeiro botão, citado no parágrafo anterior, é responsável pela criação e exibição de um diálogo que contém os elementos de interface tidos como campos privados da classe *BoolOpItf*. Ou seja, este diálogo contém duas listas, uma com grupos existentes no modelo corrente e outra com os grupos já selecionados pelo usuário até o momento de exibição do diálogo. Além disso, os

três botões mencionados permitem ao usuário inserir, remover ou alterar a ordem de seleção dos grupos contidos na lista de grupos selecionados, garantindo contudo que o número de grupos selecionados não seja maior que dois.

Os outros três botões presentes na interface do MG são responsáveis pela chamada das operações booleanas. Ou seja, se o número de grupos selecionados for igual a dois e um destes três botões for acionado, todos os procedimentos descritos no algoritmo proposto neste trabalho são realizados com as entidades presentes em cada grupo. Estes elementos de interface podem ser visualizados na Figura 4.8.

As operações booleanas são executadas chamando-se os métodos apropriados da classe *BoolOp* a partir de um objeto desta classe manipulado pela classe *BoolOpItf*. Ao final de uma operação booleana, o objeto corrente da classe *BoolOpItf* é responsável pela remoção das entidades topológicas presentes na lista de entidades a serem removidas, através de um método desta classe que promove a comunicação das operações booleanas com a estrutura de dados do modelador.

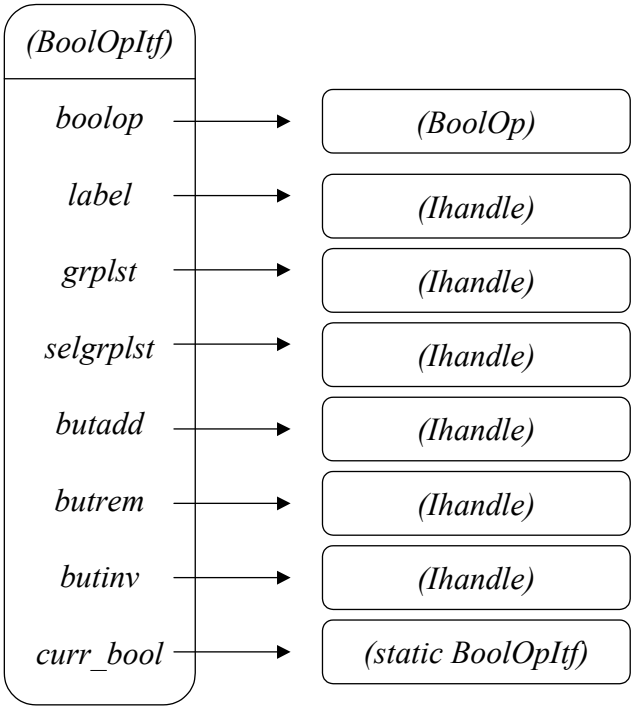
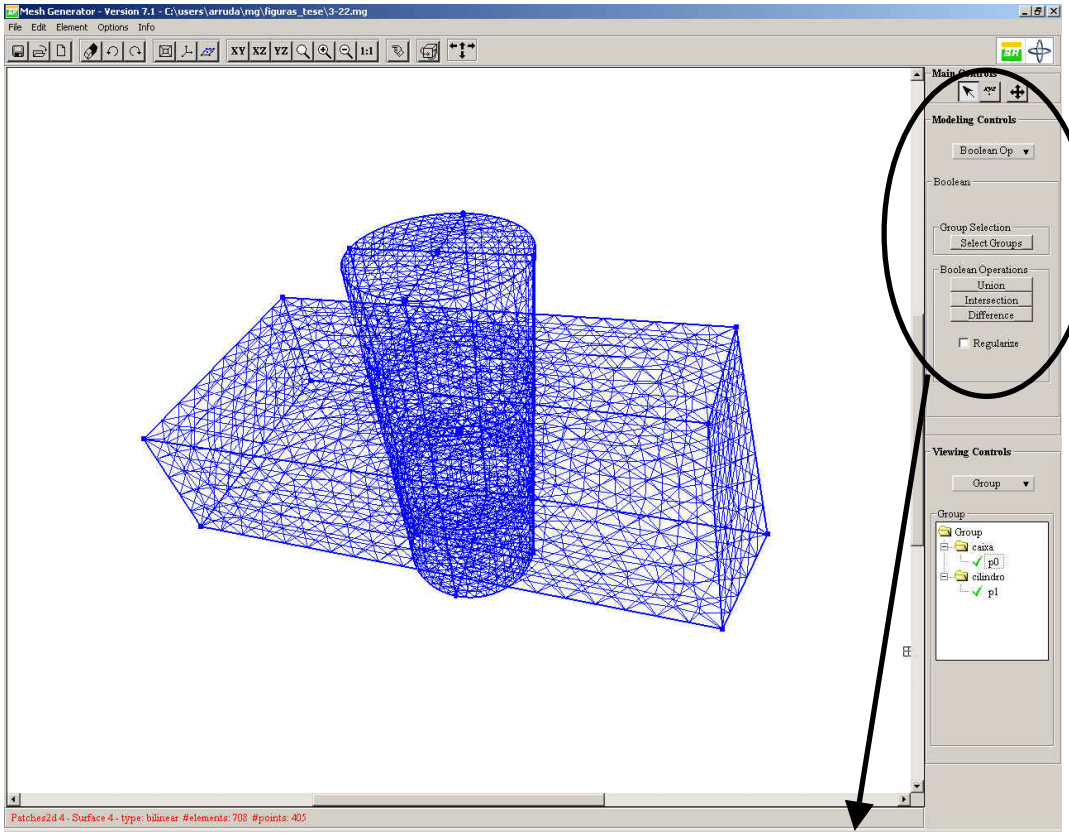


Figura 4.7 – Relações dos objetos da classe *BoolOpItf*.



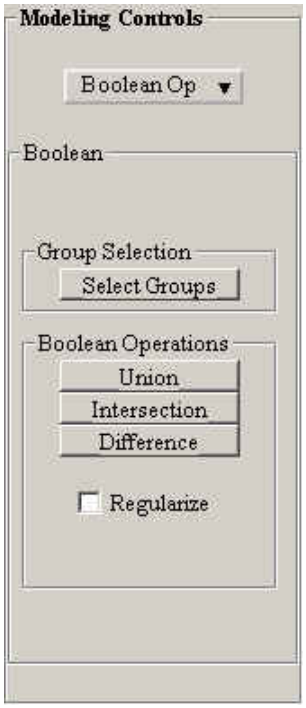
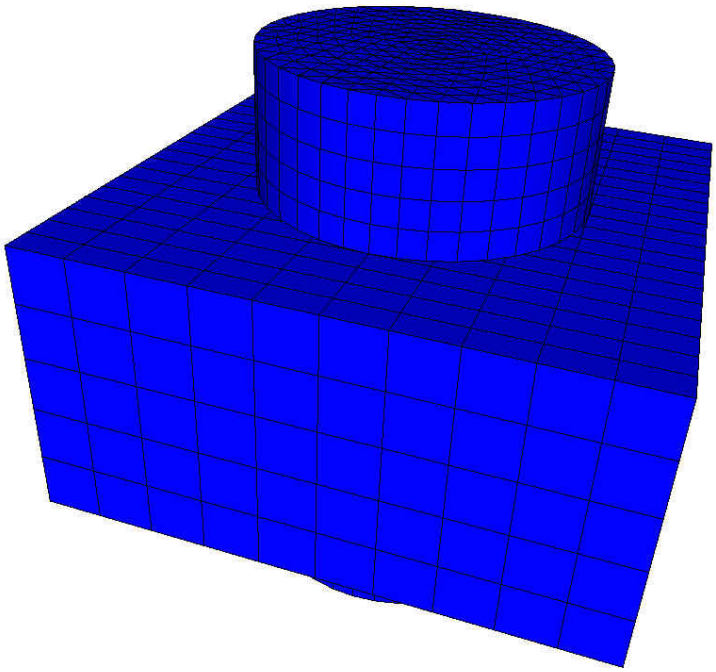


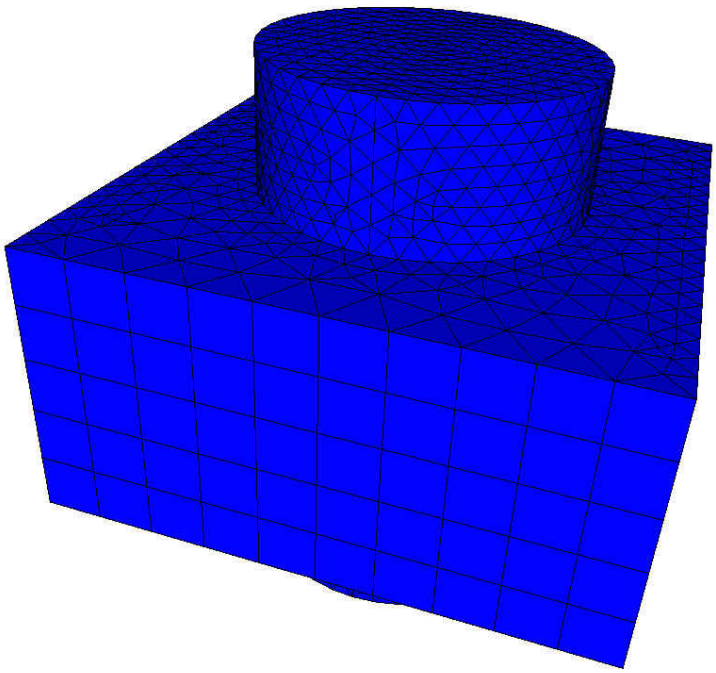
Figura 4.8 – Elementos de interface responsáveis pela chamada do algoritmo de operações booleanas.

4.3.
Pré-processamento dos parâmetros de entrada

No momento em que o usuário seleciona as entidades que irão fazer parte de cada grupo no intuito de realizar operações booleanas entre grupos, ele deve estar ciente de que o algoritmo só funcionará devidamente se forem obedecidas as suas condições de aplicabilidade, como foi visto no capítulo anterior. Isto significa que todas as interseções entre os elementos topológicos que irão participar dos grupos já deverão ter sido previamente calculadas, gerando novos elementos topológicos que apenas se tocam, e que efetivamente deverão ser passados para os grupos. A Figura 4.9 ilustra um modelo gerado no MG antes e depois do cálculo das interseções entre os elementos.



(a)



(b)

Figura 4.9 – Modelo gerado no MG: a) antes da chamada do algoritmo de interseção entre superfícies; b) depois de calculadas as interseções.

Além disso, se o usuário deseja que determinadas regiões façam parte dos grupos, então estas já deverão ter sido reconhecidas, para que se possa efetivamente passar elementos topológicos do tipo *Patch3d* (regiões) para os grupos. Do contrário, se apenas os retalhos de superfícies que formam a fronteira destas regiões forem selecionados, o algoritmo irá encarar estes retalhos de superfícies apenas como faces soltas no espaço, ignorando o fato de que formam um conjunto conexo e fechado de faces. A Figura 3.3 ilustra um exemplo desta diferença conceitual.

Neste ponto é importante se fazer uma ressalva: a interface do MG permite que o usuário escolha entre o reconhecimento automático de regiões ou o reconhecimento de regiões através da seleção explícita pelo usuário dos retalhos de superfície que formam a fronteira de uma região (Figura 4.10). A princípio, a vantagem da utilização de um ou outro procedimento depende do modelo que está sendo analisado e dos objetivos do usuário. Se é necessário que todas as regiões presentes no modelo sejam reconhecidas, ou se a seleção dos retalhos de superfície que formam a fronteira de uma ou mais regiões for muito complicada, então o reconhecimento automático de regiões torna-se bem mais útil e prático. No entanto, se num modelo grande e complexo se deseja efetuar o reconhecimento de regiões somente numa porção pequena do modelo, então a opção pelo reconhecimento manual de regiões é justificada pela economia de tempo no processamento das informações. Além disso, pode acontecer de se desejar tratar um conjunto conexo e fechado de retalhos de superfície apenas como uma casca, como foi citado acima.

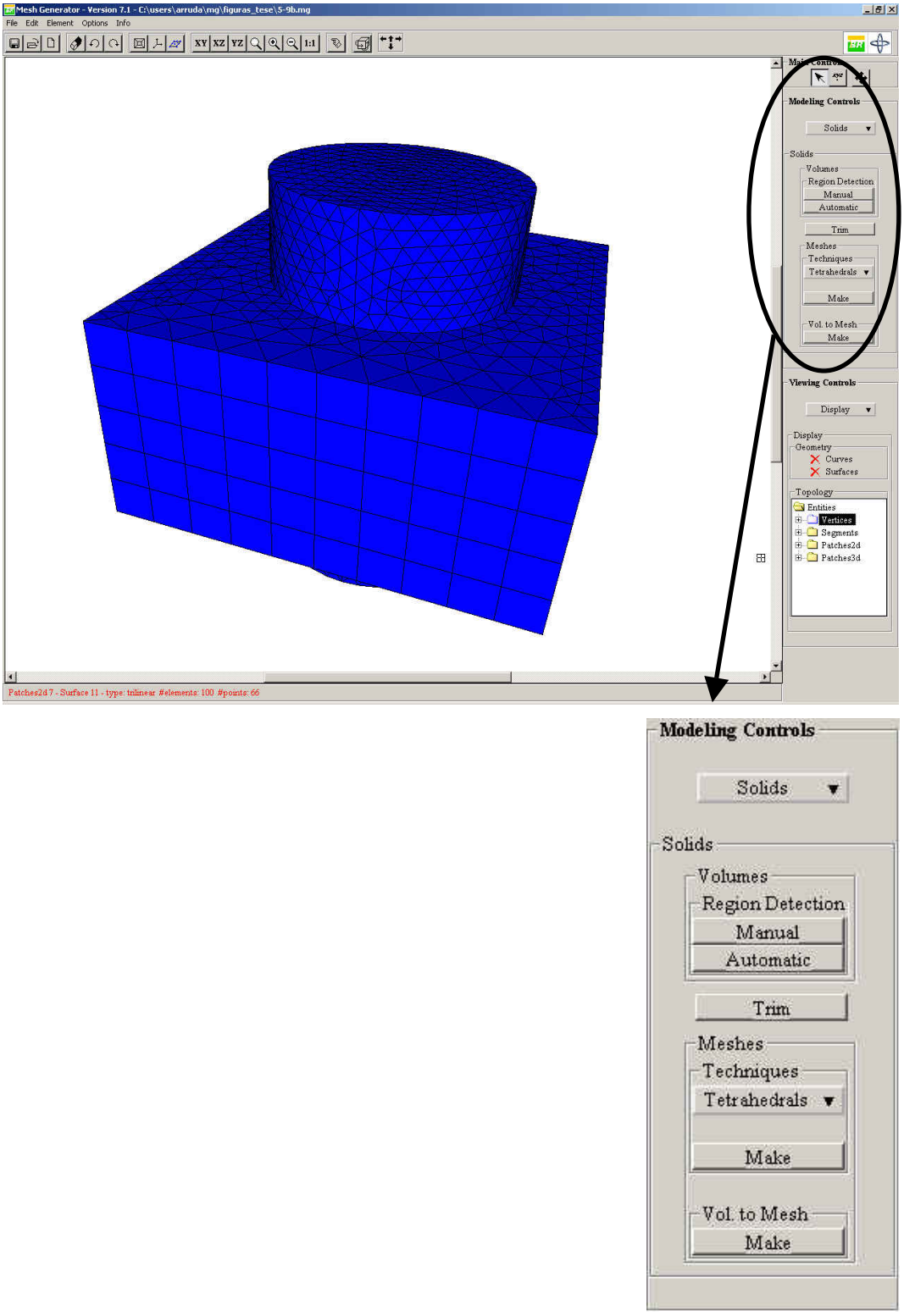


Figura 4.10 – Reconhecimento de multi-regiões: detecção automática ou seleção explícita dos retalhos de superfície que compõem a fronteira de cada região.

Porém, além dessas existe uma outra diferença entre um e outro procedimento, que está relacionada com o tratamento das informações de adjacência dos retalhos de superfície que formam uma ou mais regiões. No

reconhecimento automático de regiões, estes retalhos são passados do MG para a biblioteca CGC, que devolve ao MG as regiões formadas. Já no reconhecimento manual de regiões, o MG se encarrega de consultar as informações de adjacência na sua própria estrutura de dados para determinar a existência de regiões fechadas que utilizem somente os retalhos de superfícies selecionados pelo usuário.

Se um conjunto *A* conexo e fechado de retalhos de superfície formar uma região, mas no interior desta região houver um conjunto *B* formado por um ou mais retalhos de superfície pendentes ou soltos, caso o usuário selecione explicitamente apenas os retalhos de superfície do conjunto *A* e opte pelo reconhecimento manual de regiões, a região formada terá apenas os retalhos de superfície do conjunto *A* como parte da sua fronteira, já que neste procedimento o MG considera somente os retalhos de superfície passados como parâmetro no reconhecimento de regiões. Contudo, se o usuário optar pelo reconhecimento automático de regiões, a biblioteca CGC retornará ao MG a informação de que todos os retalhos de superfície do conjunto *A* e do conjunto *B* compõem a fronteira da região formada. Como os objetos da classe *Patch3d* do MG possuem uma única lista com as referências dos retalhos de superfície que compõem a sua fronteira (identificados pela CGC), sem informações sobre o fato de um determinado retalho de superfície ser pendente ou solto no interior daquela região, então haverá uma inconsistência entre a informação de fronteira desta região e o conceito de fronteira apresentado neste trabalho (seção 3.2), caso esta região venha a fazer parte de algum grupo utilizado na execução das operações booleanas. Para que isto não aconteça, é necessário que se adote o seguinte procedimento: caso uma determinada região seja requisitada para fazer parte de um grupo que servirá de parâmetro de entrada para as operações booleanas, deve-se selecionar explicitamente os retalhos de superfície da sua fronteira e efetuar o reconhecimento manual de regiões, sem que as estruturas internas a esta região, pendentes ou soltas, sejam selecionadas. Se o usuário desejar que as estruturas internas à região façam parte daquele grupo, então ele deve selecioná-las explicitamente e acrescentá-las ao grupo. Obviamente, se as regiões que se deseja reconhecer não possuírem estruturas internas e o usuário quiser optar pelo reconhecimento automático de regiões, isto não acarretará problema algum. Uma das soluções para este problema seria aquela apresentada no capítulo anterior, ou seja, uma modificação na interface entre o MG e a biblioteca CGC de modo que estruturas internas pendentes ou soltas

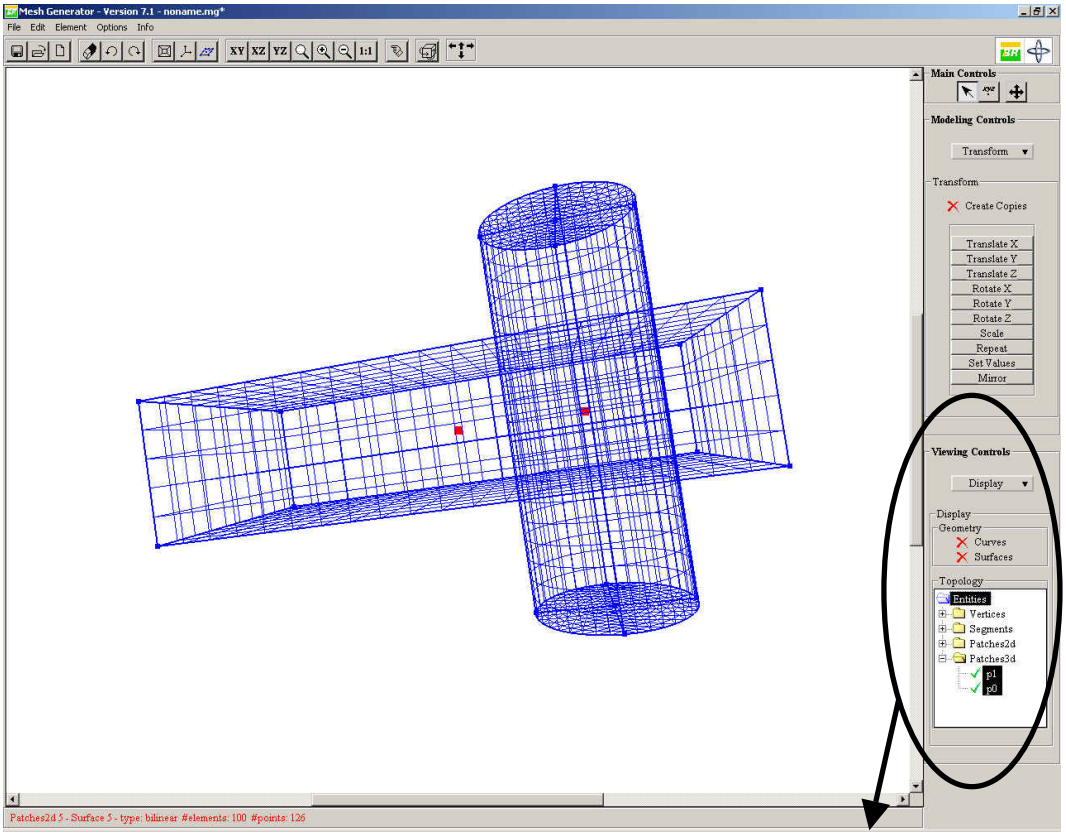
sejam identificadas e diferenciadas das outras. Infelizmente, até a conclusão deste trabalho, esta modificação ainda não havia sido implementada.

Uma modificação realizada no código fonte do MG durante o desenvolvimento deste trabalho diz respeito ao tratamento de regiões que se interceptam quando o algoritmo de interseção é chamado. Originalmente, durante a execução do algoritmo de interseção, o modelador destruía os *patches3d* cujas fronteiras se interceptavam, removendo-os da estrutura de dados. Se após a interseção entre curvas e superfícies se desejasse que as regiões continuassem existindo, o algoritmo de detecção de regiões deveria ser novamente chamado. Para que as novas regiões comportassem exatamente os mesmos volumes que as regiões originais era necessário que o usuário realizasse a detecção manual de regiões, selecionando explicitamente os novos retalhos de superfície que constituíam a fronteira das regiões. Este procedimento podia não ser trivial em alguns casos, devido à presença de retalhos de superfície difíceis de serem selecionados. No entanto, se não houvesse necessidade de que as novas regiões compreendessem os mesmos volumes que as regiões originais, o algoritmo para detecção automática de regiões poderia ser chamado.

Como as operações booleanas normalmente são aplicadas sobre regiões que se interceptam, trechos do código fonte do modelador foram modificados de forma que após o término do algoritmo de interseção os *patches3d* originais não fossem removidos, mas apenas tivessem a sua lista de *patches2d* adjacentes atualizada. Desta forma, ao manipular objetos sólidos no ambiente de modelagem do MG, o usuário passou a contar com duas opções: realizar o reconhecimento de regiões *antes* de calcular as interseções entre os retalhos de superfícies e arestas que formam a fronteira das regiões, ou calcular as interseções *antes* do reconhecimento de regiões, podendo em seguida selecionar manualmente os retalhos de superfície da fronteira de cada região ou optar pelo reconhecimento automático de regiões. As Figuras 4.11 e 4.12 ilustram a interface para as diferentes formas de se tratar o problema.

No Capítulo 2, foi comentada uma limitação do MG em relação ao cálculo da interseção entre superfícies. Quando um retalho de superfície se superpõe a outro, sem que as arestas dos seus contornos coincidam, o algoritmo de interseção falha em alguns casos. Este problema pode estar relacionado com o fato de que as próprias malhas das superfícies existentes são usadas como suporte para a definição das curvas de interseção. Quando estas curvas de interseção coincidem com as próprias arestas do contorno de uma das

superfícies, é possível que o algoritmo de interseção encontre problemas na determinação dos pontos de interseção das arestas do retalho de superfície mais interno contra o outro retalho de superfície, já que neste caso existem infinitos pontos de interseção. Assim, a determinação das curvas e regiões de *trimming* poderia ficar comprometida, e conseqüentemente também a reconstrução das malhas.



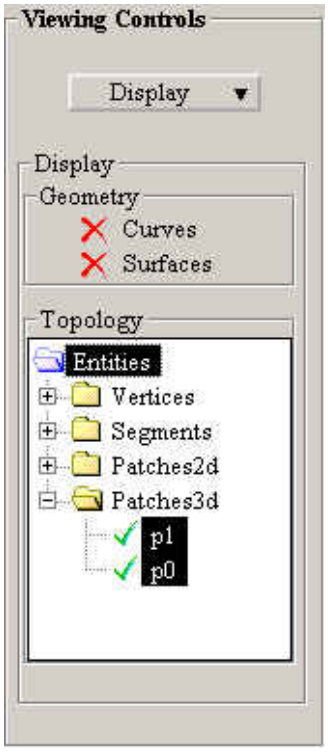


Figura 4.11 – Detecção de regiões antes da interseção.

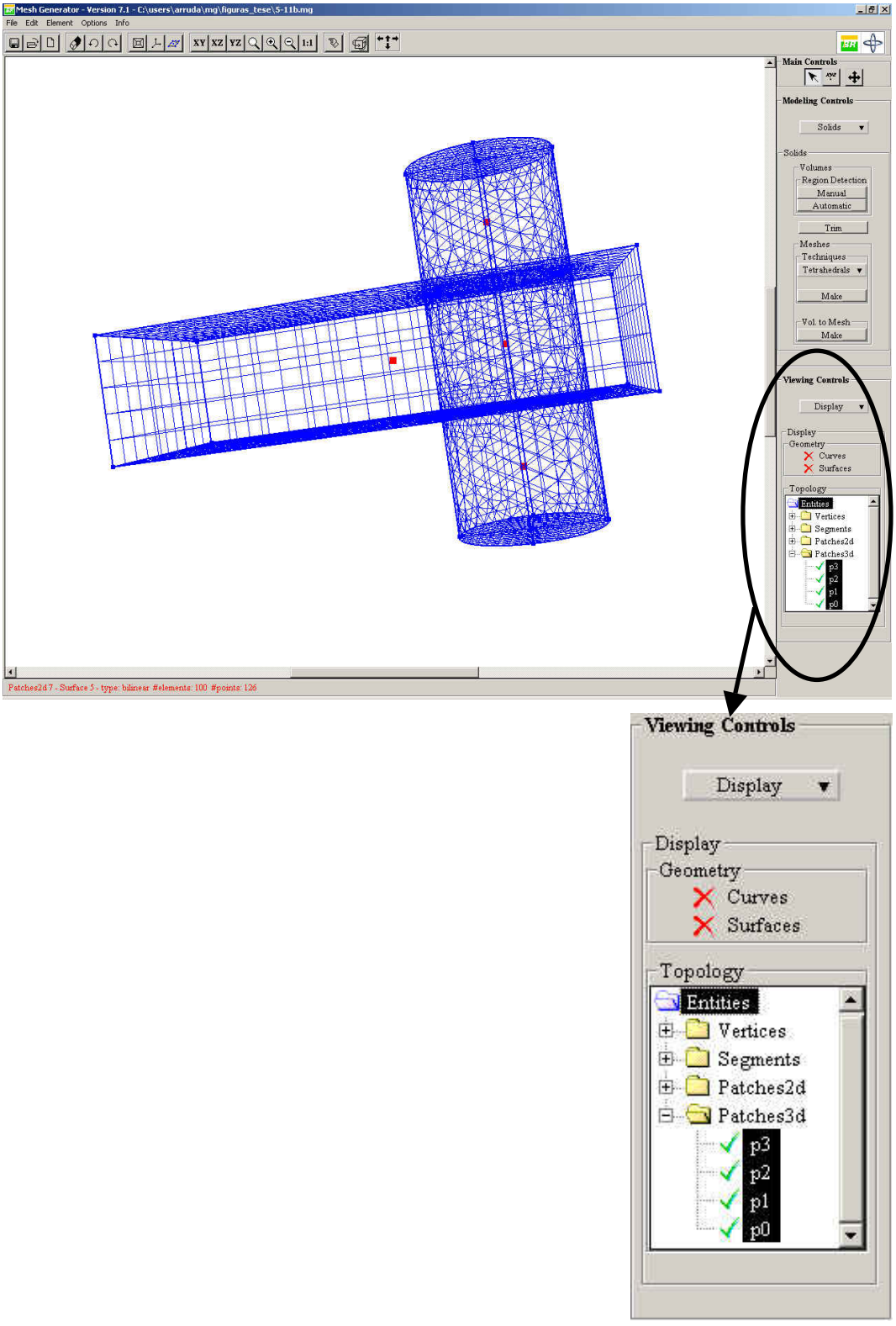


Figura 4.12 – Detecção de regiões depois da interseção.

4.4. Detalhes de implementação das operações booleanas no MG

A forma como o algoritmo de operações booleanas foi implementado no MG necessita de um maior detalhamento. É preciso que se compreenda melhor a maneira como as informações teóricas foram adaptadas para o contexto deste modelador, usufruindo facilidades já existentes que fizeram desta adaptação uma tarefa mais simples. Além disso, a forma como algumas informações de adjacência são armazenadas na estrutura de dados do MG também influenciou bastante na implementação do algoritmo da forma com ele foi descrito. Em alguns casos, houve necessidade de se realizar pequenas modificações em alguns passos do algoritmo original, mas que não comprometeram de forma muito significativa o resultado esperado da aplicação das operações booleanas sobre grupos de entidades topológicas. Algumas limitações do modelador também influenciaram o resultado final das operações booleanas, mas somente para um número reduzido de casos.

4.4.1. Armazenamento das informações dos grupos

Quando uma operação booleana é requisitada pelo usuário através de um dos botões da interface, caso já existam dois grupos de entidades topológicas selecionados, referências para estes são passadas como parâmetros para o método *addgroups* da classe *BoolOp*. O primeiro passo do algoritmo é o armazenamento de referências para todas as entidades topológicas explicitamente contidas nos grupos e para todas as entidades topológicas hierarquicamente inferiores a estas que podem ser obtidas por relações de adjacência.

A princípio, da forma como o algoritmo foi proposto, *loops* internos de faces constituídos por seqüências alternadas de vértices e arestas fechando um ciclo, vértices soltos no interior de faces, arestas soltas ou pendentes no interior de faces ou de regiões, cascas no interior de regiões constituídas por conjuntos conexos e fechados de faces e faces soltas ou pendentes no interior de regiões deveriam ser automaticamente armazenados. Contudo, na prática, a estrutura de dados do MG não permite que se obtenha todas estas entidades apenas por relações de adjacência.

O MG permite que se tenha acesso, através de relações de adjacência, às arestas pendentes ou soltas no interior de faces, bem como às arestas e vértices

pertencentes a um *loop* interno de uma face formando um ciclo fechado, contudo não há ainda suporte para vértices soltos no interior de faces. Se houver interesse na inclusão destes vértices em um dos grupos, eles devem ser explicitamente selecionados. Arestas ou faces pendentes ou soltas no interior de regiões, como já foi dito, também devem ser explicitamente selecionadas para que façam parte de um grupo. Vale lembrar que as regiões que contêm estas estruturas internas devem ser reconhecidas manualmente, por meio da seleção das faces que compõem a sua *fronteira*, da forma como ela foi definida neste trabalho na seção 3.2 (sempre que o termo *fronteira* for mencionado, deve-se ter em mente a definição adotada neste trabalho).

4.4.2. Classificação das entidades topológicas dos grupos

Esta é uma das etapas do algoritmo que teve de ser adaptada para que pudesse ser implementada no ambiente de modelagem do MG sem gerar problemas durante a determinação do resultado de uma operação booleana.

No MG, bem como na maioria dos modeladores que utilizam uma representação de fronteira, a remoção de uma entidade topológica de um modelo pode ocasionar a remoção automática de outras entidades topológicas adjacentes à que foi removida pela estrutura de dados do modelador, para manter a consistência topológica do modelo. Assim, a remoção de um vértice provoca a remoção automática de todas as arestas e faces incidentes naquele vértice. A remoção de uma aresta provoca a remoção automática do conjunto de todas as faces que compartilham a aresta, constituído pelas faces em cuja fronteira esta aresta está localizada e pelas faces que possuem esta aresta pendente ou solta no seu interior. Além disso, a remoção de uma aresta provoca também a remoção automática dos seus vértices extremos, a não ser que existam outras arestas incidentes nestes vértices. A remoção de uma face provoca a remoção automática de todas as regiões em cuja fronteira esta face está localizada.

Exceto pela remoção automática dos vértices extremos de uma aresta quando esta é removida (desde que não haja outras arestas incidindo sobre estes vértices), os outros procedimentos automáticos descritos quando uma entidade topológica é removida não influenciariam os passos do algoritmo, já que foi frisado que a ordem de remoção das entidades topológicas deve ser precisamente aquela que foi descrita, ou seja, primeiramente removem-se as

faces, depois as arestas e depois os vértices. A remoção automática dos vértices extremos de uma aresta removida, contudo, faz com que se tenha de ter cuidado na etapa de inclusão dos vértices a serem removidos na lista *ldelent*. Quando esta lista é passada para o objeto da classe *BoolOpItf* responsável pela interface entre as operações booleanas e o MG, para que ele informe para a estrutura de dados do modelador que as entidades topológicas contidas nesta lista devem ser removidas, é extremamente recomendável que não sejam passadas referências de entidades que já não fazem parte da estrutura de dados, pois isto pode ocasionar problemas, como invasão de memória. Ao mesmo tempo, esta remoção automática dos vértices de uma aresta removida deveria de certa forma simplificar os procedimentos descritos no algoritmo, pois todo vértice que possui arestas incidentes não precisaria ser armazenado nem classificado. Mas na verdade esta simplificação pode comprometer o resultado das operações booleanas em algumas situações, como pode ser visto na Figura 4.13. Como situações como estas são menos comuns de ocorrerem, optou-se por efetuar esta simplificação, classificando-se somente os vértices que não possuem arestas incidentes. Estes são os vértices que poderão ser passados para a lista *ldelent* para serem explicitamente removidos.

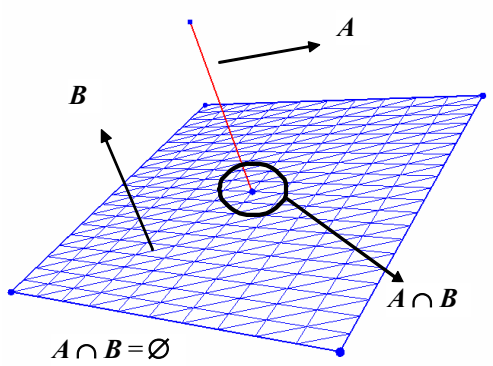
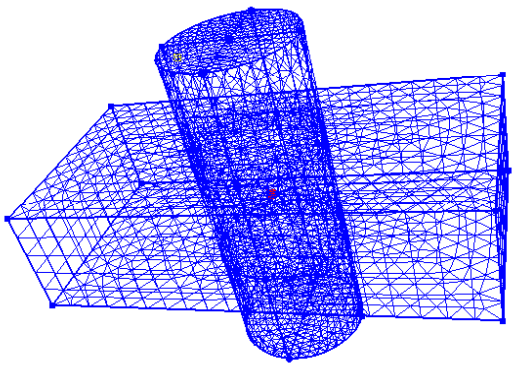


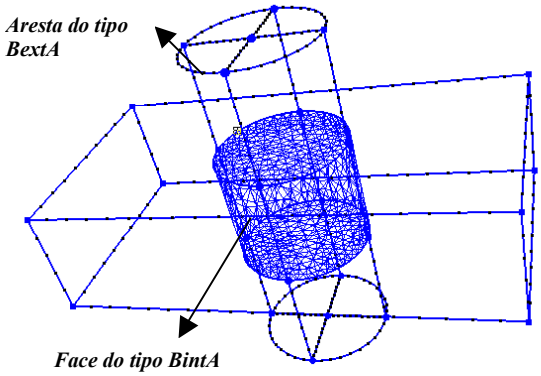
Figura 4.13 – Interseção entre duas entidades topológicas A e B. O resultado da interseção é o vértice comum a ambas as entidades, mas que não pode ser representado devido à remoção automática dos vértices de uma aresta quando ela é removida.

Existe ainda um outro problema relacionado com a remoção das arestas após a aplicação das operações booleanas sobre os grupos. No MG, a descrição geométrica de um retalho de superfície (face) está vinculada com as arestas da sua fronteira, pela própria maneira como ele é gerado. As formas de geração de

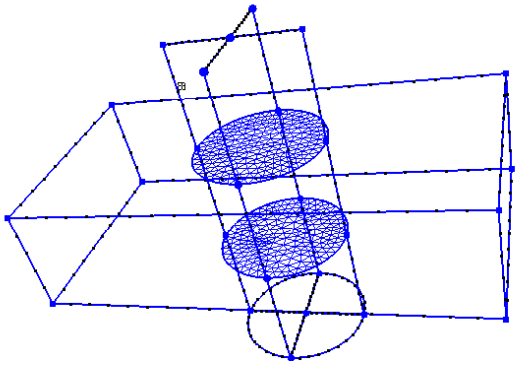
retalhos de superfície no MG foram descritas no Capítulo 2. Dessa forma, quando as interseções entre os retalhos de superfície são calculadas gerando assim novos retalhos de superfície, estes novos retalhos de superfície possuem a mesma descrição geométrica dos retalhos originais que os geraram, ou seja, eles dependem da descrição geométrica das arestas que formavam a fronteira dos retalhos de superfície originais. As arestas das fronteiras destes novos retalhos que foram geradas pela subdivisão das arestas das fronteiras dos retalhos originais contêm as mesmas informações geométricas que elas. Porém, existem arestas da fronteira dos retalhos originais que passam a não fazer parte da fronteira dos novos retalhos. Estas arestas guardam informações necessárias para a descrição geométrica dos novos retalhos de superfícies. Assim, se elas forem removidas, automaticamente os retalhos de superfícies que dependem delas para a completa caracterização da sua geometria serão removidos. Este processo pode ser visualizado na Figura 4.14. O problema maior neste caso é que situações como estas podem ocorrer com bastante freqüência, pois um retalho de superfície pertencente a um grupo pode ser classificado quanto à sua localização espacial em relação às entidades do outro grupo de forma diferente das arestas que compunham a fronteira do retalho de superfície original cuja subdivisão deu origem a este retalho. A Figura 4.14 ilustra um exemplo simples de situação em que isto ocorre.



(a)



(b)



(c)

Figura 4.14 – Influência das curvas geradoras de um retalho de superfície na sua descrição geométrica: a) dois sólidos; b) interseção entre os sólidos; c) remoção de quatro arestas ocasionando remoção automática dos retalhos de superfície.

Tendo em vista este problema, a maneira mais simples de tratá-lo, sem a necessidade de se implementar grandes artifícios para descobrir quais são as arestas responsáveis pela descrição geométrica de cada retalho de superfície, é a manutenção de todas as arestas pertencentes aos dois grupos que fizerem

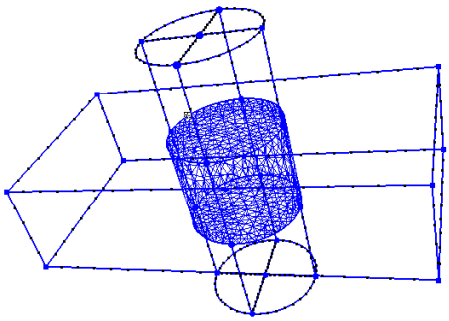
parte da fronteira de alguma face, independentemente do fato desta face ter sido removida ou não. A princípio, isto pode parecer uma simplificação grosseira, mas visualmente esta opção ainda é melhor do que deixar somente as arestas que carregam sozinhas uma parte da descrição geométrica de alguma face que não foi removida e que não fazem parte da fronteira desta face. Uma comparação entre estas duas possibilidades pode ser vista na Figura 4.15. A manutenção de todas as arestas dá uma visão geral do problema, permitindo que o usuário tenha uma noção do contorno dos retalhos de superfície originais antes da aplicação das operações booleanas, em contraste com os retalhos de superfície remanescentes no resultado destas operações.

Com esta simplificação adotada, as únicas arestas que precisam ser classificadas são aquelas que não pertencem à fronteira de nenhuma face existente nos seus respectivos grupos (explicitamente ou implicitamente através da fronteira de uma região). Ou seja, as arestas que precisam ser classificadas são aquelas que se encontram pendentes ou soltas no interior de faces ou regiões, aquelas que se encontram pendentes exteriormente a alguma face ou região, e as que se encontram soltas no espaço tridimensional exteriormente a todas as regiões.

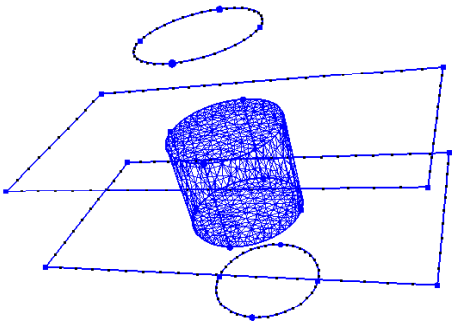
Arestas pendentes ou soltas no interior de faces fazem parte de uma lista encadeada de arames, denominada *lwire*, que faz parte da estrutura de dados de uma face (*Patch2d*). As arestas da fronteira de uma face fazem parte de uma outra lista, *lsegments*. Contudo, as arestas dos *loops* internos a uma face constituídos por ciclos fechados e alternados de arestas e vértices também fazem parte da lista *lwire*, o que não é interessante do ponto-de-vista do algoritmo proposto, já que estas arestas efetivamente fazem parte da fronteira desta face. Como não há como diferenciar as arestas pendentes ou soltas no interior de uma face das arestas de fronteira dessa face que pertencem a um *loop* interno, já que todas elas pertencem à lista *lwire*, as arestas de fronteira que pertencem a um *loop* interno também são classificadas juntamente com as outras, a não ser que elas façam parte da fronteira de uma outra face (Figura 4.16). Isto só ocasionaria algum problema no resultado de uma operação booleana se houvesse alguma situação em que as arestas de fronteira do *loop* interno de uma face tivessem de ser removidas sem que a face tivesse sido removida, pois a remoção destas arestas faria com que a face fosse automaticamente removida. Esta situação poderia ocorrer, por exemplo, numa operação de diferença do tipo *A – B* em que a remoção de uma determinada aresta do tipo *BinA* ou *INTERS* está condicionada ao fato dela fazer parte ou não

da fronteira de alguma face remanescente (Capítulo 3). Neste caso, as arestas de fronteira do *loop* interno de uma face remanescente que fossem classificadas desta forma, por serem consideradas equivalentes a arestas soltas no interior da face, seriam passadas para a lista *ldelent* erradamente, fazendo com que a face fosse automaticamente removida. Uma situação deste tipo é ilustrada na Figura 4.17. No entanto, vale lembrar que o mesmo problema ocorreria no caso de arestas efetivamente pendentes ou soltas no interior da face, já que todas estas arestas estão sendo consideradas equivalentes e recebendo o mesmo tratamento. Logo, o problema maior está no vínculo existente entre arestas localizadas no interior de faces e as faces em si. Uma solução para este problema seria alterar o código fonte do MG de tal forma que quando uma restrição interna a um retalho de superfície fosse removida, o único efeito fosse a reconstrução da malha associada àquele retalho de superfície, sem que este fosse removido.

A classificação das faces de cada grupo não sofre restrição alguma, nem cria a necessidade de se fazer adaptações ao algoritmo proposto para poder ser realizada.



(a)



(b)

Figura 4.15 – Comparação entre duas possibilidade de abordagem: a) manutenção de todas as arestas; b) manutenção somente das arestas que guardam sozinhas informações geométricas sobre os retalhos de superfície.

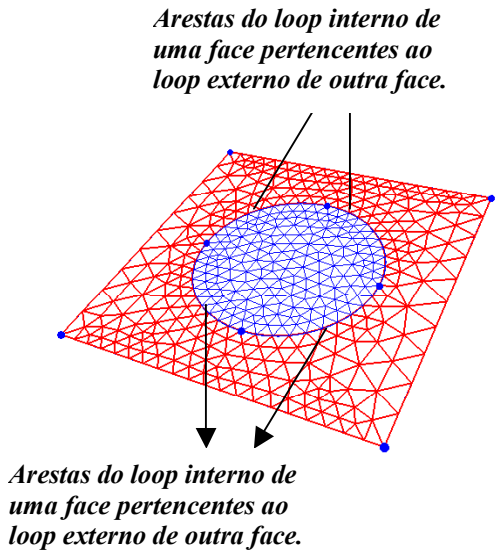


Figura 4.16 – Arestas no interior de uma face formando um *loop* interno fechado.

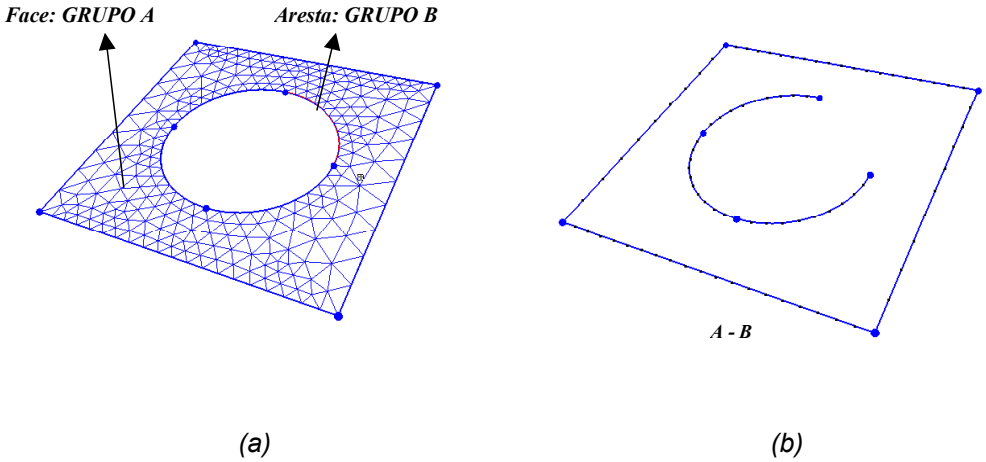


Figura 4.17 – Aresta de fronteira sendo tratada como aresta solta no interior da face.

4.4.3. Operações Booleanas e Regularização do Resultado

Tendo em vista as limitações do modelador expostas nas seções anteriores e as adaptações ao algoritmo para que ele pudesse ser implementado no MG da forma mais fiel possível à sua descrição original, é interessante verificar se os métodos públicos da classe *BoolOp* responsáveis pelas operações booleanas de união, interseção e diferença e o método público desta mesma classe responsável pela regularização do resultado também precisaram ser implementados realizando-se modificações ou aproximações no algoritmo original.

De uma forma geral, esses métodos foram implementados exatamente como foi descrito no algoritmo. O que se deve ter em mente, no entanto, é o conjunto de entidades topológicas que estão sendo tratadas quando as listas *lallvtx* e *lallseg* estão sendo percorridas. Pode-se dizer que, exceto pelas situações particulares já comentadas, em que o resultado da aplicação de alguma operação booleana pode ser diferente do esperado, o único detalhe de implementação que também poderia comprometer o resultado destas operações é o uso do método *regions*, da classe *GeomBoolOp*, que é aquele responsável pela detecção de possíveis regiões formadas a partir de um conjunto de retalhos de superfície passados como parâmetro. Este método é chamado tanto na operação de diferença quanto na regularização do resultado. Na diferença, ele é chamado após o passo de verificação e remoção de faces, para que se possa determinar se eventuais faces do tipo *BinA* que foram temporariamente mantidas como parte do resultado fazem ou não parte das fronteiras das regiões formadas com as faces remanescentes. As faces deste tipo que não fizerem parte da fronteira de nenhuma região formada devem ser removidas. Na regularização, este método é usado para se determinar quais as faces (com qualquer classificação) que fazem parte da fronteira das eventuais regiões formadas com as faces remanescentes de alguma operação booleana. As faces que não fizerem parte da fronteira de nenhuma região devem ser removidas.

O problema, neste caso, também já havia sido comentado anteriormente. Como este método envia as informações necessárias sobre os retalhos de superfície (faces) para a biblioteca CGC, para que esta retorne ao MG a informação sobre os retalhos de superfície que fazem parte da fronteira das regiões formadas, as faces pendentes ou soltas internamente às regiões formadas serão consideradas como parte da fronteira das mesmas, o que vai de encontro à definição de fronteira adotada neste trabalho. Vale notar que neste caso, o problema é mais sério, já que não pode ser resolvido instruindo-se o usuário a agir de uma determinada maneira. Para que estruturas internas não sejam consideradas como parte da fronteira de regiões quando estas regiões estão sendo selecionadas para fazerem parte de um grupo, basta que o usuário faça manualmente o reconhecimento de regiões, desconsiderando as estruturas internas. No presente caso, não há como solucionar o problema de forma análoga, já que o procedimento que está sendo realizado é totalmente automático, não dependendo da interação do usuário.

Contudo, vale ressaltar que para que estruturas pendentes ou soltas internamente a regiões façam parte do resultado de uma operação booleana, é

necessário que elas já sejam estruturas internas a alguma região nas condições iniciais do problema. Ou seja, apenas um modelo inicialmente *non-manifold* poderia recair neste problema, comprometendo o resultado das operações de diferença e regularização.

4.5. Pós-processamento do resultado

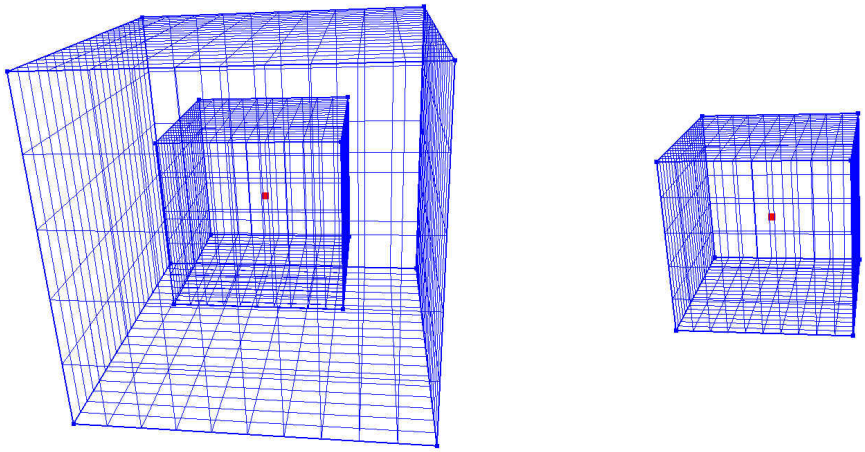
Após a aplicação das operações booleanas sobre os dois grupos e de uma eventual regularização do resultado, pode-se realizar o reconhecimento de regiões com os retalhos de superfície remanescentes. Este procedimento pode ser bastante útil, por exemplo, num processo de criação de um modelo complexo de engenharia, quando se deseja que as regiões resultantes de uma operação booleana sejam combinadas com novas regiões, novamente através de operações booleanas.

O reconhecimento automático de regiões como pós-processamento do resultado de uma operação booleana livra o usuário da tarefa de selecionar explicitamente os retalhos de superfície que formam a fronteira de cada região para efetuar o reconhecimento manual de regiões. Este inclusive pode ser um procedimento demorado e difícil, dependendo da posição relativa dos retalhos de superfície no espaço. Obviamente, o reconhecimento automático de regiões poderia ser efetuado, contudo nem sempre se deseja que todas as regiões do modelo completo sejam reconhecidas, mas apenas aquelas resultantes da aplicação das operações booleanas sobre os grupos de entidades topológicas escolhidos pelo usuário.

O único problema em se realizar o reconhecimento de regiões automaticamente com os retalhos de superfície remanescentes de uma operação booleana é que qualquer conjunto conexo e fechado de retalhos de superfície passa a ser encarado como a fronteira de uma nova região, o que nem sempre se é desejável. A Figura 4.18 mostra um exemplo de interseção entre dois sólidos em que o resultado deve ser tratado apenas como uma casca, ou seja, apenas como o conjunto de retalhos de superfície remanescentes da interseção, sem considerar o volume interno a este conjunto de retalhos de superfície. Um dos sólidos que está sendo combinado é um cubo com um vazio interno representado por uma casca cúbica. O outro sólido é um cubo que ocupa exatamente o volume do vazio interno do primeiro cubo. A interseção entre estes dois sólidos constitui-se apenas das faces da fronteira do segundo cubo. Nesse

caso, o reconhecimento automático de regiões após a interseção resulta em uma região que não pertence ao resultado da operação booleana de interseção realizada.

Como casos como o da Figura 4.18 são bastante particulares e difíceis de serem encontrados em problemas práticos, optou-se por realizar o reconhecimento automático de regiões com os retalhos de superfície remanescentes de uma operação booleana.



(a) *(b)*

Figura 4.18 – Problema com a detecção automática de regiões após a operação booleana: a) sólido com vazio interno ocupado por outro sólido; b) interseção entre os dois sólidos representada apenas por uma casca.

4.6. Eficiência e robustez do algoritmo

Como foi visto, o algoritmo divide-se em três etapas: armazenamento das informações provenientes dos dois grupos, classificação das entidades topológicas (vértices, arestas e faces) e processamento das operações booleanas com estas entidades.

A primeira etapa caracteriza-se por ser bastante rápida. Extrair informações contidas nos grupos e obter novas informações a partir destas pelas relações de adjacência são procedimentos que dependem somente de consultas a listas encadeadas e inserção de novos nós em outras listas encadeadas. O tempo de execução desta etapa varia linearmente com a quantidade de entidades topológicas contidas diretamente ou indiretamente nos dois grupos.

A segunda etapa é determinante na estimação do tempo de execução do algoritmo. Ela é a etapa lenta. Isto porque ela é a única que depende dos

algoritmos geométricos da biblioteca CGC disponibilizados pelos métodos públicos da classe *GeomBoolOp*. A detecção de ponto em retalho de superfície e a detecção de ponto em região são dois procedimentos relativamente lentos, e que são chamados muitas vezes nesta etapa. Mesmo quando os grupos não possuem vértices ou arestas soltos ou pendentes, é necessário se localizar espacialmente os vértices das fronteiras das faces de um grupo em relação às regiões e faces do outro grupo. Pode-se dizer que o tempo de execução desta etapa também depende do número de entidades topológicas presentes em cada grupo, contudo o número de entidades não é tão determinante quanto o número total de vértices não compartilhados pelos dois grupos localizados interiormente à *bounding box* de alguma face ou região do grupo que não o contém. Isto porque estes algoritmos geométricos são chamados apenas no momento em que vértices deste tipo precisam ser classificados. Uma ordem de grandeza do tempo de duração desta etapa será apresentada no próximo capítulo.

A última etapa é menos lenta que a anterior, contudo não tão rápida quanto a primeira. Como nesta etapa as entidades já estão classificadas, o que se faz efetivamente é consultar o tipo de classificação de cada entidade e determinar se ela deve ser removida ou não. Este processo seria bem mais rápido se não fosse por consultas adicionais às listas de regiões de cada grupo e às listas de faces adjacentes a cada região para se determinar quantas regiões de cada grupo compartilham uma mesma face e pela chamada do método *regions* da classe *GeomBoolOp* na operação de diferença, para se determinar se as eventuais faces do tipo *BinA* remanescentes fazem parte da fronteira de alguma região resultante.

A questão da robustez do algoritmo proposto deve ser analisada separadamente para o algoritmo teórico e para a sua adaptação ao ambiente de modelagem do MG. O algoritmo teórico exposto no capítulo anterior busca cobrir todos os casos possíveis em que dois grupos de entidades topológicas em qualquer número e com qualquer dimensão são combinados através das operações booleanas, desde que obedecidas as suas condições de aplicabilidade. Arestas e faces pendentes ou soltas e vértices soltos interiormente ou exteriormente às regiões presentes, regiões de um mesmo grupo ou de grupos distintos com faces, arestas ou vértices comuns em suas fronteiras, regiões com múltiplas cascas, faces com múltiplos *loops* e superposição de faces são exemplos de casos tratados pelo algoritmo proposto. Logo, pode-se dizer que o algoritmo teórico é bastante robusto para os objetivos a que se propõe. Já a versão implementada no MG não possui um nível de

abrangência tão grande, já que vários casos particulares não podem ser tratados apropriadamente devido às limitações atuais do modelador expostas neste capítulo.

Vale ressaltar, no entanto, que a robustez do algoritmo está associada ao tipo de tratamento que se resolveu atribuir às entidades topológicas presentes nos grupos. Isto significa que a maneira como as informações de entrada são manipuladas pelo algoritmo também integra o conjunto de contribuições deste trabalho. Não há na literatura de modelagem geométrica uma maneira formal e padronizada de se tratar entidades topológicas em domínios *non-manifold* quando operações booleanas são aplicadas sobre elas, como há em domínios *manifold*. Logo, este trabalho propõe apenas uma possível forma de abordagem do problema, tendo em vista a sua aplicação no campo da modelagem aplicada à engenharia. Uma maneira mais simplificada de se tratar o problema, por exemplo, seria desconsiderar quaisquer entidades topológicas internas a quaisquer regiões presentes nos dois grupos, mantendo assim o ponto-de-vista de *lugar geométrico* que costuma estar atrelado às operações booleanas. Poderia também ser feita a restrição de não permitir que regiões de um mesmo grupo compartilhem uma mesma face, ou ainda de não poder haver vértices, arestas ou faces completamente soltos no espaço. A superposição de faces também poderia ser proibida.

Todas estas simplificações sem dúvida reduziriam o esforço computacional necessário para o processamento das informações e diminuiriam o número de situações topológicas particulares em que o resultado de uma operação booleana poderia ficar comprometido. Contudo, desejou-se tratar a questão das operações booleanas em domínios *non-manifold* da forma mais abrangente possível.

5
Exemplos

Este capítulo contém exemplos das técnicas de modelagem que foram descritas anteriormente e implementadas no MG. Diversos modelos são gerados a partir do conjunto de operações booleanas disponíveis. Desta forma, pode-se ter uma noção mais nítida da importância de tais operações na modelagem de objetos complexos, no que diz respeito à facilidade de criação dos mesmos e economia de tempo.

Casos patológicos, em que o algoritmo de interseção de superfícies falha, também são apresentados para análise dos problemas encontrados e das causas dos mesmos.

5.1.
Geração de modelos *manifold*

Nesta seção, alguns modelos *manifold* gerados no MG com as operações booleanas serão mostrados. Mesmo em modelos relativamente simples, o uso das operações booleanas se faz de grande utilidade, pois a remoção das entidades indesejadas é feita automaticamente, evitando que o usuário tenha de selecionar explicitamente as entidades topológicas que deseja eliminar, tarefa que em muitas situações pode não ser trivial.

A Figura 5.1a mostra inicialmente dois cilindros se interceptando, cada um sendo representado por uma região (*patch3d*). São criados dois grupos, um para cada região. A união entre os grupos é então realizada, formando um único sólido, que é representado por uma nova região (Figura 5.1b). Esta nova região é acrescentada a um novo grupo. São então inseridos quatro paralelepípedos que atravessam este sólido longitudinalmente (Figura 5.1c). Cada paralelepípedo constitui uma região distinta, e estas quatro regiões são então acrescentadas a um novo grupo. Estes dois últimos grupos são combinados, na ordem em que foram criados, para se realizar a operação de diferença. O resultado desta operação é a criação de quatro furos no sólido obtido pela união dos dois cilindros (Figura 5.1d).

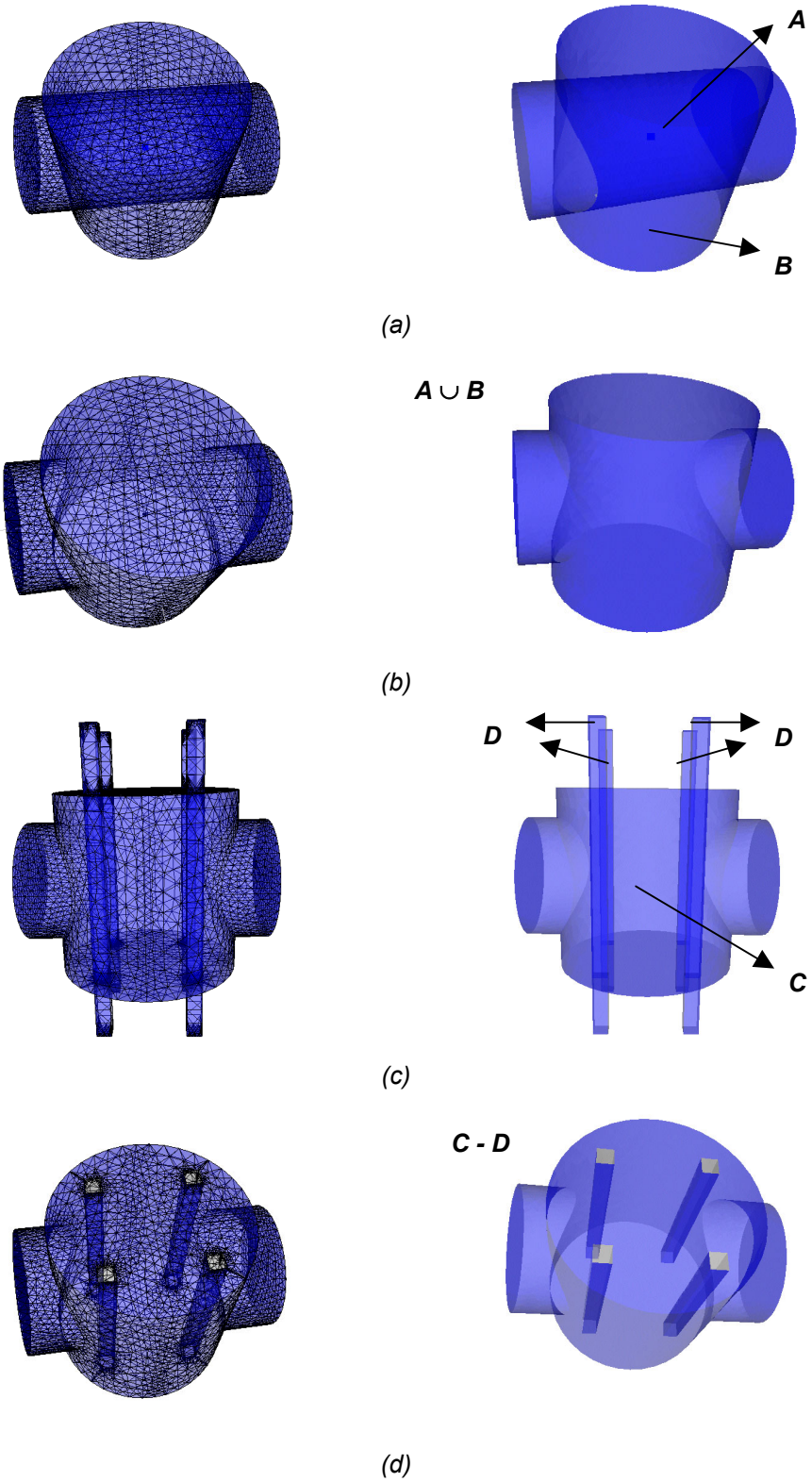


Figura 5.1 – Operações booleanas de união e diferença: a) dois cilindros se interceptando; b) união entre os cilindros; c) quatro paralelepípedos; d) diferença entre o resultado da letra (b) e os paralelepípedos.

A Figura 5.2a mostra novamente dois cilindros que se interceptam. Na Figura 5.2b é mostrada a interseção entre estes cilindros. Na mesma figura, é mostrado um cubo que foi inserido no modelo, em que a maior parte do seu volume é interior ao sólido gerado pela interseção entre os cilindros. A Figura 5.2c mostra a diferença entre os dois sólidos (interseção entre os cilindros *menos* cubo). Pode-se visualizar nesta figura o surgimento de quatro orifícios, ocasionados pelas partes do cubo que eram externas ao outro sólido.

Este exemplo é interessante para ser analisado mais detalhadamente. Chamando-se a região formada pela interseção entre os dois cilindros de *A* e o cubo de *B*, pode-se notar que as faces remanescentes da operação de diferença são do tipo *AoutB* ou *BinA*. Neste exemplo, não existem faces do tipo *INTERS*, e as faces de *B* que eram externas a *A*, ou seja, as faces do tipo *BoutA* representando as quinas do cubo, foram removidas. As faces *BinA* foram mantidas pois fazem parte da fronteira das regiões formadas após a operação de diferença. Vale notar que existe um total de seis regiões formadas no resultado.

As Figuras 5.3a e 5.3b mostram novamente dois cilindros que se interceptam e a união entre estes cilindros. Na Figura 5.3c, um cubo é inserido no modelo de forma a interceptar uma parte do sólido resultante da união entre os cilindros. Na Figura 5.3d, é calculada a interseção entre este sólido e o cubo. Na Figura 5.4a é calculada a diferença entre este sólido e o cubo e na Figura 5.4b é calculada a diferença entre o cubo e este sólido.

As Figuras 5.5a e 5.5b mostram a vista frontal e superior de um cilindro e um prisma se interceptando. A Figura 5.5c mostra a diferença entre o prisma e o cilindro e a Figura 5.5d mostra a diferença entre o cilindro e o prisma.

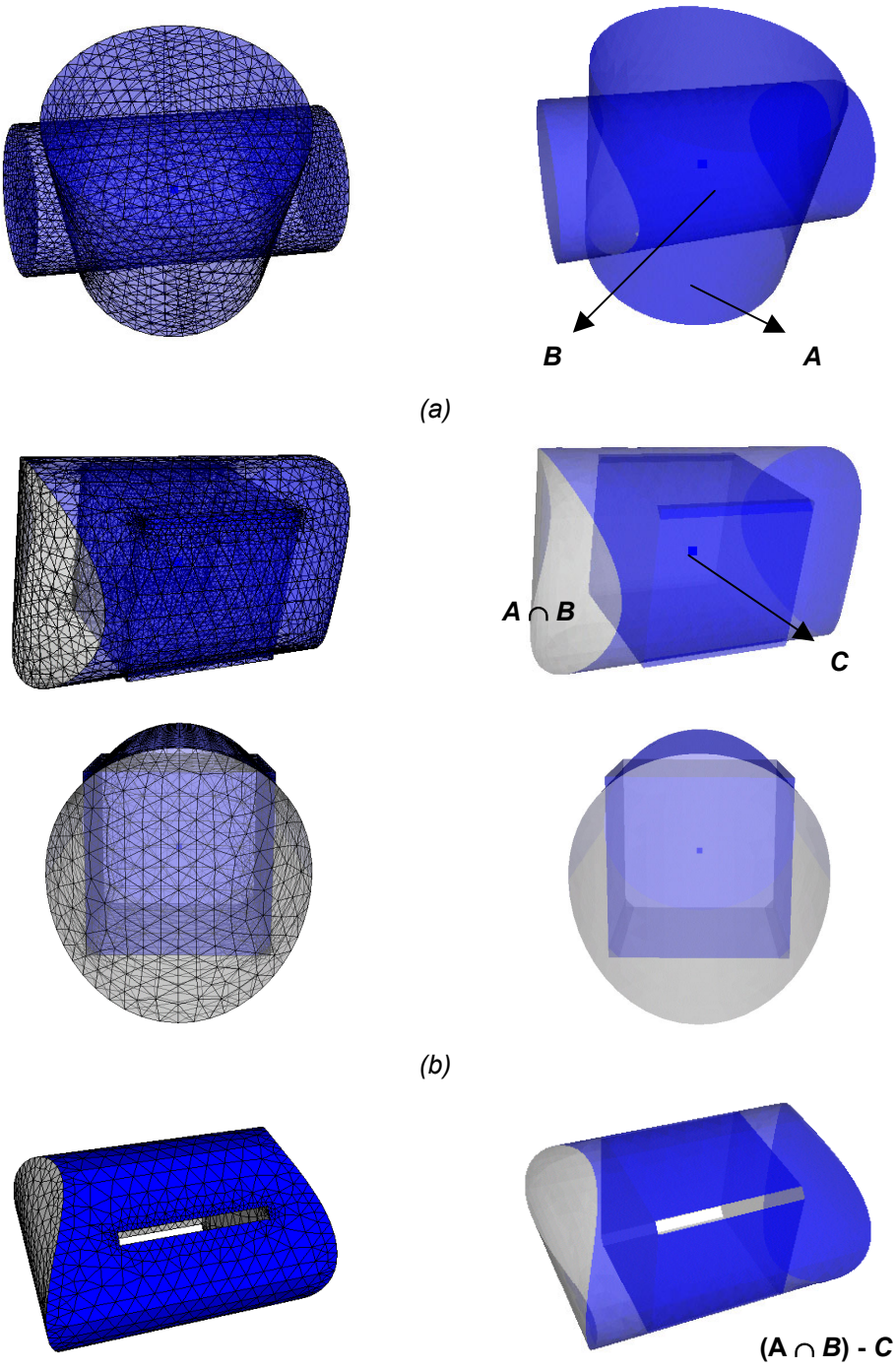
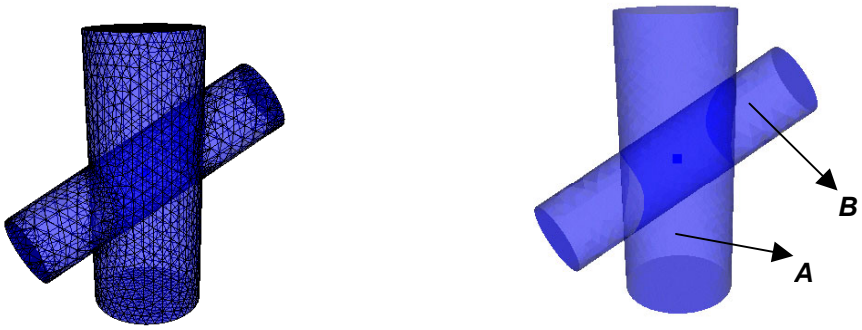
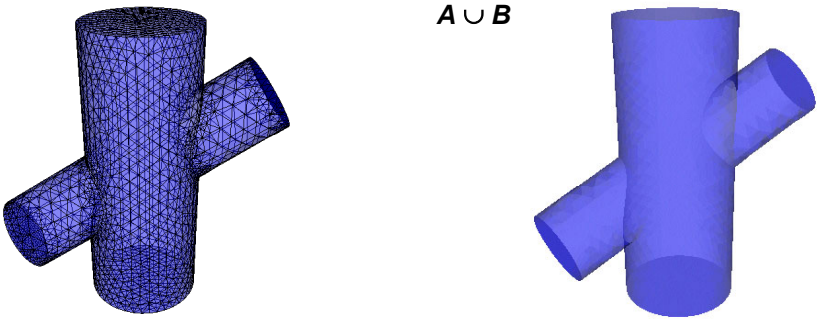


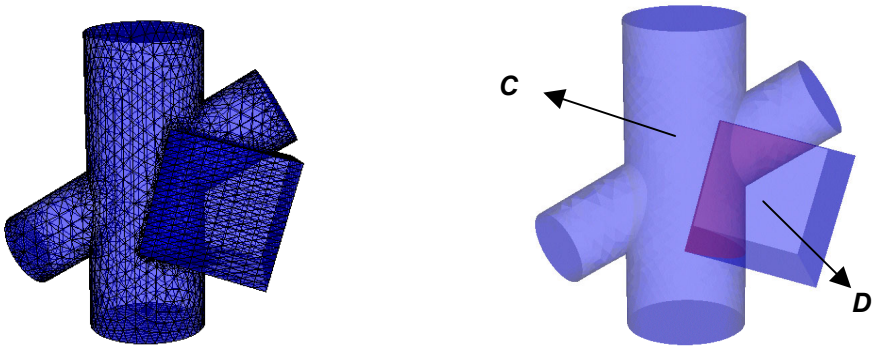
Figura 5.2 – Interseção e diferença: a) dois cilindros; b) interseção entre os cilindros e cubo a ser subtraído; c) diferença entre os dois sólidos da letra (b).



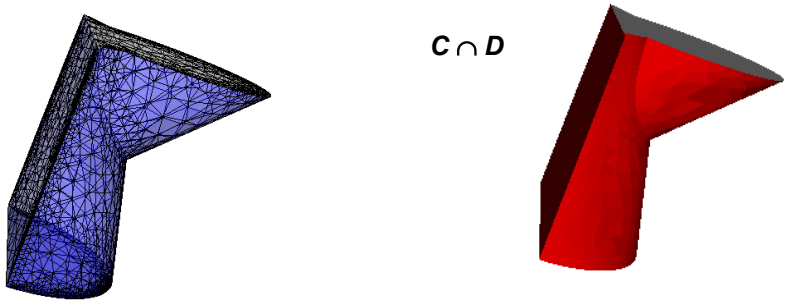
(a)



(b)

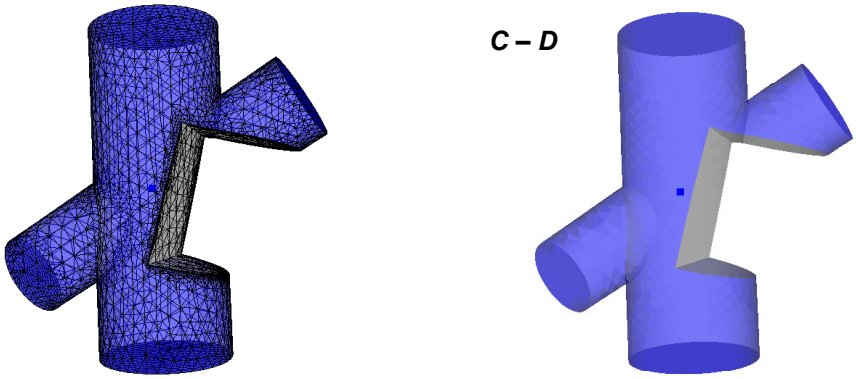


(c)

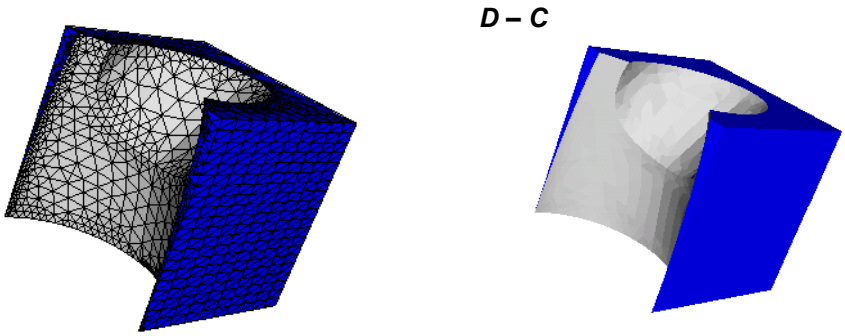


(d)

Figura 5.3 – União e interseção: a) dois cilindros; b) união entre os cilindros; c) cubo interceptando os cilindros; d) interseção entre os dois sólidos.

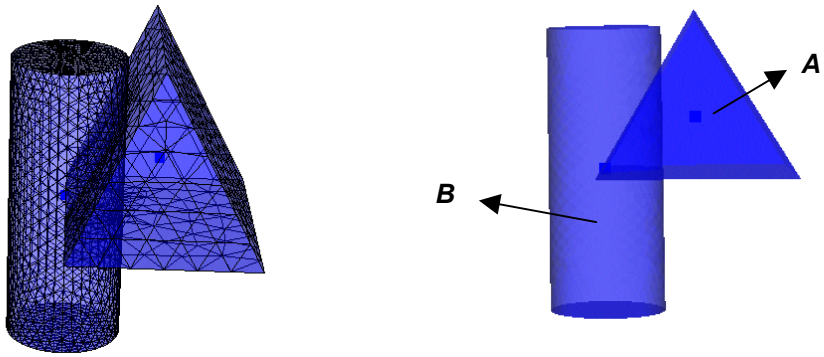


(a)

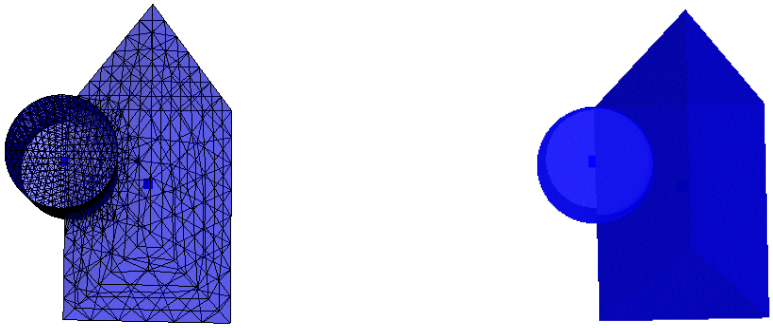


(b)

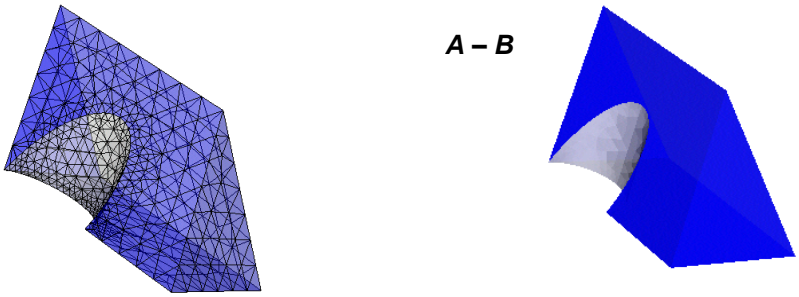
Figura 5.4 – Diferenças entre os sólidos *C* e *D* da Figura 5.3: a) *C* – *D*; b) *D* – *C*.



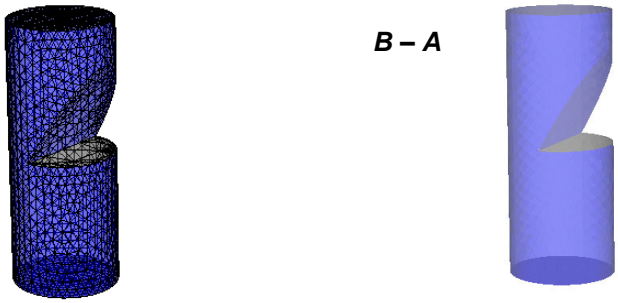
(a)



(b)



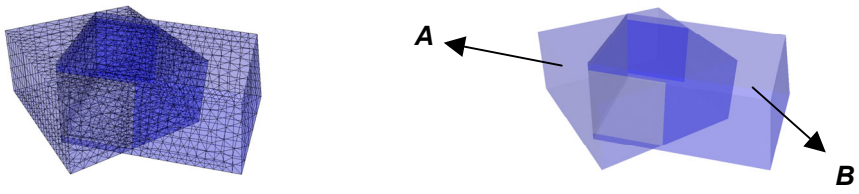
(c)



(d)

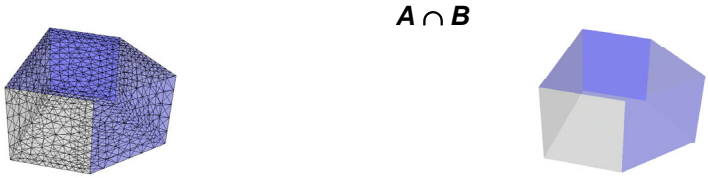
Figura 5.5 – Cilindro e prisma: diferenças: a) vista frontal; b) vista de cima; c) prisma menos cilindro; d) cilindro menos prisma.

A Figura 5.6a mostra dois paralelepípedos se interceptando obliquamente. As Figuras 5.6b, 5.6c, 5.6d e 5.6e mostram respectivamente as operações de união, interseção, diferença *A* – *B* e diferença *B* – *A* entre estes sólidos.

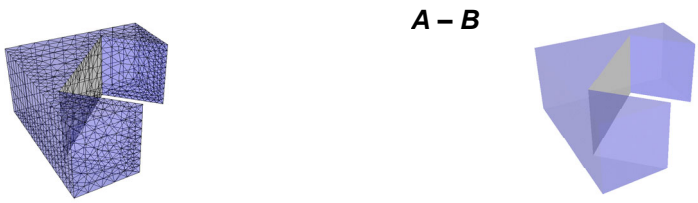


(a)

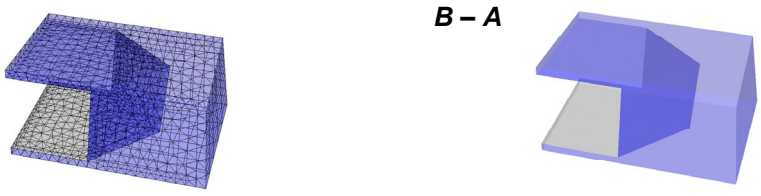
(b)



(c)



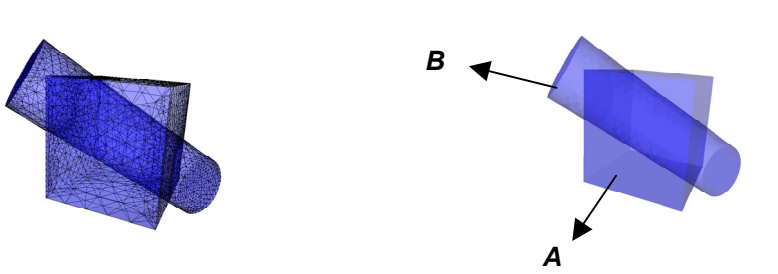
(d)



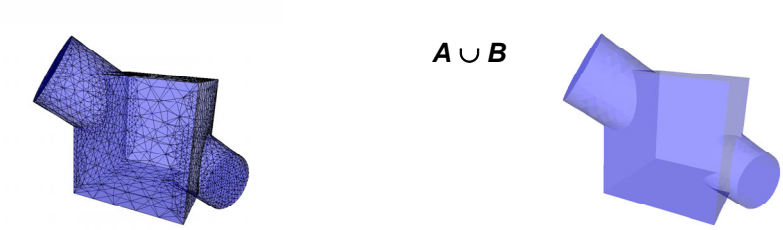
(e)

Figura 5.6 – Dois paralelepípedos se interceptando: a) disposição espacial; b) união; c) interseção; d) diferença *A* – *B*; e) diferença *B* – *A*.

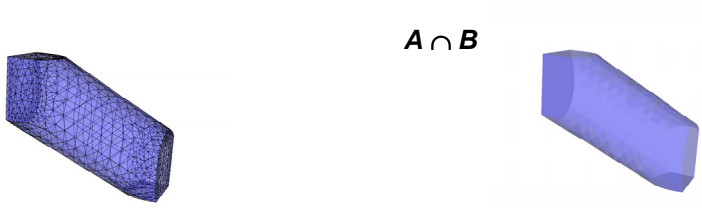
A Figura 5.7a mostra um cilindro interceptando um cubo obliquamente. As Figuras 5.7b, 5.7c, 5.7d e 5.7e mostram respectivamente as operações de união, interseção, diferença *A* – *B* e diferença *B* – *A* entre estes sólidos.



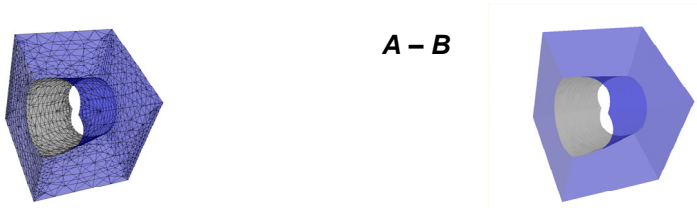
(a)



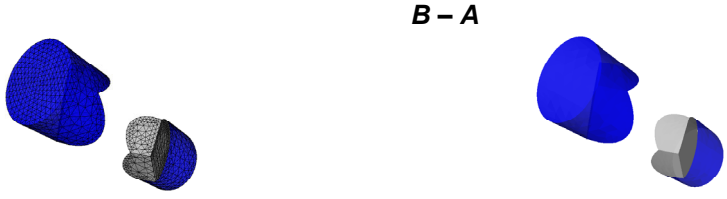
(b)



(c)



(d)



(e)

Figura 5.7 – Cilindro cortando cubo obliquamente: a) disposição espacial; b) união; c) interseção; d) diferença *A* – *B*; e) diferença *B* – *A*.

A Figura 5.8 ilustra um caso interessante para ser analisado. Neste modelo, a detecção de regiões foi realizada após o cálculo das interseções entre as superfícies, resultando em quatro regiões distintas que apenas se tocam em suas fronteiras.

Na Figura 5.8a pode-se vislumbrar o modelo completo, onde se nota dois paralelepípedos maiores com uma cavidade cada um. Nestas cavidades, encaixam-se dois paralelepípedos menores.

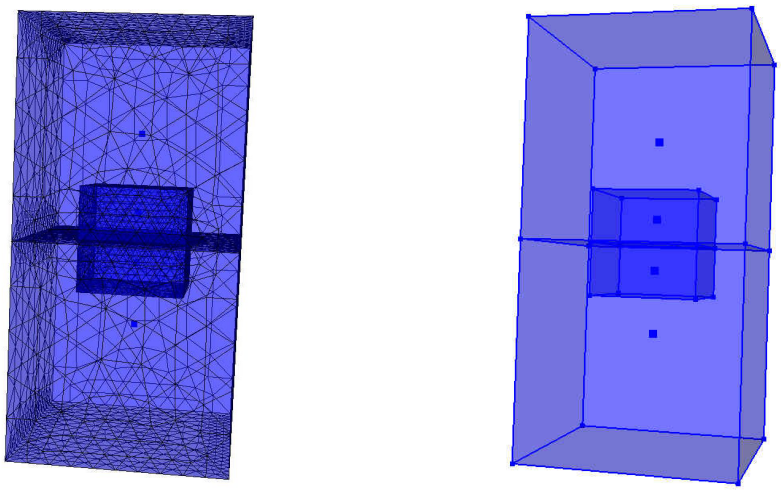
Nas Figuras 5.8b e 5.8c podem-se visualizar os grupos de entidades que serão combinados. Cada grupo é composto por três regiões, sendo que os dois paralelepípedos menores estão contidos em ambos os grupos.

A Figura 5.9 e a Tabela 5.1 mostram a numeração e a classificação de todas as faces presentes no modelo. Algumas informações relevantes associadas a algumas faces também são mostradas na Tabela 5.1. A Figura 5.10 mostra o resultado da aplicação das três operações booleanas sobre os dois grupos.

Na Figura 5.10a, nota-se que, na união entre os grupos, as faces do tipo *INTERS* de número 6, 7, 8, 9, 10, 12, 13, 14, 15 e 16 são removidas por fazerem parte da fronteira de mais de uma região de um mesmo grupo. A face de número 11 é removida por fazer parte da fronteira de uma região de cada grupo, sendo que os vetores normais a esta face em relação a cada região que ela delimita possuem orientações opostas.

Na Figura 5.10b, nota-se que, na interseção entre os grupos, a única face do tipo *INTERS* que é removida é a face de número 11. Isto porque é a única face deste tipo que faz parte da fronteira de mais de uma região dos dois grupos simultaneamente.

Na Figura 5.10c, nota-se que, na diferença entre os grupos (*A* – *B*), as faces do tipo *INTERS* de número 12, 13, 14, 15 e 16 são removidas por fazerem parte da fronteira de mais de uma região do grupo *B*. A face de número 17 (também do tipo *INTERS*) é mantida por fazer parte da fronteira de uma região de cada grupo, sendo que os vetores normais a esta face em relação a cada região que ela delimita possuem orientações opostas.



(a)

(b)

(c)

Figura 5.8 – Quatro paralelepípedos se tocando: a) disposição espacial; b) grupo *A*; c) grupo *B*.

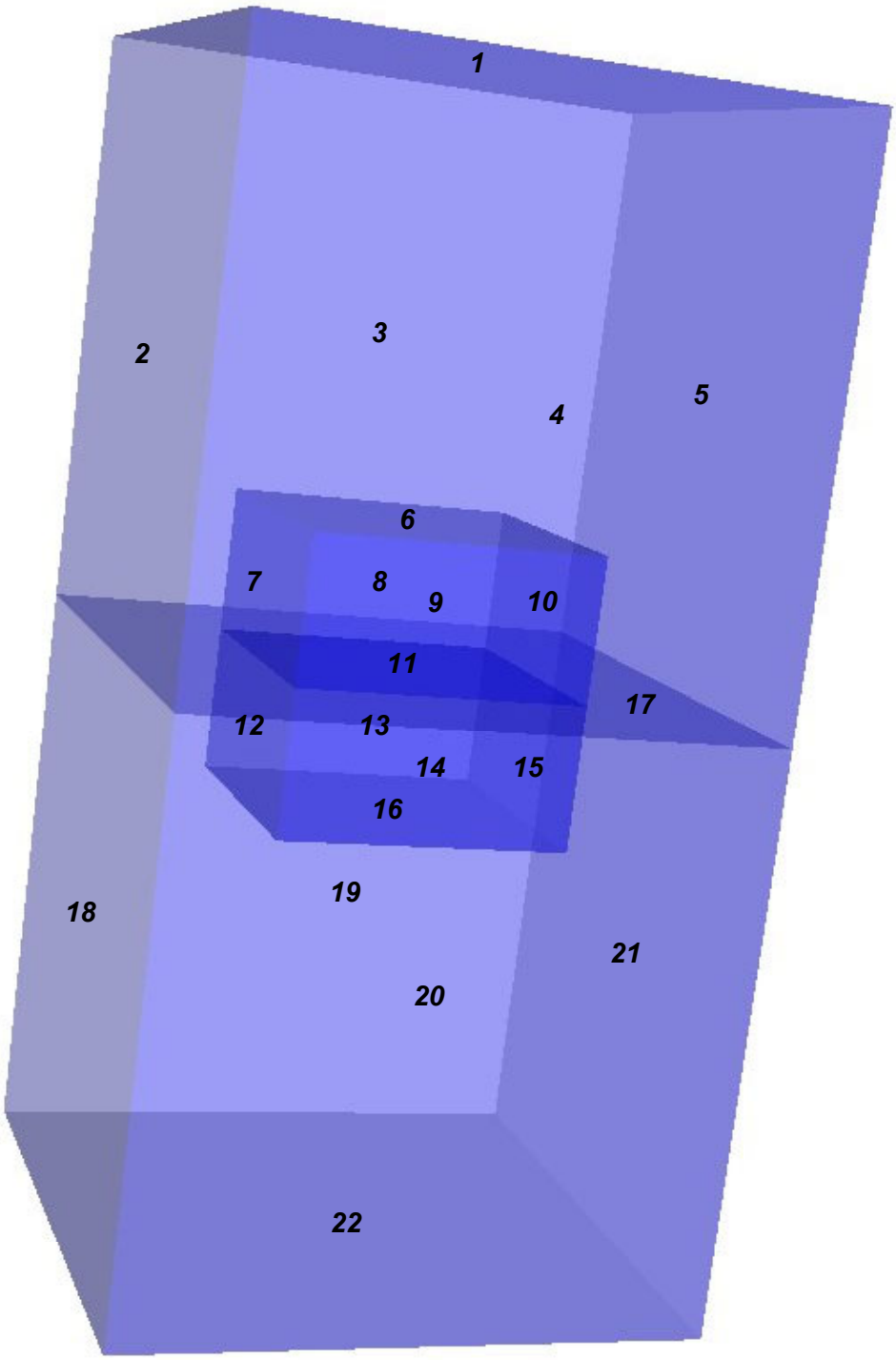


Figura 5.9 – Numeração das faces do modelo da Figura 5.8.

Tabela 5.1 – Classificação das faces do modelo da Figura 5.8.

Número da face	Classificação	Observações
1	*AoutB*	
2	*AoutB*	
3	*AoutB*	
4	*AoutB*	
5	*AoutB*	
6	*INTERS*	2 regiões de *A* e 1 região de *B*
7	*INTERS*	2 regiões de *A* e 1 região de *B*
8	*INTERS*	2 regiões de *A* e 1 região de *B*
9	*INTERS*	2 regiões de *A* e 1 região de *B*
10	*INTERS*	2 regiões de *A* e 1 região de *B*
11	*INTERS*	2 regiões de *A* e 2 regiões de *B*
12	*INTERS*	2 regiões de *B* e 1 região de *A*
13	*INTERS*	2 regiões de *B* e 1 região de *A*
14	*INTERS*	2 regiões de *B* e 1 região de *A*
15	*INTERS*	2 regiões de *B* e 1 região de *A*
16	*INTERS*	2 regiões de *B* e 1 região de *A*
17	*INTERS*	1 região de *A* e 1 região de *B*
18	*BoutA*	
19	*BoutA*	
20	*BoutA*	
21	*BoutA*	
22	*BoutA*	

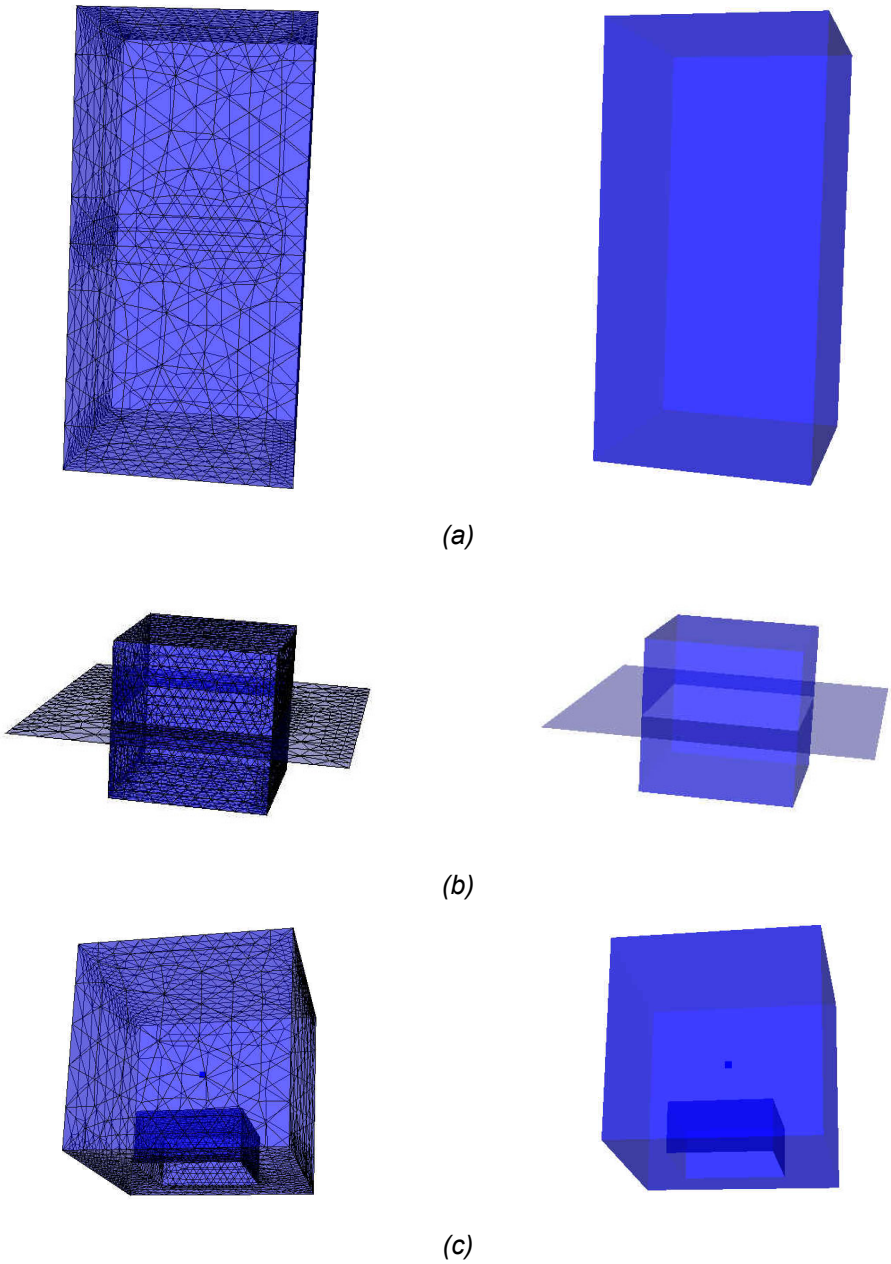


Figura 5.10 – Operações booleanas aplicadas aos grupos da Figura 5.8: a) união; b) interseção; c) diferença.

5.2. Geração de modelos a partir de entidades topológicas com diferentes dimensões

Um dos objetivos do algoritmo proposto neste trabalho é permitir que entidades topológicas com diferentes dimensões possam ser combinadas a partir das operações booleanas. Esta possibilidade garante que o algoritmo seja aplicável em sistemas de modelagem bidimensionais, que trabalham com faces, arestas e vértices, mas não trabalham com regiões.

A Figura 5.11a mostra uma face interceptando um cubo. Após a chamada do algoritmo de interseção do MG, esta face é dividida em cinco novas faces, uma interna e quatro externas ao cubo. Estas cinco faces são acrescentadas a um grupo e a região representada pelo cubo é acrescentada a outro grupo. A Figura 5.11b mostra a interseção entre os dois grupos e a Figura 5.11c mostra a diferença entre o grupo formado pelas faces e o grupo formado pelo cubo.

A diferença entre o cubo e as faces é um exemplo de situação em que as limitações do MG não permitem que o resultado seja representado corretamente. O resultado esperado para esta operação de diferença seria simplesmente o conjunto de faces, arestas e vértices pertencentes à fronteira do cubo. Contudo, este não é o resultado obtido no MG.

Primeiramente, a face interna ao cubo, que seria classificada como *BinA*, seria erradamente mantida no resultado. Isto porque quando fosse chamado o método *regions*, da classe *GeomBoolOp*, que retorna as listas de faces pertencentes à fronteira de cada região formada no resultado (determinadas pela CGC), esta face faria parte da lista de faces da fronteira do cubo, pois, como já foi comentado, ainda não há implementado no MG um recurso para se diferenciar uma face interior a uma região das faces pertencentes à fronteira desta região. Além disso, as quatro arestas que são compartilhadas pelo cubo, pela face interna ao cubo e pelas faces externas ao cubo, por não serem classificadas na versão do algoritmo implementada no MG, já que fazem parte da fronteira de alguma face, também não seriam removidas. Ainda que fosse possível identificar e remover estas arestas, isto faria com que as faces da fronteira do cubo, que possuem estas arestas soltas no seu interior, fossem automaticamente removidas, pois elas são necessárias para a completa descrição geométrica destas faces.

Vale observar também que a união entre os dois grupos não ocasionaria a remoção de nenhuma das entidades presentes no modelo, já que a face interior ao cubo não pertence à fronteira de nenhuma região do seu grupo (seção 3.5.3.1).

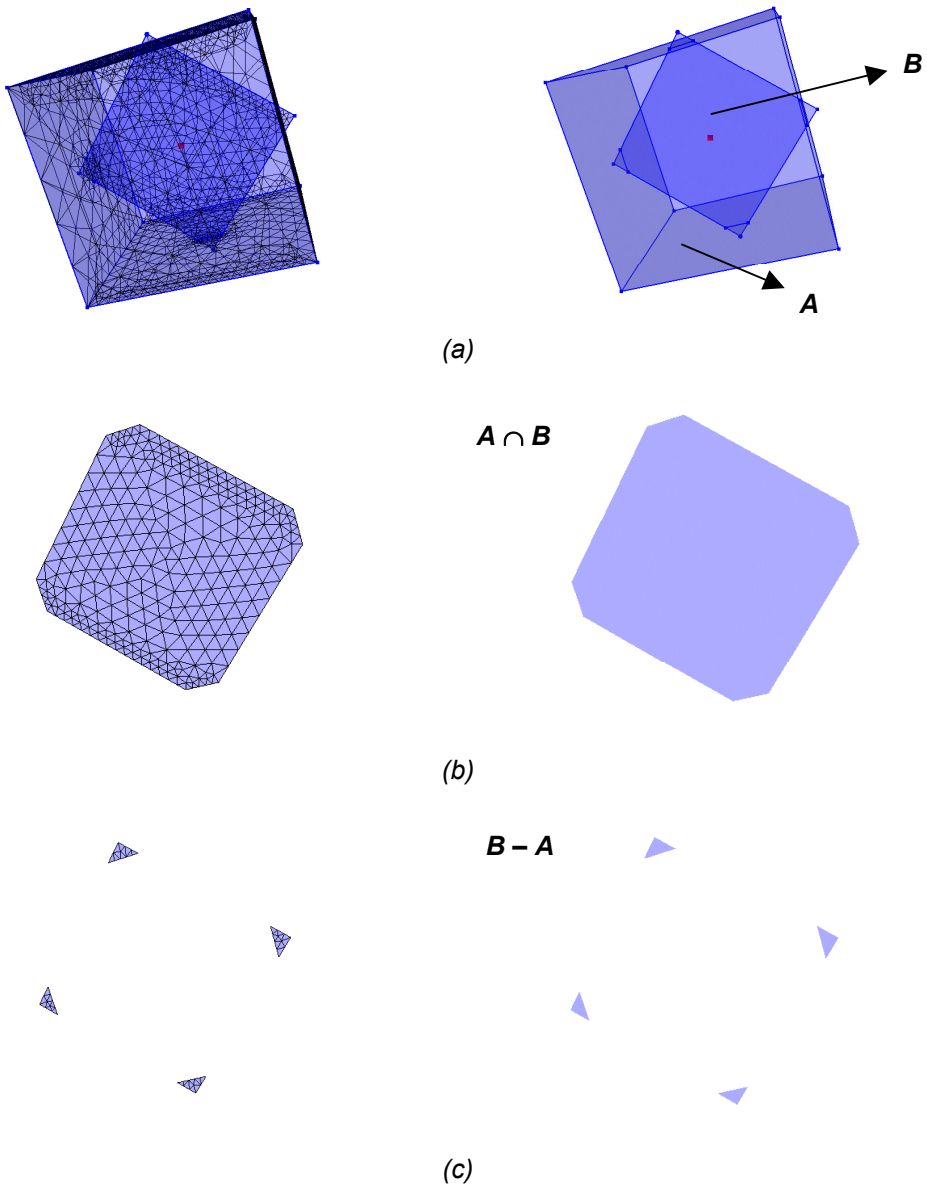


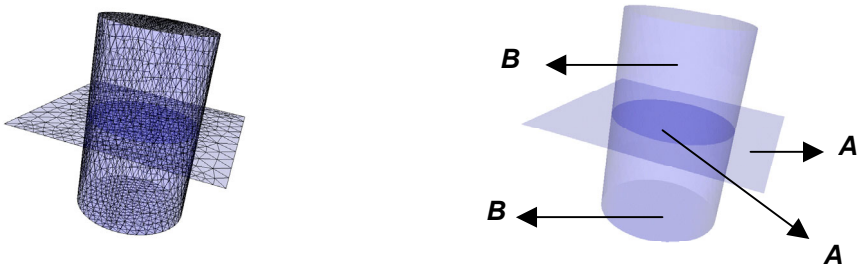
Figura 5.11 – Operações booleanas com entidades de diferentes dimensões.

A Figura 5.12a mostra um retalho de superfície plano cortando um cilindro. Neste caso, o reconhecimento de regiões foi realizado após o cálculo das interseções entre superfícies, de modo que o cilindro completo passou a ser representado por duas regiões distintas. Estas duas regiões formam um grupo, e os dois novos retalhos de superfície que compõem o retalho de superfície plano original formam o outro grupo. Na Figura 5.12b, é mostrada a união entre os grupos, com a remoção do retalho de superfície que separa as duas regiões que compõem o cilindro. Na Figura 5.12c, é mostrada a interseção entre os grupos, em que o único retalho de superfície remanescente é aquele que foi removido na operação de união. Na Figura 5.12d, é feita a diferença *A* – *B* entre os dois grupos, e na Figura 5.12e é feita a diferença *B* – *A* entre os dois grupos.

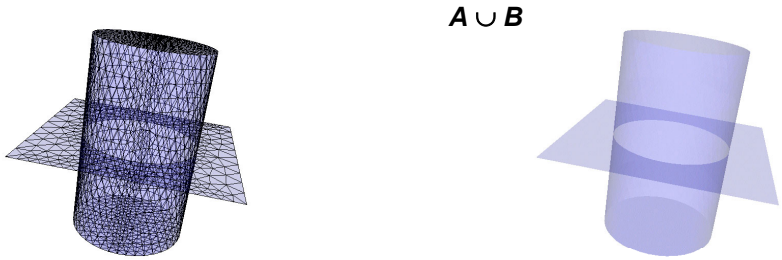
Este caso é relativamente simples. Basta notar que o retalho de superfície que divide o cilindro em duas regiões é compartilhado por ambos os grupos, o que faz com que seja classificado como *INTERS*. Como ele pertence à fronteira de mais de uma região de um dos grupos, ele é removido na união.

Na interseção, este retalho é o único que permanece no resultado, pois todos os outros são do tipo *AoutB* ou *BoutA*.

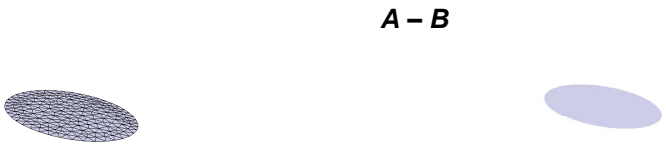
Na diferença *A* – *B*, este retalho é removido pois pertence à fronteira de mais de uma região do grupo *B*. Já na diferença *B* – *A*, ele é removido por pertencer à fronteira de mais de uma região de *B* e de nenhuma região de *A* (seção 3.5.3.3).



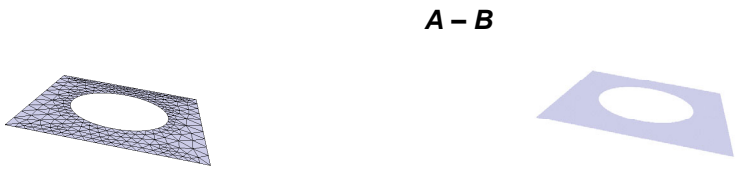
(a)



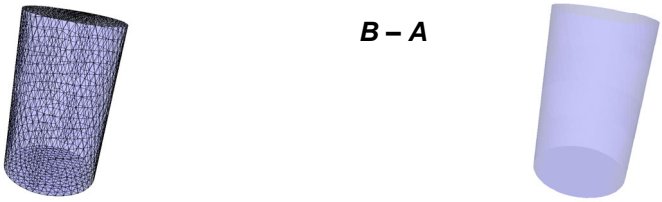
(b)



(c)



(d)



(e)

Figura 5.12 – Plano cortando cilindro: a) disposição espacial; b) união; c) interseção; d) diferença *A* – *B*; e) diferença *B* – *A*.

5.3. Geração de modelos *non-manifold*

Nesta seção são apresentados casos de modelos em que as informações de entrada já representam situações *non-manifold* ou modelos *manifold* em que o resultado da aplicação de alguma operação booleana é *non-manifold*.

A Figura 5.13a mostra dois cubos, sendo um deles totalmente interno ao outro. Um arame parte de uma aresta da fronteira do cubo mais externo, propagando-se internamente até um ponto que é interno aos dois cubos. Após as interseções entre curvas e superfícies terem sido calculadas, este arame é dividido em dois novos arames, um deles interno somente ao cubo mais externo, e outro interno aos dois cubos.

Escolhe-se um grupo *A* formado pelo cubo externo e pelos dois arames. Em seguida, o cubo interno é acrescentado a um grupo *B*. A Figura 5.13b mostra a união entre os cubos. Todas as faces do cubo interno, que são do tipo *BinA* e pertencem à fronteira de uma região do grupo *B*, são removidas. Os dois arames, por sua vez, permanecem no resultado. A Figura 5.13c mostra a interseção entre os grupos, em que todas as faces do cubo externo, que são do tipo *AoutB*, são removidas, juntamente com o arame que é externo ao grupo *B*. A Figura 5.13d mostra a diferença *A – B*, em que todas as faces dos dois cubos são mantidas, porém as faces do cubo interno passam agora a representar parte da fronteira do cubo externo, constituindo um vazio interno a este cubo. O arame interno aos dois cubos é removido, e o outro é mantido.

A Figura 5.14a mostra dois cubos dispostos igualmente aos cubos da figura anterior. Contudo, o cubo interno contém uma aresta solta em uma das faces da sua fronteira. Na Figura 5.14b, a união entre os grupos é mostrada. Todas as faces do cubo interno, que são do tipo *BinA* e fazem parte da fronteira de uma região do grupo *B*, são removidas. Como a face que continha a aresta solta no seu interior foi removida, então esta aresta também é removida. Na Figura 5.14c, é mostrada a interseção entre os grupos. Apenas as faces do cubo externo são removidas.

As duas diferenças não são mostradas pelo seguinte motivo: a diferença *A – B* deveria manter todas as faces das fronteiras dos dois cubos, mas removendo a aresta solta no interior da face do cubo interno. Isto porque as faces do cubo interno representariam agora um vazio interno dentro do cubo externo. Mas, no MG, se esta aresta for removida, necessariamente a face que a

contém também é removida. A diferença *B* – *A* remove todas as entidades dos dois grupos.

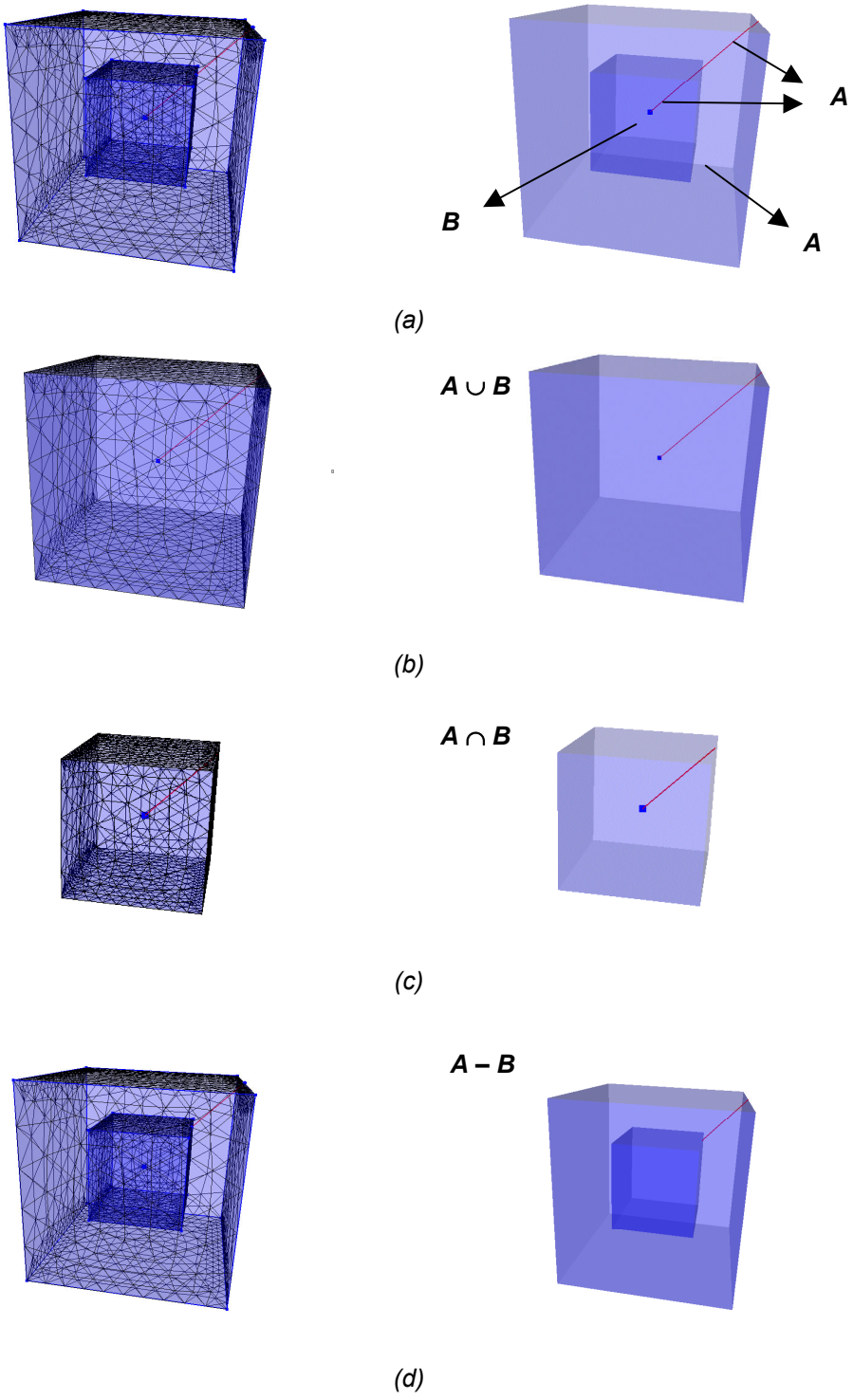
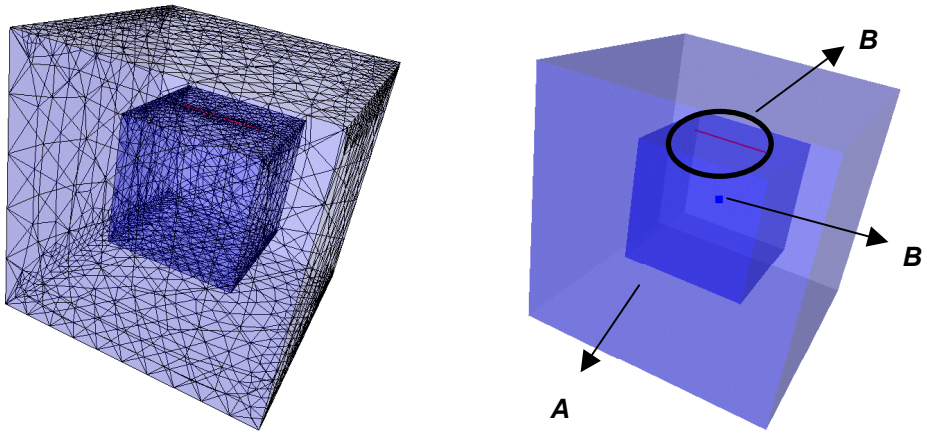
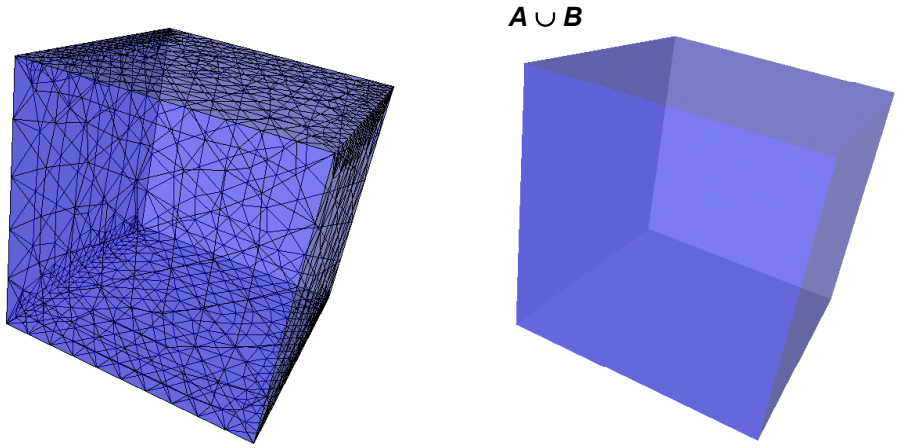


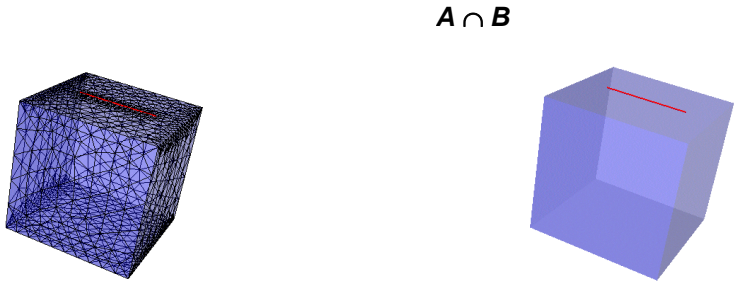
Figura 5.13 – Região com arame interno sendo combinada com outra região: a) disposição espacial; b) união; c) interseção; d) diferença *A* - *B*.



(a)



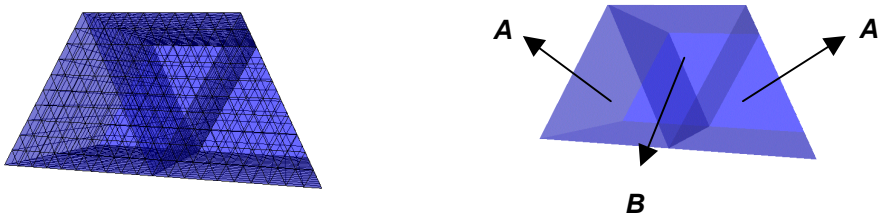
(b)



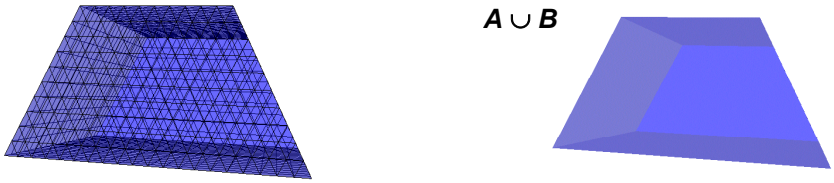
(c)

Figura 5.14 – Aresta solta no interior de uma face: a) cubo com outro cubo interno em que uma das faces contém uma aresta solta no seu interior; b) união entre os grupos; c) interseção entre os grupos.

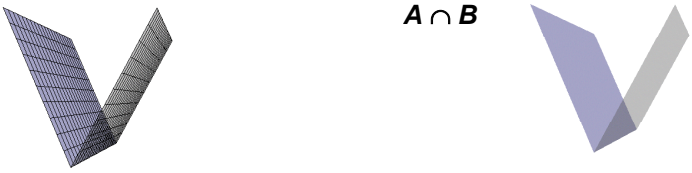
A Figura 5.15a mostra um sólido composto por três regiões, com várias arestas *non-manifold* sendo compartilhadas por mais de duas faces cada uma. As duas regiões que não possuem uma face em comum compõem o grupo *A*, e a região entre essas duas compõe o grupo *B*. A união entre os dois grupos, mostrada na Figura 5.15b, promove a remoção das faces compartilhadas pelos dois grupos. Essas faces são do tipo *INTERS* e cada uma faz parte da fronteira de uma região de cada grupo. Os vetores normais destas faces em relação às regiões que as compartilham possuem orientações opostas, o que faz com que elas sejam removidas. A Figura 5.15c mostra a interseção entre os grupos, em que as únicas faces remanescentes são aquelas que foram removidas na união, ou seja, as faces do tipo *INTERS*, já que todas as outras faces são do tipo *AoutB* ou *BoutA*. A Figura 5.15d mostra a diferença *A* – *B*, que mantém somente as duas regiões do grupo *A*, e a Figura 5.15e mostra a diferença *B* – *A*, que mantém somente a única região do grupo *B*.



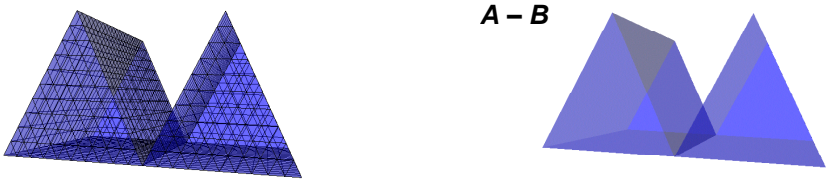
(a)



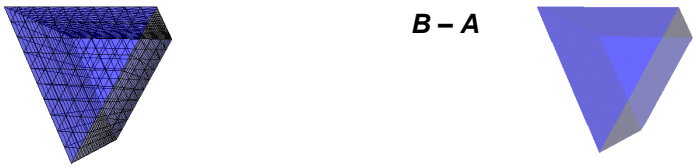
(b)



(c)



(d)



(e)

Figura 5.15 – Três prismas: a) disposição espacial; b) união; c) interseção; d) diferença *A* – *B*; e) diferença *B* – *A*.

5.4. Regularização do resultado

Esta seção contém exemplos de operações booleanas regularizadas entre grupos de entidades topológicas. A regularização é um procedimento bastante utilizado quando se deseja obter resultados constituídos somente por volumes preenchíveis.

A Figura 5.16a mostra dois paralelepípedos se interceptando. A Figura 5.16b mostra a união entre estes sólidos. A Figura 5.16c mostra um novo paralelepípedo que foi inserido no modelo interceptando o sólido obtido na figura anterior. A interseção entre estes dois novos sólidos é mostrada na Figura 5.16d, onde se pode notar que uma face que era comum a ambos os sólidos ficou pendente no resultado. O processo de regularização é então aplicado ao resultado da interseção, restando apenas o sólido *manifold* da Figura 5.16e.

A Figura 5.17a mostra dois cubos compartilhando uma face. Além disso, existem duas faces pendentes, uma internamente a um dos cubos e outra internamente ao outro cubo. O grupo *A* é então formado por um dos cubos e pelas duas faces pendentes. O grupo *B* é formado pelo outro cubo. Na Figura 5.17b, é mostrada a união entre os grupos. A face compartilhada, que pertence a uma região de cada grupo e possui vetores normais com sentidos opostos em relações às duas regiões, é removida. As faces pendentes são mantidas, uma delas por ser do tipo *AoutB* e a outra por ser do tipo *AinB* e não pertencer à fronteira de nenhuma região do grupo *A*. Na Figura 5.17c, é mostrada a interseção entre os grupos. Apenas a face compartilhada (do tipo *INTERS*) e a face pendente do tipo *AinB* são mantidas, pois todas as outras são externas. Na Figura 5.17d, pode-se visualizar a diferença *A – B*, em que são mantidas todas as faces do tipo *AoutB* (que formam a fronteira do cubo) e a face pendente internamente ao cubo do grupo *A*.

A aplicação da regularização nos resultados da união, interseção e diferença entre os grupos só produz o efeito desejado no caso da interseção, removendo todas as entidades remanescentes, pois nenhuma região é formada. No caso da união e da diferença, ocorre um problema já mencionado que é a manutenção das estruturas internas no resultado da regularização, pois como o reconhecimento de regiões é feito através da RED na biblioteca CGC, esta retorna ao MG a informação de que estas estruturas internas fazem parte da fronteira das regiões formadas.

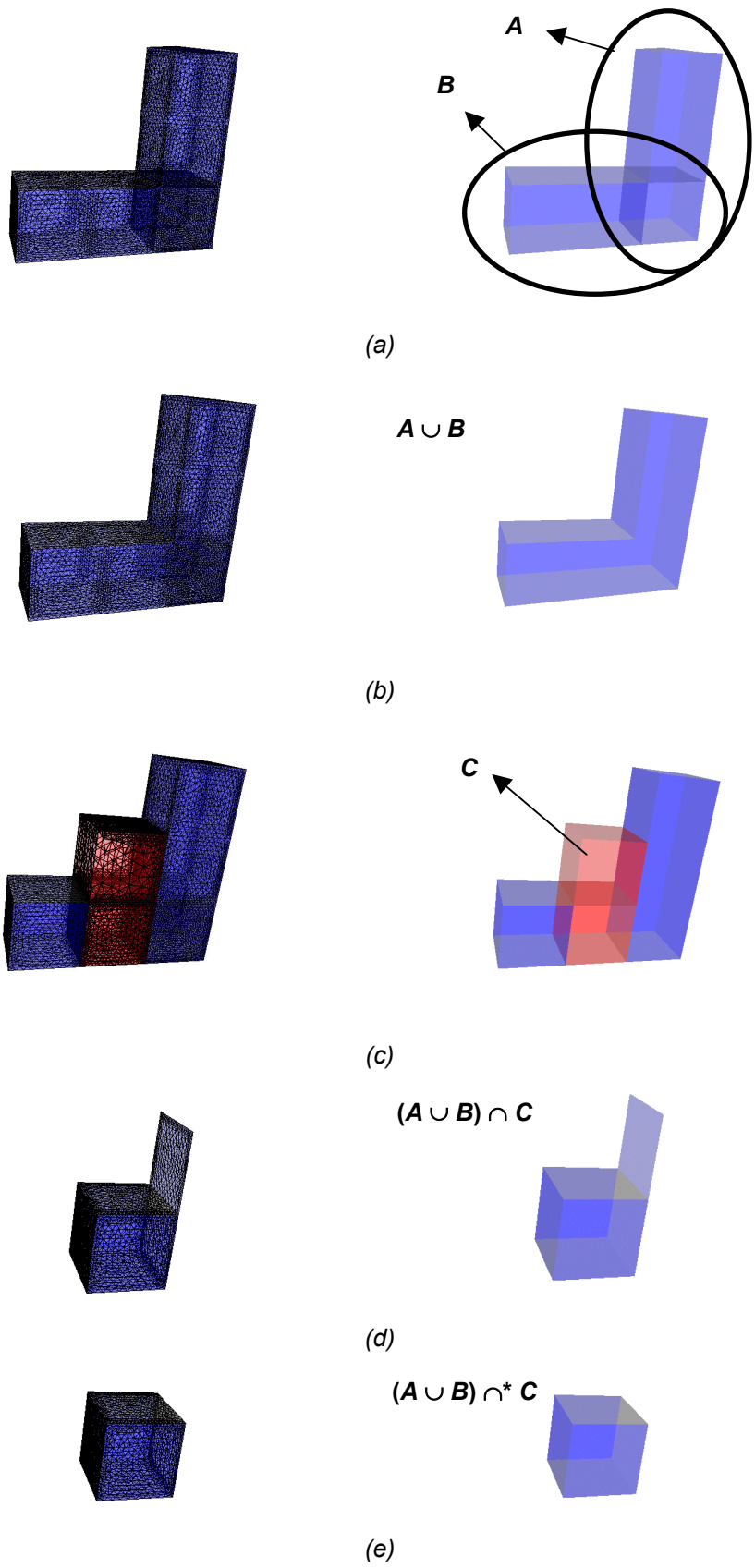
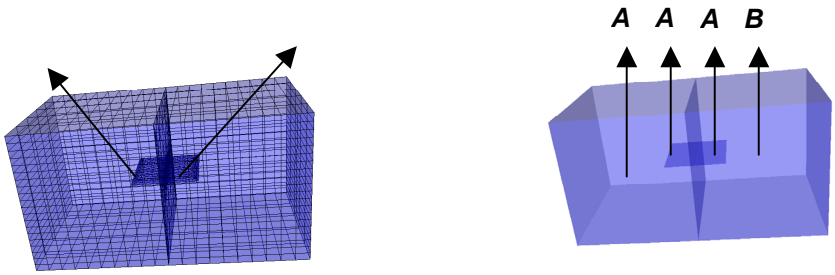
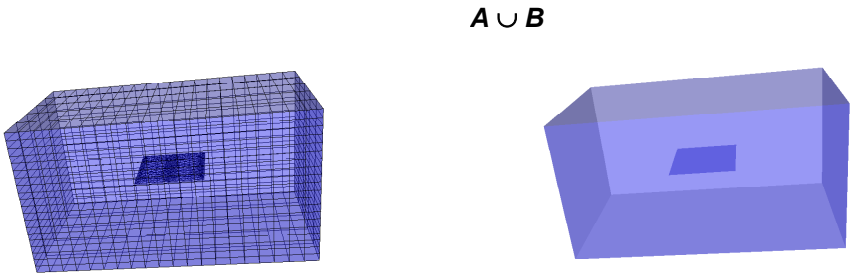


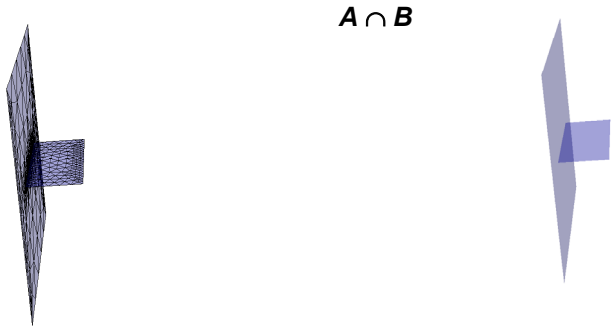
Figura 5.16 – União, interseção e regularização do resultado: a) dois paralelepípedos (A e *B*); b) *A* ∪ *B*; c) novo paralelepípedo (*C*) inserido no modelo; d) (*A* ∪ *B*) ∩ *C*; e) (*A* ∪ *B*) ∩* *C*.



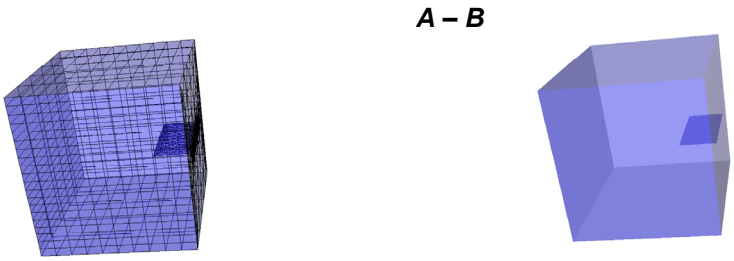
(a)



(b)



(c)



(d)

Figura 5.17 – Sólido com estrutura interna - problemas na regularização: a) dois cubos com faces internas; b) união; c) interseção; d) diferença.

5.5. Problemas com o pré-processamento dos parâmetros de entrada

Os modelos que foram apresentados nas seções anteriores podem ser considerados relativamente simples. No entanto, ilustram de forma satisfatória a aplicação do algoritmo proposto nos grupos de entidades topológicas.

A criação de modelos mais complexos com aplicação prática no ramo da engenharia, como peças mecânicas e máquinas industriais, pode ser realizada através de uma seqüência de operações booleanas em objetos geometricamente mais simples. O resultado de uma operação booleana costuma ser usado como parâmetro de entrada na aplicação de uma nova operação booleana, que irá combinar este objeto com outro para formar um objeto mais complexo.

Contudo, para que todo o processo de criação de um modelo complexo seja realizado de forma eficaz, é necessário que o pré-processamento dos parâmetros de entrada seja efetuado antes da aplicação de uma nova operação booleana sobre os objetos de interesse. Isto significa que todas as regiões devem ser reconhecidas e que todas as interseções entre curvas e superfícies devem ser calculadas.

Como um dos objetivos da existência das operações booleanas em um sistema de modelagem é precisamente facilitar a criação de objetos mais complexos a partir de primitivas mais simples, torna-se conveniente que o usuário do sistema possa instanciar quaisquer primitivas, com qualquer localização espacial, para serem combinadas por meio destas operações. Não cabe ao usuário preocupar-se com detalhes relativos à maneira como os dados de entrada são tratados no pré-processamento do modelo.

Logo, os algoritmos utilizados no pré-processamento do modelo devem ser robustos o suficiente para suportar quaisquer parâmetros de entrada que o usuário venha a definir, e gerar um novo modelo consistente topologicamente para que as operações boolenas possam ser aplicadas corretamente.

Porém, o tratamento de casos particulares, como a superposição de faces ou duas regiões que se tocam de forma que uma aresta da fronteira de uma delas é interior a uma face da outra, não são triviais. Problemas práticos em que estes tipos de situações ocorrem não são incomuns. A superposição de faces, por exemplo, ocorre com bastante freqüência em problemas que utilizam primitivas que contenham faces planas.

Algumas tentativas de se criar objetos complexos no MG utilizando somente as operações booleanas foram realizadas, porém sempre havia alguma etapa de modelagem em que o pré-processamento dos parâmetros de entrada era realizado de forma incorreta ou causava problemas de execução do programa, como invasão de memória. Dois casos serão mostrados a seguir.

Na Figura 5.18, pode-se visualizar um cilindro cortando um cubo. A criação automática de um cilindro no MG (ferramenta *Wizard*) é realizada de tal forma que as bases do cilindro e a sua superfície lateral são divididas em quatro faces cada uma, como se pode notar na própria figura. A particularidade deste caso está no fato de que as arestas da fronteira das faces laterais do cilindro tocam o interior de quatro faces do cubo. Quando é requerido o cálculo das interseções entre as superfícies presentes, um problema é detectado.

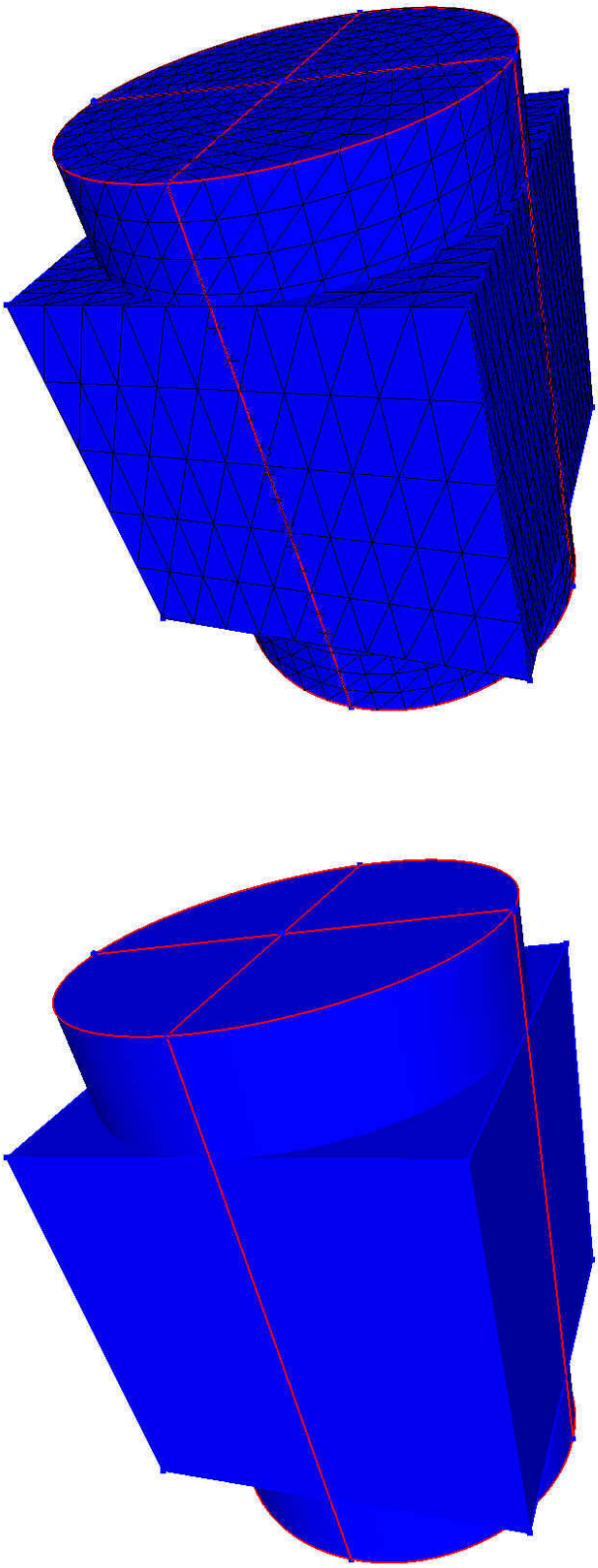


Figura 5.18 – Problemas no cálculo da interseção de superfícies.

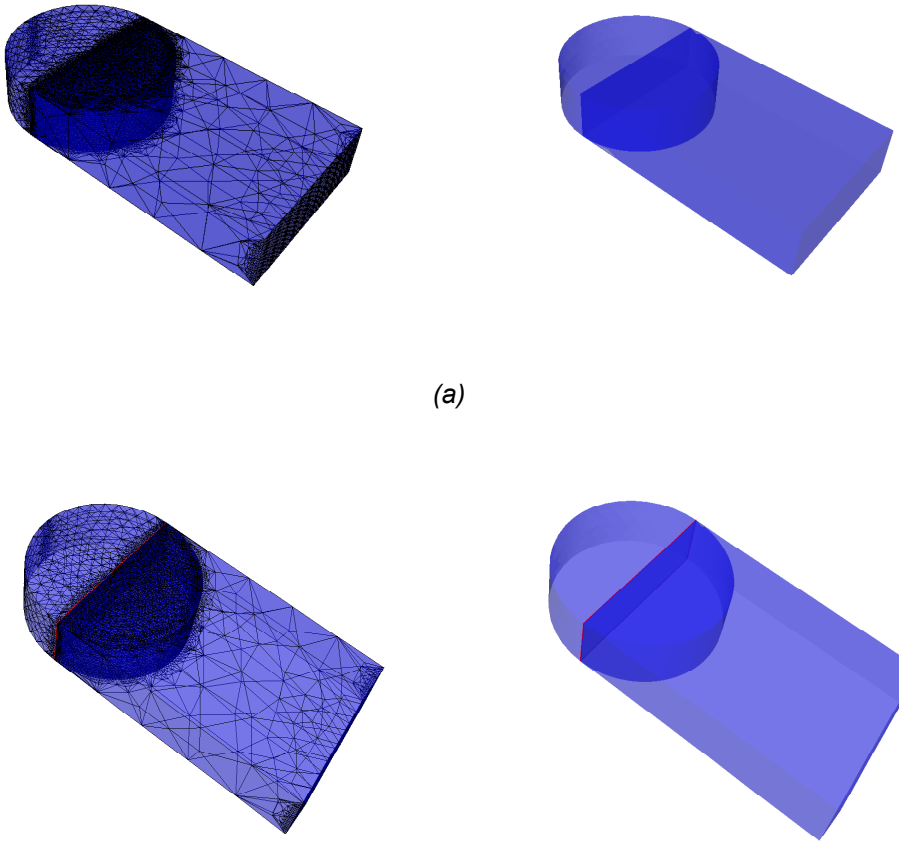
A Figura 5.19a mostra uma cilindro e um paralelepípedo que se interceptam. Pode-se notar que as bases do cilindro e duas faces do paralelepípedo possuem partes superpostas. Além disso, duas arestas de cada base do cilindro são coincidentes com quatro arestas do paralelepípedo e duas arestas das faces laterais do cilindro são coincidentes com duas arestas do paralelepípedo (Figura 5.19b). Nesta figura, a interseção entre as superfícies já foi calculada.

A Figura 5.20 mostra um problema que surgiu após o cálculo das interseções. Novos retalhos de superfície foram criados, como pode ser visualizado na Figura 5.20a, ocupando a área que era comum às faces superpostas do cilindro e do paralelepípedo. Contudo, estes novos retalhos de superfície foram criados apenas como subdivisões das faces do paralelepípedo que se superpunham às faces do cilindro, como pode ser visto na Figura 5.20b. Os retalhos de superfície que representavam as faces das bases do cilindro continuaram existindo, ou seja, continuaram existindo faces superpostas após a interseção, ao invés de existir uma única face que fosse comum aos dois sólidos (Figuras 5.20c e 5.20d). O motivo disto é a amarração das informações geométricas de cada retalho de superfície com as arestas dos retalhos de superfícies originais que existiam antes da interseção ser calculada. Por exemplo, a descrição geométrica da face destacada na Figura 5.20b depende das arestas do paralelepípedo que servem de diâmetro para a base do cilindro. A criação de uma face comum aos dois sólidos requereria a manutenção das informações geométricas necessárias para caracterizar os retalhos de superfícies de ambos os sólidos. As malhas destas três faces superpostas podem ser visualizadas na Figura 5.21.

A união entre os dois sólidos é mostrada na Figura 5.22a. No caso, todas as faces superpostas são mantidas. Isto decorre do seguinte fato: como os vértices das fronteiras destas faces coincidem, um ponto interior a cada uma delas é testado em relação às regiões do outro grupo. Como estes pontos não são classificados como sendo interiores, então eles são automaticamente classificados como exteriores. Porém, o reconhecimento de regiões falha. Só se conseguiu fazer com que uma região fechada fosse detectada quando a face do paralelepípedo que estava superposta às faces do cilindro foi removida manualmente.

Inseriu-se então no modelo um novo cilindro, com raio da base menor que o anterior, e cujos eixos de simetria coincidiam com o cilindro anterior (Figura 5.22b). A inserção deste cilindro teve como objetivo criar um furo no sólido

existente. Este furo é criado fazendo-se a diferença entre o sólido existente e o novo cilindro inserido (Figura 5.23a). Na Figura 5.23b, é mostrada em detalhe a má qualidade das malhas geradas nas proximidades das arestas das faces laterais do cilindro mais externo.



(a)

(b)

Figura 5.19 – Cilindro e paralelepípedo se interceptando: a) disposição espacial; b) arestas coincidentes.

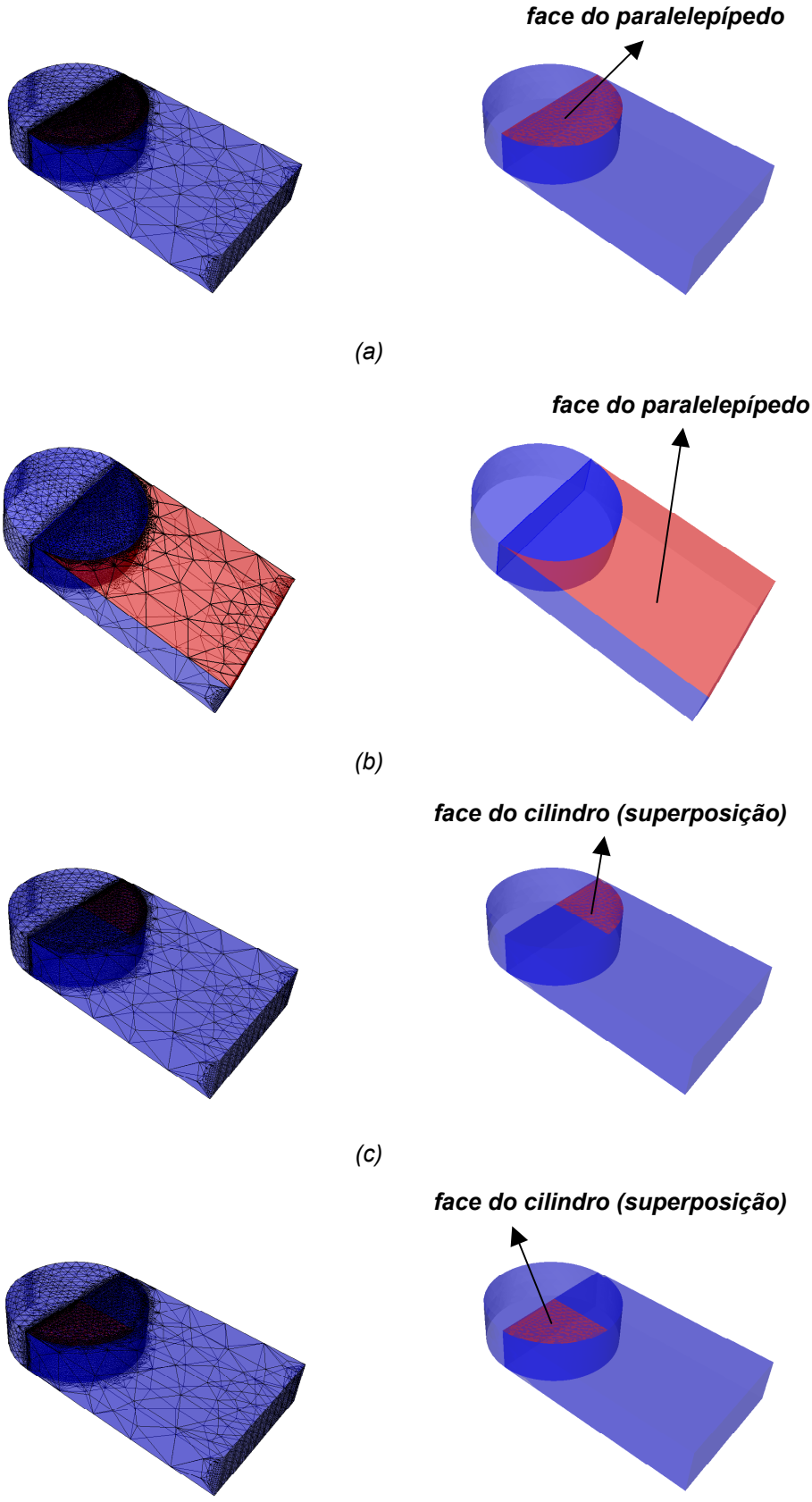


Figura 5.20 – Faces superpostas: a) e b) subdivisões da face original do paralelepípedo; c) e d) faces da base do cilindro.

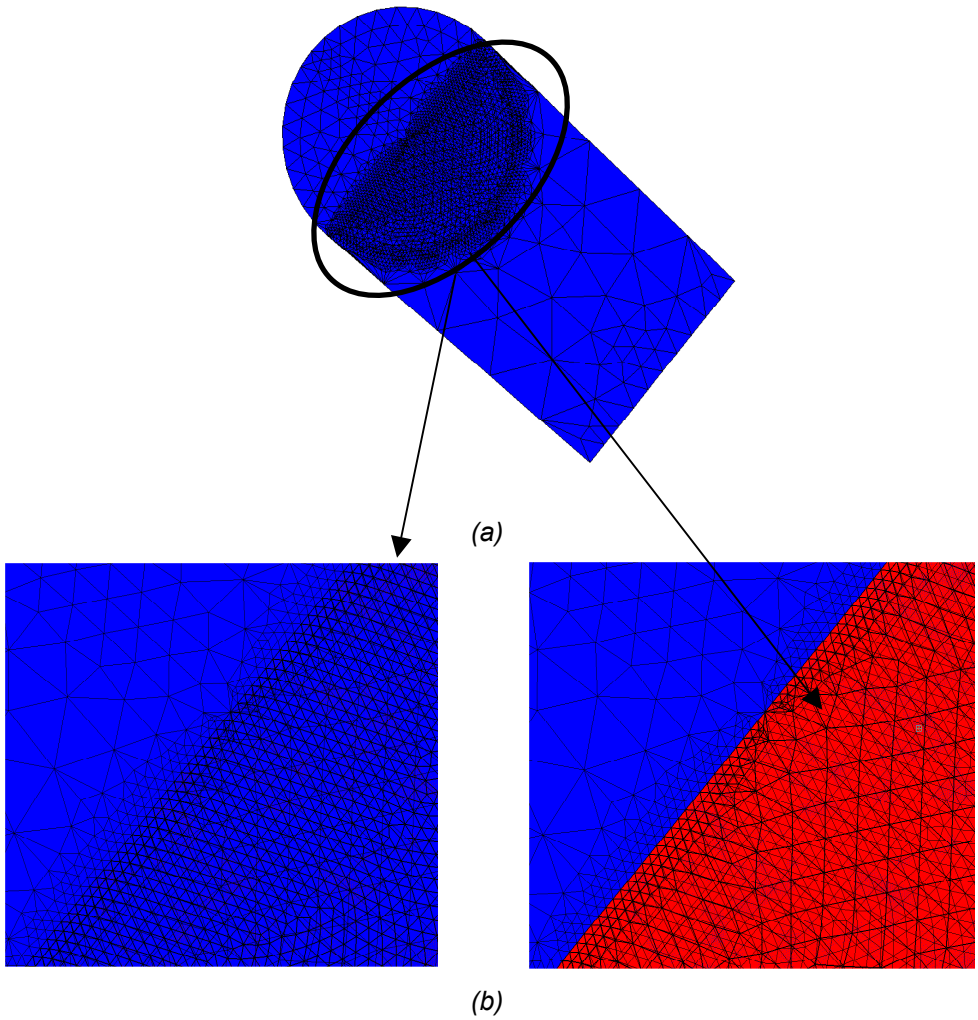
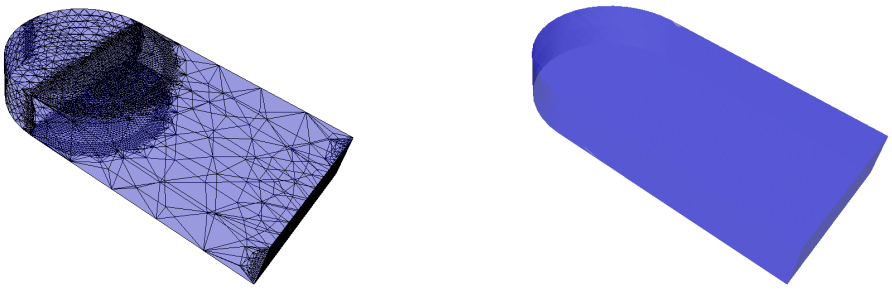
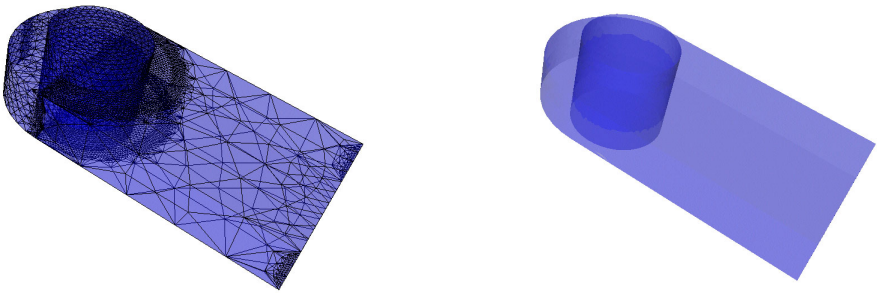


Figura 5.21 – Malhas das faces superpostas dos dois sólidos: a) malha relativa à face do paralelepípedo; b) malha relativa às faces do cilindro.

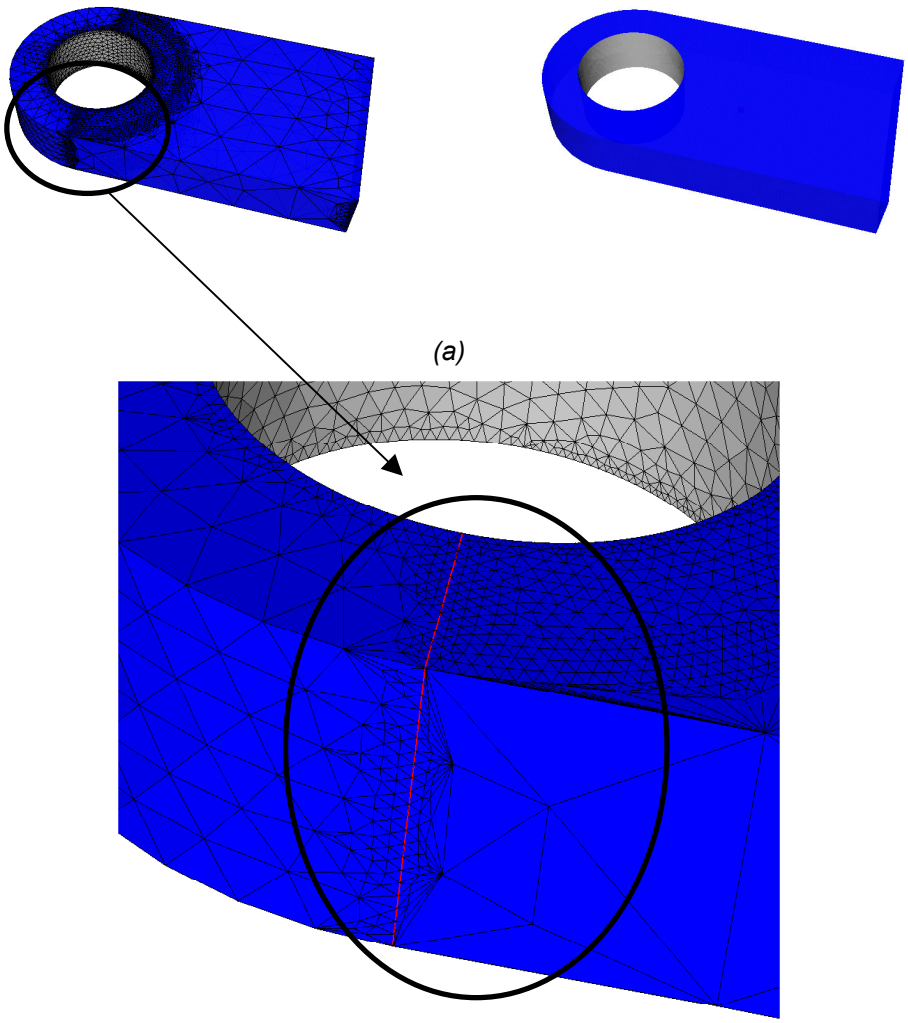


(a)



(b)

Figura 5.22 – União e inserção de um novo cilindro.



(a)

(b)

Figura 5.23 – Diferença e má qualidade das malhas geradas após a interseção de superfícies.

5.6. Ordem de grandeza do tempo de execução do algoritmo

Como foi dito no capítulo anterior, o algoritmo em si divide-se basicamente em três etapas: armazenamento das informações dos grupos, classificação das entidades topológicas e aplicação da operação booleana desejada.

É preciso lembrar, no entanto, que o pré-processamento das informações de entrada também é uma etapa necessária. Esta também é uma etapa que pode demandar um tempo relativamente grande, dependendo do número de retalhos de superfície que se interceptam no modelo.

Em relação ao algoritmo descrito, também foi dito que a etapa determinante no tempo de execução do mesmo é a de classificação das entidades topológicas, pois é a única etapa que depende de algoritmos geométricos (detecção de ponto em região e de detecção de ponto sobre retalho de superfície). É importante ressaltar que, nesta etapa, o número de vértices de um grupo contidos no interior da *bounding box* de alguma região do outro grupo é o fator mais importante para se determinar o tempo que o algoritmo irá demandar. Isto porque vértices deste tipo são aqueles que irão ser testados contra as regiões do outro grupo. Obviamente, se o número de regiões de cada grupo for grande, isto também irá ocasionar uma maior lentidão do algoritmo, já que os vértices de um grupo são testados contra todas as regiões do outro grupo, pelo menos até encontrarem uma dentro da qual se localizem.

Apenas para se ter uma noção da ordem de grandeza do tempo demandado pelo algoritmo para aplicar operações booleanas num modelo, dois exemplos serão mostrados. A máquina onde estes exemplos foram gerados foi um PENTIUM 4, com uma CPU de 1,80 GHz de velocidade e 512 Mb de memória RAM.

O primeiro exemplo é o modelo da Figura 5.7. As entidades topológicas presentes no modelo após o cálculo das interseções são as seguintes:

- 30 vértices
- 57 arestas
- 32 faces
- 2 regiões

Como este modelo não possui arestas nem vértices soltos ou pendentes, apenas as faces são classificadas. A classificação das faces é a seguinte:

- 6 faces do tipo *AoutB*
- 16 faces do tipo *BoutA*
- 6 faces do tipo *AinB*
- 4 faces do tipo *BinA*

Ou seja, um total de dez faces são interiores, e conseqüentemente seus vértices deverão ser testados contra a região do outro grupo. Os vértices das faces exteriores ou são coincidentes com vértices de outras faces ou estão fora da *bounding box* da região do outro grupo, logo não demandam muito tempo para serem classificados.

A operação booleana a ser realizada não influencia de modo significativo no tempo total de execução do algoritmo. Neste caso, o tempo total foi de doze segundos.

O segundo exemplo é mostrado na Figura 5.24a. Um dos grupos é formado somente por um cubo. O outro grupo é formado por quatro cilindros que interceptam o cubo. As entidades topológicas presentes no modelo após o cálculo das interseções são as seguintes:

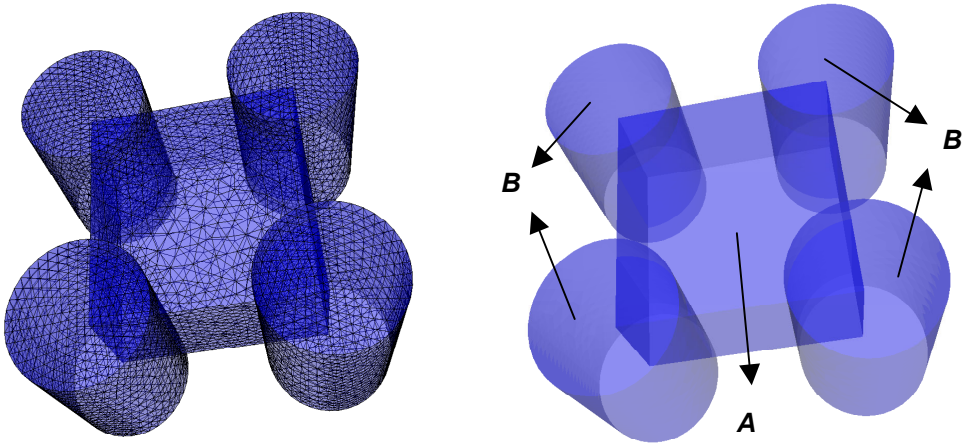
- 64 vértices
- 132 arestas
- 78 faces
- 5 regiões

Este modelo também não possui vértices ou arestas pendentes ou soltos, então apenas as faces são classificadas. A classificação é a seguinte:

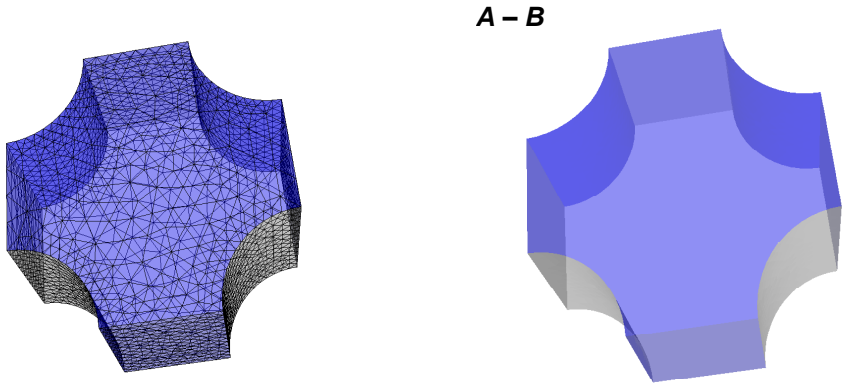
- 6 faces do tipo *AoutB*
- 52 faces do tipo *BoutA*
- 16 faces do tipo *AinB*
- 4 faces do tipo *BinA*

Neste caso, existe um total de 20 faces interiores. Novamente, os vértices das faces exteriores ou são coincidentes com vértices de outras faces ou estão fora da *bounding box* das regiões do outro grupo, logo não demandam muito tempo para serem classificados.

Neste caso, foi realizada uma operação de diferença entre os dois grupos (Figura 5.24b). O tempo total gasto pelo algoritmo foi de dezessete segundos.



(a)



(b)

Figura 5.24 – Quatro cilindros interceptando um cubo: a) disposição espacial; b) diferença.

6
Conclusões

Este trabalho apresenta um algoritmo para operações booleanas em ambientes de modelagem baseados em representação de fronteira (B-Rep) e com domínio representacional *non-manifold*. A idéia principal é permitir a aplicação das operações booleanas em entidades topológicas de qualquer dimensão, em qualquer número e com qualquer descrição geométrica.

Operações booleanas constituem o paradigma de modelagem de sistemas com representação CSG. Estes sistemas permitem que se combinem objetos sólidos primitivos por meio das operações booleanas para formar objetos sólidos mais complexos. Contudo, as primitivas sólidas possuem formas pré-definidas e sua caracterização geométrica completa depende apenas da instanciação de certos parâmetros pelo usuário do sistema.

Se os objetos sólidos passíveis de serem representados só puderem ser *manifold*, então o conjunto das operações booleanas não é fechado. Isto porque mesmo quando os objetos que estão sendo combinados são *manifold*, não há garantia alguma de que o resultado da operação booleana também será *manifold*. Para que as operações booleanas possam ser inseridas em sistemas de modelagem com domínios *manifold*, é necessário que se adote algum procedimento que evite o subconjunto de resultados *non-manifold* que podem surgir após a aplicação de uma operação booleana sobre dois sólidos. Uma maneira é simplesmente restringir os resultados que podem ser gerados, ignorando-se aqueles que não podem ser representados pelo sistema. Este procedimento pode ser eficiente, mas é também um tanto limitante. Outra maneira é a criação de artifícios que simulem uma topologia *manifold* em resultados que sejam *non-manifold*, como a duplicação de arestas ou vértices *non-manifold*. Este tipo de procedimento é menos restritivo que o anterior, mas pode ocasionar problemas de consistência entre topologia e geometria.

Requicha e Voelcker [11] propuseram uma maneira de se evitar um subconjunto dos resultados *non-manifold* que podem surgir pela aplicação das operações booleanas entre sólidos. São as operações booleanas regularizadas, que permitem que somente resultados com volumes preenchíveis sejam representados. Neste processo, entidades topológicas de mais baixa dimensão

que se encontram soltas ou pendentes são desconsideradas no resultado final. Mesmo assim, o processo de regularização ainda não garante o fechamento das operações booleanas, pois ainda é possível se obter resultados *non-manifold* a partir da aplicação de uma operação booleana regularizada entre dois sólidos.

A utilização de sistemas de modelagem com domínios *non-manifold* permite que se represente qualquer resultado de uma operação booleana, fazendo assim com que elas passem a formar um conjunto fechado de operações. A vantagem da modelagem *non-manifold* está no fato de que ela agrega todos os tipos de representação que a antecederam historicamente, ou seja, a modelagem por arames, a modelagem por superfícies e a modelagem de sólidos.

Em ambientes de modelagem que se baseiam em representação de fronteira (B-Rep), o uso da topologia como elemento-chave é justificado por três razões primordiais: estabilidade do sistema de modelagem, erros numéricos passíveis de existirem na representação geométrica de curvas e superfícies e separação das informações topológicas e geométricas como base para uma maior organização e sistematização na implementação do sistema de modelagem. A topologia de adjacência é a mais utilizada no campo da modelagem geométrica. Ela permite que se tenha acesso à conectividade dos elementos topológicos presentes.

Baseando-se nestas informações, buscou-se elaborar um algoritmo que permitisse a utilização de uma ferramenta típica de sistemas de modelagem CSG em sistemas de modelagem B-Rep. Buscou-se também estabelecer uma maneira completa e consistente de se tratar as operações booleanas em domínios *non-manifold*. Além disso, uma das principais premissas do algoritmo proposto é a utilização de informações essencialmente topológicas, evitando-se ao máximo a dependência da geometria dos modelos durante os passos de modelagem.

Para que as operações booleanas pudessem ser aplicadas não somente entre dois sólidos, mas entre conjuntos de entidades topológicas quaisquer, criou-se o conceito de grupo, que passou a representar um conjunto de entidades que necessitam ser agrupadas para que possam receber o mesmo tratamento, de acordo com o objetivo do usuário do sistema de modelagem. Um grupo pode conter um número qualquer de entidades topológicas, e estas podem ter qualquer dimensão. Deste modo, as operações booleanas não ficam restritas a apenas dois sólidos.

Neste contexto, criou-se uma metodologia de abordagem em relação às estruturas pendentes ou soltas internamente às regiões. Como em diversos problemas de engenharia estruturas internas representam restrições ao domínio destes problemas, influenciando, por exemplo, na geração de malhas de elementos finitos superficiais ou sólidas, buscou-se padronizar um procedimento que respeitasse a existência destas estruturas.

Para que o algoritmo proposto pudesse ser implementado e testado, utilizou-se um modelador geométrico tridimensional pré-existente, o MG, que atende aos requisitos mencionados quanto à forma de representação por fronteira (B-Rep) e domínio representacional *non-manifold*. Este modelador já possuía algumas ferramentas necessárias para garantir que as condições de aplicabilidade do algoritmo pudessem ser respeitadas, como a interseção entre superfícies com geometria arbitrária e o reconhecimento automático de multi-regiões. O enfoque híbrido do modelador, que possui uma estrutura de dados própria que se intercomunica com a estrutura *Radial Edge* de Weiler [17] implementada na biblioteca de classes CGC, permite que se usufrua das funcionalidades desta biblioteca apenas nas etapas de modelagem em que a costura topológica do modelo se faz necessária, como no reconhecimento de regiões.

Contudo, algumas limitações da estrutura de dados do modelador não permitem que as operações booleanas possam ser implementadas exatamente como proposto, além de não permitirem que se possam testar alguns casos patológicos, como a superposição de superfícies.

6.1. Principais contribuições

Operações booleanas são ferramentas de modelagem bastante exploradas. Podem-se encontrar diversos programas e aplicativos que utilizam as operações booleanas como uma forma de se combinar sólidos mais simples para gerar sólidos mais complexos. Mesmo em sistemas de modelagem B-Rep, nos quais a implementação destas operações torna-se algo um pouco mais complicado, as operações booleanas podem existir como forma de modelagem complementar [52].

Este trabalho reúne os conceitos fundamentais de modelagem *non-manifold*, estruturas de dados topológicas e operações booleanas entre sólidos para apresentar um algoritmo robusto e eficiente para se combinar entidades

topológicas quaisquer por meio das operações booleanas. As principais contribuições deste trabalho são:

- Apresentação de um algoritmo para realização de operações booleanas em ambientes de modelagem B-Rep com domínio *non-manifold*.
- Sistematização do tratamento de entidades soltas ou pendentes interiormente a faces e regiões.
- Possibilidade de se aplicar o algoritmo a grupos de entidades topológicas, onde cada grupo pode conter um número qualquer de entidades.
- Tratamento diferenciado para faces, arestas e vértices, garantindo assim a possibilidade de se gerar resultados com qualquer dimensão.
- Utilização de informações essencialmente topológicas no algoritmo, o que garante a sua aplicabilidade em modelos com geometria qualquer.
- Inclusão de mais uma ferramenta de modelagem dentro do modelador MG, tornando-o mais completo e eficiente.

6.2. Sugestões para trabalhos futuros

O algoritmo proposto pode ser considerado como genérico no sentido de não depender do tipo de estrutura de dados utilizado pelo sistema de modelagem para poder ser descrito. Os elementos topológicos que são manipulados - *vértices*, *arestas* e *faces* - são elementos utilizados por qualquer sistema de modelagem baseado em representação de fronteira.

Para que o algoritmo possa ser aplicado, é necessário um pré-processamento das informações de entrada, de forma que as condições de aplicabilidade do algoritmo sejam respeitadas. Isto significa que o sistema de modelagem que utilizar este algoritmo deve possuir ferramentas para o cálculo de interseções entre curvas e superfícies e para o reconhecimento de regiões.

O algoritmo consiste basicamente no armazenamento, classificação e consulta a entidades topológicas destes três tipos, para determinar quais devem permanecer e quais devem ser eliminadas no resultado de uma operação booleana qualquer.

Destas três etapas mais gerais, a única que utiliza informações geométricas do modelo é a classificação das entidades topológicas. Esta etapa requer a chamada de funções para se determinar se um ponto é interior ou

exterior a uma região, e se um ponto pertence ou não a um retalho de superfície. Para se manter a generalidade do algoritmo, entretanto, é necessário que estas funções aceitem como parâmetros de entrada retalhos de superfície com geometria arbitrária (incluindo os que formam as fronteiras das regiões).

Tendo em vista estas informações, uma sugestão seria implementar o algoritmo de operações booleanas proposto como uma biblioteca de classes extensível. Esta biblioteca poderia conter uma única classe, *BoolOp*, responsável pela aplicação das operações booleanas sobre as entidades topológicas. Poderia haver um método responsável pelo armazenamento das informações dos grupos, que seriam passadas para este método, por exemplo, como vetores de informações geométricas sobre as entidades topológicas. A descrição geométrica deveria ser comum para o cliente da biblioteca e para a biblioteca. Isto poderia ser feito através de uma interface comum, baseada, por exemplo, numa representação NURBS [27,28]. A biblioteca poderia conter uma estrutura de dados topológica própria, que utilizasse as informações geométricas recebidas para criar os elementos topológicos. Além disso, métodos *virtuais puros* (métodos de uma classe que não são implementados dentro do escopo desta classe, mas que necessariamente devem ser implementados dentro do escopo das classes derivadas desta) responsáveis pelos algoritmos geométricos de detecção de ponto em região e de detecção de ponto sobre face poderiam ser declarados, para que o cliente da biblioteca pudesse implementar uma classe derivada da classe *BoolOp* onde estes métodos seriam implementados. Os métodos responsáveis pelas operações booleanas retornariam vetores com as informações geométricas das entidades que deveriam ser removidas.

Poderiam ser pesquisados algoritmos de detecção de ponto em região e de ponto sobre retalho de superfície mais eficientes para serem usados na etapa de classificação das entidades topológicas do algoritmo proposto neste trabalho. Dependendo da complexidade do modelo em termos de número de entidades topológicas presentes, esta etapa pode demandar um tempo excessivo.

Também podem ser feitas algumas sugestões relativas ao modelador MG. Quanto à sua estrutura de dados, novas classes e métodos poderiam ser criados para que as informações provindas da biblioteca CGC pudessem ser devidamente interpretadas.

Uma classe representando uma casca poderia ser criada, de tal forma que os objetos desta classe possuíssem uma lista de listas de referências para entidades topológicas quaisquer (objetos da classe *Topology*), cada lista representando uma casca de uma determinada região. Poderia haver uma

referência para uma *parentShell*, semelhante ao *parentLoop* existente para um *patch2d*, representando uma eventual casca mais externa que contém aquela casca no seu interior. Para manter a consistência, os objetos da classe *patch3d* possuiriam uma lista de cascas associadas àquele *patch3d*. Isto permitiria que estruturas soltas internamente a uma região, bem como vazios internos, pudessem ser identificados.

Na lista de *patches3d* de um determinado *patch2d*, um mesmo *patch3d* poderia aparecer duas vezes no caso do *patch2d* estar pendente internamente a este *patch3d*. Isto inclusive faz sentido, já que este retalho de superfície é visitado duas vezes quando se percorre a lista de retalhos de superfície associada a esta região na biblioteca CGC. Ou seja, na estrutura de dados da *Radial Edge*, os dois *usos* desta face estão associados a uma mesma casca. Isto permitiria identificar os retalhos de superfície pendentes internamente às regiões.

As arestas pertencentes à fronteira de um determinado *patch2d* formando um *loop* interno conexo e fechado poderiam ser diferenciadas das arestas soltas ou pendentes internamente ao *patch2d*, sendo armazenadas numa outra lista, ou mesmo na lista *_lsegments*, juntamente com outras as arestas da fronteira deste *patch2d*. Poderia haver também uma lista de vértices representando vértices soltos no interior do *patch2d*.

Outra mudança que poderia ser feita é a desvinculação de um *patch2d* das arestas (objetos da classe *Segment*) originalmente utilizadas para gerar aquele *patch2d*. Quando um *patch2d* fosse interceptado por outros que o dividissem em dois ou mais novos *patches2d*, as malhas associadas aos novos *patches2d* poderiam ser re-geradas de forma que dependessem exclusivamente das novas arestas das fronteiras destes *patches2d*.

Além disso, o algoritmo de interseção de superfícies poderia ser estendido de forma que a superposição de faces fosse devidamente tratada. Isto poderia ser feito identificando-se o problema da superposição antes do cálculo da interseção, por meio de testes geométricos, usando-se uma tolerância conveniente para o modelo em estudo. Ao ser solicitado o cálculo da interseção entre as superfícies superpostas, uma malha única seria gerada. Isto seria de grande utilidade na geração de modelos que utilizam somente faces planas, onde esta situação pode ocorrer com mais freqüência.

Referências Bibliográficas

1 **ACIS 3D Geometric Modeler**.

Disponível em <http://www.spatial.com/components/acis>

2 **Parasolid: Powering the Digital Enterprise**.

Disponível em <http://www.ugs.com/products/open/para*solid*>

3 **Pro/ENGINEER API Toolkit**.

Disponível em <http://www.ptc.com/appserver/it/icm/cda/icm01_list.jsp?*group*=201&num=1&show=y&keyword=403>

4 MACNEAL, R.H. **MSC/PATRAN 6 & 7 FEA USER'S MANUAL**. MacNeal Swhwendler Corporation, 1996.

Disponível em

<http://www.engineering-e.com/software/packagedetail.cfm?packageid=39>

5 O'LEARY, A. **ANSYS Modeling and Meshing Guide – Release 5.4**. SAS IP Incorporated, 1998.

Disponível em <http://www.ansys.com/ServSupp/Library/library.html>

6 LIRA, W. W. M. **Modelagem Geométrica para Elementos Finitos usando Multi-Regiões e Superfícies Paramétricas**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil, 2002.

7 COELHO, L. C. G.; GATTASS, M.; FIGUEIREDO, L. H. **Intersecting and Trimming Parametric Meshes on Finite-Element Shells**. International Journal for Numerical Methods in Engineering, vol. 47, no. 4, pp. 777-800, 2000.

8 COELHO, L. C. G. **Modelagem de Cascas com Interseções Paramétricas**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 1998.

9 HOFFMANN, C. M. **Geometric & Solid Modeling: An Introduction**. Purdue University, Indiana, 1989.

10 MÄNTYLÄ, M. **An Introduction to Solid Modeling Computer**. Science Press, Rockville, Maryland, 1988.

11 REQUICHA, A.G.; VOELCKER, H.B. **Constructive Solid Geometry**. Technical Report Technical Memo no. 25, Production Automation Project, University of Rochester, Rochester, New York, 1977.

12 ROSSIGNAC, J.R.; O'CONNOR, M.A. **A Dimensional-independent Model for Pointsets with Internal Structures and Incomplete Boundaries**. Geometric Modeling for Product Engineering, pp. 145-180, North Holland, 1990.

13 ROSSIGNAC, J.R.; REQUICHA, A.G. **Constructive Non-regularized Geometry**. Computer Aided Design, vol. 23, no. 1, pp. 21-32, 1991.

14 LASZLO, M.J. **A Data Structure for Manipulating Three-dimensional Subdivisions**. PhD Thesis, Department of Computer Science, Princeton University, 1987.

15 LIENHARDT, P. **Extension of the Notion of Map Subdivisions of a Three-dimensional Space**. In STACS'88 Proceedings of the Cinquième Symposium sur les Aspects Thèoriques de L'Informatique, Bordeaux, 1988.

16 CAVALCANTI, P.R.; CARVALHO, P.C.P.; MARTHA, L.F. **Non-manifold Modeling: An Approach Based on Spatial Subdivision**. Computer-Aided Design, vol. 29, no. 3, pp. 209-220, 1997.

17 WEILER, K. **Topological Structures for Geometric Modeling**. Ph.D. Thesis, Rensselear Polytechnic Institute, Troy, N.Y., 1986.

18 WEILER, K. **The Radial-Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Representation**. Geometric Modeling for CAD Applications, North Holland, pp. 3-36, 1988.

19 CAVALCANTI, P. R. **Criação e Manutenção de Subdivisões do Espaço**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 1992.

20 WATSON, D.F. **Computing the n-dimensional Delaunay Triangulation with Application to Voronoi Polytopes**. The Computer Journal, vol. 24, no. 2, pp. 167-172, 1981.

21 YERRY, M.A.; SHEPHARD, M.S. **Automatic Three-Dimensional Mesh Generation by Modified-Octree Technique**. International Journal for Numerical Methods in Engineering, vol. 20, pp. 1965-1990, 1984.

22 LOHNER, R.; PARIKH, P., **Generation of Three-Dimensional Unstructured Grids by the Advancing-Front Method**. International Journal for Numerical Methods in Fluids, vol. 8, pp. 1135-1149, 1988.

23 MOLLER, P.; HANSBO, P. **On Advancing Front Mesh Generation in Three Dimensions**. International Journal for Numerical Methods in Engineering, vol. 38, pp. 3551-3569, 1995.

24 PERAIRE, J.; PEIRO, J.; FORMAGGIA, L.; MORGAN, K.; ZIENKIEWICZ, O.C. **Finite Euler Computation in Three-Dimensions**. International Journal for Numerical Methods in Engineering, vol. 26, pp. 2135-2159, 1988.

25 CAVALCANTE, Neto, J. B. **Geração de Malha e Estimativa de Erro para Modelos Tridimensionais de Elementos Finitos com Trincas**. Tese de Doutorado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Civil, 1998.

26 MIRANDA, A. C.; MARTHA L.F. **Uma Biblioteca Computacional para Geração de Malhas Bidimensional e Tridimensional de Elementos Finitos**. XXI CILAMCE, Rio de Janeiro, Brasil, vol. 03, pp.09.1-09.15, 2000.

27 PIEGL, L. **On NURBS: A Survey**. IEEE Comput. Graph. and Appl., vol. 10, no. 1, pp. 55-71, 1991.

28 PIEGL, L.; TILLER, W. **The Nurbs Book**. 2nd ed. Springer-Verlag, 1999.

29 **NLib – NURBS Library**. *Solid* Modeling Solutions.

Disponível em <http://www.smlib.com/nlib.html>

30 **SOLIDS++**.

Disponível em <http://www.integrityware.com/products/*SOLIDS*++/*solids*++.html>

31 LAVOIE, P. **An introduction to NURBS++**. 1999.

Disponível em <http://libnurbs.sourceforge.net/user.pdf>

32 DE FIGUEIREDO, L. H. **Adaptive Sampling of Parametric Curves**. Graphics Gems V, pp. 173-178, 1995.

33 FARIN, G.E. **Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide**. 3rd ed., Boston: Academic Press, 1993.

34 MARTHA, L. F.; MENEZES, I. F. M.; LAGES, E. N.; PARENTE Jr., E.; PITANGUEIRA, R. L. S. **An OOP Class Organization for Materially Nonlinear Finite Element Analysis**. XVII CILAMCE, Pádova, Itália, pp. 229-232, 1996.

35 MIRANDA, A.C.; MARTHA, L.F. **Mapeamento Transfinito Tridimensional**. XX CILAMCE, São Paulo, Brasil, vol. 1, pp. 1-13, 1999.

36 CAVALCANTE NETO, J. B.; WAWRZYNEK P.A.; CARVALHO, M.T.M.; MARTHA, L.F.; INGRAFFEA, A.R., **An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks**. In Fifth US National Congress on Computational Mechanics, Colorado, USA, 1999.

37 POTYONDY, D.O. **A Software Framework for Simulating Curvilinear Crack Growth in Pressurized Thin Shells**. Ph.D. Thesis, School of Civil Engineering, Cornell University, 1993.

38 VIANNA, A.C. **Modelagem Geométrica Completa para Modelos Bidimensionais de Elementos Finitos**. Dissertação de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1992.

39 SAMET, H. **The Quadtree and Related Hierarchical Data Structure**. ACM Computer Surveys, vol. 16, no. 2, pp. 187-260, 1984.

40 WAWRZYNEK, P.A. **Discrete Modeling of Crack Propagation: Theoretical Aspects and Implementation Issues in Two and Three Dimensions**. Ph.D. Thesis, School of Civil Engineering, Cornell University, 1991.

41 CARVALHO M. T. M. **Uma Estratégia para o Desenvolvimento de Aplicações Configuráveis em Mecânica Computacional**. Tese de Doutorado, PUC-Rio, Departamento de Engenharia Civil, 1995.

42 LIRA W. W. M. **Um Sistema Integrado Configurável para Simulações de Problemas em Mecânica Computacional**. Dissertação de Mestrado, PUC-Rio, Departamento de Engenharia Civil, 1998.

43 COMER D. **The Ubiquitous B-tree**. ACM Computing Surveys, vol. 11, no. 2, pp. 121-131, 1979.

44 BECKMANN N.; KRIEGEL H.P.; SCHNEIDER R.; SEEGER B. **The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles**. In Proceedings of the ACM SIGMOD Conference on Management of Data, pp. 322-332, May 1990.

45 DE BERG, M.; VAN KREVELD, M.; OVERMARS, M.; SCHWARZKOPF, O. **Computational Geometry: Algorithms and Applications**. Springer-Verlag, Berlin, 1997.

46 PREPARATA, F.P.; SHAMOS, M.I. **Computational Geometry: An Introduction**. Springer Verlag, New York, 1990.

47 **Doxygen**.

Disponível em <http://www.stack.nl/~dimitri/doxygen/manual.html>

48 **IUP – Portable User Interface**.

Disponível em <http://www.tecgraf.puc-rio.br/iup>

49 STROUSTRUP B. **The C++ Programming Language**. 3 ed., Addison-Wesley, 1997.

50 DEITEL, H. M.; DEITEL, P. J. **C++ Como Programar**. 3 ed., Bookman, Porto Alegre, 2001.

51 SHEWCHUK, J. R. **Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates**. Discrete & Computational Geometry, vol. 18, pp. 305-363, 1997.

52 **SMLib – Solid Modeling Library**. *Solid* Modeling Solutions.

Disponível em <http://www.smlib.com/smlib.html>

53 WEILER, K. **Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments**. IEEE Computer Graphics and Applications, pp. 21-40, 1985.

54 LIENHARDT, P. **N-Dimensional Generalized Combinatorial Maps and Cellular Quasi-Manifolds**. Journal on Computational Geometry and Applications, pp. 275-324, 1994.

55 BAUMGART, B. **Winged-edge Polyhedron Representation**. Stanford Artificial Intelligence Report No. 5, pp. 253-272, September, 1979.

56 CASTANHEIRA, D. B. S. **Projeto Operações Booleanas de Sólidos**. Estudos em Computação Multimídia, Universidade de Brasília, 2004.

Disponível em <http://www.geocities.com/danbalby/index_port.html>

Apêndice

Inicialmente este capítulo visa mostrar a evolução dos sistemas de modelagem, estabelecer os conceitos e premissas básicos de modelagem geométrica e apresentar os principais problemas encontrados na modelagem de sólidos.

A seguir são apresentados os conceitos de representações *manifold* e *non-manifold*, no contexto da modelagem geométrica aplicada à engenharia. São dados exemplos de objetos sólidos de ambos os tipos e é feita uma análise no que diz respeito à abrangência dos dois tipos de representação, suas restrições, seus domínios e suas dificuldades de implementação.

A seção seguinte descreve de uma forma geral os conceitos de topologia e geometria, apresenta os diferentes tipos de topologia existentes, determina a suficiência de uma representação topológica e mostra as principais relações de adjacência entre elementos topológicos. Em seguida, apresenta um dos elementos mais importantes na implementação de um modelador que seja capaz de reproduzir de forma correta, rápida e eficiente os objetos que se desejam analisar: as estruturas de dados topológicas. Elas são a base de todas as ferramentas oferecidas por qualquer modelador para que a criação e a manipulação de modelos seja feita de maneira coerente e sem ambigüidades. Comentam-se também os principais tipos de estruturas de dados topológicas utilizados para domínios bi e tridimensionais.

Posteriormente, analisam-se os esquemas mais utilizados para representar sólidos, com ênfase naqueles mais utilizados na maioria dos modeladores: CSG (*Constructive Solid Geometry*) e B-Rep (*Boundary Representation*). Nesta seção cada um destes esquemas de representação é apresentado, e estabelece-se uma comparação entre eles, do ponto-de-vista de implementação e de utilização, mostrando-se vantagens e desvantagens de cada um.

A.1.
Conceitos Fundamentais

Um modelo é, em geral, um objeto construído artificialmente para facilitar a visualização, observação ou análise de um outro objeto. Diferentes áreas

produzem modelos para que fenômenos relativos aos objetos que realmente se deseja estudar possam ser analisados sem a necessidade de que o objeto inclusive exista ou seja diretamente observável. Podem-se citar modelos físicos, modelos moleculares, modelos matemáticos ou desenhos de engenharia.

Modelos computacionais consistem em informações armazenadas em arquivos ou na memória de um computador que podem ser utilizadas para realizar tarefas semelhantes aos outros tipos de modelos citados acima de forma confiável e eficiente. A quantidade de informações que podem ser armazenadas num modelo computacional depende do tipo de perguntas às quais se deseja obter respostas satisfatórias. Este trabalho está inserido num contexto em que os problemas que se tentam resolver através de modelos são essencialmente geométricos.

Modelagem geométrica, como descrita por Weiler [17], envolve a criação, manipulação, manutenção e análise das representações das formas geométricas de objetos bi e tridimensionais. Possui inúmeras aplicações entre as mais diversas áreas, como a produção de filmes, design e análise de peças mecânicas industriais, visualização científica, dentre várias outras, sendo a reprodução de objetos para análise em engenharia, particularmente a análise pelo MEF, o foco deste trabalho.

O desenvolvimento de algoritmos computacionais que sejam capazes de processar estas representações de forma eficiente e não ambígua é um dos principais desafios dos profissionais que lidam com modelagem geométrica. Deseja-se tornar o processamento das informações relativas aos objetos em estudo uma tarefa mais eficaz, robusta e econômica.

As formas de representações de sólidos e conseqüentemente a quantidade e os tipos de informações a serem armazenadas podem diferir bastante entre dois sistemas de modelagem geométrica, de acordo com as limitações de cada um e com as necessidades que ambos se propõem a suprir.

Uma característica significativa inerente à modelagem geométrica é que as técnicas para armazenamento e processamento de informações geométricas são relativamente independentes do tipo de aplicação desejada: métodos semelhantes podem ser usados para construir modelos de máquinas, peças mecânicas, e utensílios domésticos.

A.2.
Evolução Histórica

Diversas formas de modelagem geométrica surgiram ao longo das últimas décadas, diferindo na quantidade e tipo de informações diretamente disponíveis sendo representadas, e nas outras informações que podem ser derivadas destas. Cada uma das formas de modelagem apresenta características particulares como nível de complexidade, limitações, volume de memória necessário para armazenamento das informações, eficiência na busca e manipulação destas informações, dentre outras, que podem torná-las mais vantajosas ou desvantajosas de acordo com o tipo de objetivo almejado.

Weiler [17] classifica historicamente a evolução dos sistemas de modelagem da seguinte forma:

- Modelagem por arames (*wireframe modeling*)

Uma das primeiras técnicas de modelagem geométrica que existiram, representa os objetos por arestas e pontos na superfície do mesmo (Figura A.1a). Este tipo de representação pode ser ambíguo em alguns casos.

- Modelagem por superfícies (*surface modeling*)

Tipo de modelagem um pouco mais avançada que a anterior por representar a descrição matemática das superfícies que formam o contorno dos objetos (Figura A.1b). Este tipo de técnica permite a visualização gráfica e controle numérico de máquinas industriais para confecção de modelos cuidadosamente construídos, contudo oferece poucos testes de integridade.

- Modelagem de sólidos (*solid modeling*)

Técnica de modelagem mais recente que as anteriores, contém informações implícitas ou explícitas sobre o fechamento e a conectividade de objetos sólidos, ganhando uma importância cada vez maior em diversas aplicações (Figura A.1c). Esta técnica possui várias vantagens sobre as anteriores, sobretudo por garantir que qualquer modelo gerado irá formar objetos com volumes fechados e com contorno definido que se assemelham mais aos volumes fisicamente realizáveis na prática. Esta técnica permite que se diferencie o interior dos objetos do seu exterior, criando assim a possibilidade de se determinar propriedades inerentes aos objetos como volume e centro de gravidade. Além disso, existe a disponibilização de ferramentas que, à medida em que criam e manipulam os sólidos, mantêm a integridade do modelo.

- Modelagem geométrica *non-manifold*

É a forma de modelagem mais recente, que elimina as restrições normalmente associadas às formas de modelagem de sólidos *manifold* agregando todas as capacidades dos três tipos de formas de modelagem previamente analisados e estendendo o domínio de cada uma delas. Este tipo de técnica permite representar objetos com estruturas internas ou pendentes de dimensão inferior (Figura A.1d). O potencial deste tipo de representação ainda não foi totalmente explorado, e sistemas baseados nesta forma de modelagem vêm sendo desenvolvidos em número crescente. Uma discussão mais detalhada sobre formas de representação *manifold* e *non-manifold* será apresentada na próxima seção.

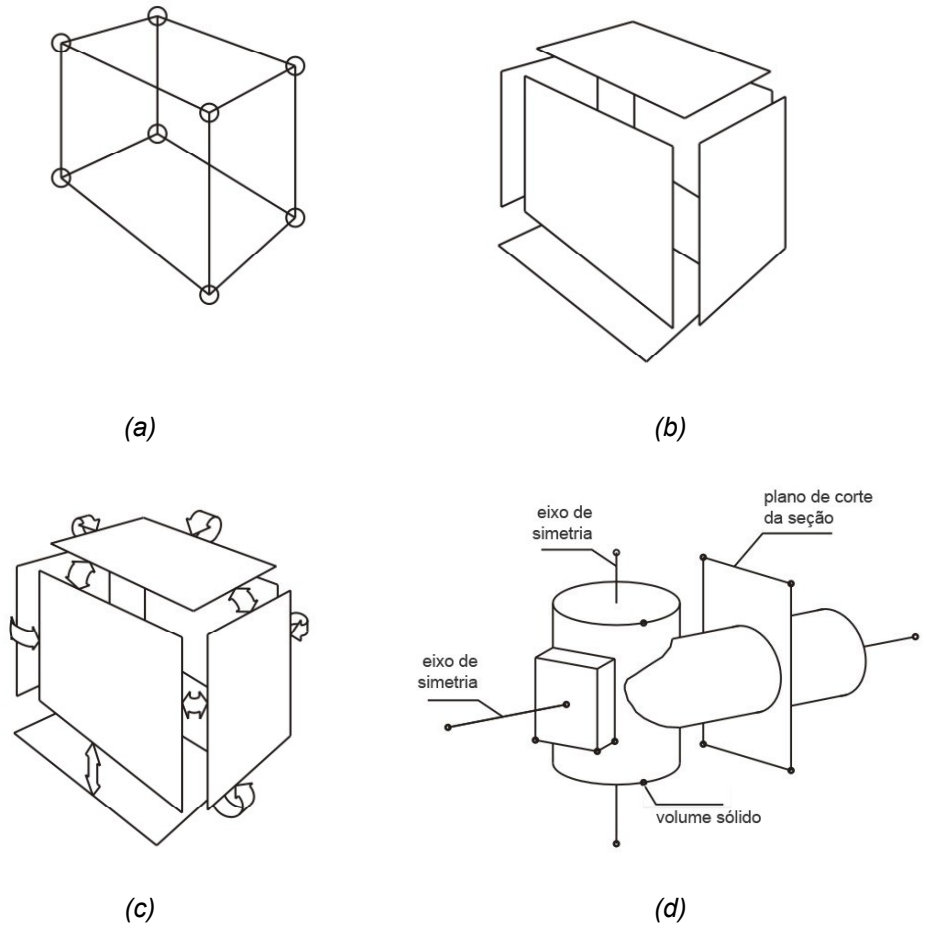


Figura A.1 – Formas de modelagem: a) por arames; b) por superfícies; c) modelagem de sólidos; d) modelagem *non-manifold*.

A.3.
Problemas da modelagem de sólidos

Modelos de sólidos devem ser completos e representarem com exatidão os objetos de estudo para que possam ter um campo de aplicação amplo. A seguir, são apresentados alguns problemas enfrentados na modelagem de sólidos [10].

A.3.1.
Completude

Modelos gráficos bidimensionais consistem em objetos bidimensionais como linhas, arcos, texto e outras notações, para representar computacionalmente objetos físicos. Contudo, os computadores não possuem a capacidade de interpretação necessária para que estes modelos gráficos possam servir como modelos de sólidos.

Um dos problemas dos modelos gráficos bidimensionais é o fato de ser perfeitamente possível gerar desenhos que não representem nenhuma forma real (Figura A.2). Outro problema é que até hoje ainda é inviável a inferência algorítmica de todas as informações tridimensionais necessárias a partir de desenhos bidimensionais.

Se a terceira coordenada puder ser inserida para que se possa gerar uma representação tridimensional dos objetos a partir de um modelo gráfico bidimensional, passa-se a ter um modelo de arames (*wireframe model*), como exposto anteriormente. Um modelo deste tipo reduz consideravelmente a possibilidade de gerar desenhos inconsistentes. Contudo, mesmo um conjunto de linhas tridimensionais pode representar um objeto sólido de forma ambígua, como pode ser observado na Figura A.3.

O problema com estes modelos gráficos é que eles não conseguem representar de forma completa as informações geométricas necessárias para que perguntas de caráter geométrico arbitrárias possam ser respondidas.

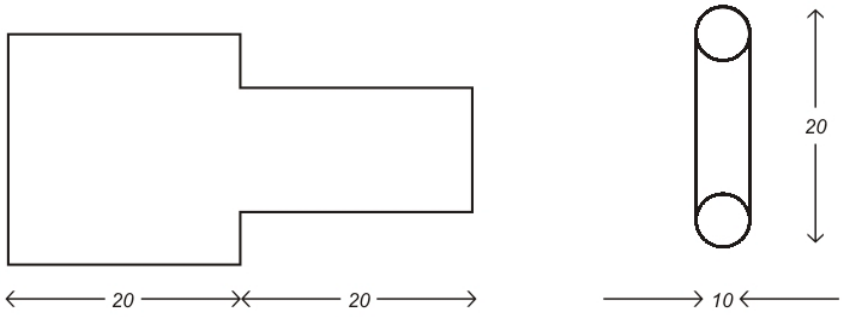


Figura A.2 – Desenho sem sentido [10].

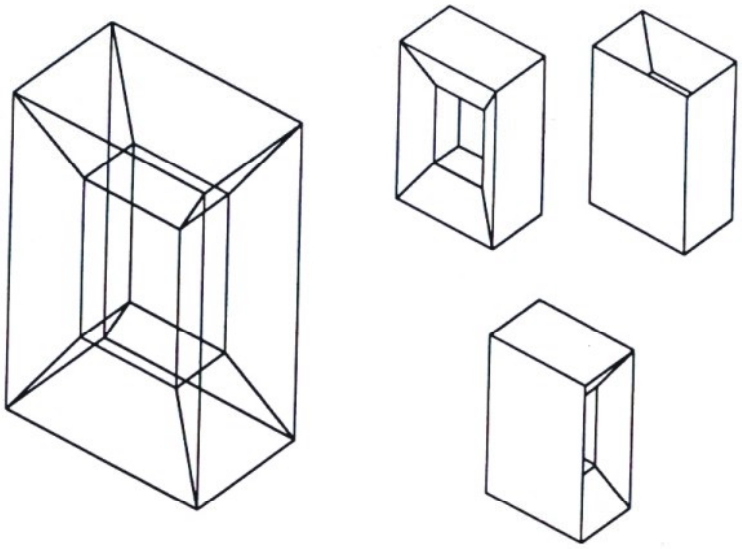


Figura A.3 – Objeto sólido ambíguo [10].

A.3.2. Integridade

Modelos poliedrais fornecem informações suficientes sobre as partes ocultas dos objetos, o que faz com que sejam utilizados nas metodologias de computação gráfica ao invés dos modelos gráficos. Tais modelos são constituídos de primitivas bidimensionais – faces poligonais – ao invés de linhas.

Algoritmos para remover linhas ocultas partem do pré-suposto de que os polígonos não se interceptam a não ser nas suas arestas ou vértices comuns. Modelos poliedrais apropriadamente construídos irão obedecer a este critério, mas não se pode garantir que qualquer modelo irá satisfazê-lo.

A integridade de modelos sólidos é um dos principais problemas a serem abordados em modelagem de sólidos. Um modelo sólido cuja construção seja

excessivamente complexa não possui interesse prático. Testes para checagem de integridade podem ser muito complicados, e ainda requerem que o usuário determine quais são as ações corretas a serem executadas.

A.3.3. Complexidade e abrangência geométrica

Aliado ao problema da integridade surge o problema da complexidade dos modelos. Mesmo modelos relativamente simples, como o da Figura A.4, podem ser formados por centenas ou milhares de polígonos. Gerar tais informações pode ser complicado, trabalhoso e pouco eficiente.

Em relação à abrangência geométrica, modelos poliedrais podem não ser suficientes para representar objetos com geometrias precisas e complexas, como no caso da indústria automobilística. A dificuldade de lidar com geometria de sólidos é proporcional à complexidade dos modelos matemáticos utilizados para representá-la.

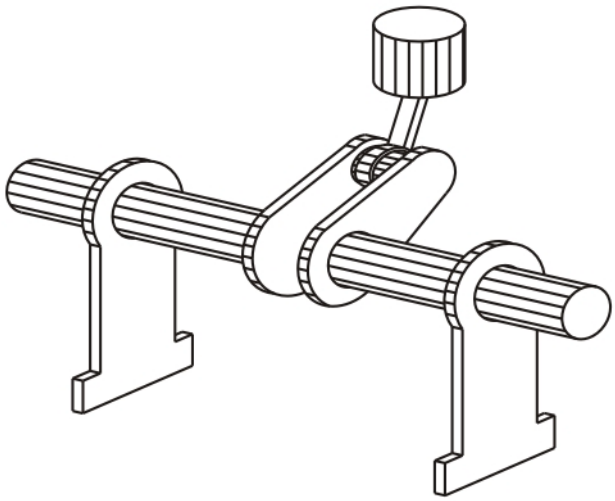


Figura A.4 – Modelo poliédrico [10].

A.3.4. Natureza dos algoritmos geométricos

Espera-se que modelos sólidos sejam capazes de responder de forma algorítmica a perguntas típicas de natureza geométrica passíveis de serem realizadas em aplicações para engenharia, tais como: Como á e forma do objeto? Qual é o peso, a área de superfície, o centro de gravidade do objeto?

Este objeto consegue suportar esta carga? Como este objeto pode ser manufaturado a partir de um conjunto de processos de manufatura disponíveis?

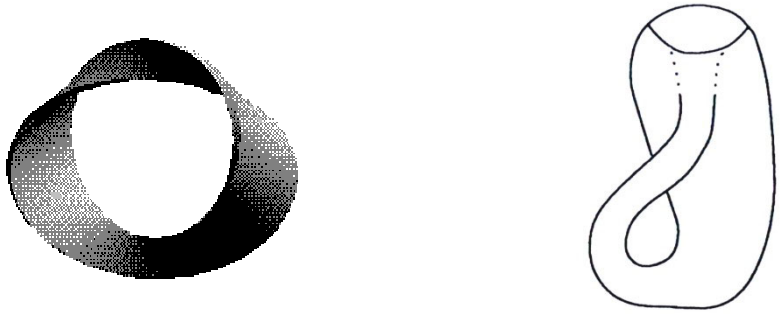
As respostas a tais perguntas podem ser uma imagem, um número, uma constante booleana (verdadeiro ou falso) ou mesmo um outro modelo sólido que represente o resultado de um processo, como no caso de uma pergunta do tipo: qual o efeito deste processo de manufatura aplicado a este objeto?

Pode-se observar, então, que um modelador geométrico deve incluir ferramentas que permitam não apenas simular objetos físicos, mas também processos físicos. E ainda, que tais processos físicos constituam um conjunto fechado de operações, ou seja, que seja possível aplicar um processo num modelo gerado a partir da aplicação prévia de um processo físico a um outro modelo. Todas estas operações devem garantir a manutenção da consistência dos modelos.

A.4. Modelagem Geométrica *manifold* e *non-manifold*

Numa representação de sólidos *manifold* (*2-manifold*), todo ponto numa superfície é bidimensional, isto é, todo ponto tem uma vizinhança que é homeomorfa a um disco bidimensional. Isto quer dizer que, se analisada localmente numa área pequena o suficiente no entorno de um ponto dado, uma superfície existente num espaço tridimensional pode ser considerada “chata” ou plana. Pode-se dizer que deformando a superfície localmente para um plano, ela não rasga ou passa a possuir pontos coincidentes.

Uma superfície *manifold* é orientável se for possível distinguir dois lados diferentes. Escolhe-se um ponto p dessa superfície e define-se arbitrariamente um sentido (horário ou anti-horário). Mantendo-se esta orientação, move-se ao longo de qualquer caminho fechado dessa superfície. Se existe um caminho tal que seja possível retornar a p com uma orientação oposta à escolhida, então a superfície é não-orientável, caso contrário é orientável. Na Figura A.5a pode-se vislumbrar a faixa de Möbius (Möbius strip) e na Figura A.5b a garrafa de Klein (*Klein bottle*), que são exemplos de superfícies não-orientáveis. Como exemplos de superfícies orientáveis pode-se citar a esfera e o toro.



(a) *(b)*

Figura A.5 **–** Exemplos de superfícies não orientáveis: a) Faixa de Möbius; b) Garrafa de Klein [9].

As superfícies de um sólido devem ser orientáveis e fechadas para que haja uma nítida distinção entre interior, fronteira e exterior. Enquanto as superfícies *manifold* de um sólido podem se constituir de diversos pedaços, estes devem ser conectados para formar uma superfície fechada.

Non-manifold é um termo de modelagem geométrica que se refere a situações topológicas que não são *2-manifold*. Num ambiente *non-manifold*, a vizinhança ao redor de um ponto qualquer numa superfície não precisa ser um disco bidimensional, ou seja, mesmo analisando localmente uma área bem pequena ao redor do ponto, ela pode não ser “chata” ou plana. Um cone tocando uma outra superfície em um único ponto, mais de duas faces se encontrando numa aresta comum ou arames (arestas soltas) emanando de um ponto numa superfície são situações topológicas *non-manifold*. Algumas delas podem ser vistas na Figura A.6. Cavalcanti [19] define um sólido *non-manifold* como a imersão de vários sólidos *manifold* no $\Re^3$, permitindo que as variedades se auto-interceptem, mas restringindo-se as interseções às dimensões 0 ou 1. Os objetos são variedades topológicas, mas seu mergulho no $\Re^3$ permite a coincidência geométrica de estruturas topológicas distintas. Alguns exemplos de sólidos *non-manifold* são mostrados na Figura A.7.

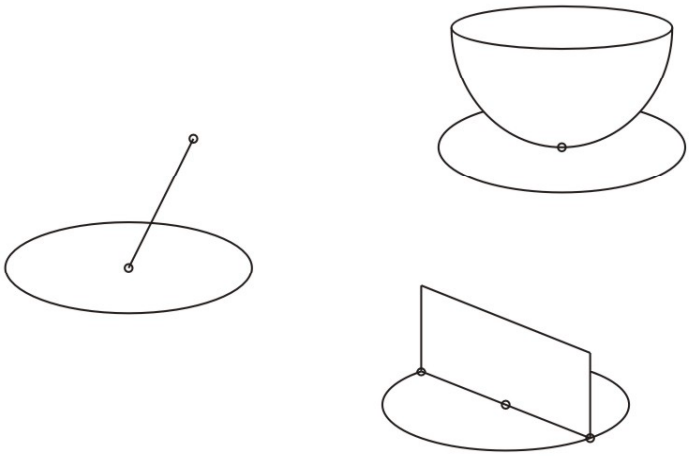


Figura A.6 – Situações topológicas *non-manifold* [17].

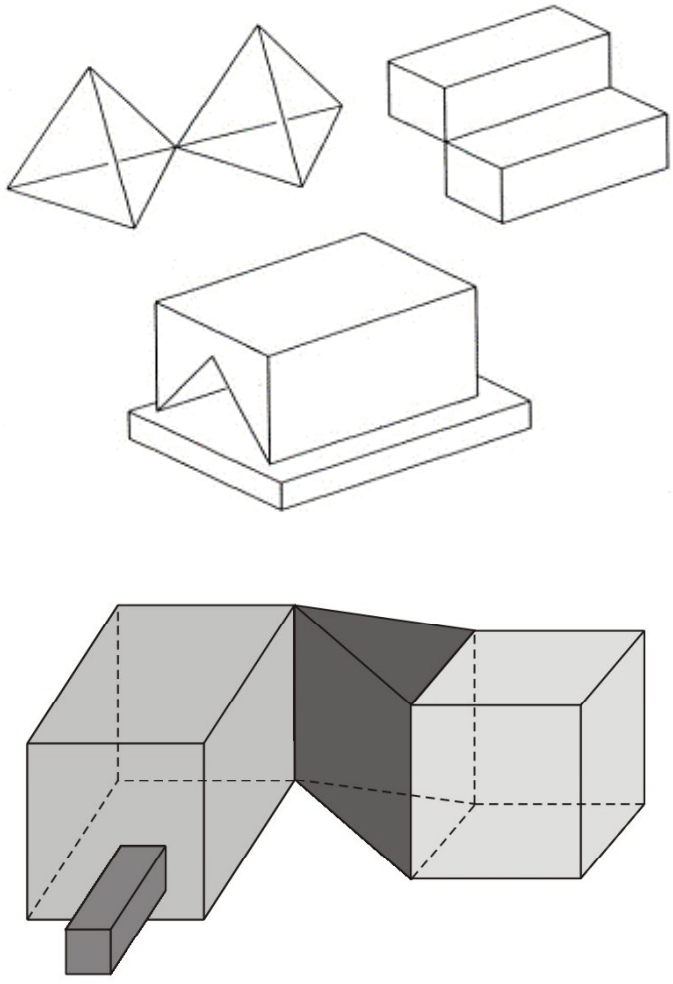


Figura A.7 – Exemplo de sólidos *non-manifold* [10,19].

O conjunto das operações booleanas não é fechado em representações do tipo *manifold*. As operações booleanas regularizadas, a partir do momento em

que permitem somente resultados com volumes preenchíveis, desconsiderando estruturas pendentes de mais baixa dimensão, evitam um subconjunto dos resultados *non-manifold* que podem ser gerados pela aplicação das operações booleanas em sólidos *manifold*. Mesmo assim, a aplicação destas operações, regularizadas ou não, em alguns sólidos *manifold* produz resultados *non-manifold* não representáveis, portanto, em ambientes de modelagem *manifold*. As Figuras A.8 e A.9a ilustram exemplos deste tipo.

No caso particular da Figura A.9a, em que foi realizada a operação booleana regularizada de união entre dois suportes em forma de L, surge o problema de que a aresta (P, Q) é adjacente a quatro faces. Segundo Hoffmann [9], três tipos de abordagem para tratar de estruturas *non-manifold* foram desenvolvidos:

1 - Os objetos de estudo devem ser *manifold*s, logo operações em sólidos com resultados *non-manifold* não são permitidas ou são consideradas como erros.
2 - Os objetos são topologicamente *manifold*s, mas sua inserção no espaço tridimensional permite coincidência geométrica de estruturas topologicamente diferentes.
3 - Objetos *non-manifold* são permitidos, tanto como dados de entrada (*input*) quanto resultados (*output*).

A primeira abordagem pode ser satisfatória para várias aplicações, contudo ela restringe muito o domínio dos objetos com que se deseja trabalhar, e também uma boa parcela dos resultados passíveis de se obter. Em particular, modeladores que possuem facilidades voltadas para a aplicação das operações booleanas em sólidos, como é precisamente o caso discutido neste trabalho, passam a não poder usufruir de boa parte da capacidade proporcionada por estas operações. Além disso, dependendo da estruturação interna do modelador, mesmo que os resultados produzidos pela aplicação das operações booleanas em sólidos *manifold* sejam igualmente *manifold*, pode haver passos intermediários que produzam resultados parciais *non-manifold*, o que se torna um inconveniente para o usuário.

Na segunda abordagem, deve haver uma interpretação topológica das estruturas *non-manifold* presentes. No caso da Figura A.9a, deve-se interpretar a aresta *non-manifold* PQ como duas arestas distintas que são geometricamente coincidentes. As duas possibilidades existentes encontram-se nas Figuras A.9b e A.9c.

Uma das maneiras de se decidir por qual estrutura optar é fazer com que a triangulação das superfícies não gere triângulos degenerados. No caso da Figura A.9c, como os pontos P1 e P2 são topologicamente diferentes, uma aresta será gerada entre estes dois pontos na triangulação. Contudo, como os pontos são geometricamente coincidentes, a aresta terá tamanho nulo e triângulos degenerados serão criados. A estrutura apresentada nesta figura também conduz a problemas de ordem geométrica mais difíceis e compromete a robustez do modelador, sobretudo quando se passa para o domínio das superfícies curvas.

A terceira abordagem permite a existência de estruturas *non-manifold* como vértices e arestas. Apesar de requerer um volume de memória maior para ser implementada, possui algoritmos mais simples já que não necessita de testes para verificar a presença de entidades topológicas indesejáveis e para verificação de ambigüidades.

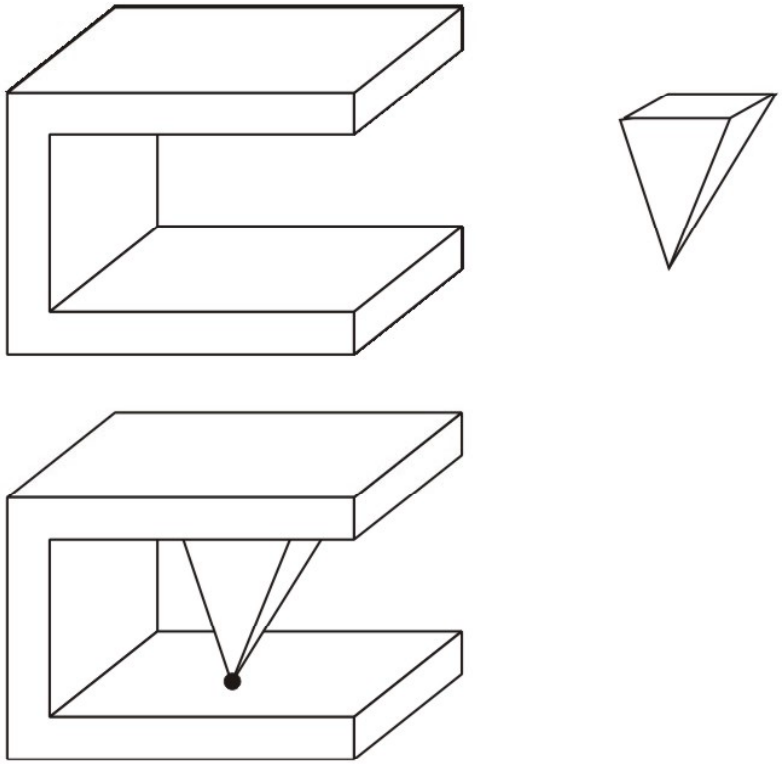
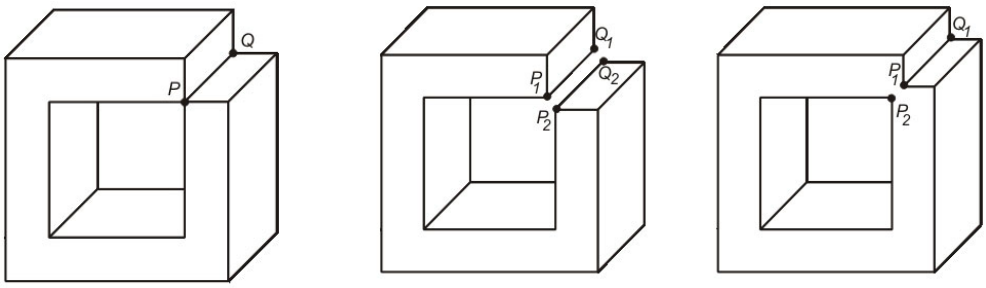


Figura A.8 – União regularizada de dois objetos *manifold* gerando um resultado *non-manifold* [17].



(a) *(b)* *(c)*

Figura A.9 – Aproximações para situações *non-manifold*: a) objeto *non-manifold* gerado pela união regularizada de dois suportes em forma de L; b) e c) topologias possíveis [9].

Em termos gerais, representações *non-manifold* são mais flexíveis, podem representar uma variedade maior de objetos e podem suportar uma maior quantidade de aplicações do que representações *manifold*, ao custo de uma estrutura de dados maior. Operações booleanas, tanto em representações *manifold* como *non-manifold*, necessitam detectar e lidar com resultados *non-manifold* até certo grau. Contudo, numa representação *non-manifold* tais resultados são representados e manipulados uniformemente e de maneira nítida. Por isso, se é desejado que as operações booleanas formem um conjunto fechado de operações, representando de forma precisa e fiel os resultados *non-manifold* eventualmente presentes, é necessário que se opte por um tipo de representação *non-manifold*. Esta também se torna necessária se a análise das estruturas internas ao volume de um objeto e as relações entre elas são importantes, como em objetos compostos e nas malhas de elementos finitos. Um dos aspectos mais relevantes das representações *non-manifold* talvez seja o de agregarem num só tipo de representação as três formas de modelagem já apresentadas: por arames, por superfícies e modelagem de sólidos. Tal uniformidade oferece vantagens significativas na criação, implementação e manutenção de sistemas de modelagem geométrica.

Representação topológicas do tipo *manifold* ainda podem ser preferíveis no caso de haver prioridade quanto ao espaço de armazenamento das informações, e quando as vantagens adicionais proporcionadas por um sistema *non-manifold* não são requeridas.

A.5.
Topologia em Modelagem Geométrica

Semanticamente pode-se definir topologia como o estudo das formas. Contudo, tal definição pode-se estender ou modificar de acordo com a área de estudo: topologia algébrica, topologia combinatória, topologia diferencial, topologia de conjuntos de pontos, dentre outros tipos. Para os fins a que se destina este trabalho, enfoque será dado à topologia de conjuntos de pontos. Para que este tipo de topologia possa ser perfeitamente compreendido, alguns conceitos e definições devem ser apresentados, assim como o ambiente onde se irá trabalhar.

A.5.1.
Topologia e geometria

A geometria completa de um objeto pode ser definida como o conjunto de informações essenciais para definir a forma deste objeto e a localização espacial de todos os seus elementos constituintes.

Topologia é uma abstração, pode ser definida como um conjunto coerente de informações obtidas a partir da descrição geométrica completa de um objeto. Tal conjunto de informações é invariante após a aplicação de transformações afins ao objeto. Ou seja, propriedades que se modificam devido a transformações afins (homeomorfismos, como será definido mais tarde) não fazem parte deste conjunto. Pode-se notar, então, que a topologia é incompleta na descrição de um objeto, ela não possui informações suficientes para que o objeto possa ser modelado de forma única e completa. Por outro lado, a geometria completa daquele objeto permite que ele seja perfeitamente modelado. Isto significa que não se pode obter todas as informações geométricas a partir da topologia de um objeto, mas o contrário é válido, apesar de nem sempre as informações geométricas estarem numa forma conveniente para que se possam derivar as informações topológicas.

A.5.2.
Conceitos fundamentais e definições matemáticas

Cavalcanti [19], Weiler [17,53] e Hoffmann [9] apresentaram alguns conceitos e definições formais relativos à topologia que serão utilizados ao longo deste trabalho:

A.5.2.1.
Conceitos da teoria de grafos

Um vértice é um ponto único associado a uma posição tridimensional única no espaço de modelagem. Uma aresta é um segmento de uma curva limitado por dois vértices. A princípio, estes dois vértices seriam distintos, o que implicaria no fato de cada aresta possuir sempre dois vértices distintos e haver apenas uma única aresta entre dois vértices quaisquer, contudo tal definição formal pode ser abandonada quando se trabalha com sistemas de modelagem que permitem situações como *self-loops* (arestas que se auto-interceptam) ou *multigraphs* (multigrafos ou situação topológica em que várias arestas possuem os mesmos dois vértices). Um grafo é um conjunto de vértices e arestas distintas que utilizam estes vértices. Tais arestas podem ser segmentos retos ou curvos de curvas espaciais delimitando o contorno de superfícies. Vértices que compartilham uma aresta entre si são considerados adjacentes um ao outro.

Um grafo é considerado conexo se quaisquer dois vértices são unidos por algum caminho (*path*), que é uma seqüência alternada de vértices e arestas que começa e termina nos dois vértices e onde cada aresta no caminho é incidente a cada vértice antes e depois da mesma na seqüência. A conectividade de um grafo é o número mínimo de vértices que, quando removidos juntamente com suas arestas incidentes, resulta num grafo desconexo.

Um *self-loop* é uma situação onde uma aresta liga um vértice a ele mesmo, ou seja, quando uma aresta possui um único vértice associado a ela. Um *multigraph* é um grafo em que é permitido que várias arestas liguem os mesmos dois vértices. Grafos que permitem tanto *self-loops* como *multigraphs* são chamados *pseudographs* (pseudografos).

A.5.2.2.
Conceitos topológicos

Um homeomorfismo é um tipo de transformação contínua e cuja transformação inversa também é contínua. Intuitivamente homeomorfismos podem ser encarados como deformações elásticas que preservam relações de adjacência. Topologia pode então ser definida como o estudo das propriedades que são invariantes por homeomorfismos.

Um disco aberto é uma porção de um espaço bidimensional que se localiza no interior de um círculo de raio positivo centrado em um determinado ponto, excluindo-se a circunferência que a delimita. Uma bola aberta ou esfera

aberta é o equivalente tridimensional ao disco aberto e constitui-se do conjunto de pontos interiores a uma esfera centrada num ponto e com um raio maior que zero, excluindo-se a casca que delimita a esfera.

Um subconjunto de um espaço topológico é conexo em arco (*arcwise-connected*) se para qualquer par de pontos desse subconjunto existe um caminho contínuo entre eles totalmente contido nesse subconjunto. Desta maneira, pode-se definir uma superfície como um espaço conexo em arco que é topologicamente bidimensional. Vale ressaltar que, apesar de uma superfície ser localmente bidimensional, ela pode ser curva e geometricamente existir num espaço tridimensional.

Uma superfície é limitada se toda ela puder estar contida numa bola aberta. A fronteira de uma superfície pode ser uma curva aberta ou fechada, ou mesmo um único ponto na superfície. Uma superfície é fechada se ela é limitada e não tem fronteira, como por exemplo uma esfera. Um plano é uma superfície ilimitada e sem fronteira, portanto não é uma superfície fechada.

Uma vizinhança de um ponto é definida como sendo uma bola aberta centrada neste ponto. Um ponto de um espaço topológico é considerado um ponto de aderência de um subconjunto desse espaço se toda vizinhança desse ponto contém pelo menos um ponto desse subconjunto.

Um subconjunto de um espaço topológico é fechado quando ele contém todos os seus pontos de aderência. Se o complemento desse subconjunto (todos os pontos do espaço exceto os que pertencem ao próprio subconjunto) for fechado, então esse subconjunto é aberto (o que significa que todo ponto desse subconjunto possui uma vizinhança completamente contida nele).

O *fecho* de um subconjunto A de um espaço topológico, denotado por $F(A)$, é o conjunto de todos os pontos de aderência de A. O *interior* desse subconjunto, denotado por $I(A)$, é o conjunto dos pontos de A que não são pontos de aderência do complemento de A. A *fronteira* de A, denotada por ∂A, é o conjunto dos pontos do espaço topológico que são pontos de aderência de A e do complemento de A.

Um subconjunto A de um espaço topológico é conexo quando pode ser dividido em dois subconjuntos B e C com $A = B \cup C$, $B \neq \phi$ e $C \neq \phi$ de forma que:

· $\exists b \in B$, com $b \in F(C)$;

· $\exists c \in C$, com $c \in F(B)$.

Um subconjunto de um espaço topológico é dito *limitado* quando ele está contido numa bola aberta. Um conjunto fechado e limitado é dito *compacto*.

A *regularização* de um conjunto de pontos *A*, denotada por *R(A)*, é definida como sendo *R(A) = F(I(A))*. Os conjuntos que satisfazem *R(A) = A* são ditos *regulares*.

Uma superfície *2-manifold* (ou simplesmente *manifold*, como será utilizado no resto desta dissertação), como dito anteriormente, é uma superfície topologicamente conexa e bidimensional em que cada ponto tem uma vizinhança que é topologicamente equivalente (homeomorfa) a um disco aberto. Uma superfície *manifold* pode ser fechada ou não.

De uma forma geral, um *n-manifold M* num espaço Euclidiano E^m, onde $m \geq n$, é um subespaço que é localmente homeomorfo a E^n. Isto é, para cada ponto *p* de *M*, existe uma vizinhança *U* de *p* que é homeomorfa a E^n. Um *n-manifold* com fronteira é um subespaço que é localmente homeomorfo ao semi-espaço positivo

$E^{n+} = \{(x_1, ..., x_n) \in E^n / x_1 \geq 0\}$

O hiperplano $x_1 = 0$ é a fronteira de E^{n+}.

Ressalta-se que, num *n-manifold M* com fronteira, pode-se distinguir entre pontos interiores e de fronteira: um ponto $p \in M$ é interior se existe uma vizinhança *U* de *p* que é homeomorfa a E^n. Um ponto $p \in M$ é um ponto de fronteira se existe uma vizinhança *U* de *p* que é homeomorfa à vizinhança do ponto (0, ..., 0) em E^{n+}. Em contraste, um *n-manifold* consiste apenas de pontos interiores.

Pode-se demonstrar que a fronteira de um *n-manifold* com fronteira é homeomorfa a um *(n-1)-manifold* sem fronteira.

Para os objetivos a que esta dissertação se propõe a cumprir, apenas o conceito de 2-*manifold* (ou simplesmente *manifold*) é necessário.

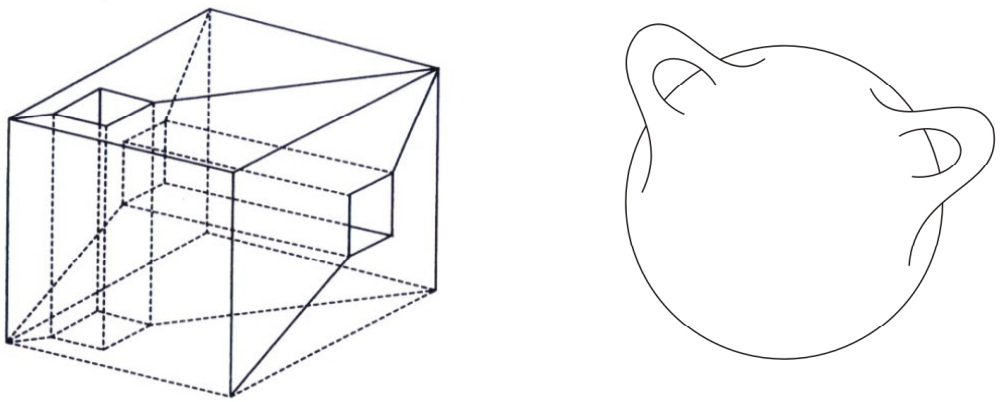
Um grafo pode ser *imerso* em (ou *mapeado* sobre) uma superfície se é desenhado nessa superfície sem que nenhum par de arestas se intercepte, exceto nos seus vértices incidentes. Um *grafo planar* é aquele que pode ser imerso numa superfície planar. Um grafo também pode ser imerso num espaço tridimensional, contendo ou não superfícies limitadas, desde que as propriedades de não-interseção sejam respeitadas, isto é, desde que nenhum par de elementos se intercepte a não ser em elementos de fronteira de mais baixa dimensão comuns.

Faces são subconjuntos conexos da superfície definida por um grafo imerso em uma superfície. Cada face é um componente conexo do conjunto obtido pela subtração dos vértices e arestas do grafo imerso da superfície. A

fronteira de uma face corresponde a todas aquelas arestas e vértices do grafo imerso que tocam a face em toda a sua extensão. A face não contém a sua fronteira. Quando um grafo está imerso numa superfície *manifold* orientável cada aresta do grafo é usada exatamente duas vezes quando se percorre as arestas que delimitam cada face de modo que o vértice final de uma aresta seja o vértice inicial da próxima aresta.

Uma face é *simplesmente conexa* quando possui uma fronteira única e conexa. Uma face *multiplamente conexa* possui uma fronteira formada por dois ou mais componentes desconexos, como no caso de uma face com um orifício interno.

Uma *alça* (*handle*) pode ser formada cortando-se dois buracos na superfície do objeto e construindo-se um tubo para ligá-los. A *ordem* (*genus*) de um grafo pode ser definida como o número mínimo de alças que precisam ser adicionadas a uma esfera para que o objeto seja homeomorfo a ela, ou seja, para que o grafo possa ser imerso na superfície resultante sem que arestas se interceptem em lugares diferentes dos seus vértices comuns. Um objeto com a forma de uma rosquinha ou um toro são homeomorfos a uma esfera com uma alça, logo possuem ordem 1. O objeto na Figura A.10a possui dois buracos, sendo topologicamente equivalente a uma esfera com duas alças, como mostrado na Figura A.10b.



(a) *(b)*

Figura A.10 – Exemplos de homeomorfismo: a) objeto com dois buracos e faces homeomorfas a discos; b) esfera com alças (ordem 2) [9].

A.5.2.2.1.
A Fórmula de Euler – Poincaré

As informações topológicas associadas a um sólido podem ser inconsistentes, de forma que as relações prescritas de adjacência entre os elementos topológicos como vértices, arestas e faces podem não ser satisfeitas. Isto se agrava se as informações topológicas são derivadas de informações geométricas aproximadas, devido aos problemas de ponto flutuante que eventualmente aparecem.

A fórmula de Euler – Poincaré descreve uma relação quantitativa entre os elementos topológicos que é necessária, mas não suficiente, para garantir a consistência topológica do modelo analisado.

O caso mais simples é o de sólidos que apresentam uma superfície fechada e orientável e sem buracos ou vazios internos. Assume-se que cada face possui um único *loop* (seqüência contínua formando uma volta completa) de vértices adjacentes, ou seja, que a face é homeomorfa a um disco fechado. Para sólidos deste tipo, o número *V* de vértices, *E* de arestas, e *F* de faces satisfazem a seguinte fórmula de Euler:

$$V - E + F - 2 = 0 \tag{2.1}$$

Tal fórmula é facilmente demonstrável por indução para um sólido qualquer obedecendo às condições impostas acima. Considerando-se o caso de sólidos que tenham buracos, mas que ainda sejam limitados por uma superfície única e conexa, em que cada face seja homeomorfa a um disco, passa-se a considerar a ordem (*G*) da superfície, cuja definição já foi apresentada anteriormente, na fórmula acima. Desta forma os elementos topológicos passam a obedecer a fórmula de Euler-Poincaré:

$$V - E + F - 2(1 - G) = 0 \tag{2.2}$$

O objeto da Figura A.10a é um exemplo de sólido que obedece a tais restrições e à fórmula acima.

O próximo passo é aumentar o nível de generalização permitindo que o sólido tenha vazios internos. Estes vazios são limitados por superfícies *manifold* fechadas e separadas, chamadas cascas (*shells*). O número de cascas será denotado por S. Em seguida, retira-se a restrição de que cada face seja limitada

por um único *loop* de vértices, desde que cada face possa ser mapeada para o plano. Na Figura A.11, a face mostrada possui 4 diferentes *loops*, sendo um constituído por um único vértice e outro por dois vértices conectados por uma aresta. Sendo *L* o número de total de *loops* presentes em todas as faces, a relação entre todos os elementos topológicos pode ser finalmente generalizada da seguinte forma:

$$V - E + F - (L - F) - 2(S - G) = 0 \tag{2.3}$$

Na Figura A.12 pode-se vislumbrar um sólido que ilustra essa relação.

Apesar de todo sólido *manifold* ter que satisfazer a fórmula de Euler-Poincaré estendida, nem toda superfície satisfazendo a essa relação será a superfície de um sólido *manifold*. A superfície mostrada na Figura A.13 possui o mesmo número de vértices, arestas e faces de um cubo e não é a superfície de um sólido *manifold*, já que possui uma face pendente ligada à parte piramidal por uma aresta *non-manifold*.

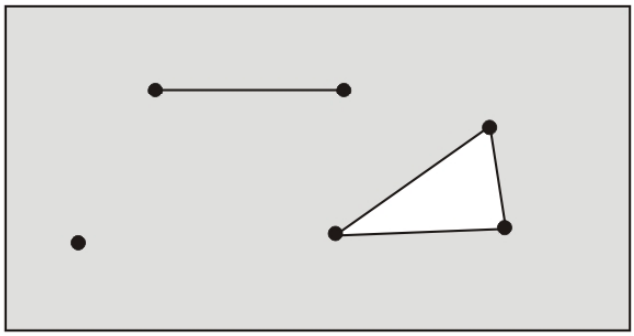


Figura A.11 – Face com quatro *loops* [9].

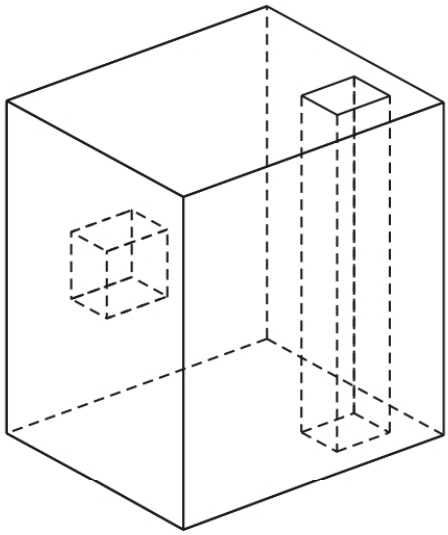


Figura A.12 – Sólido que obedece à equação de Euler-Poincaré estendida [9].

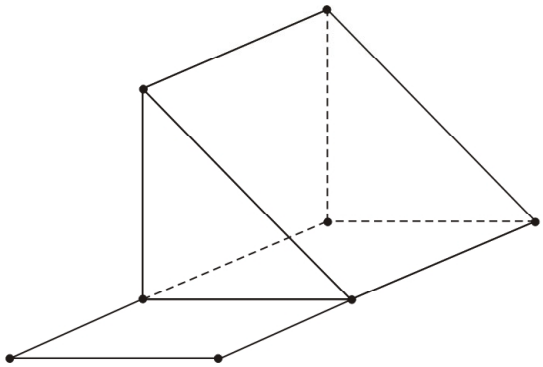


Figura A.13 – Superfície com uma aresta *non-manifold* [9].

A.5.2.3. Conceitos gerais de modelagem

Modelagem geométrica também é uma área com sua própria terminologia. A seguir encontram-se os principais termos de interesse prático para este trabalho e suas definições formais.

Non-manifold, como descrito anteriormente, é um termo de modelagem geométrica referente a situações topológicas que não são *manifold* [17]. Num ambiente de modelagem *non-manifold*, a superfície ao redor de um ponto pode não ser localmente plana, ou seja, não é necessário que o ponto tenha uma vizinhança que seja homeomorfa a um disco aberto. Exemplos de situações topológicas com tais características podem ser vistos nas Figuras A.6, A.7, A.8 e

A.9a. *Representações non-manifold* são aquelas que permitem condições topológicas *non-manifold*.

Segundo a terminologia sugerida por Weiler [17], pode-se classificar as arestas da seguinte forma: *arames* são arestas imersas no espaço sem que façam parte da fronteira de nenhuma face. Uma *lâmina* é uma aresta utilizada somente uma vez na fronteira de uma única face. Uma *aresta manifold* é aquela que é usada exatamente duas vezes na fronteira de uma única face ou exatamente uma vez na fronteira de duas faces. Uma *aresta non-manifold* é aquela que é utilizada três ou mais vezes nas fronteiras de uma ou mais faces. Uma *aresta suporte* (*strut edge*) é uma aresta *manifold* que pertence à fronteira de uma face e que possui um vértice que não tem nenhuma outra aresta incidente. Uma *aresta istmo* (*isthmus edge*) é uma aresta *manifold* que pertence à fronteira de uma única face, mas cujos vértices possuem ambos outras arestas incidentes.

Relações de adjacência são informações a respeito de quais elementos topológicos (e em quem ordem), tais como vértices, arestas e faces, se tocam. Elas são a base da *topologia de adjacência*, que trata das adjacências físicas dos elementos topológicos imersos no espaço ou nas superfícies dos objetos. Este tipo de topologia é o mais utilizado e é o que será estudado neste trabalho.

As *operações booleanas regularizadas*, foco deste trabalho, são aquelas que restringem o resultado da sua aplicação somente a objetos com volumes preenchíveis. Isto significa que estruturas pendentes de mais baixa dimensão não fazem parte deste resultado, apesar de poder haver resultados *non-manifold*.

A.5.3. Tipos de topología

Apesar de no contexto da modelagem geométrica comumente se pensar em topologia como um conjunto de relações de adjacência entre elementos de composição como vértices, arestas e faces, esta abordagem refere-se apenas a um tipo de topologia de conjuntos de pontos possível, a *topologia de adjacência*.

Outros tipos de topologia, como a topologia de nós e a topologia que determina a quantidade de contorções num objeto com ordem (*genus*) maior que zero, também fazem parte deste universo. Quando dois tipos de topologia não possuem nenhuma relação entre as suas informações, diz-se que são topologias ortogonais.

Este trabalho restringe-se ao uso da topologia de adjacência, que já provou ser bastante eficiente em diversas aplicações em modelagem geométrica. Desta forma, as relações de adjacência são extraídas das informações geométricas associadas aos modelos. Pode-se, então, simplificar a terminologia de tal forma que as relações de adjacência entre elementos topológicos passe a ser referida apenas como *topologia* do modelo e as informações geométricas completas, incluindo a descrição matemática das superfícies e curvas e a localização de pontos no espaço, como *geometria* do modelo.

A.5.4. O uso da topologia na modelagem geométrica

Por que o uso da topologia em ambientes de modelagem se faz tão útil? Por que não utilizar as informações geométricas completas para se obter as relações de adjacência desejáveis? Quais são as vantagens de se criar um nível de abstração para manipular os elementos topológicos e analisar a proximidade entre estes elementos?

Perguntas como estas possuem respostas sustentadas pelas idéias de economia de tempo e complexidade dos algoritmos. Quando a topologia se caracteriza como um conjunto de informações unificadas, coerentes e organizadas com alto nível de abstração, pode-se obter, de forma simples e rápida, informações de extrema importância relativas aos elementos topológicos sem haver a necessidade de se fazer uma consulta global à geometria do modelo. Na manipulação de uma pequena porção do objeto em estudo, é bastante útil que se obtenha diretamente as informações sobre as porções adjacentes sem ter que passar por todas as informações associadas ao objeto. Isto melhora de forma significativa a eficiência do sistema de modelagem [17].

Utilizar a topologia como base ou elemento-chave de um sistema de modelagem significa disponibilizar explicitamente as informações topológicas bem como criar uma estrutura de dados e algoritmos que a utilizam baseados nestas informações. A abordagem mais comum organiza os elementos topológicos segundo uma hierarquia decrescente, colocando os elementos dimensionalmente superiores acima dos dimensionalmente inferiores (Figura A.14).

Existem três razões primordiais para justificar o uso da topologia como base de um sistema de modelagem, além daquelas já mencionadas [17].

A primeira diz respeito à estabilidade do sistema de modelagem. Enquanto a geometria de superfícies curvas ainda é um campo de estudo em andamento, de forma que diversas formas de representação matemática dessas superfícies existem ou estão sendo implementadas ou estendidas, a topologia dos elementos que compõem estas superfícies é algo que sofre muito pouca variação de acordo com o tipo de geometria utilizada. Isto significa que um sistema baseado na representação topológica dos elementos precisa sofrer muito poucos ajustes no caso da mudança do tipo de representação geométrica adotado. Pode-se inclusive adotar um sistema com mais de um tipo de enfoque geométrico utilizando-se a mesma topologia.

A segunda diz respeito à quantidade de erros numéricos passíveis de existirem na representação geométrica de superfícies curvas. Como não existe uma uniformização dos processos associados ao uso de tolerâncias e aproximações nos algoritmos geométricos, muitas vezes as relações de adjacência podem ser comprometidas se forem extraídas da geometria do modelo. Desta forma, por problemas de precisão, retalhos de superfície que deveriam ser adjacentes podem apresentar um pequeno espaço entre si. Se as informações de adjacência já são conhecidas no momento da criação do modelo, a combinação da topologia com as técnicas de representação geométrica proporcionam um meio de representar as propriedades desejadas do objeto de forma consistente, a menos, eventualmente, de algumas poucas imprecisões geométricas.

A terceira razão é o fato de que a separação das informações topológicas e geométricas num sistema de modelagem geométrica torna a implementação desse sistema algo mais organizado e sistemático, de forma que a criação, consulta, manipulação e análise de modelos se torna mais simples.

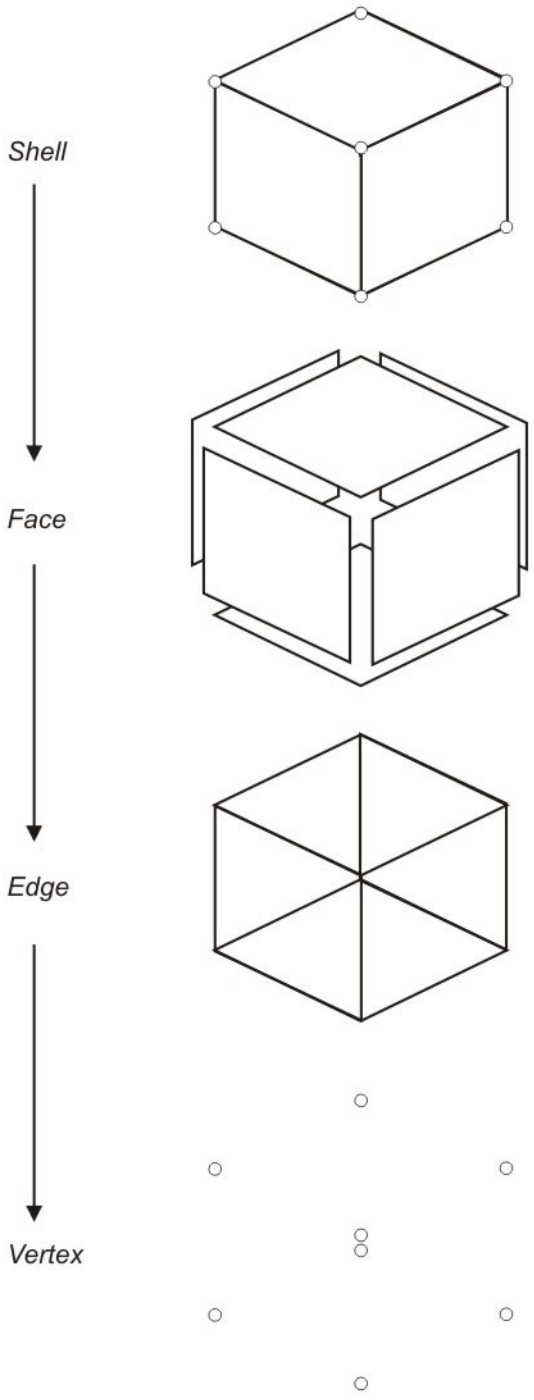


Figura A.14 – Representação hierárquica decrescente dos elementos topológicos [17].

A.5.5.
Suficiência de uma topologia

Na topologia de adjacência que considera vértices, arestas e faces como os três elementos de adjacência, existem nove tipos de relações de adjacência possíveis, mostradas na Figura A.15. Uma representação topológica com informações bastantes para recriar todas estas nove relações sem erros nem ambigüidades pode ser considerada como suficiente. É necessário, para que se possa caracterizar uma representação como sendo suficiente, que se saiba o *domínio* com o qual se vai trabalhar, ou seja, quais são os objetos e processos físicos disponíveis e quais são as limitações ou restrições a que o modelo tem de se submeter.

Não é necessário que um sistema de modelagem armazene todos os tipos de relações de adjacência existentes, pois isto representaria um gasto adicional de espaço que não justificaria a economia de tempo que tal facilidade proporcionaria. Existe um conjunto mínimo suficiente de relações de adjacência que devem ser armazenadas para que todas as outras possam ser derivadas delas de forma eficiente. Determinar tal conjunto é um dos objetivos na criação de um sistema de modelagem. Vários sistemas adotam este conjunto mínimo associado a uma ou mais relações de adjacência diferentes, para que todas as informações necessárias possam estar disponíveis quando se desejar. Isto porque cada relação de adjacência envolve somente dois elementos topológicos, enquanto o número de elementos presentes é de três ou mais. A unicidade de cada elemento topológico é de extrema importância, visto que atributos não-topológicos específicos de cada aplicação podem ser associados a cada elemento. Se a combinação de relações de adjacência escolhida for suficiente, então deixa de haver a necessidade de se confiar na representação geométrica, passível de erros e imprecisões, para se obter as informações topológicas remanescentes.

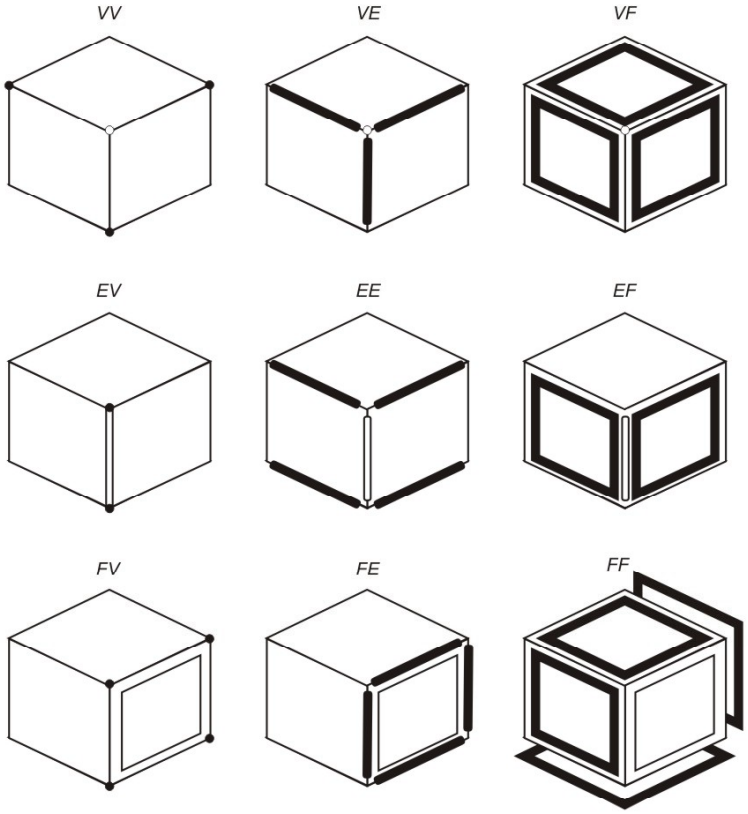


Figura A.15 – Nove relações de adjacência possíveis entre vértices, arestas e faces [17].

A.5.5.1. Topologia suficiente como base de um sistema de modelagem

Quando a topologia é usada como base de um sistema de modelagem, o ideal é que seja completamente independente da parte geométrica desse sistema. Isto porque alterações na representação geométrica não irão provocar mudanças de grande porte na estrutura do sistema. Desta forma, o ideal é que a representação topológica usada como base do sistema seja suficiente. Testes de consistência do modelo também serão mais confiáveis e completos evitando assim imprecisões de ordem geométrica.

Em sistemas de modelagem baseados em representações de fronteira (B-Rep), como é o caso do modelador tridimensional MG utilizado na elaboração deste trabalho, a utilização da topologia de adjacência como base da estrutura de modelagem é altamente recomendável, pois permite a implementação de algoritmos mais simples e eficientes. É crucial, entretanto, que a representação topológica usada seja suficiente, de forma a tornar a estrutura do sistema independente das informações geométricas.

A.5.5.2.
Domínio topológico

Uma representação topológica envolve não somente as informações que serão armazenadas e as estruturas de dados utilizadas para organizar estas informações, mas também os tipos de procedimentos que serão disponibilizados no ambiente de modelagem para lidar com estas informações. Um dos passos mais importantes na implementação de um sistema de modelagem é a definição do domínio topológico onde os modelos serão criados e manipulados [17].

O *domínio* é o conjunto de possibilidades para as quais a representação é válida. Ou seja, o domínio é a especificação dos tipos de objetos e tipos de processos que poderão existir no ambiente a ser criado. É o primeiro passo a ser dado na idealização de qualquer sistema de modelagem, de forma a satisfazer as necessidades a que se propõe este sistema. Considerando a quantidade de esforço requerido para construção de um sistema de modelagem robusto, não se pode correr o risco de basear sua implementação numa estrutura de representação que é insuficiente sobre o domínio que pretende cobrir.

A *precisão* de uma representação depende da completa especificação do domínio para o qual ela se destina a ser útil, da prova de suficiência deste domínio e da existência de operadores que comprovadamente cubram todo o domínio sem que sejam capazes de criar ou manipular as informações fora do domínio da representação.

O domínio deve ser especificado da forma mais completa possível, definindo-se um ambiente inicial seguido de uma série de restrições àquele ambiente. As restrições podem ser:

- *restrições representacionais*, que limitam as condições topológicas dos modelos, basicamente informando o que pode ou não ser representável naquele domínio. Como exemplo, podem-se citar restrições quanto ao número de vazios internos que pode haver num sólido, quanto ao número de *loops* existentes em cada face ou quanto à ordem (*genus*) de uma superfície.
- *restrições procedurais*, que limitam a forma como as condições topológicas serão representadas, sem restringir propriamente o que é ou não representável. Por exemplo, pode-se limitar a quantidade de alças presentes numa face como sendo zero, sem significar que uma face com alças não pode ser representável, apenas que a alça deve possuir uma fronteira bem definida. Deste modo, qualquer objeto não-

representável a princípio pode ser transformado em algo representável sem alterar a forma do objeto.

A.5.5.3. Tipos de suficiência

Suficiência topológica é a capacidade de representar de forma completa e não ambígua as adjacências topológicas de um modelo qualquer. A completude é a habilidade de se obter qualquer relação de adjacência desejada a partir das informações diretamente disponíveis e a não ambigüidade é a propriedade que determina a existência de um único modelo a partir de um conjunto de relações de adjacência pré-estabelecido, ou seja, é uma relação biunívoca entre a representação topológica e o objeto que está sendo representado.

Pode-se classificar a suficiência topológica em dois tipos: teórica e prática. A suficiência teórica é o mínimo de informação necessária para representar sem ambigüidades uma topologia de adjacência completa, e a suficiência prática é o mínimo necessário numa representação prática de modelagem geométrica.

Em termos de suficiência teórica, deve-se estabelecer um conjunto de relações de adjacência que possua a habilidade de representar exata e completamente a topologia de um grafo mapeado, não permitindo o uso da geometria dos elementos para derivar alguma informação topológica adicional. Combinações de relações de adjacência individualmente insuficientes podem ser utilizadas para alcançar a suficiência da representação, desde que envolvam todos os tipos de elementos topológicos utilizados.

Uma das questões mais importantes a ser levantada quando se fala em suficiência prática de uma representação topológica é a identificação ou rotulação (*labeling*) individual dos elementos topológicos presentes. Isto para que informações adicionais não ligadas à topologia destes elementos possam ser associadas a cada elemento de acordo com a aplicação. Weiler [17] afirma que uma representação topológica incluindo *n* tipos de elementos para os quais *labels* são requisitados deve permitir a derivação de pelo menos *n-1* relações de adjacência envolvendo todos os *n* tipos de elementos. Isto é interessante pois mostra que em termos práticos não há diferença entre se selecionar um subconjunto de *n-1* relações de adjacência individualmente insuficientes ou relações de adjacência individualmente suficientes (desde que envolvam todos os n tipos de elementos), já que *n-1* relações de adjacência serão necessárias de qualquer forma.

Por exemplo, em um ambiente de modelagem contendo os três tipos básicos de elementos topológicos (vértices, arestas e faces), é necessário que se possa derivar pelo menos duas relações de adjacência para que se possa inter-relacionar todos os três tipos de elementos, já que cada relação de adjacência pode individualmente referir-se a dois tipos de elementos no máximo.

A.5.6. Topologia em representações de sólidos *manifold*

Em modelagem geométrica *manifold*, as devidas restrições devem ser impostas ao domínio representacional das possibilidades topológicas para que apenas sólidos com superfícies *manifold* sejam fisicamente representáveis sem ambigüidades.

Num espaço Euclidiano tridimensional, apenas sólidos com superfícies *manifold* compactas e orientáveis podem ser gerados. Objetos *non-manifold* como os da Figura A.7 são excluídos do tipo de representação aqui exposto, portanto.

Em processos práticos industriais, por exemplo, a representação do tipo *manifold* se faz bastante útil, já que objetos *non-manifold* com arestas ou faces pendentes não caracterizam formas comumente encontradas nestes processos, pelo fato de tais elementos representarem estruturas de dimensão inferior infinitamente delgadas não manufaturáveis ou produzíveis mecanicamente. Desta forma, apenas objetos com volumes sólidos definidos são desejados.

Restrições como estas influenciam na seqüência de operações de modelagem permitidas, já que em nenhum passo intermediário do processo de modelagem deve haver objetos *non-manifold*.

A habilidade de representar grafos de fronteira como pseudografos que permitam *self-loops* e multigrafos é desejável porque situações como estas ocorrem naturalmente como resultado de operações de modelagem aplicadas aos objetos *manifold*, particularmente aquelas envolvendo as operações booleanas (ver Figura A.16). Tais situações podem ser contornadas dividindo-se cada *self-loop* e multigrafo em mais de uma aresta, contudo isto requer um trabalho computacional bem maior para identificar problemas como estes e resolvê-los, de forma que uma das ferramentas mais poderosas dos sistemas de modelagem de sólidos que é a sua capacidade de preservar e rapidamente disponibilizar informações sobre a consistência topológica das superfícies representadas fica comprometida. Isto porque o número de elementos

necessários para representar objetos como estes cresce desnecessariamente, diminuindo a eficiência do sistema de modelagem.

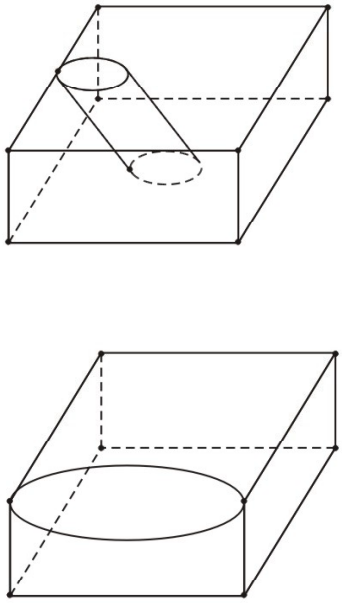


Figura A.16 – *Self-loops* e multigrafos gerados pela aplicação das operações booleanas a sólidos: a) *self-loop* criado pela subtração de um cilindro de um paralelepípedo; b) multigrafo criado pela subtração de uma esfera de um paralelepípedo [17].

A restrição geométrica mais importante na implementação da geometria relativa a uma topologia *manifold* é que as superfícies *manifold* não podem se interceptar a não ser nas suas fronteiras. Isto é necessário para manter a superfície homeomorfa a um disco aberto como prescrito na definição de um *manifold*. Toda imersão de um grafo numa superfície deve ser um *manifold* orientável num espaço Euclidiano tridimensional. Isso significa que não é permitido que alguma face se auto-intercepte ou intercepte outras faces, obrigando a topologia de adjacência a conter todas as informações de interseção através das informações de adjacência.

As faces não devem possuir alças, para garantir que um número arbitrário de alças não seja adicionado à superfície de um sólido sem mudar a estrutura do seu grafo de fronteira, forçando a topologia de adjacência a conter toda a informação sobre ordem (*genus*), e mantendo a validade da fórmula de Euler-Poincaré. Toda face deve ser mapeável num plano, ou seja, deve ser topologicamente plana, não podendo conter alças. Se alças pudessem ser acrescentadas arbitrariamente às faces, informações globais como a ordem da superfície deveriam estar contidas na descrição geométrica da mesma, ao invés

de fazerem parte da estrutura topológica. Desta forma, estar-se-ia abrindo mão de uma das grandes vantagens dos sistemas topologicamente estruturados, que é a de não necessitarem da consulta das informações geométricas do modelo para poderem caracterizar completamente o mesmo.

Em termos gerais, *manifolds* compactos e orientáveis e grafos imersos e conexos caracterizam as principais restrições para garantir um domínio topologicamente suficiente para representações *manifold* e a validade da equação de Euler-Poincaré.

A.5.7. Topologia em representações de sólidos *non-manifold*

Situações *non-manifold* aparecem naturalmente como resultados de operações de modelagem aplicadas a sólidos, sobretudo como resultado das operações booleanas, ainda que os sólidos com os quais se trabalhe sejam *manifold*. Representações *non-manifold* permitem a análise das estruturas internas dos objetos e uma representação uniforme de quaisquer combinações das formas de modelagem já estudadas, por arames, por superfícies, modelagem de sólidos e modelagem *non-manifold*.

O desenvolvimento de sistemas de modelagem que trabalhem com representações *non-manifold* é relativamente recente. A implementação de topologias de adjacência para domínios *non-manifold* ainda não é uma área totalmente explorada. Ainda que resultados *non-manifold* provenientes da aplicação das operações booleanas já tivessem sido observados e operadores de baixo nível (denominados operadores de Euler, que serão analisados na próxima seção) para manipular as informações topológicas de forma a manter a consistência do modelo já tivessem sido idealizados para condições *non-manifold*, muito ainda há para explorar neste ramo da modelagem geométrica. Trabalhos relacionados com este tipo de representação podem ser encontrados em [12,13,14,15,16,17,18].

Algumas áreas de aplicação da modelagem geométrica se utilizam da representação *non-manifold*, usufruindo das vantagens adicionais oferecidas para ampliar o conjunto de objetos e processos de modelagem disponíveis de forma a diversificar o universo representacional dos modelos gerados.

Na área de *modelagem*, o fato de uma representação *non-manifold* agregar todas formas de modelagem pré-existentes facilita na transição de modelos gerados por arames para modelos gerados por superfícies e destes

para modelos sólidos, incluindo a detecção de regiões sem a necessidade de reestruturação interna. Representações *non-manifold* também permitem o armazenamento de informações geométricas arbitrárias, como eixos de simetria e planos de corte, juntamente com a descrição da forma do objeto, num único modelo. Objetos heterogêneos constituídos de vários materiais, como os utilizados em aeronaves e outras aplicações, podem ser modelados com relações de adjacência explicitamente disponíveis no modelo. Além disso, implementações fechadas das operações booleanas são possíveis.

Na área de *análise*, malhas de elementos finitos para aplicações que utilizam o MEF podem ser geradas na mesma representação que a original utilizada na geração dos modelos, de tal modo que pode haver uma intercomunicação entre o modelador e os resultados obtidos para que o modelo seja atualizado automaticamente. Isto significa que existe a possibilidade de se implementar ferramentas de criação e análise simultânea do modelo, otimizando o processo. Resultados *non-manifold* das operações booleanas são representáveis e pode-se efetuar a análise de pontos, curvas, áreas e volumes de *overlap* (que se interpenetram).

Objetos heterogêneos podem ser representados diretamente. Isto significa que regiões com volumes comuns, áreas com fronteiras comuns, faces coincidentes e outros tipos de situações topológicas como estas podem ocorrer. Estruturas internas dos objetos podem ser diretamente representadas. Esquemas de representação de regiões que correspondem a interseções de regiões constituídas de materiais diferentes podem ser criados, como por exemplo a anexação de um atributo à região que estabeleça qual material predomina na operação de interseção.

Weiler [17] descreve o domínio topológico e geométrico de uma representação *non-manifold* de forma completa, proporcionando assim uma visão geral dos objetos e processos físicos que podem ser representados num sistema de modelagem baseado em representação de fronteira (B-Rep) e que tenha como elemento-chave a topologia de adjacência dos elementos presentes. Como este trabalho desenvolve-se num ambiente de modelagem *non-manifold* com as características mencionadas, vale ressaltar alguns pontos relativos a este domínio para que se possa vislumbrar o universo representacional dentro do qual as operações booleanas foram implementadas.

- *Superfícies non-manifold* – adota-se uma representação topológica *non-manifold* que englobe todas as formas de modelagem pré-existentes: por arames, por superfícies e modelagem de sólidos, com a

única restrição de que todas as informações sobre os elementos topológicos e sobre as interseções entre eles sejam representadas explicitamente na estrutura de grafo imersa num espaço Euclidiano tridimensional. As operações booleanas podem ser implementadas como um conjunto fechado de operações e estruturas internas dos objetos podem ser diretamente representáveis.

- *Faces manifold* – Uma face é definida como a porção conexa e limitada de uma superfície, sem incluir a sua fronteira. Enquanto a superfície como um todo pode ser *non-manifold*, cada face de um objeto deve ser *manifold*, ou seja, nenhuma face pode se auto-interceptar ou interceptar outras faces a não ser em suas fronteiras. Uma face *non-manifold* pode ser representada garantindo a existência de uma fronteira ao longo de todos os pontos e curvas *non-manifold*.
- *Faces mapeáveis num plano* – É requerido que toda face seja mapeável num plano sem que seja necessário cortá-la ou criar novas fronteiras na mesma. Isto força a estrutura topológica a armazenar toda informação sobre ordem (*genus*). Pode-se dizer que não é permitido que faces contenham alças.
- *Propriedades de não-interseção* – Regiões e faces não podem se interceptar a não ser ao longo de suas fronteiras. Arestas não podem se interceptar a não ser em seus pontos extremos nem podem interceptar faces a não ser ao longo de suas fronteiras. Vértices devem ser distintos espacialmente. Pode-se dizer que dois elementos topológicos só podem se interceptar num nível de hierarquia pelo menos um nível abaixo do nível do elemento hierarquicamente inferior.
- *Finitude* – Vértices estão em posições finitas do espaço, arestas são finitas em comprimento, faces são finitas em sua área de superfície e regiões fechadas são finitas em volume. Pode-se dizer que as formas permitidas devem ser representadas por um número finito de elementos topológicos.
- *Pseudografos* – Pseudografos são permitidos, o que significa que pode haver *self-loops* e multigrafos. Isto permite a existência de arestas curvas sem restrições na sua geometria.
- *Grafos desconexos* – Grafos desconexos são permitidos. Isto permite faces com múltiplas fronteiras desconexas e objetos com um ou mais vazios internos.

- *Grafos rotulados* – Todos os grafos são rotulados, ou seja, todos os elementos topológicos possuem uma identificação única, que permite que informações não-geométricas e não-topológicas sejam associadas a estes elementos.

A.5.8. Estruturas de dados topológicas

Diversas formas de implementação da topologia associada a modelos geométricos tridimensionais já foram realizadas. Algumas delas, que serão analisadas com mais detalhes nesta seção, consagraram-se ao longo do tempo no campo da modelagem geométrica por caracterizarem-se como formas concisas, organizadas, robustas, eficientes e suficientes de representação das informações topológicas de adjacência associadas aos modelos analisados. Cada uma delas se faz útil em domínios que podem ser *manifold* ou *non-manifold*.

Os três principais tipos de elementos topológicos – vértices, arestas e faces – e as informações geométricas associadas a eles constituem as bases de uma estrutura de dados baseada numa representação de fronteira. Todos os modelos de fronteira representam faces em termos de nós explícitos de uma estrutura de dados. A partir daí, muitas alternativas para representar a geometria e a topologia de um modelo de fronteira são possíveis.

Neste trabalho, uma importância maior é dada aos modelos baseados em arestas. Como referência para modelos baseados em vértices pode-se citar a estrutura *G-Map* de Lienhardt [15,54].

A.5.8.1. Modelos de fronteira baseados em polígonos

Um modelo de fronteira que possua apenas faces planas é chamado de modelo poliédrico. O fato de todas as arestas serem segmentos de reta simplifica bastante a implementação de uma estrutura de dados baseada nesta representação. No caso mais simples, faces são representadas como polígonos, cada um constituído de uma seqüência de coordenadas espaciais. Um sólido consiste de um conjunto de faces agrupadas, por exemplo, através de uma tabela de identificadores de faces ou através de uma lista encadeada de nós de faces [10].

Como variante, pode-se considerar os vértices como entidades independentes da estrutura de dados de fronteira. Isto significa que os vértices podem ser implementados como estruturas que armazenam as suas coordenadas espaciais bem como outras informações associadas às arestas e faces adjacentes a eles. Desta forma, identificadores de vértices ou ponteiros para nós de vértices podem ser associados às faces, de modo que cada face contenha uma lista de vértices que a defina. Esta lista deve estar ordenada, por exemplo, no sentido horário (ou anti-horário) dos vértices, vistos por um observador olhando de fora do sólido que contém a face.

Um fato interessante deste tipo de abordagem é que não existe informação alguma de superfície associada às faces; por elas serem planares, suas geometrias estão totalmente determinadas pelas coordenadas dos seus vértices. Se equações das faces forem necessárias para algum algoritmo geométrico, elas podem ser calculadas.

A escolha das informações que serão armazenadas explicitamente e das que serão deixadas como implícitas (ou seja, para serem calculadas quando necessário) é uma questão importante na implementação da estrutura de dados. Pode-se, como mencionado, armazenar os vértices (e conseqüentemente suas coordenadas) associados às faces e obter as equações destas faces quando necessário, ou armazenar as equações das faces explicitamente deixando as coordenadas dos vértices como informações implícitas, o que caracteriza um tipo de modelagem peculiar apropriada para objetos convexos, apenas.

A.5.8.2. Modelos de fronteira baseados em arestas

Se superfícies curvas estão presentes no modelo, a inclusão explícita de nós de arestas na estrutura de dados torna-se útil, seja para armazenar informações das curvas de interseção de faces, ou porque nós de arestas são úteis para se gravar relações de adjacência. Em modelos de fronteira baseados em arestas, a fronteira de uma face é representada em termos de uma seqüência fechada e orientada de arestas, ou seja, um *loop*. Os vértices da face são obtidos através das arestas, igualmente.

Cada aresta possui uma *orientação*, que se caracteriza pela ordenação dos seus vértices (inicial e final), e as faces são novamente orientadas segundo uma lista ordenada no sentido horário ou anti-horário das suas arestas segundo

um observador externo ao sólido. Cada aresta pertence simultaneamente a duas faces, com orientações distintas em cada uma delas.

A.5.8.2.1. A estrutura de dados *Winged-edge*

O esquema formalizado mais antigo de representação da fronteira de um poliedro e da sua topologia é a estrutura de dados *winged-edge*, desenvolvida originalmente por Baumgart [55] para aplicação na área de visão computacional em robótica. Ela descreve objetos poliedrais *manifold* através do armazenamento de informações sobre arestas, vértices e faces.

As informações topológicas armazenadas na estrutura da *winged-edge* para cada aresta é composta de adjacências da aresta em relação a outras arestas, vértices e faces. O nome "*winged-edge*" resulta da aparência gráfica da arestas adjacentes quando desenhadas em relação à aresta de referência (Figura A.17). Para que estas referências aos elementos topológicos adjacentes possam ser armazenadas, é necessário se estar trabalhando num ambiente de modelagem rotulado, onde cada um dos três elementos são unicamente identificados.

A inclusão de nós explícitos que armazenem informações sobre os três elementos básicos faz com que a *winged-edge* seja uma estrutura de dados mais elaborada. Por exemplo, para se implementar algoritmos como os de remoção de faces ocultas e sombreamento, informações explícitas sobre vizinhança entre faces pode ser armazenada associando-se cada aresta com os identificadores das duas faces que ela separa.

A importância da orientação de arestas e faces se faz nítida na implementação da estrutura da *winged-edge*. Isto porque levam-se em conta as arestas imediatamente anteriores e imediatamente posteriores à aresta de referência em relação às duas faces que ela limita. Além disso, a aresta de referência é usada uma única vez em seu sentido positivo (do vértice inicial para o vértice final) e uma única vez no seu sentido negativo (do vértice final para o inicial), uma para cada face que ela separa. Pode-se atribuir os qualificativos *face esquerda* e *face direita* referentes àquela aresta orientada baseando-se num observador externo ao sólido que tais faces delimitam.

As informações topológicas são estruturadas da seguinte maneira: cada face precisa armazenar somente um identificador de uma aresta arbitrária e um *flag* (indicador) que indique a orientação desta aresta naquela face (positiva ou

negativa). Como cada vértice é adjacente a um ciclo ordenado de arestas, cada estrutura de vértice também necessita armazenar somente um identificador de uma aresta qualquer deste ciclo. Por fim, cada estrutura de aresta armazena as seguintes informações: vértices incidentes (identificados por vértice *inicial* e vértice *final*), faces adjacentes *esquerda* e *direita* (ou *horária* e *anti-horária*), arestas antecessoras e sucessoras no sentido horário e arestas antecessoras e sucessoras no sentido anti-horário. As informações geométricas normalmente se resumem às coordenadas dos vértices e das equações das faces, de tal forma que a normal a cada face esteja apontada para fora do sólido. Pode haver informações geométricas como as equações paramétricas das curvas que contêm as arestas no espaço tridimensional, dentre outras, no caso de sólidos com superfícies curvas.

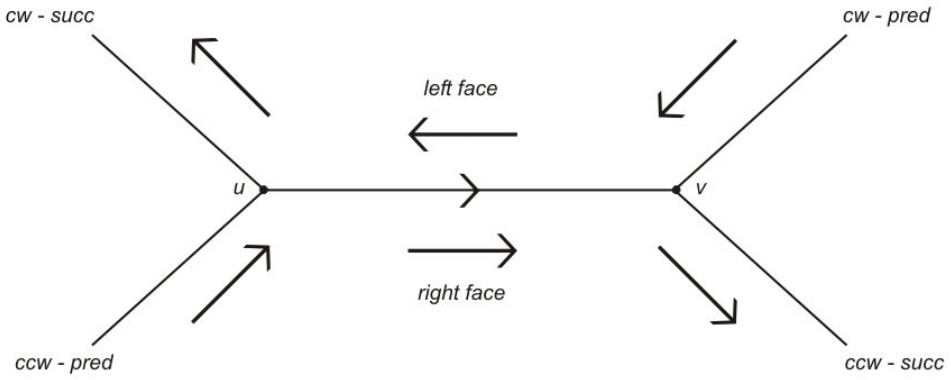


Figura A.17 – Estrutura de dados *winged-edge* [9].

Weiler [17] discute sobre uma outra estrutura de dados chamada *winged-edge modificada*, que é praticamente idêntica à *winged-edge*, com a diferença de que cada ponteiro para aresta adjacente (anteriores e posteriores, no sentido horário ou anti-horário) são acompanhados por campos que indicam exatamente que lado da aresta apontada está sendo referido (por exemplo, indicando o sentido positivo ou negativo daquela aresta). Weiler propõe ainda duas outras estruturas de dados baseadas em aresta, *vertex-edge* e *face-edge*, e demonstra a suficiência de cada uma destas quatro formas de representação.

A.5.8.2.2.
Extensões para grafos desconexos

As estruturas de dados mencionadas acima assumem que os grafos imersos de fronteira utilizados são grafos conexos. Esta restrição pode ser retirada para que objetos que contenham faces com múltiplas fronteiras e vazios internos possam ser representados. Para isso, é necessária a extensão das estruturas de dados apresentadas, com a criação de novos tipos de elementos topológicos para representar tais objetos.

Existem várias maneiras de lidar com objetos que possuem faces multiplamente conexas, ou seja, faces que contenham mais de um contorno de fronteira mas que ainda possuam uma única área de superfície conexa. Uma destas maneiras é a criação de uma aresta artificial, ou aresta auxiliar, que une um contorno de fronteira a outro contorno da mesma face (Figura A.18). Esta aresta auxiliar possui a mesma face adjacente em ambos os lados. Este artifício não é desejável pois além de necessitar que o sistema de modelagem possua algoritmos que determinem explicitamente a maneira exata de conectar os contornos através das arestas auxiliares, aumenta consideravelmente a quantidade de arestas presentes no modelo, o que pode ser ruim quando este é submetido a operações que subdividam as arestas auxiliares, como por exemplo as operações booleanas.

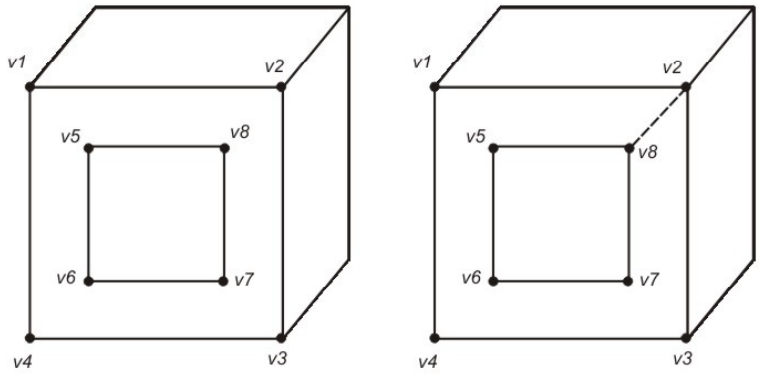


Figura A.18 – Aresta auxiliar para conectar contornos separados de uma face [17].

Uma alternativa bastante comum é a implementação de uma estrutura de dados de *loop*, que representa uma seqüência ordenada de arestas que pertencem a um contorno de uma face. Desta forma, cada face guarda uma lista

de *loops* referentes a todos os contornos associados àquela face. Cada *loop* armazena um identificador de uma aresta (ou semi-aresta, como será definido mais tarde quando outras estruturas de dados forem apresentadas) pertencente ao seu ciclo. As próprias informações armazenadas na aresta são suficientes para se obter toda a definição do contorno a que pertence. Naturalmente, ponteiros guardados na estrutura de aresta que antes apontavam para faces passam agora a apontar para *loops*.

No caso de objetos que possuem um ou mais vazios internos, mas que ainda constituem um único volume conexo, uma outra abordagem torna-se necessária. Diferentemente das faces multiplamente conexas descritas acima, mais de uma superfície precisa ser representada. Isto requer mudanças na estrutura de dados acima do nível de face, sendo este problema normalmente solucionado considerando-se uma lista de cascas separadas para um objeto. Um tipo de implementação bastante eficiente é a criação de árvores binárias de cascas em que cada nó possui um ramo contendo as cascas que são interiores àquela contida no nó de referência e outro ramo contendo as cascas que são exteriores à mesma. Isto permite a representação de sólidos com vários vazios, sólidos contidos nestes vazios, vazios contidos dentro destes sólidos, e assim por diante.

A.5.8.2.3. A estrutura de dados *Half-edge*

Para muitas aplicações, é necessário que se tenha uma estrutura de dados capaz de representar modelos não-intuitivos, para que se possa armazenar todos os passos intermediários de uma seqüência de operações utilizadas na descrição de um modelo. Na verdade, se modelos não-intuitivos não puderem ser representados, as operações de manipulação de objetos presentes num sistema de modelagem seriam de uso bastante limitado. A própria criação de novas entidades como polígonos, arestas e vértices dentro de um modelo pré-existente pode levar a casos especiais que precisam ser representados. Do mesmo modo, um algoritmo que permita a realização das operações booleanas num sistema de modelagem depende da criação de modelos intermediários muito pouco intuitivos quando manipula estruturas de dados baseadas em representações de fronteira (B-Rep), como é o caso do algoritmo proposto neste trabalho. Uma estrutura de dados completa também precisa de que sejam armazenadas as informações geométricas do modelo, que podem ser somente

as coordenadas do vértices no caso de modelos poliedrais, ou equações de curvas e superfícies no caso de modelos com superfícies curvas.

A estrutura de dados *half-edge* [10] é uma variação da estrutura *winged-edge* apresentada anteriormente. Ela é implementada segundo uma hierarquia de cinco níveis topológicos: sólido, face, *loop*, *half-edge* e vértice. A seguir são apresentadas as definições e características destes elementos para a estrutura de dados *half-edge* [10]:

- *Sólido* – O nó sólido constitui a raiz de toda a estrutura de dados da *half-edge*. A partir do ponteiro para um sólido, pode-se acessar quaisquer elementos da estrutura topológica, como faces, arestas (que constituem um nó à parte da estrutura de dados, como será exposto a seguir) e vértices, através de listas duplamente encadeadas. Todos os sólidos também são agrupados por uma lista duplamente encadeada, ou seja, cada sólido possui ponteiros para o sólido anterior e o sólido posterior.
- *Face* – O nó face representa uma face planar dos objetos representados pela estrutura de dados *half-edge*. Numa implementação mais completa e abrangente desta representação, as faces podem conter múltiplas fronteiras, de forma que cada face é associada a uma lista de *loops*, cada um representando uma fronteira da face. Como todas as faces são planares, um dos *loops* pode ser chamado de fronteira *externa*, enquanto os outros representam os *buracos* de uma face. Isto pode ser feito através de dois ponteiros, um dos quais aponta para o *loop* externo e outro que aponta para o primeiro *loop* na lista duplamente encadeada que engloba todos os *loops* da face.
- *Loop* – o nó *loop*, como exposto acima, representa uma fronteira conexa de uma face. Possui um ponteiro para a face que o contém, um ponteiro para uma das *half-edges* que formam a sua fronteira e ponteiros para os *loops* anterior e posterior daquela face.
- *Half-edge* (*semi-aresta*) – O nó *half-edge* descreve uma linha que forma um *loop*. Consiste num ponteiro para o *loop* que o contém e um ponteiro para o vértice inicial da linha na direção do *loop*. Possui também ponteiros para as *half-edges* anterior e posterior daquele *loop*, formando uma lista duplamente encadeada de *half-edges* de um *loop*. Desta forma, o vértice final de uma linha é tido como vértice inicial da próxima *half-edge*.

- *Vértice* – O nó vértice contém um vetor de 4 números reais que correspondem às coordenadas homogêneas de um ponto no espaço Euclidiano tridimensional. Há também dois ponteiros para os vértices anterior e posterior formando uma lista duplamente encadeada dos vértices de um sólido.

A Figura A.19 dá uma visão geral da estrutura de dados *half-edge*. Para que esta estrutura esteja completa, no entanto, é necessário introduzir mais um nó, para que as informações topológicas formem um conjunto unificado e não-ambíguo na representação dos objetos. O conceito de semi-aresta (*half-edge*) já está bem definido, mas é necessário que se crie um nó de aresta (*edge*) para que se possa avaliar as relações entre as faces de um sólido sem que para isso elas tenham que fazer referência a um mesmo nó de vértice. O nó *aresta* associa as duas semi-arestas entre si. Ela combina as duas metades de uma aresta para formar a aresta em si. Possui ponteiros para as duas semi-arestas (esquerda e direita, por assim dizer). Há também uma lista duplamente encadeada de arestas, com ponteiros para a aresta anterior e posterior.

Em relação aos elementos anteriormente citados, deve-se acrescentar um ponteiro na estrutura da *half-edge* para a aresta que a contém e um ponteiro na estrutura de vértice para uma das *half-edges* que emana do mesmo. A Figura A.20 mostra um esquema da relação entre as arestas e semi-arestas.

O caso especial de um *loop* vazio (constituído por somente um vértice, mas nenhuma aresta) pode ser representado. Para isto, na estrutura de dados *half-edge* o ponteiro para vértice aponta para este vértice único e o ponteiro para aresta é nulo. Os ponteiros para as semi-arestas anterior e posterior apontam para a própria semi-aresta em questão.

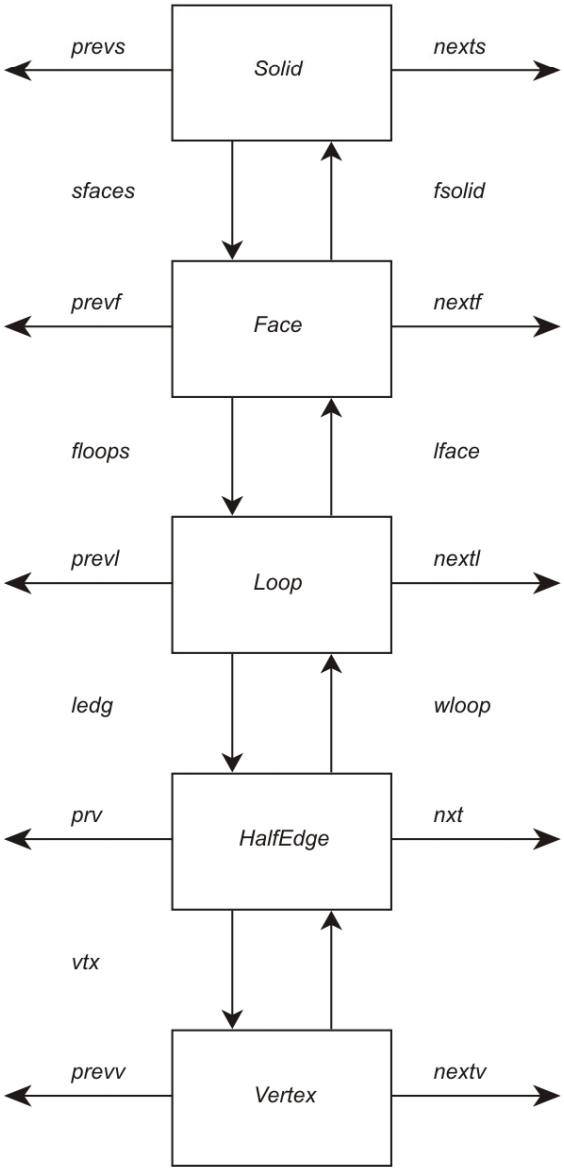


Figura A.19 – Estrutura de dados *half-edge* [10].

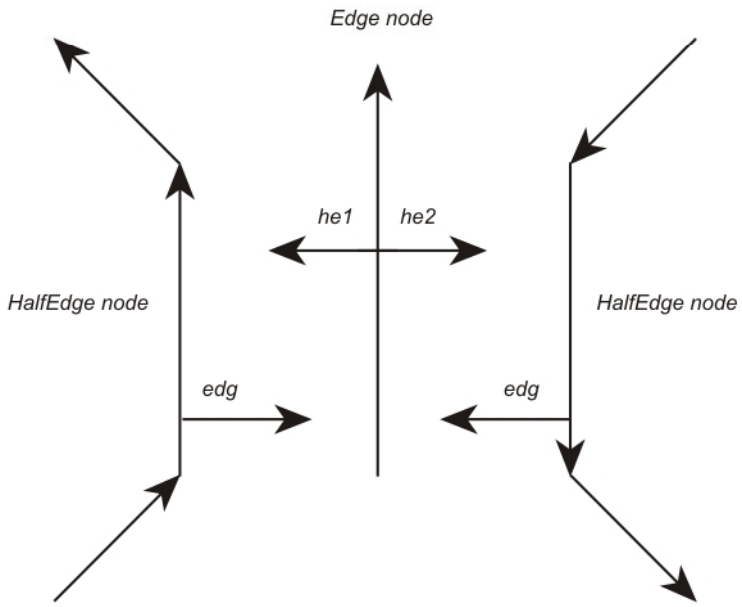


Figura A.20 – Relação entre aresta e semi-aresta na estrutura de dados *half-edge* [10].

A.5.8.2.4. A estrutura de dados *Radial Edge*

As estruturas de dados apresentadas até agora caracterizam-se pela sua utilidade para representar objetos em ambientes de modelagem *manifold*. Contudo, deseja-se estudar mais detalhadamente ambientes cujo domínio representacional não se restringe a superfícies *manifold*, ou seja, deseja-se analisar representações *non-manifold*.

O domínio *non-manifold* pode ser qualificado como mais amplo que o domínio *manifold*, simplesmente porque este último se restringe a representar junções de superfícies que são topologicamente bidimensionais, enquanto o domínio *non-manifold* suporta junções mais complexas. Contudo, várias similaridades também podem ser encontradas. Muitas questões no projeto de sistemas de modelagem *manifold* também existem no projeto de um sistema *non-manifold*, não apenas no mesmo nível dimensional, mas também de maneira semelhante em níveis mais altos.

Primeiramente convém apresentar e definir os elementos topológicos envolvidos numa representação *non-manifold* baseada numa representação de fronteira (B-Rep). Alguns destes elementos já foram definidos para ambientes de modelagem *manifold*, sendo necessário apenas acrescentar às suas descrições

as características adicionais atribuídas aos mesmos para condições *non-manifold*. Outros elementos são inteiramente novos, e serão completamente definidos frisando-se sua importância na simplificação e eficiência dos algoritmos que manipulam as relações de adjacência entre os elementos [17].

Um *modelo* é um espaço topológico tridimensional único, consistindo de uma ou mais regiões do espaço. É o nó raiz de toda estrutura de dados, de onde se pode acessar qualquer elemento topológico presente num modelo geométrico.

Uma *região* é um volume no espaço. Existe sempre pelo menos uma região num modelo. Apenas uma região num modelo pode ter extensão infinita, todas as outras possuem extensão finita, e quando mais de uma região existe num modelo, todas elas têm fronteira.

Uma *casca* é uma porção conexa da fronteira orientada de uma região. Uma única região pode ter várias cascas, como no caso de um sólido com vazios internos. Uma casca pode ser um conjunto conexo de faces que formam uma região fechada ou pode ser um conjunto aberto de faces adjacentes, um arame, uma combinação destes ou mesmo um único ponto.

Um *loop* é uma fronteira conexa de uma única face. *Loops* normalmente são seqüências alternadas de arestas e vértices, mas podem ser um único vértice. *Loops* também são orientáveis mas não orientados, já que eles limitam uma face que pode ser usada por duas cascas ou mais. Logo, é o *uso* de um *loop* que é orientado.

Uma *aresta* é uma porção de um *loop* entre dois vértices. Topologicamente, uma aresta é uma curva de fronteira que pode servir como parte da fronteira de um *loop* para uma ou mais faces que se encontram naquela aresta. Uma aresta é orientável, embora não orientada. É o *uso* de uma aresta que é orientado.

Um *vértice* é um ponto topológico único no espaço. Um único vértice pode servir como fronteira de uma face ou como fronteira de uma casca completa.

A seguir são definidos os quatro elementos topológicos chamados de *usos* de alguns elementos já descritos: faces, *loops*, arestas e vértices. O uso pode ser visto como a ocorrência de um elemento topológico em um relacionamento de adjacência com um elemento de dimensão superior.

Um *uso de face* (*face-use*) é um dos dois usos (lados) de uma face. Pode ser descrito como o uso de uma face por uma casca. Usos de faces são orientados com respeito à geometria da face.

Um *uso de loop* (*loop-use*) é um dos usos de um *loop* associado com um dos dois usos de uma face. É orientado com respeito ao uso de face associado a ele.

Um *uso de aresta* (*edge-use*) é uma curva de fronteira orientada de um uso de *loop* associado a um uso de face. Representa o uso da aresta por aquele uso de *loop*, ou, para o caso de um arame, pelos seus vértices extremos. A orientação é especificada de acordo com a geometria da aresta.

Um *uso de vértice* é uma estrutura que representa o uso de um vértice por uma aresta como um ponto extremo, por um *loop* no caso deste ser constituído por um único vértice, ou por uma casca no caso desta ser constituída por um único vértice. Pode-se dizer que os usos de vértices são utilizados para captarem condições *non-manifold* nos vértices.

Uma estrutura de dados que represente diretamente os *usos* dos elementos topológicos tais como faces, *loops*, arestas e vértices, da forma como foi descrito acima, simplifica o acesso aos elementos desejados eliminando a necessidade de procedimentos de tomada de decisões durante um laço que percorra os elementos topológicos. Os usos correspondem a posicionamentos específicos na lista de adjacências, logo são unicamente definidos, e como tal não dão margem a ambigüidades.

Condições *non-manifold* em arestas e vértices aparecem muitas vezes como resultados de operações comuns de modelagem, como as operações booleanas, ainda que os objetos sobre os quais se esteja operando sejam *manifold*. Casos comuns de arestas *non-manifold* são aqueles em que mais de duas faces possuem uma aresta em comum, e sólidos conectados por um único vértice caracterizam tal vértice como sendo um vértice *non-manifold* (Figura A.21).

A estrutura de dados *Radial Edge*, proposta por Weiler [17], é conhecida por este nome porque ela armazena explicitamente a lista de faces ordenadas radialmente ao redor de uma aresta (Figura A.22). Algumas características desta estrutura são apresentadas a seguir:

As relações de adjacência hierarquicamente decrescentes (de elementos dimensionalmente superiores para elementos dimensionalmente inferiores) e as relações de adjacência hierarquicamente crescentes (dos elementos dimensionalmente inferiores para os dimensionalmente superiores) são diretamente representadas nas estruturas de dados.

A relação de adjacência que consiste na lista desordenada de arestas incidentes num vértice é representada no intuito de capturar as adjacências de

dois volumes separados que se tocam num único ponto *non-manifold*, assim como para capturar as arestas adjacentes a um arame. Já que o vértice é a única entidade comum entre estas estruturas adjacentes nestas situações, as estruturas de vértice e uso de vértice são os lugares lógicos para armazenar tais informações de adjacência.

A relação de adjacência que consiste na lista ordenada de *loops* ao redor de uma aresta é representada. Isto é necessário pois o mesmo volume pode ser adjacente a uma aresta através de várias direções de uma só vez.

Os usos de faces, *loops*, arestas e vértices são diretamente representados.

Arames são representados por dois usos de aresta, um para cada extremidade da aresta. A conectividade com outras arestas é mantida através da estrutura de uso de vértice.

A Figura A.23 permite visualizar a descrição hierárquica da estrutura de dados *Radial Edge*, partindo de níveis mais altos em dimensão (regiões) para os mais baixos (vértices). Todos os elementos topológicos são mantidos em listas circulares duplamente encadeadas e possuem ponteiros para atributos.

Uma descrição detalhada da estrutura de dados *Radial Edge* pode ser encontrada em Weiler [17] e Cavalcanti [19], onde são também encontrados trechos de códigos em linguagem Pascal e C, respectivamente, com as definições dos tipos estruturados que representam os elementos topológicos. Weiler também demonstra a suficiência e a completude desta estrutura de dados para ambientes de modelagem *non-manifold* sujeitos às restrições já apresentadas neste trabalho.

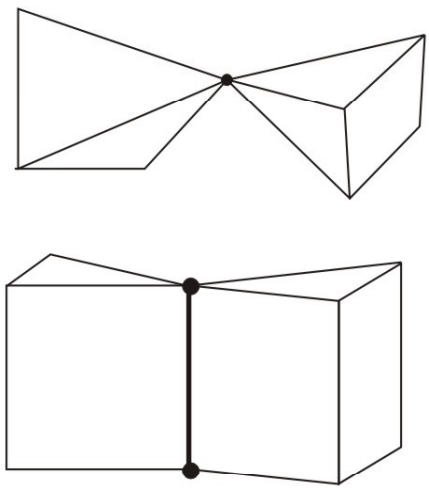


Figura A.21 – Condições *non-manifold* em um vértice e em uma aresta [17].

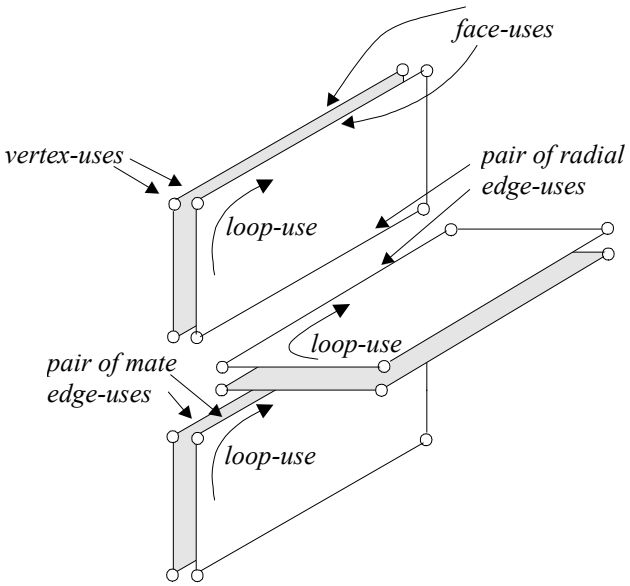


Figura A.22 – Elementos topológicos na RED [6].

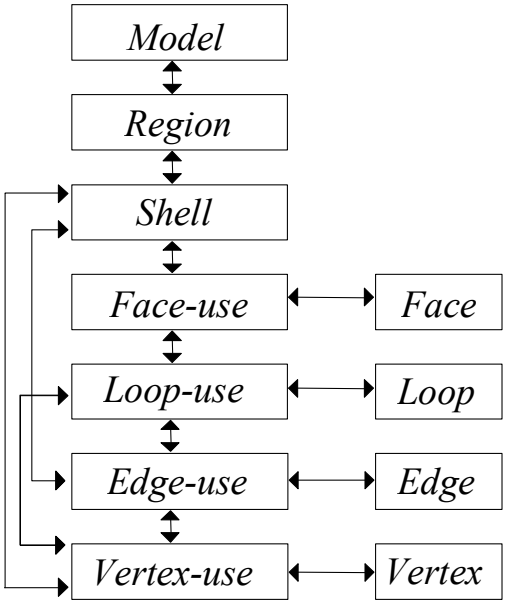


Figura A.23 – Descrição hierárquica da RED [6].

A.5.9.
Operadores de *Euler* e Operadores *Non-Manifold*

Estruturas de dados topológicas como as apresentadas na seção anterior são um tanto complexas para serem manipuladas diretamente. Quando um modelo está sendo gerado num ambiente de modelagem, operações de criação e remoção de entidades topológicas são constantemente realizadas. Durante o processo de construção de objetos num sistema de modelagem geométrica,

cada etapa intermediária requer a inserção, modificação ou eliminação de elementos topológicos, o que pode ser uma tarefa não trivial, já que a manipulação de um elemento normalmente ocasiona modificações na estrutura topológica do modelo como um todo, provocando alterações em outros elementos topológicos de mesmo nível de hierarquia ou de níveis superiores ou inferiores.

Conceitualmente, *operadores de Euler* podem ser definidos como um conjunto de operadores de relativo alto nível que manipulam estas estruturas de dados topológicas. Eles criam e modificam consistentemente a topologia de objetos com superfícies *manifold*, sem entrar nos detalhes do formato de armazenamento dos dados. Os tipos de elementos que podem ser criados ou destruídos seguem a terminologia apresentada na seção anterior: cascas, faces, *loops*, arestas, vértices e também a *ordem* (*genus*). Estes operadores foram originalmente criados por Baumgart [55] como uma ferramenta para manipular as estruturas de dados da *winged-edge*. Contudo, eles podem utilizados com quaisquer estruturas de dados dentre as descritas na seção anterior. Eles também possuem a vantagem de oferecer uma interface de programação que permite que a estrutura de dados seja trocada sem a necessidade de reescrever toda a aplicação, além do fato de que todo operador de Euler é inversível, ou seja, pode ter um operador inverso que permite a restauração da estrutura topológica do modelo antes da aplicação daquele operador.

Tradicionalmente os operadores de Euler possuem nomes na forma mxky, onde *m* significa *make* (criar) e *k* significa *kill* (destruir). As letras *x* e *y* indicam o tipo do elemento topológico que está sendo tratado. No caso mais simples, apenas um elemento de cada tipo é criado ou destruído, mas algumas vezes vários elementos do mesmo tipo são criados ou destruídos, como será visto a seguir. Cada elemento é representado por uma letra diferente: *S*, *F*, *L*, *E*, *V* e *G* representando, respectivamente, *shell* (casca), face, *loop*, *edge* (aresta), vértice e *genus* (ordem).

Operadores de Euler são usados como uma linguagem intermediária em alguns sistemas de modelagem. Eles fornecem um nível de abstração bastante desejável para os algoritmos que são implementados num nível mais alto. Assim, a princípio, a representação mais básica pode ser modificada com pouco impacto sobre a implementação do sistema de modelagem. Outra vantagem é que estes operadores garantem a consistência topológica durante o processo de modelagem. Isto pode ser vantajoso quando a topologia exata do resultado de uma operação de modelagem pode estar em cheque devido a imprecisões

numéricas do modelo. Os operadores de Euler, na medida em que provocam mudanças nas quantidades de elementos topológicos de cada tipo durante cada etapa de modelagem, garantem o balanceamento da equação de Euler-Poincaré, de forma que as estruturas de dados estão sempre restritas a representarem uma topologia *manifold* válida em cada passo.

Segundo Weiler [17], cinco dos operadores básicos de Euler que serão apresentados, MSFLV, MEV, ME, GLUE (funcionalidade mais genérica) e KE são suficientes para criar qualquer topologia, mas outros costumam ser implementados para aumentar a conveniência e a flexibilidade do processo de construção de superfícies. A Tabela A.1 resume os operadores de Euler descritos por Weiler. Alguns operadores mais genéricos cujos nomes não seguem a terminologia exposta serão detalhados abaixo:

O operador GLUE ("*Glue Faces*") junta duas faces com um único *loop* cada uma, eliminando ambas as faces e *loops* e um conjunto de arestas e vértices, com o efeito de juntar os volumes que as duas faces estão limitando. Ele possui duas variantes: *kflevmg* (*kill face*, *loop*, *edge*, *vertex*, *make genus*) e *kflevs* (*kill face*, *loop*, *edge*, *vertex*, *shell*). O seu inverso é o operador UNGLUE ("*Unglue Faces*").

O operador ESQUEEZE ("*Edge Squeeze*") é também conhecido como "*Kill Edge*, *Vertex*". Ele "espreme" as extremidades da aresta especificada até se encontrarem, eliminando a aresta e um vértice enquanto preserva as adjacências.

O operador composto MME ("*Make Multiple Edge*s") cria uma corrente conexa com um número determinado de arestas que recebe como parâmetro, começando num vértice também passado como parâmetro.

O operador composto ESPLIT ("*Edge Split*") divide a aresta especificada em duas arestas conexas, criando um vértice entre elas.

O operador LMOVE ("*Loop Move*") move o *loop* especificado da sua face corrente para uma outra face passada como parâmetro.

As Figuras A.24, A.25 e A.26 mostram exemplos de aplicação dos operadores contidos na Tabela A.1.

Tabela A.1 – Operadores de Euler [17].

construtivos	destrutivos	compostos	variados
MSFLV	*KSFLEV*	*MME*	*LMOVE*
MEV	*ESQUEEZE (KEV)*	*ESPLIT*	
mefl	*KE*	*KVE*	
mekl	*kefl*		
meksfl	*keml*		
GLUE	*kemsfl*		
kflevmr	*UNGLUE*		
kflevs	*mflevkg*		
	mflevs		

Os operadores de Euler são flexíveis, pois eles constituem operadores que manipulam sistematicamente o modelo de um grafo imerso aresta por aresta, fornecendo checagem de integridade topológica automaticamente. Praticamente qualquer outro tipo de operador de modelagem comumente encontrado pode ser implementado a partir dos operadores de Euler, incluindo primitivas paramétricas e varreduras.

Operadores booleanos também podem ser implementados usando-se os operadores de Euler. Contudo, em muitas aplicações que utilizam os operadores de Euler, opta-se por não implementar as operações booleanas no nível destes operadores, preferindo-se manipular diretamente a estrutura de dados do sistema. Isto para que os testes de integridade topológica possam ser realizados somente na etapa final de modelagem, eliminando as restrições que poderiam aparecer no caso destes testes serem realizados a cada etapa do processo de modelagem. A realização de testes de integridade a cada passo também pode ser um fator limitante em termos de eficiência do algoritmo.

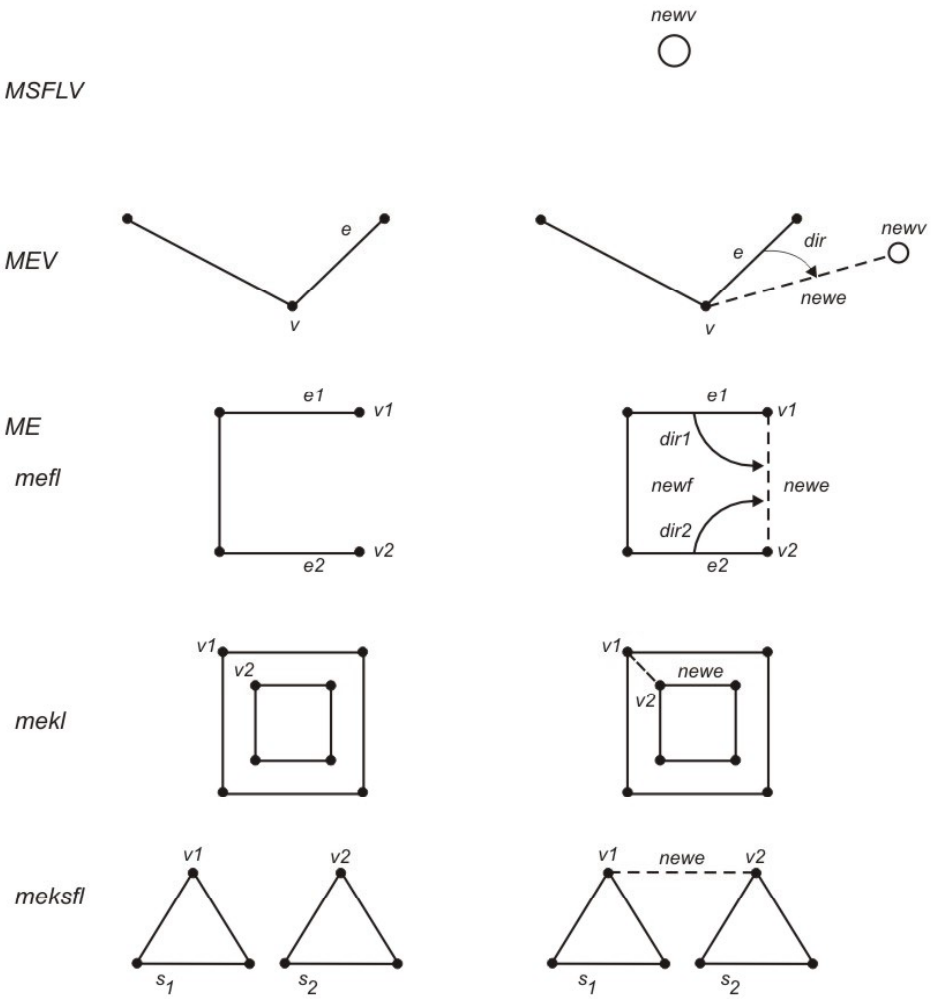


Figura A.24 – Aplicação dos operadores de Euler [17].

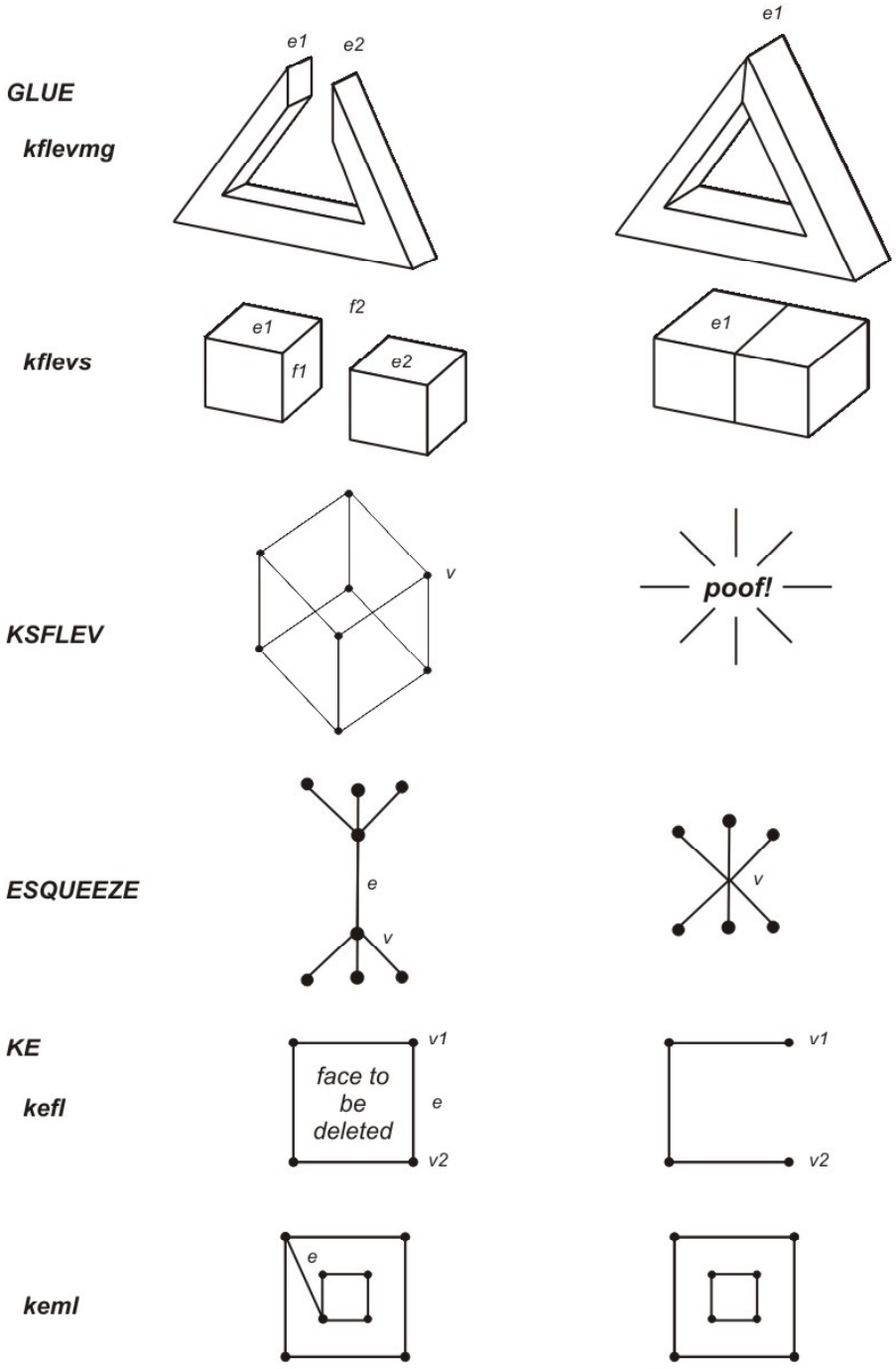


Figura A.25 – Aplicação dos operadores de Euler [17].

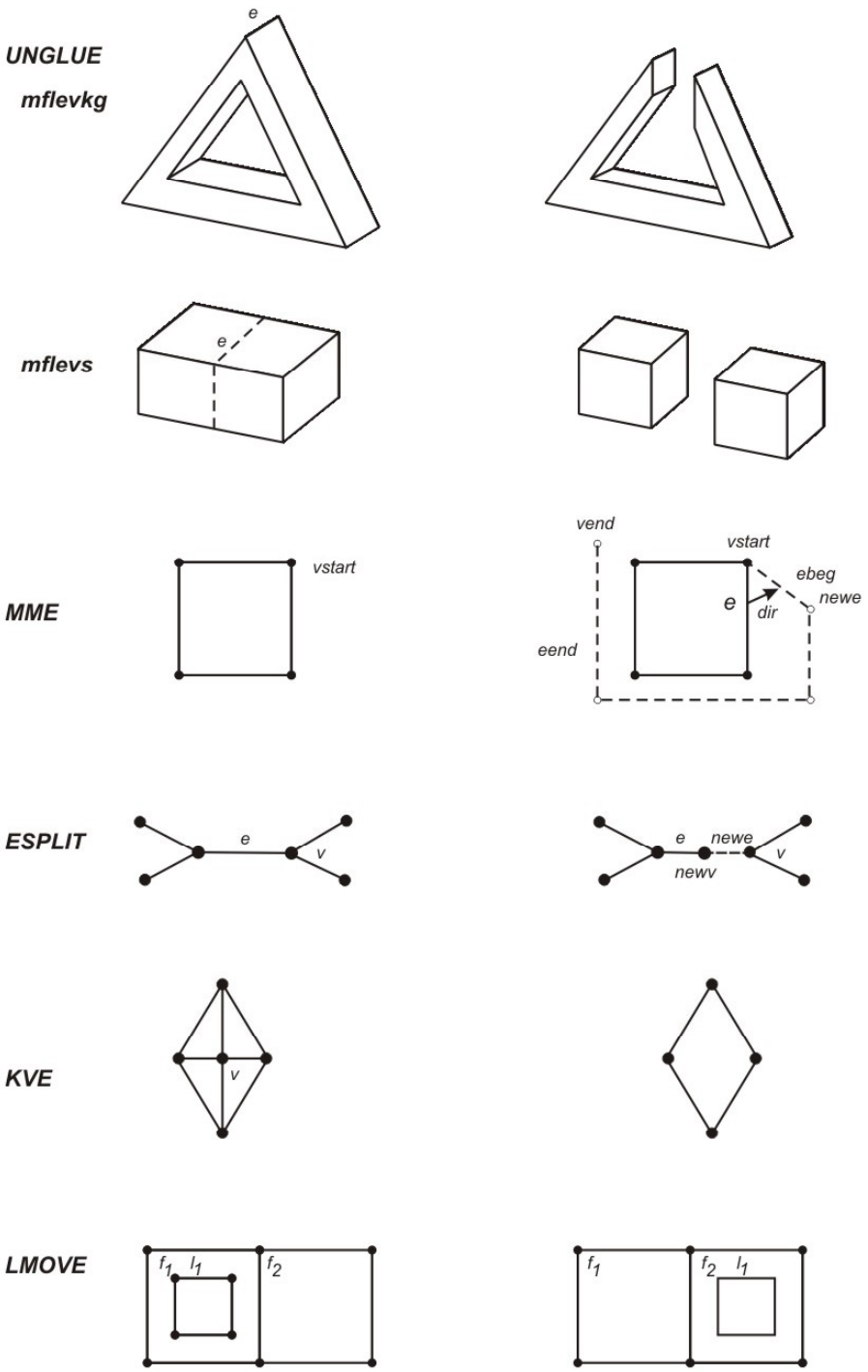


Figura A.26 – Aplicação dos operadores de Euler [17].

Este trabalho se desenvolve num ambiente de modelagem que utiliza os operadores de Euler e os operadores *non-manifold*, que serão descritos a seguir, para manter a consistência topológica dos modelos, contudo optou-se pela alternativa descrita acima, implementando-se as operações booleanas através da manipulação direta da estrutura de dados do modelador.

Operadores para construir, modificar e percorrer representações *non-manifold*, que também isolem as funcionalidades de modelagem de mais alto nível dos detalhes de implementação da estrutura de dados utilizada também se fazem necessários. A vantagem de operadores *non-manifold* é que eles impõem muito poucas restrições em relação à sua aplicação durante o processo de modelagem.

Weiler [17] introduziu um conjunto de operadores que fornecem um meio, de relativo alto nível, para manipular a *Radial Edge*. Estes operadores podem lidar com situações *manifold* ou *non-manifold*, pois em alguns casos informações completamente diferentes são necessárias para se construir um modelo sem ambigüidades. As versões *manifold* separadas destes operadores mostram-se totalmente compatíveis com funções pré-existentes de mais alto nível criadas originalmente para situações *manifold* utilizando os operadores de Euler, apesar de algumas informações adicionais serem requeridas para o ambiente *non-manifold*. Quando nenhuma diferença de especificação é requerida, operadores genéricos são utilizados, lidando com situações *manifold* ou *non-manifold*.

Os operadores de Weiler também podem ser classificados como construtivos ou destrutivos, de acordo com o número de entidades topológicas presentes no modelo antes e depois da sua aplicação. Como no caso dos operadores de Euler, existem também operadores compostos, que podem ser implementados como uma seqüência de aplicações de outros operadores mais simples. A Tabela A.2 mostra os operadores de Weiler, inclusive com os subcasos dos operadores gerais que requerem ações diferentes de acordo com a situação topológica em que eles são aplicados.

A nomenclatura destes operadores segue a mesma linha de raciocínio dos operadores do Euler, com algumas informações adicionais: os elementos topológicos podem ser *M*, *R*, *S*, *F*, *L*, *E* e *V*, significando modelo, região, *shell* (casca), face, *loop*, *edge* (aresta) e vértice. O traço baixo serve para distinguir os nomes dos operadores *non-manifold* dos nomes dos operadores de Euler. As versões estritamente *manifold* de alguns operadores estão precedidas pela letra *M*. Subcasos estão em letras minúsculas. As Figuras A.27, A.28, A.29, A.30 e A.31 ilustram a aplicação destes operadores para algumas situações de modelagem. Weiler demonstra ainda que um conjunto mínimo de cinco operadores é suficiente para construir qualquer modelo: *M_MR*, *M_SV*, *M_E*, *M_F* e *K_E*.

Em [17], [18] e [19] podem-se encontrar descrições mais detalhadas e completas dos operadores *non-manifold* introduzidos por Weiler, inclusive a

extensão de alguns deles para lidar com casos particulares, acesso aos operadores, manipulação dos mesmos e exemplos de uso.

Tabela A.2 – Operadores *non-manifold* de Weiler [17].

	gerais	*non-manifold*	*manifold*
construtivos	*M_MR*	*M_EV*	*MM_EV*
	M_SV	*M_E*	*MM_E*
		me	*mefl*
		meks	*mekl*
	M_RSFL	*M_F*	
		mfl	
		mflrs	
destrutivos	*K_V*	*K_F*	
	kvfle	*kflrs*	
	kve	*kfl*	
	kvl	*kflms*	
	kvlms		
	kvs		
	kvrsfl		
	kvsfle		
	kvrsfle		
	K_E		
	ke		
	kems		
	keml		
	kefl		
	keflms		
	K_M		
	G_V		
	gvksv		
	gvkv		
	G_E		
	geke		
	gekfle		
	gekev		
	geksev		
	G_F		
	gfksflev		
	gfkflev		
compostos	*ESPLIT*		
	ESQUEEZE		
	esqeezekev		
	esqeezeke		

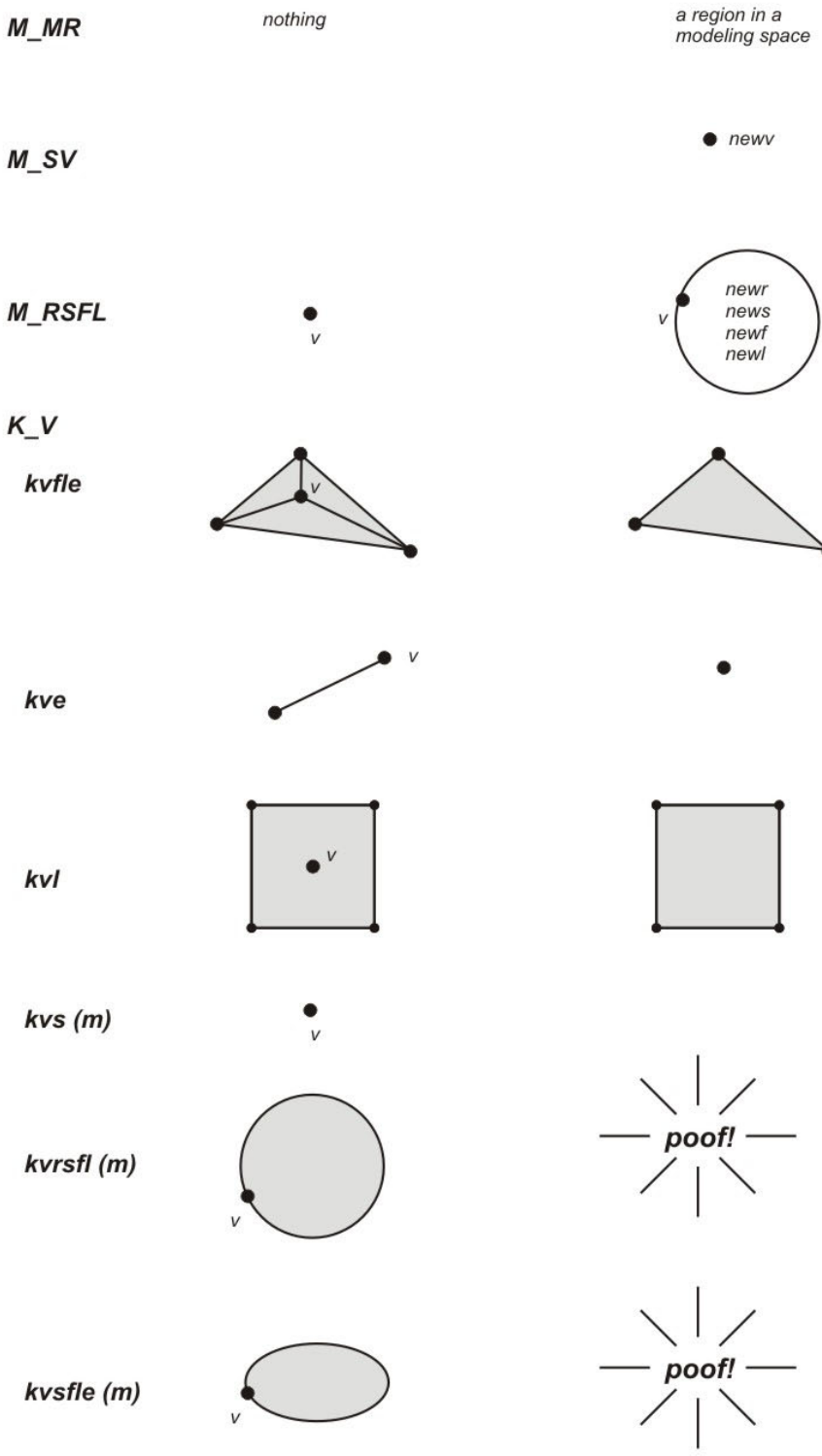


Figura A.27 – Aplicação dos operadores *non-manifold* de Weiler [17].

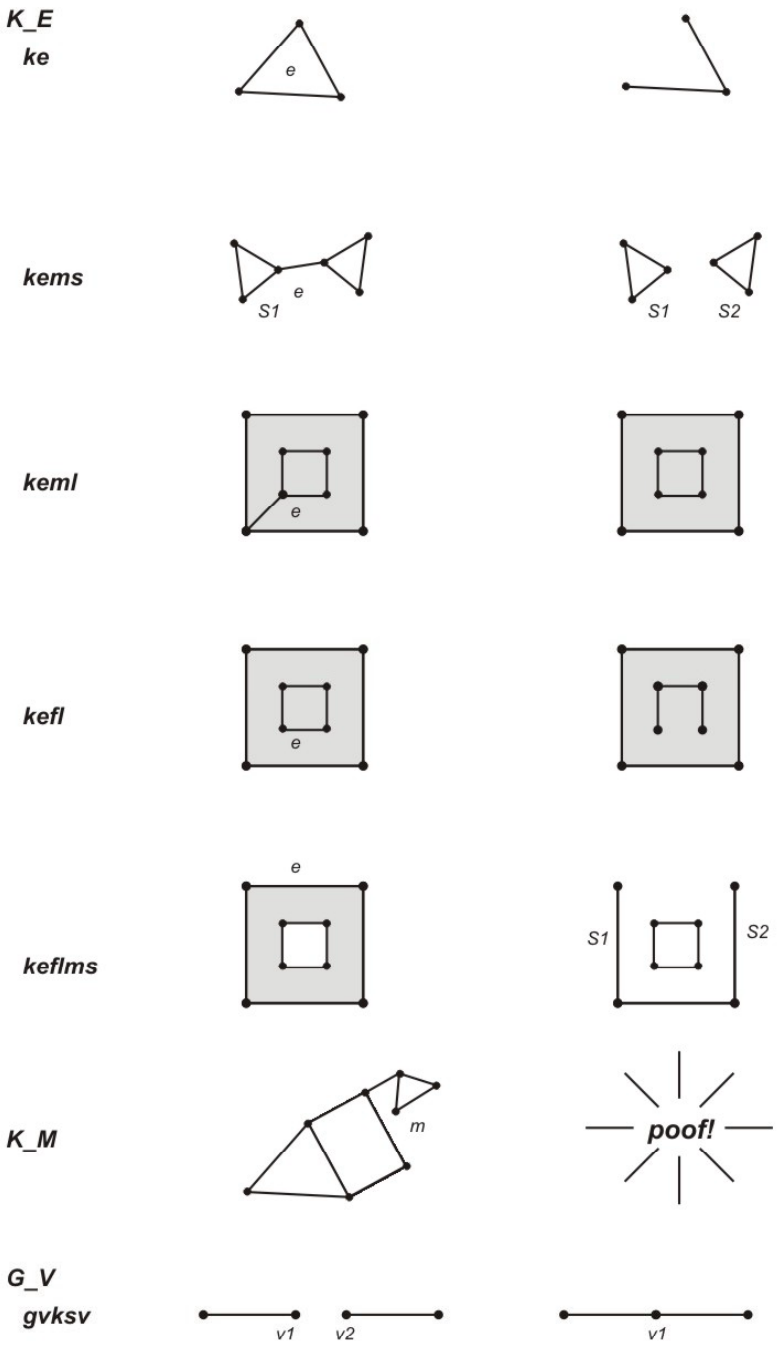


Figura A.28 – Aplicação dos operadores *non-manifold* de Weiler [17].

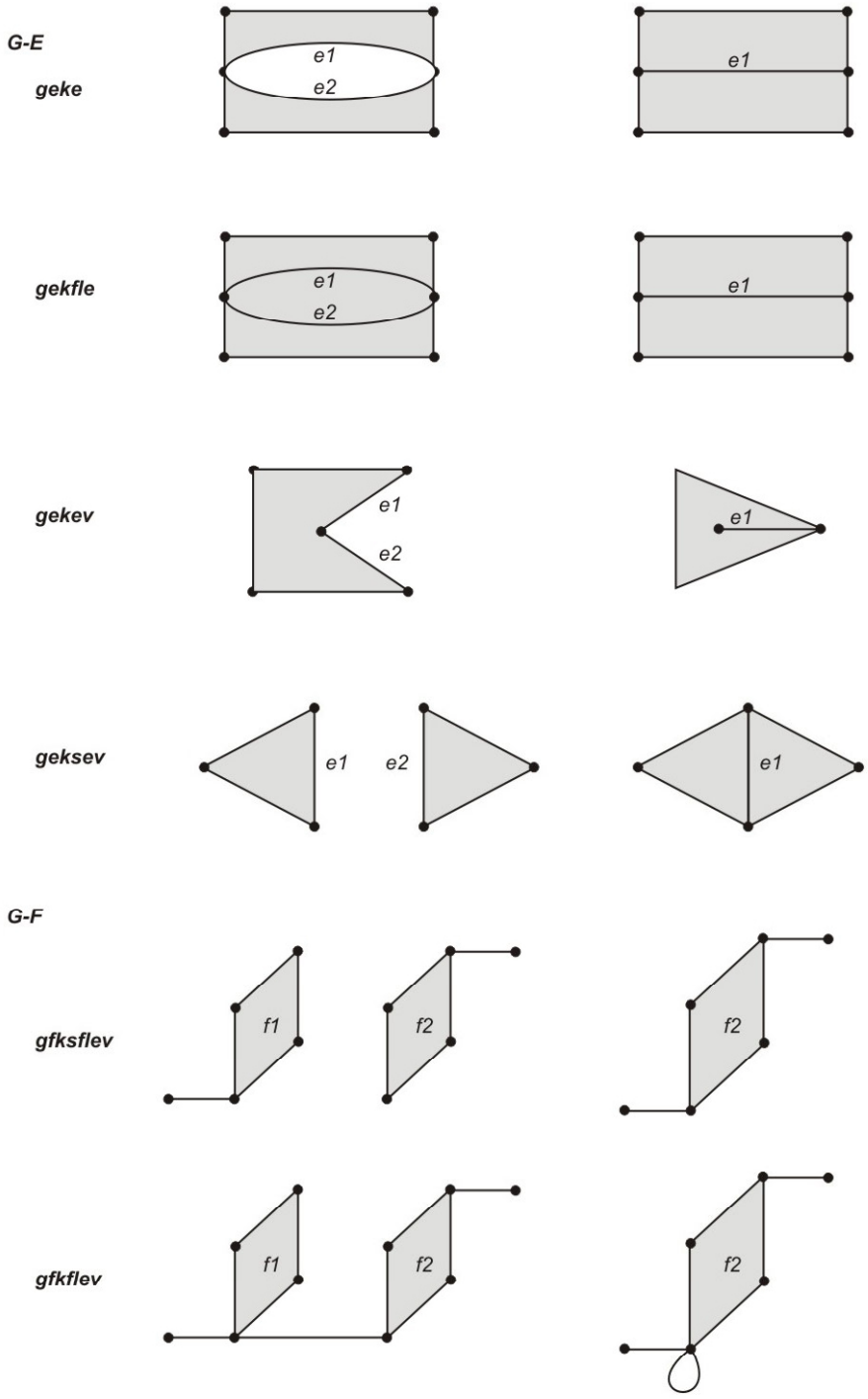


Figura A.29 – Aplicação dos operadores *non-manifold* de Weiler [17].

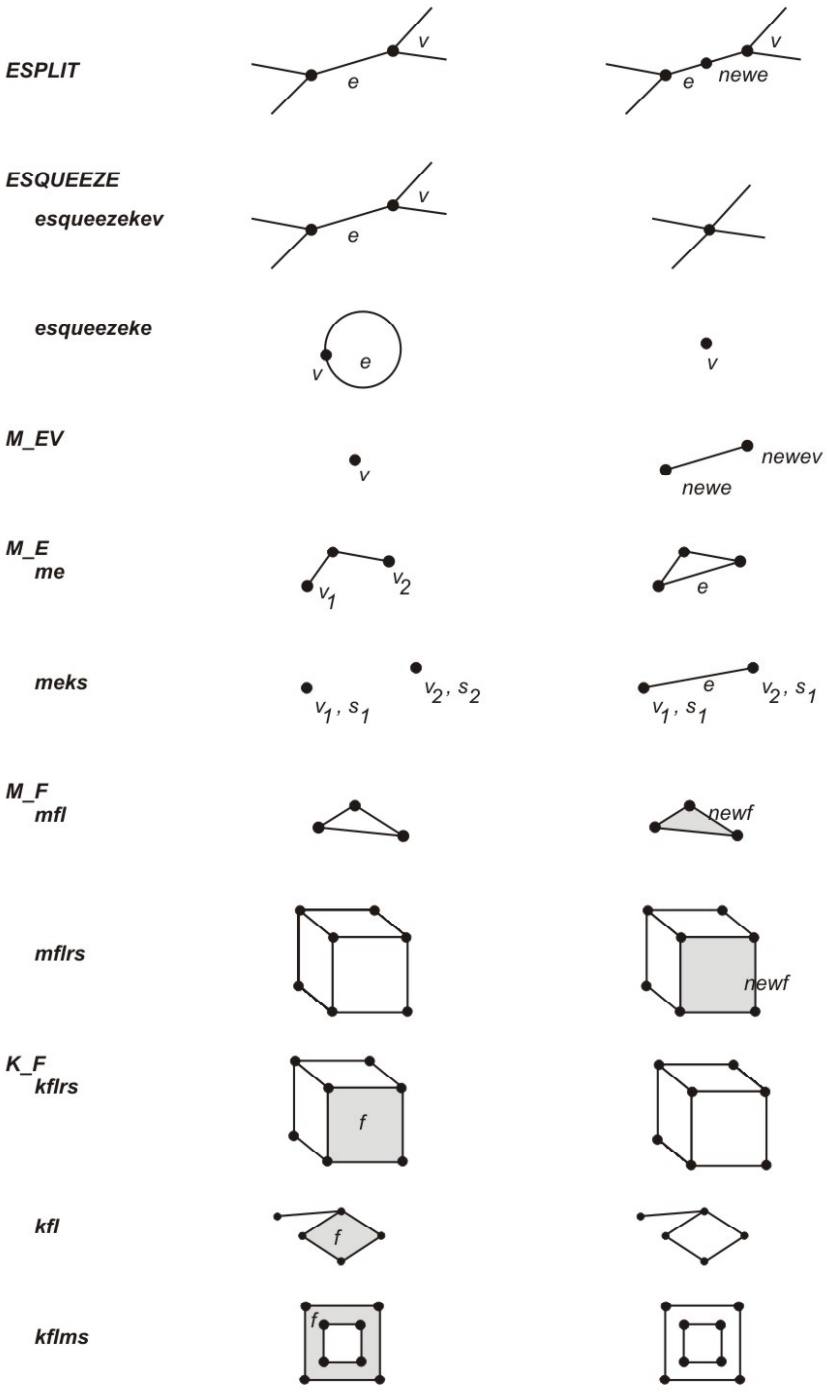


Figura A.30 – Aplicação dos operadores *non-manifold* de Weiler [17].

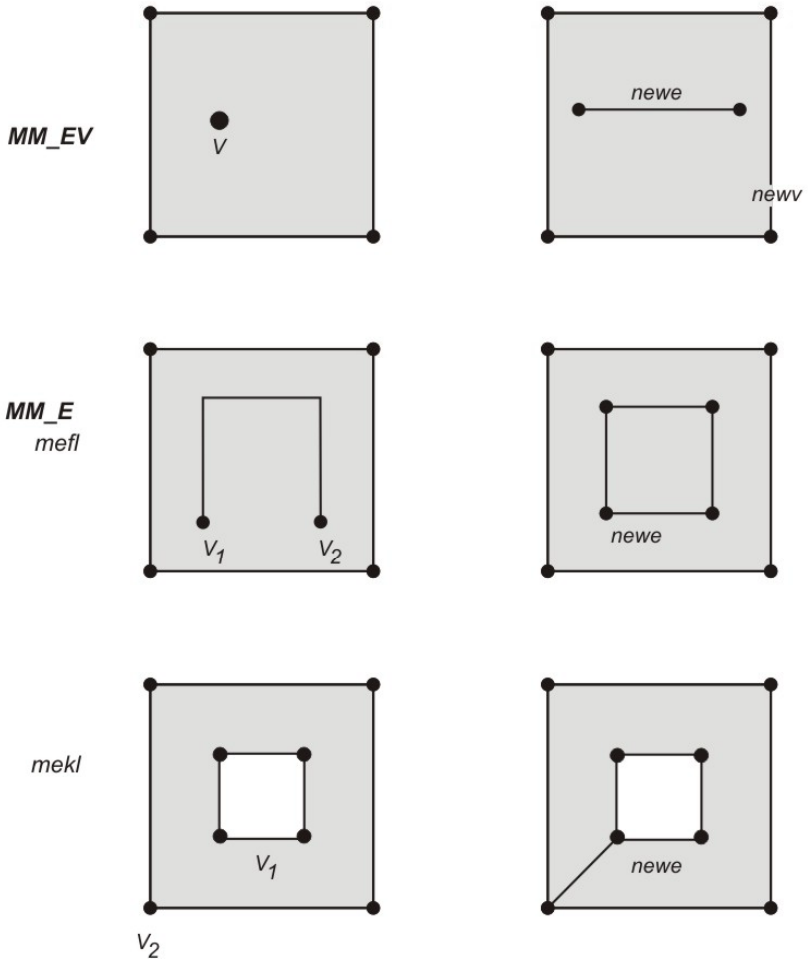


Figura A.31 – Aplicação dos operadores *non-manifold* de Weiler [17].

A.6. Tipos de representação de sólidos

Até agora falou-se em modelagem geométrica através de um ponto-de-vista centrado na existência e na inter-relação entre elementos topológicos, como vértices, arestas, faces, *loops*, cascas e regiões. Tal ponto-de-vista reflete uma maneira específica de tratar os objetos imersos num espaço Euclidiano tridimensional, através dos elementos que os compõem e das relações de adjacência entre estes elementos. Pode-se dizer que a forma como os modelos foram apresentados constitui um tipo de representação totalmente fundamentado no conceito de topologia. Este tipo de representação chama-se *representação de fronteira*, ou B-Rep (*Boundary Representation*), e caracteriza apenas uma forma possível de representar as informações relativas a um modelo geométrico.

Outras formas comumente encontradas em sistemas de modelagem são *CSG* (*Constructive Solid Geometry*) [9,10], modelos de decomposição e modelos híbridos, que contêm múltiplas representações simultâneas.

Os modelos mais usados para representar sólidos têm sido CSG e B-Rep, que se caracterizam por serem bastante úteis e de certa forma, complementares. Muitos modeladores, contudo, já apresentam formas de representação híbrida, permitindo uma maior flexibilidade na construção dos modelos, deixando ao usuário a tarefa de decidir qual a melhor forma de representar o objeto de interesse. O modelador MG, utilizado na implementação do algoritmo proposto neste trabalho, utiliza uma abordagem B-Rep na representação das informações dos modelos. A inserção das operações booleanas neste modelador permite que possamos passar a classificá-lo como um modelador híbrido.

CSG e B-Rep possuem vantagens e desvantagens inerentes a cada um. Um objeto CSG, por exemplo, é sempre válido no sentido de que sua superfície é fechada e orientável e fecha um volume, desde que as primitivas sejam válidas neste sentido. Um objeto B-Rep, por outro lado, é facilmente desenhado num sistema de visualização gráfica. Por este motivo é que existe essa tendência a se querer tirar proveito das vantagens de ambas as formas de representação, através de modeladores híbridos.

A.6.1. Modelos de decomposição

Modelos de decomposição exercem um papel coadjuvante na modelagem geométrica devido a certas limitações. Entretanto, muitos possuem aplicações importantes, como por exemplo na área de análise numérica e em bancos de dados geográficos.

Basicamente, os modelos de decomposição descrevem um sólido através da combinação de blocos básicos que são mantidos juntos. Nas áreas de visão computacional, processamento de imagens e robótica, estruturas de dados chamadas *Quadtrees* ou *Octrees* constituem métodos hierárquicos que auxiliam na discretização do domínio.

O esquema de decomposição mais simples é a *subdivisão uniforme* do espaço numa grade de cubos de tamanho especificado. Os cubos que interceptam o interior do sólido são *marcados*, de forma que o sólido seria então representado pelos cubos marcados. Um exemplo deste tipo de decomposição pode ser visto na Figura A.32. O tamanho dos cubos define o grau de exatidão

na representação do sólido. A princípio, apenas os cubos marcados precisam ser armazenados.

Quando um grau de exatidão maior é requerido, pode ser que o número de cubos que tenham de ser armazenados na estrutura de dados se torne excessivamente grande e isto passe a ser um inconveniente. *Octrees* resolvem este problema agregando cubos marcados em cubos maiores. Conceitualmente, o espaço é particionado em várias grades, cada uma com um tamanho de malha que é o dobro da anterior. A Figura A.33 ilustra esta idéia.

Modelos de decomposição constituem um tipo de representação bastante geral, simples e que permite a utilização de uma série de algoritmos eficientes. Contudo, são modelos que requerem uma quantidade de memória considerável para o armazenamento das informações quando se necessita de grande precisão na descrição geométrica do modelo. Outros problemas também aparecem quando se trabalha com espaços digitalizados, como por exemplo, a impossibilidade de se inverter uma operação de rotação.

Existem alguns tipos de decomposição espacial em que os elementos espaciais não possuem uma relação específica implícita com o sistema de coordenadas, o que facilita a rotação dos objetos representados. Ao custo de adjacências mais complexas e processamento de formas geométricas, pode-se quebrar esta relação e permitir elementos com formas irregulares. Na análise pelo MEF, elementos triangulares e tetraédricos são usados, como mostrado na Figura A.34.

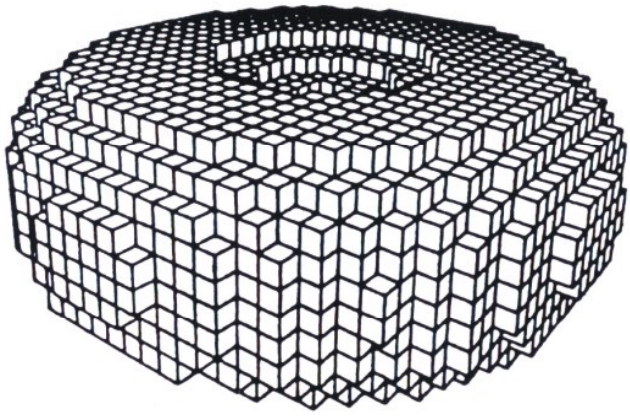


Figura A.32 – Subdivisão uniforme do espaço [10].

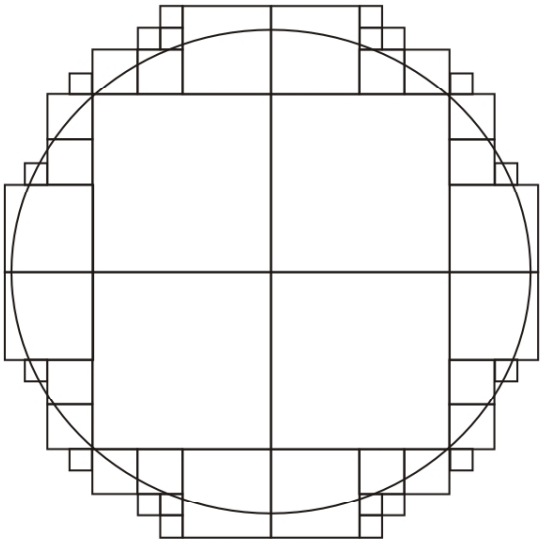


Figura A.33 – Subdivisão do espaço por Octree [9].

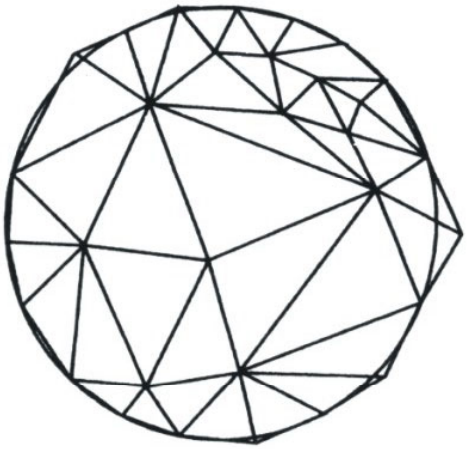


Figura A.34 – Subdivisão irregular do espaço [9].

A.6.2.
Modelos de fronteira (B-Rep)

Modelos de fronteira representam os objetos através das superfícies que os delimitam. Um B-Rep pode ser definido como um grafo com nós correspondendo às faces, arestas e vértices que definem a fronteira topológica do sólido. A descrição da superfície de um sólido constitui-se de duas partes, uma topológica e outra geométrica. A descrição topológica diz respeito à conectividade e orientação dos vértices, arestas e faces. A descrição geométrica

permite a imersão dessa superfície no espaço. Historicamente, esta representação evoluiu de uma descrição de poliedros.

Basicamente, as informações topológicas contêm especificações sobre vértices, arestas e faces de forma abstrata, indicando suas incidências e adjacências. As informações geométricas especificam, por exemplo, as equações das superfícies que contêm as faces, e das curvas que contêm as arestas. O armazenamento explícito das informações topológicas ocasiona um gasto elevado de memória.

A Figura A.35 ilustra os componentes básicos de um modelo de fronteira. Na figura, a superfície do objeto é dividida em um conjunto fechado de faces (Figura A.35a), cada qual sendo representada pelo seu polígono delimitador (Figura A.35b), que por sua vez é representado em termos de arestas e vértices (Figura A.35c).

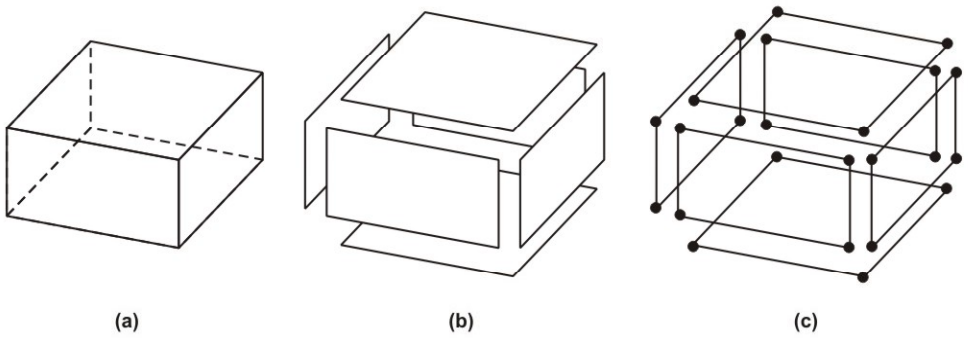


Figura A.35 – Componentes básicos de um modelo de fronteira [10].

Na seção anterior foram apresentadas as formas de representação *manifold* e *non-manifold*, algumas estruturas de dados topológicas e os operadores que manipulam estas estruturas. Todos estes tópicos dizem respeito ao tipo de representação B-Rep, já que tratam diretamente os elementos topológicos, que constituem o elemento-chave dos sistemas de modelagem que utilizam a técnica B-Rep no armazenamento das informações dos modelos.

Operações booleanas, que são típicas de modelos CSG, podem ser implementadas em sistemas de modelagem que utilizam a técnica B-Rep ao custo de um processamento adicional para a obtenção do resultado.

A.6.3.
Modelos construtivos

Na geometria construtiva de sólidos, um sólido é representado pela combinação de sólidos mais simples, chamados *primitivas*. A combinação destes sólidos mais simples é realizada por meio das operações booleanas aplicadas a eles. Tanto a superfície quanto o interior de um objeto são definidos, porém implicitamente.

Para que se possa entender as propriedades e algoritmos associados a este tipo de representação, deve-se conhecer as operações básicas contidas na mesma, incluindo classificação de pontos, curvas e superfícies em relação a um sólido, detecção de redundâncias na representação e métodos de aproximação sistemática de objetos [9].

A.6.3.1.
Primitivas

Objetos primitivos são selecionados a partir de um universo de formas geométricas possíveis. Uma forma geométrica é instanciada determinando-se valores para certos parâmetros. Três abordagens para definição de primitivas podem ser encontradas:

- Cada primitiva é selecionada a partir de um conjunto de geometrias sólidas pré-definidas, e valores são atribuídos a certos parâmetros que controlam a geometria final do objeto. Este é o tipo de abordagem utilizada numa representação CSG. As primitivas básicas comumente encontradas são: cilindros, toros, esferas, cones, paralelepípedos, dentre outras. Os parâmetros podem ser o raio de uma esfera, a altura e o raio de um cilindro, a aresta lateral de um cubo, etc.
- Uma primitiva é criada pela *varredura* de um contorno de uma superfície ao longo de uma curva espacial. Tanto o contorno da superfície quanto a curva são definidos por parâmetros.
- Todas as primitivas são *semi-espaços* algébricos, ou seja, conjuntos de pontos da forma

  $\{(x, y, z) / f(x, y, z) \leq 0\}$

onde f é um polinômio irredutível, ou seja, não-fatorável. Os coeficientes do polinômio podem ser considerados como os parâmetros de forma da primitiva.

Em modeladores CSG, a primeira abordagem costuma ser utilizada. A instanciação dos parâmetros que definem a geometria completa do objeto pode

ser atrasada, sendo possível a criação de objetos genéricos. Contudo, tais objetos não podem ser visualizados ou convertidos para uma representação B-Rep, já que diferentes parâmetros podem levar a formas geométricas totalmente diversas.

A.6.3.2. *Undo* e *Redo*

Na modelagem de sólidos, cada passo da construção de um objeto envolve modificações no modelo corrente, que podem estar sujeitas a erros. Além disso, usuários podem querer testar várias alternativas, modificá-las, corrigir falhas, e assim por diante. A capacidade de se desfazer uma operação de modelagem ou de refazê-la pode ser algo imprescindível em alguns sistemas.

Dependendo do tipo de representação com que se está trabalhando e da complexidade do modelo, desfazer uma operação ou uma seqüência de operações pode ter um custo muito alto em termos de tempo. Normalmente, para que não seja necessária a realização explícita da inversão de uma operação para que se possa chegar ao modelo existente antes da aplicação desta operação, opta-se por armazenar um histórico de operações que levaram ao modelo corrente. A maneira mais eficiente para se armazenar este histórico de modo que se tenha acesso a qualquer etapa de modelagem e de tal forma que se possa reproduzir toda a seqüência de operações realizadas é através de uma *árvore*.

A capacidade de *refazer* (*redo*) um modelo a partir da raiz da árvore até certo ponto é algo menos complexo do que se *desfazer* (*undo*) uma operação, e conseqüentemente a operação de refazer é mais rápida. Isto porque em muitos casos, operações de modelagem requerem a execução de algoritmos que checam a validade do modelo corrente, e sendo assim, uma operação que já foi feita antes não necessita novamente destes testes de validade.

Em representações puramente CSG, as operações de *undo* e *redo* costumam ser bem simples, por motivos que serão expostos a seguir. Em representações B-Rep, modificações locais, tais como alteração da geometria de uma aresta do contorno de uma face ou a extrusão de uma face são fáceis de serem desfeitas, contudo modificações globais, tais como as operações booleanas, não são. Para se resolver o problema de operações difíceis de serem desfeitas, pode-se optar pelo *check point*, ou seja, o armazenamento das informações do modelo imediatamente antes da operação ser realizada e

imediatamente após. Desta forma, desfazer torna-se apenas uma questão de recuperação do modelo anterior. Obviamente, o custo de se realizar um *check point* deve ser avaliado em comparação com o custo de se inverter uma operação, pois nem sempre este processo de inversão é algo complexo.

O armazenamento do histórico de operações numa árvore permite a reconstrução do modelo desde o início, e o acesso a qualquer nível de modelagem que se deseje. Pode-se inclusive elaborar uma maneira de se atribuir uma identificação a cada nó da árvore de modo que se possa acessar diretamente uma etapa desejada sem a necessidade de se realizar uma série de operações de *undo* e *redo*.

A.6.3.3.
A Representação CSG

Um objeto CSG é construído a partir de primitivas padronizadas, utilizando-se as operações booleanas e movimentos rígidos, tais como translação e rotação. As primitivas padronizadas podem ser paralelepípedos, esferas, cilindros, cone e toros. Elas são genéricas no sentido de representarem formas geométricas que precisam ser instanciadas pelo usuário. Naturalmente, outros tipos de primitivas podem ser permitidas, desde que possam ser generalizadas da mesma forma, ou seja, a partir de parâmetros cujos valores devem ser definidos pelo usuário.

Cada primitiva possui um sistema de coordenadas local, a partir do qual os valores dos parâmetros podem ser instanciados. Os sistemas de coordenadas locais devem poder ser relacionados uns com os outros por meio de um sistema de coordenadas global, que serve para mapear a localização das primitivas no espaço tridimensional. A princípio, as primitivas devem possuir valores finitos para os seus parâmetros, ou seja, apenas sólidos com dimensões finitas podem ser representados. Casos particulares de primitivas que constituem semi-espaços, ou seja, com dimensões infinitas, podem ser tratados, pelo menos como passos intermediários, no processo de definição de sólidos mais complexos.

A.6.3.4.
As operações booleanas regularizadas

Após instanciar as primitivas, estas podem ser combinadas através da utilização das operações booleanas regularizadas. São elas: a *união*

regularizada (∪*), a *interseção regularizada* (∩*) e a *diferença regularizada* (-*). O processo de regularização consiste em eliminar do resultado da aplicação da operação booleana todas as entidades de dimensão inferior que ficaram pendentes, ou seja, considerar apenas os volumes preenchíveis. Utilizando a terminologia já apresentada neste trabalho, pode-se afirmar que a regularização de um conjunto de pontos é definida como o *fecho do interior* deste conjunto de pontos. A Figura A.36 ilustra a aplicação da operação regularizada de interseção entre dois sólidos.

Na prática, contudo, as operações booleanas regularizadas são implementadas classificando-se os elementos de superfície resultantes da aplicação da operação e eliminando-se os de mais baixa dimensão. A eliminação destas estruturas pode ser desejável para definir apenas objetos sólidos como resultados das operações, contudo, em muitas aplicações, pode ser necessário retê-las, mesmo quando estão no interior dos objetos. É o caso de sistemas que utilizam a análise pelo MEF, onde tais estruturas de mais baixa dimensão podem representar restrições quanto à discretização do domínio.

As Figuras A.37 e A.38 mostram exemplos de aplicação das operações booleanas em algumas primitivas básicas.

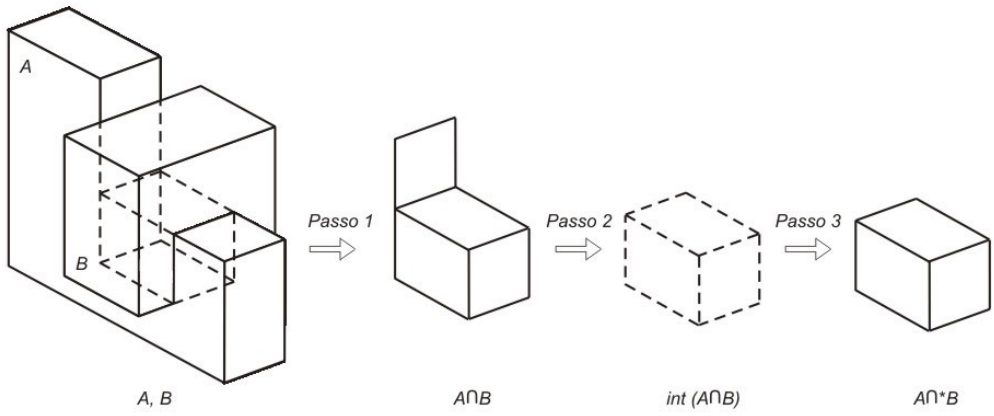


Figura A.36 – Interseção regularizada entre sólidos [9].

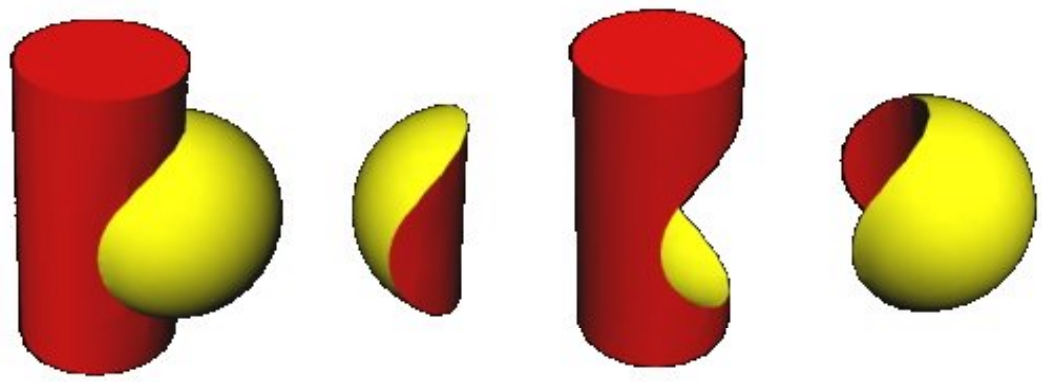


Figura A.37 – Operações booleanas aplicadas a dois sólidos: união, interseção e diferenças [56].

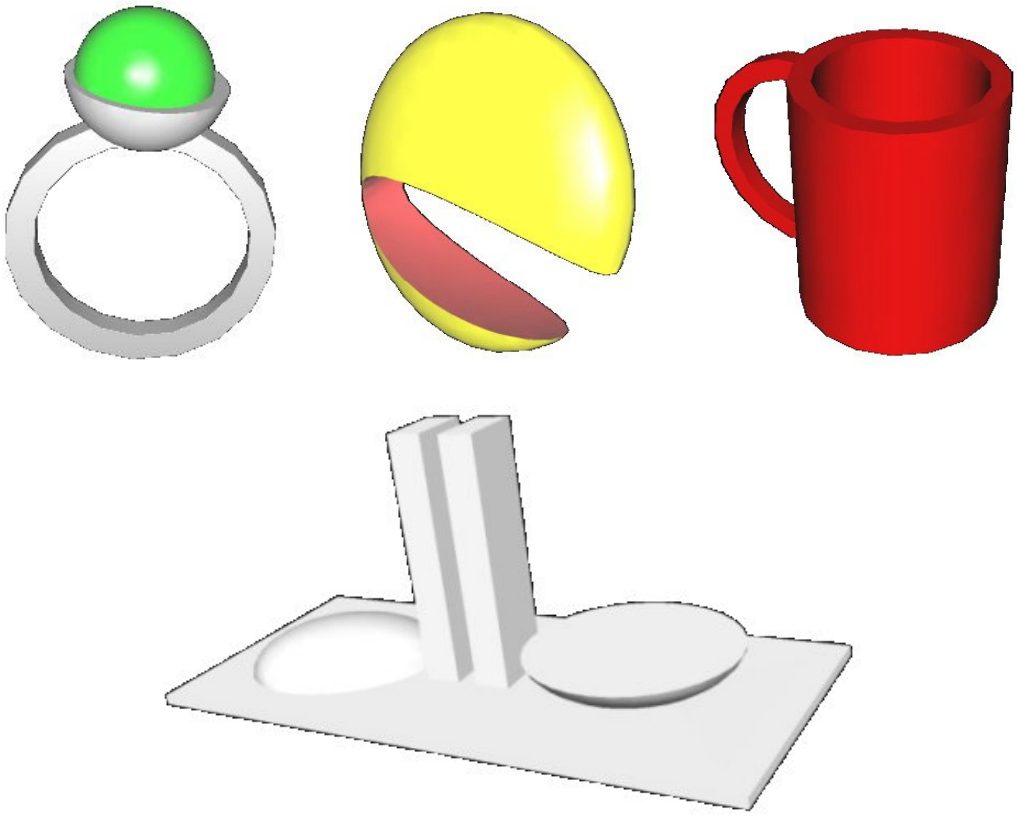


Figura A.38 - Sólidos gerados por meio de operações booleanas em primitivas básicas [56].

A.6.3.5. Construção de um objeto CSG

Selecionar uma primitiva e instanciá-la é um processo que pode ser realizado através de uma interface simples com a estrutura de dados do

modelador. Apenas para uniformizar procedimentos deste tipo, pode-se adotar uma convenção simples que determine qual é a primitiva com a qual se deseja trabalhar e quais são os valores dos parâmetros requeridos para a definição geométrica completa desta primitiva. Assim, por exemplo, para se obter um paralelepípedo cujos comprimentos das arestas são 3, 5 e 7, pode-se especificar o comando *block* (3, 5, 7), onde os comprimentos podem estar expressos num sistema de unidades quaisquer ou podem ser dados explicitamente. Da mesma forma, outras operações como os movimentos rígidos de sólidos podem ser expressos desta maneira, desde que os devidos parâmetros sejam instanciados, como por exemplo *y-translate* (*block* (3, 5, 7), 3), que representa uma translação do paralelepípedo acima mencionado na direção do eixo y em 3 unidades. Associações de operações deste tipo com as operações booleanas permitem que se construa um sólido CSG completo.

O suporte da Figura A.39, por exemplo, pode ser gerado através da expressão

(*block* (1, 4, 8) ∪* *x-translate* (*block* (8, 4, 1), 1) -* *x-translate* (*y-translate* (*z-cilinder* (1, 1), 2), 5)

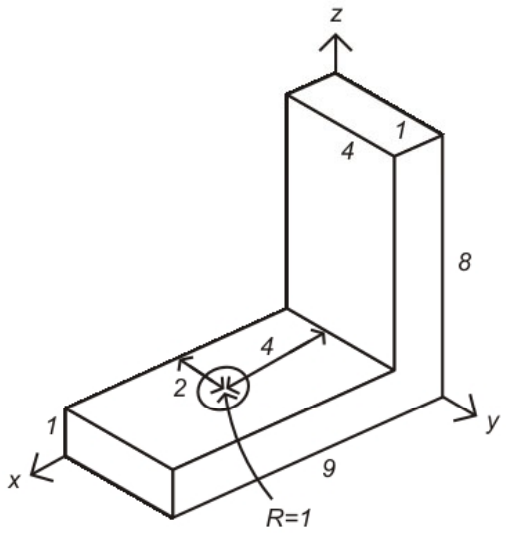


Figura A.39 – Suporte [9].

A estrutura que representa um sólido CSG é uma árvore em que as operações booleanas e de movimentos rígidos utilizadas para construí-lo são

hierarquizadas. Tal estrutura é chamada *árvore* CSG (CSG *tree*). Cada operação realizada é representada por um nó interno (que não é folha) da árvore, e cada primitiva por um nó folha. A Figura A.40 ilustra graficamente uma árvore CSG usada na construção de um sólido em forma de anel.

A geometria construtiva de sólidos precisa fazer uso de uma outra representação, tida como a sua representação interna, para que sólidos assim representados possam ser visualizados. Os dados necessários à visualização de um sólido CSG serão processados a partir da sua árvore. Para obtenção de tais dados, a árvore pesquisada deve manter informações adequadas para tal, que podem variar dependendo da representação interna utilizada. Um nó interno deve manter uma referência para a operação booleana ou de movimento rígido à qual se refere. Um nó folha deve manter a estrutura da primitiva correspondente na representação interna, ou então informações a serem utilizadas na construção da mesma, dependendo de qual seja ela. O processamento de uma árvore CSG se dá em geral pela busca em profundidade. Isto porque, pelo uso de tal metodologia, as operações são percorridas na ordem de aplicação das mesmas. Assim, dados relativos às primitivas serão obtidos nos nós folhas e combinações serão realizadas nos nós internos.

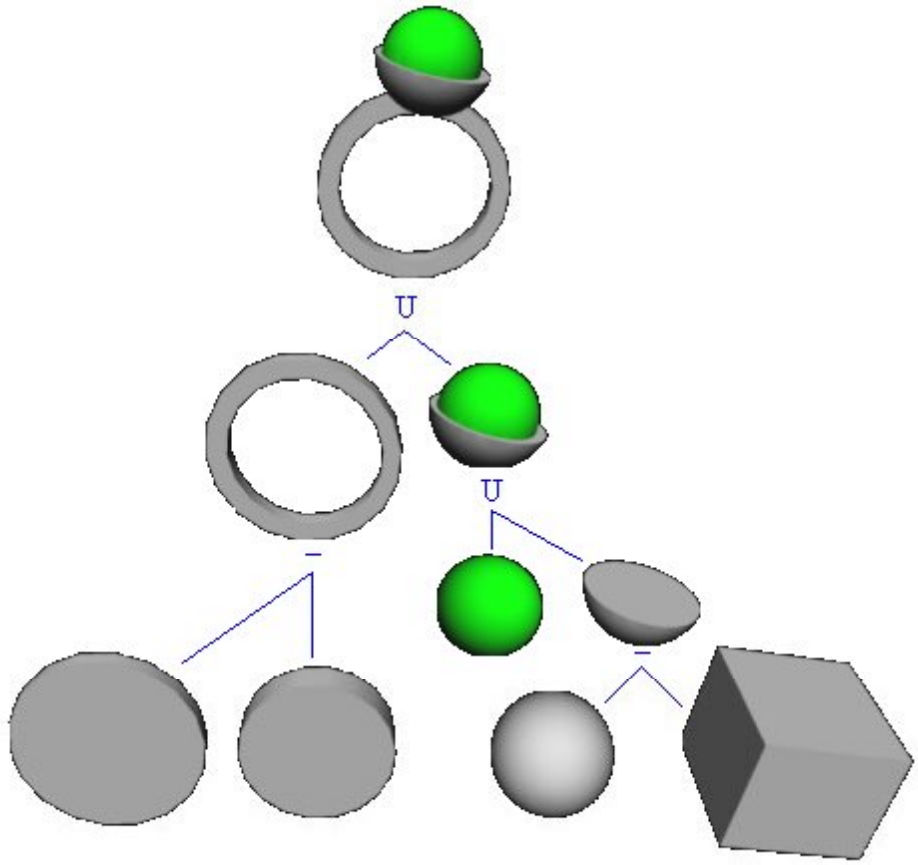


Figura A.40 – Árvore CSG [56].

Pode-se utilizar a representação B-Rep como representação interna de sólidos CSG, o que é uma abordagem bastante robusta e flexível para a representação de sólidos compostos pelo uso das operações booleanas. Quando se faz uso dela, o sólido B-Rep correspondente a uma árvore CSG é obtido quando sua visualização é requerida. Ele é construído a partir da combinação direta das primitivas conforme as operações realizadas. Para tal, a árvore é percorrida fazendo-se busca em profundidade, de forma que primitivas em B-Rep sejam criadas quando nós folhas forem visitados e os sólidos obtidos a partir dos nós descendentes sejam combinados (conforme a operação correspondente) quando nós internos forem visitados. Contudo, informações topológicas requerem uma avaliação da árvore, o que é bastante caro em termos de tempo de execução. Em contrapartida, o gasto para armazenamento de uma árvore CSG é mínimo. Algoritmos de desenho tipo *scan line* conseguem facilmente gerar uma imagem a partir de uma árvore CSG.

A expressão que descreve o suporte da Figura A.39 pode ser facilmente transformada numa árvore CSG, como mostrado na Figura A.41. Neste exemplo, os dois blocos que foram unidos estão se tocando, e a altura do cilindro é exatamente igual à espessura do suporte. Na prática, tais informações podem causar problemas devido a imprecisões de ordem numérica. É recomendável que se permita a existência de superposições durante as operações de união.

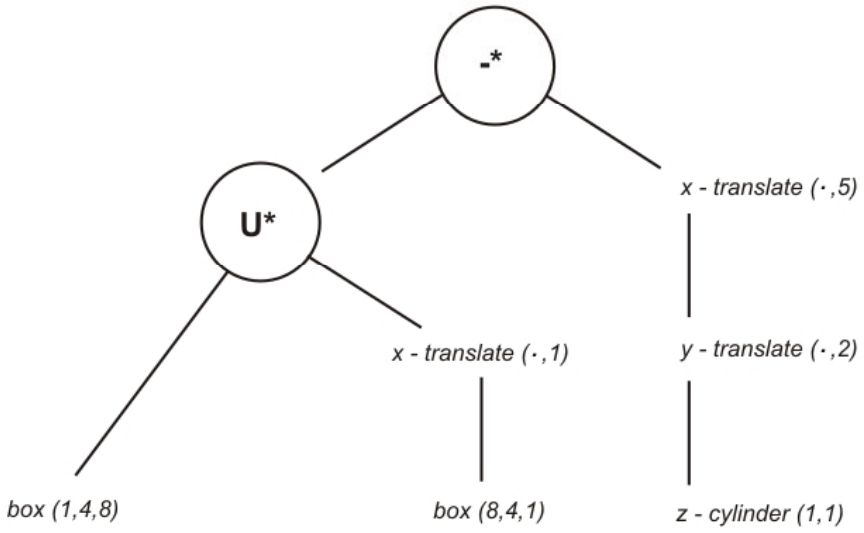


Figura A.41 – Árvore representando a expressão CSG [9].

A.6.3.6.
Classificação de pontos, curvas e superfícies em relação a sólidos

Tendo construído um objeto CSG, pode-se questionar a sua geometria de diversas formas. Alguns tipos mais elementares de questões dizem respeito à classificação de pontos, curvas e superfícies contidas no espaço tridimensional em relação ao sólido representado por uma árvore CSG.

Algoritmos que determinam a classificação de pontos em relação a sólidos, baseados nos conceitos de vizinhança já apresentados neste trabalho, buscam solucionar casos não elementares como aqueles de pontos que se localizam sobre a superfície de duas primitivas que se combinam para formar um sólido CSG, mas que são interiores ao sólido CSG resultante. Para poder solucionar problemas como estes, são considerados os tipos de vizinhança do ponto em questão em relação às primitivas originais, no sentido de determinar se o ponto se localiza sobre uma face, sobre uma aresta ou sobre um vértice do contorno destas primitivas, e ainda a *orientação* das faces destas primitivas.

Algoritmos de classificação de curvas em relação a sólidos seguem uma organização bastante semelhante aos de classificação de pontos. Eles subdividem a curva em segmentos que podem estar dentro, fora ou sobre a superfície das primitivas. Para classificar uma curva em relação a uma primitiva, pode ser necessária a parametrização da curva e a substituição da forma paramétrica nas equações implícitas das superfícies que envolvem a primitiva para se obter os pontos de interseção que serão necessários para a *segment*ação da curva para posterior classificação.

A classificação de superfícies em relação a sólidos também segue a mesma linha de raciocínio dos algoritmos usados para classificar pontos e curvas. Uma superfície irá interceptar um sólido num número de áreas. Cada área é limitada por segmentos de curva, onde cada segmento está na interseção da superfície com uma das primitivas do sólido. Um procedimento genérico para a obtenção destes segmentos e a partir deles as respectivas áreas é o seguinte:

- Intercepta-se a superfície com cada uma das primitivas que foram combinadas para formar o sólido.
- Classificam-se as curvas resultantes, conseqüentemente determinando-se as arestas delimitadoras das áreas de superfície que estão dentro ou fora do sólido, ou ainda na sua superfície.

- Combinam-se os segmentos, apropriadamente orientados, construindo-se assim uma representação de fronteira das respectivas áreas de superfície.

A classificação de superfícies em relação a sólidos pode ser utilizada para se elaborar um método de conversão de uma representação CSG para uma representação B-Rep. Esta conversão pode ser baseada no seguinte raciocínio: considera-se todos os pares de primitivas usadas para formar o objeto CSG que se interceptam, obtendo para cada par, um conjunto de curvas de interseção. Classificando-se cada curva em relação ao sólido, pode-se determinar quais são os que se localizam sobre a superfície do sólido. Cada segmento será uma aresta da representação B-Rep. Estes segmentos definem também, na superfície das primitivas, as áreas que serão as faces da representação B-Rep do sólido. Considerando-se as vizinhanças, podem-se extrair as informações topológicas necessárias para se determinar as adjacências das diversas faces.

A.6.3.7. Redundâncias e aproximações em árvores CSG

Uma pesquisa geométrica numa árvore CSG pode crescer linearmente, ou mesmo quadraticamente, com o número de primitivas utilizadas para construir o sólido final. Isto faz com que se deseje investigar se todos os nós contidos na árvore são estritamente necessários para a composição do objeto resultante, ou seja, se não existe alguma sub-árvore que seja redundante. Redundância, neste caso, representa uma contribuição nula para a geometria final do objeto.

Um tipo de redundância comum é o de uma sub-árvore que representa o espaço vazio. Diz-se que uma sub-árvore deste tipo define o *objeto nulo* (Λ). Um algoritmo de detecção de Λ permite detectar se dois objetos se interferem: Se T_1 e T_2 forem duas árvores CSG representando os objetos, eles não irão se interferir se $T_1 \cap^* T_2 = \Lambda$.

De uma forma geral, uma sub-árvore *T'* de uma árvore *T* é redundante se a substituição desta sub-árvore *T'* pelo objeto nulo Λ ou pelo seu complemento Ω não alterar a geometria do objeto definido por *T*.

Redundâncias podem ocorrer, por exemplo, devido à construção de uma árvore CSG *T* por modificação de uma árvore T_1. Pode-se aproveitar a sub-árvore T_1 devido a uma possível semelhança entre os objetos resultantes. Possivelmente, o objeto definido por T_1 possui uma parte desnecessária para o objeto definido por *T,* e o projetista pode simplesmente cortar esta parte por uma

operação de diferença, por exemplo. Mas isto significa que a sub-árvore de T_1 que define esta parte é redundante para o objeto definido por *T,* e poderia ser eliminada. Caso esta sub-árvore seja uma estrutura complexa, isto pode comprometer a eficiência do algoritmo na construção do objeto definido por *T*.

Uma abordagem usual para a detecção de redundâncias é a aproximação de objetos CSG envolvendo-os por uma forma geométrica simples, e derivando critérios para redundância baseados nestas aproximações. Assume-se, para isto, que o objeto CSG está completamente contido na forma geométrica aproximada. Estabelece-se uma classe Σ de formas aproximadas (por exemplo, esferas ou paralelepípedos). O algoritmo tem início aproximando-se todas as primitivas *P* na árvore CSG por uma forma $\sigma(P) \in \Sigma$. Percorrendo-se a árvore das folhas até a raiz, fazem-se todas as aproximações em todos os nós interiores pelas seguintes regras:

1. Se $T = T_1 \cup^* T_2$, então $\sigma(T) = \sigma(\sigma(T_1) \cup^* \sigma(T_2))$.
2. Se $T = T_1 \cap^* T_2$, então $\sigma(T) = \sigma(\sigma(T_1) \cap^* \sigma(T_2))$.
3. Se $T = T_1 -^* T_2$, então $\sigma(T) = \sigma(T_1)$.

Todos os nós correspondentes a movimentos rígidos são distribuídos pelas folhas, ou seja, todas as primitivas devem estar corretamente posicionadas no espaço com respeito ao sistema de coordenadas do sólido final. A Figura A.42 ilustra um exemplo onde as formas geométricas de aproximação utilizadas são retângulos.

Este algoritmo permite estabelecer um critério para se saber se uma primitiva numa sub-árvore em *T* é redundante. Sendo *T'* uma sub-árvore de *T*, então se $\sigma(T') \cap^* \sigma(T) = \phi$, então a sub-árvore *T'* não contribui para a forma final do objeto e pode ser retirada de *T*. Na Figura A.42, a primitiva *D* é redundante, logo pode ser eliminada.

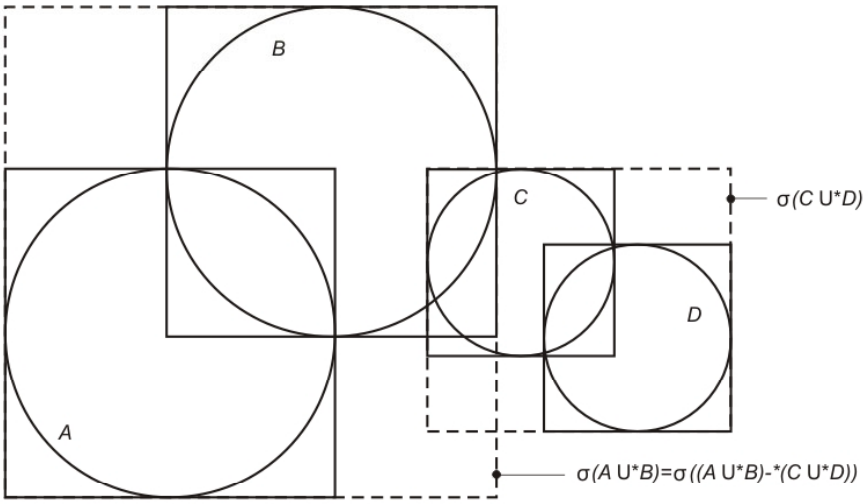


Figura A.42 – Aproximação de ($A \cup^* B$) – ($C \cup^* D$) [9].

A.6.4. Modelos Híbridos

Nas seções anteriores, pôde-se perceber que as três formas de representação de sólidos apresentadas possuíam vantagens e desvantagens que tornavam-nas mais ou menos adequadas de acordo com o tipo de aplicação. Isto motiva a criação de modeladores que abriguem as três formas de representação simultaneamente.

Um modelador híbrido busca escolher a forma mais adaptável a cada tarefa. É necessário que ele garanta consistência dos modelos levando-se em conta o espaço de modelagem em que este está inserido. Algoritmos de conversão entre as formas de representação também estão presentes. Pode-se haver inclusive vários tipos de representação de fronteira num mesmo modelador, como por exemplo, uma representação *winged-edge* para procedimentos de modelagem e outra para fins de visualização.

Alguns problemas inerentes aos modeladores híbridos podem ser enumerados, no entanto. As conversões entre as formas de representação constituem um deles. Alguns tipos de conversões são mais simples que outras. Modelos CSG, por exemplo, são facilmente convertidos para quaisquer outras formas de representação. Contudo, converter modelos B-Rep para modelos CSG é uma tarefa complexa e em muitos casos sem solução. A conversão de modelos B-Rep para modelos de decomposição também é simples. Os modelos de decomposição podem ser convertidos para modelos B-Rep ou CSG no caso de uma imagem binária precisar ser mapeada contra informações armazenadas

como B-Rep ou CSG. No entanto, esta não constitui uma forma de conversão muito explorada nos modeladores de sólidos em geral. Modeladores de sólidos B-Rep ou CSG costumam utilizar modelos de decomposição apenas como representações transitórias, isto é, modelos de decomposição são criados apenas para resolver certos problemas, e descartados depois.

A consistência do modelo constitui outro problema existente em modeladores híbridos. Normalmente, a garantia de consistência é realizada limitando-se as funcionalidades disponíveis no modelador. Pode-se citar como exemplo o fato dos modelos CSG não incluírem superfícies paramétricas. Isto significa que se um modelo B-Rep criado pela conversão de um modelo CSG for modificado para incluir superfícies paramétricas, a representação CSG original do objeto modelado não pode ser mantida consistente. Normalmente, para que um modelador híbrido possa garantir a consistência entre todas as formas de representação, ele deve abrigar operações em sólidos que possam ser realizadas em qualquer uma destas formas. As operações booleanas ocupam uma posição privilegiada em relação a este aspecto, pois podem ser mapeadas por todas as principais formas de representação de sólidos.

As principais arquiteturas de modeladores híbridos são mostradas nas Figuras A.43a e A.43b. Normalmente, existe uma forma de representação que é tida como a *principal*, enquanto as outras são secundárias. As formas de representação secundárias, no entanto, usualmente não estão disponíveis para acesso direto pelo usuário, servindo apenas como formas de representação auxiliar.

O modelador MG, utilizado neste trabalho como ambiente de modelagem onde as operações booleanas foram implementadas, utiliza uma representação B-Rep. Contudo, a nova interface do programa permite que o usuário possa gerar um modelo da mesma forma como o faria num modelador CSG, ou seja, a partir da aplicação das operações booleanas em primitivas para se chegar ao objeto desejado. Apesar da interface ter sido adaptada para este fim, fazendo com que o usuário possa explicitamente decidir se deseja criar o objeto por meio da inserção de curvas e superfícies ou por meio das operações booleanas, nenhuma representação interna CSG foi implementada em paralelo. As informações são manipuladas diretamente, através das relações de adjacência entre os elementos topológicos que constituem o modelo.

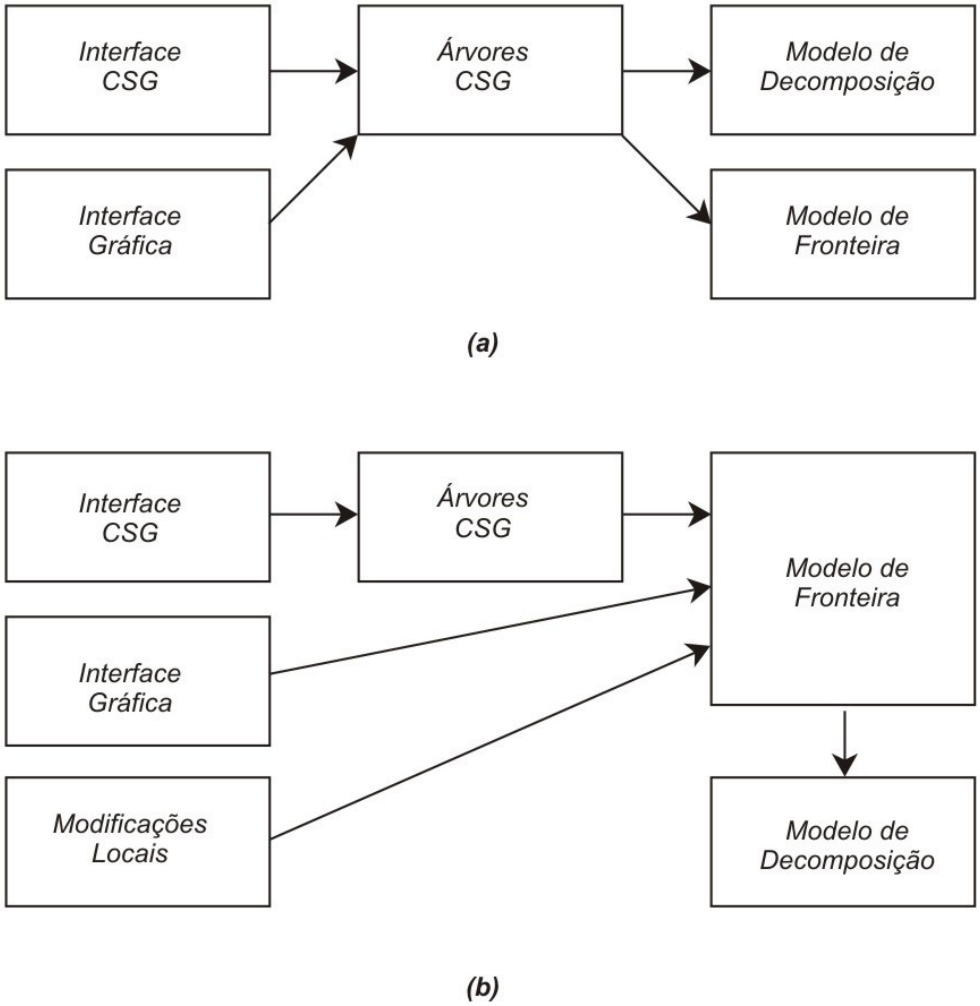


Figura A.43 – Principais arquiteturas dos modeladores híbridos: a) CSG como representação primária; b) B-Rep como representação primária [10].