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.

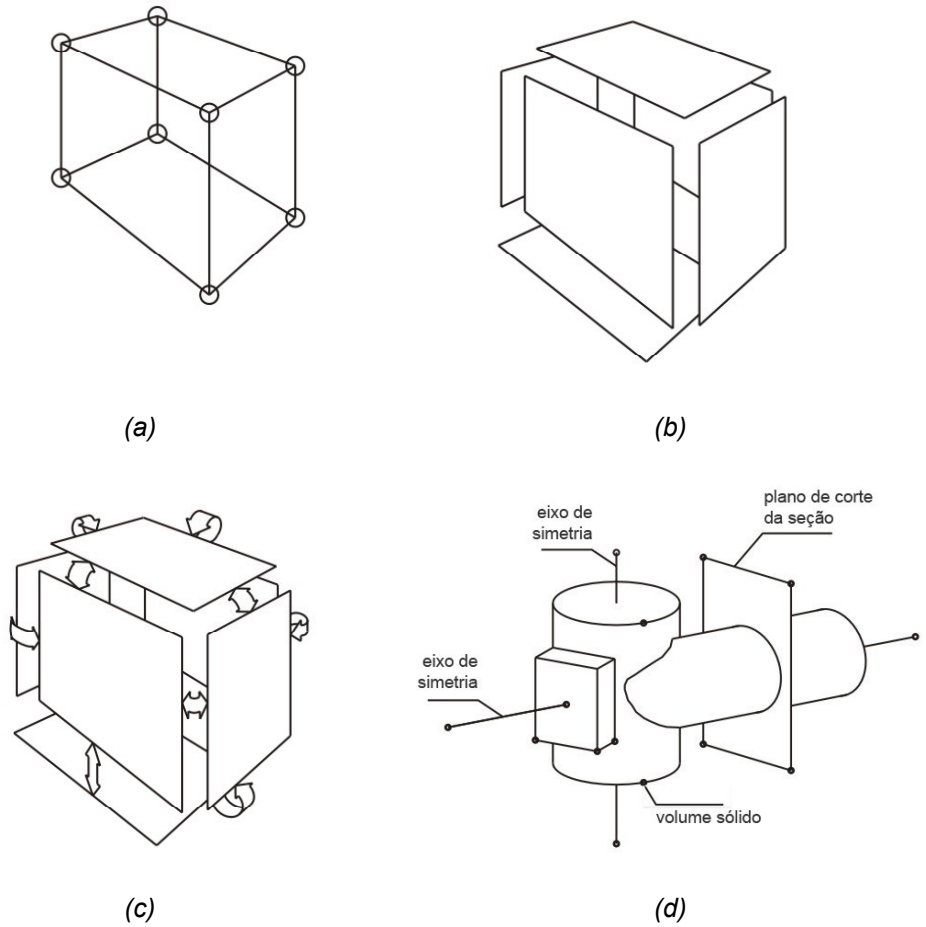


Figura A.1 – Formas de modelagem: a) por arames; b) por superfícies; c) modelagem de sólidos; d) modelagem *non-manifold*.

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.

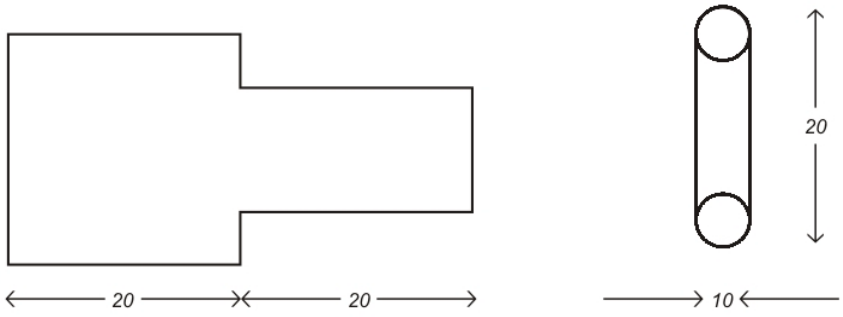


Figura A.2 – Desenho sem sentido [10].

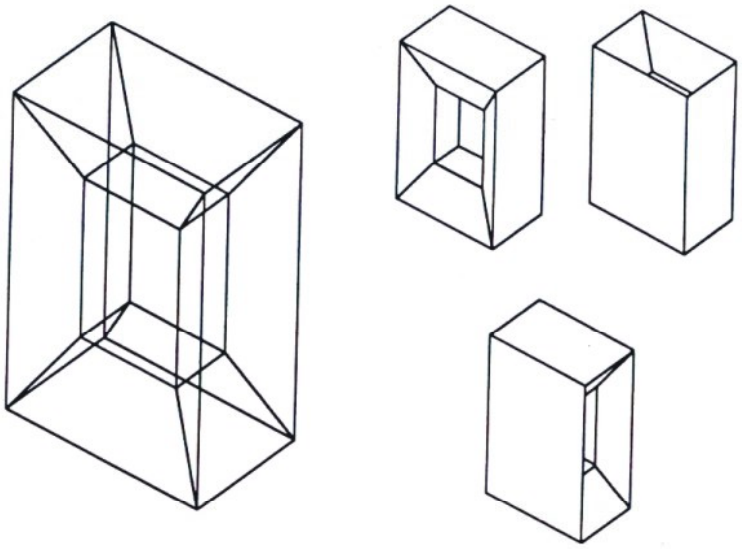


Figura A.3 – Objeto sólido ambíguo [10].

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.

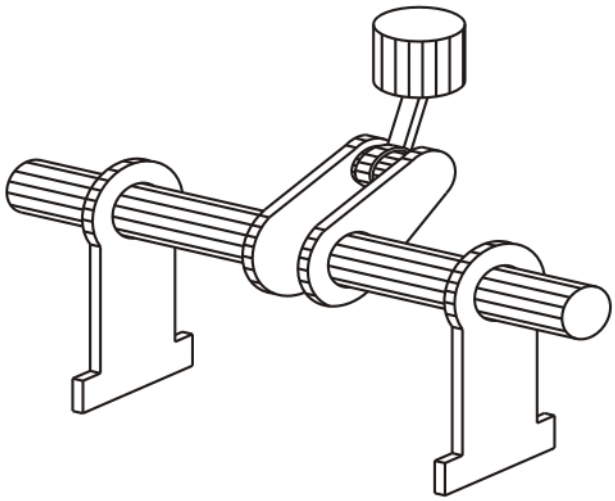


Figura A.4 – Modelo poliédrico [10].

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?

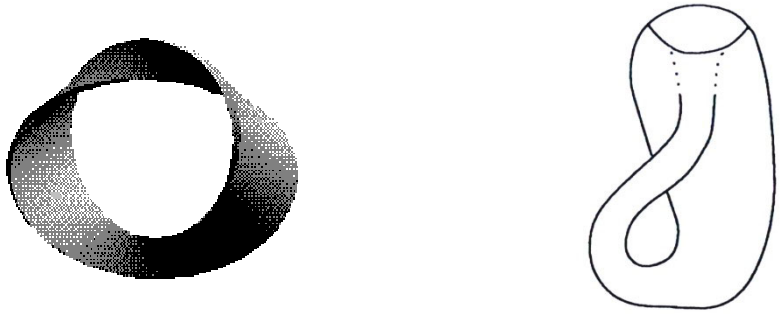
As respostas a tais perguntas podem ser uma imagem, um número, uma constante booleana (verdadeiro ou falso) ou mesmo um outro modelo sólido que represente o resultado de um processo, como no caso de uma pergunta do tipo: qual o efeito deste processo de manufatura aplicado a este objeto?

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.

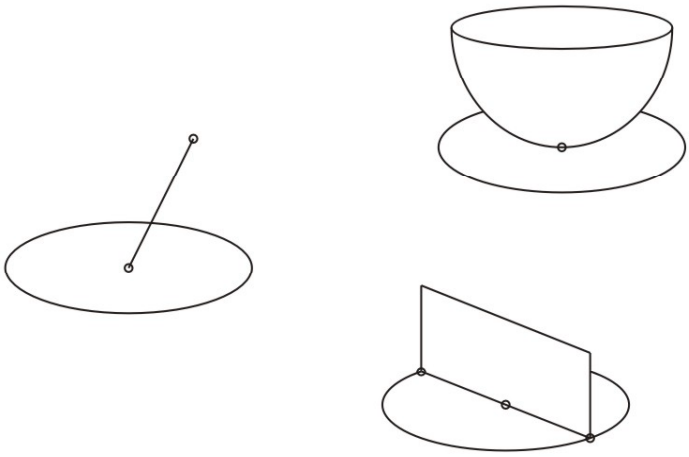


Figura A.6 – Situações topológicas *non-manifold* [17].

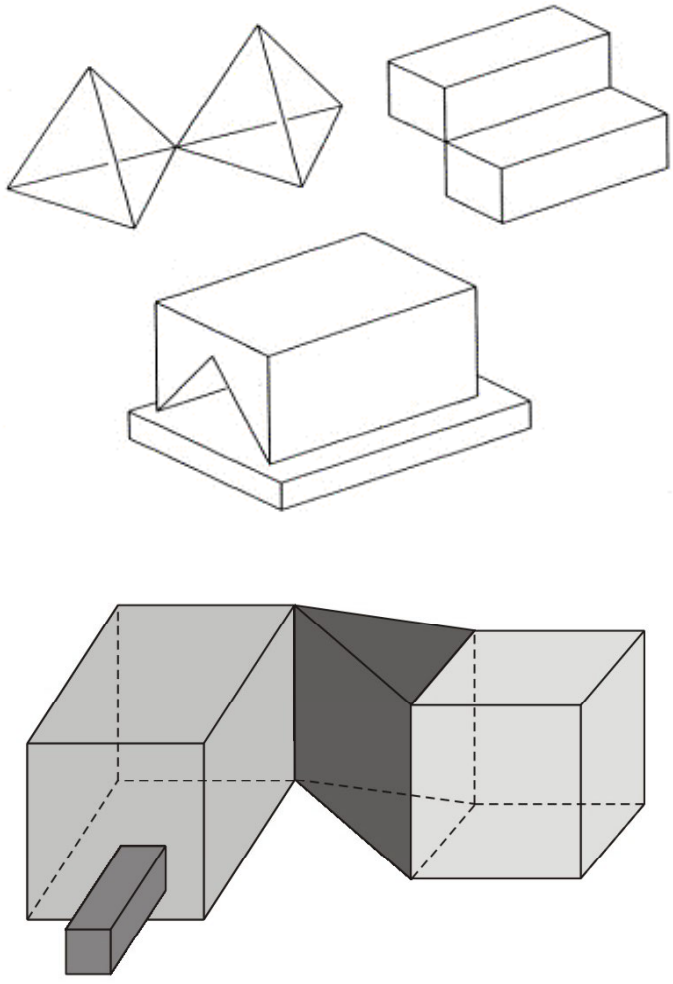


Figura A.7 – Exemplo de sólidos *non-manifold* [10,19].

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.

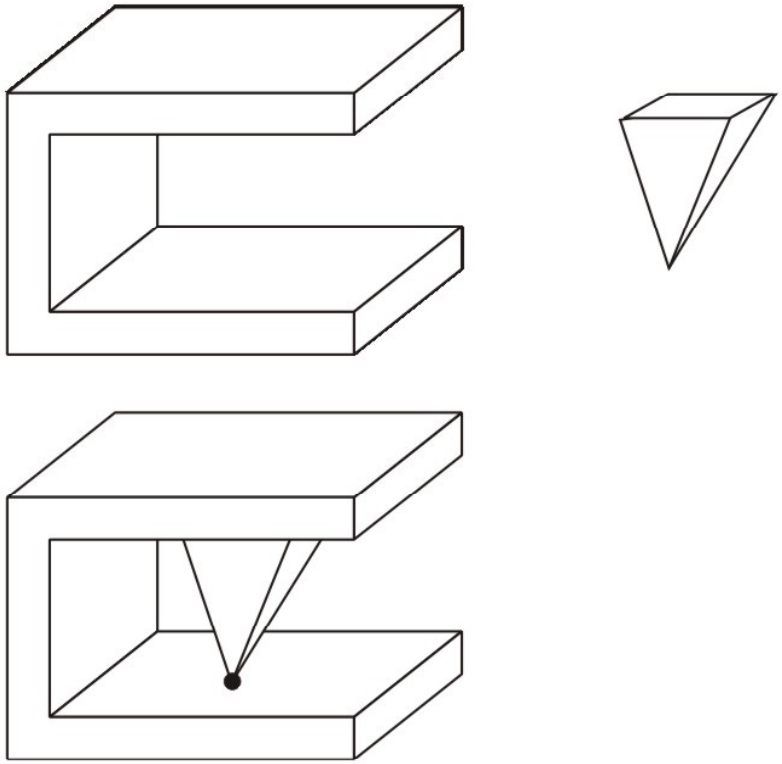
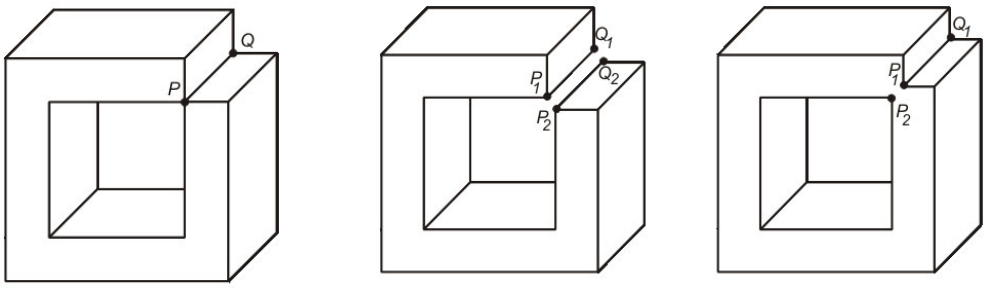


Figura A.8 – União regularizada de dois objetos *manifold* gerando um resultado *non-manifold* [17].



(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.

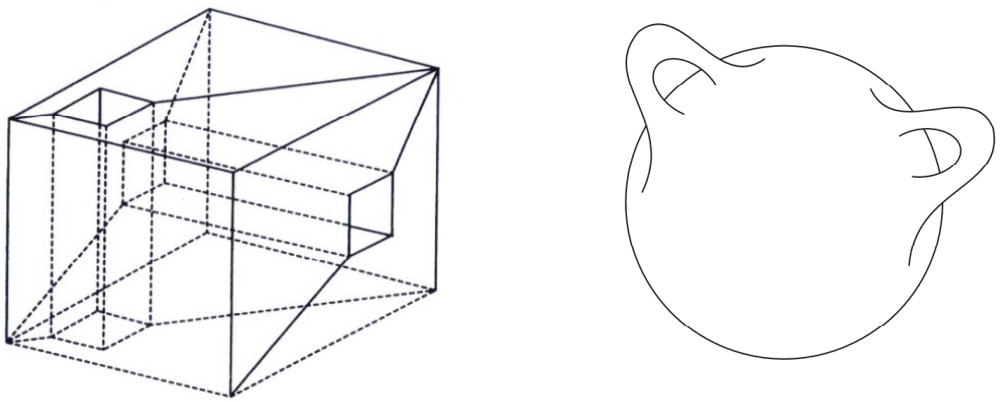
Um grafo pode ser *imerso* em (ou *mapeado* sobre) uma superfície se é desenhado nessa superfície sem que nenhum par de arestas se intercepte, exceto nos seus vértices incidentes. Um *grafo planar* é aquele que pode ser imerso numa superfície planar. Um grafo também pode ser imerso num espaço tridimensional, 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*.

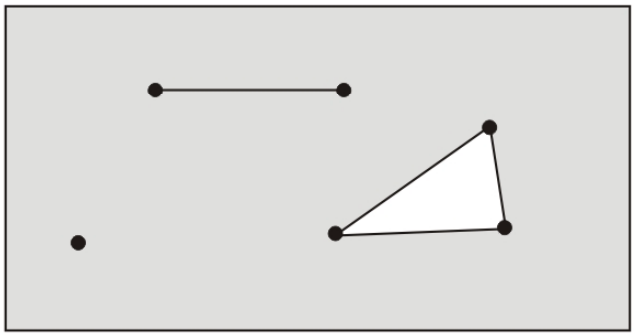


Figura A.11 – Face com quatro *loops* [9].

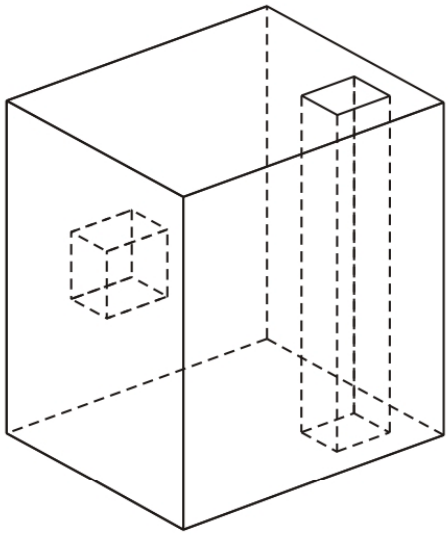


Figura A.12 – Sólido que obedece à equação de Euler-Poincaré estendida [9].

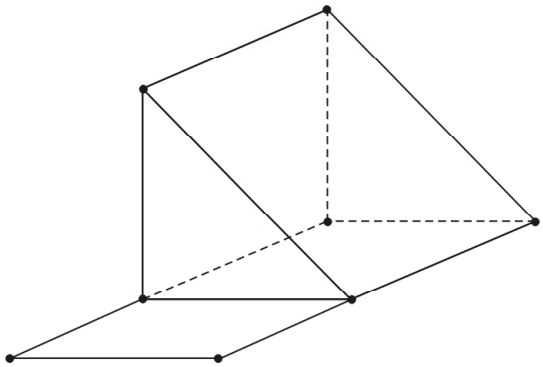


Figura A.13 – Superfície com uma aresta *non-manifold* [9].

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.

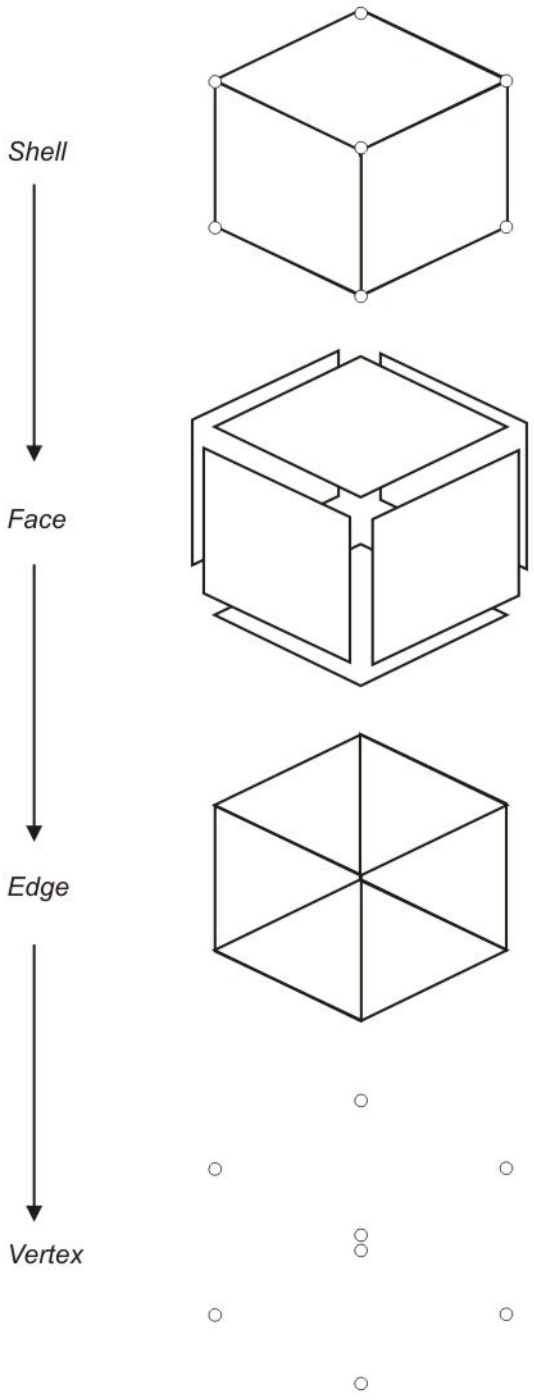


Figura A.14 – Representação hierárquica decrescente dos elementos topológicos [17].

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.

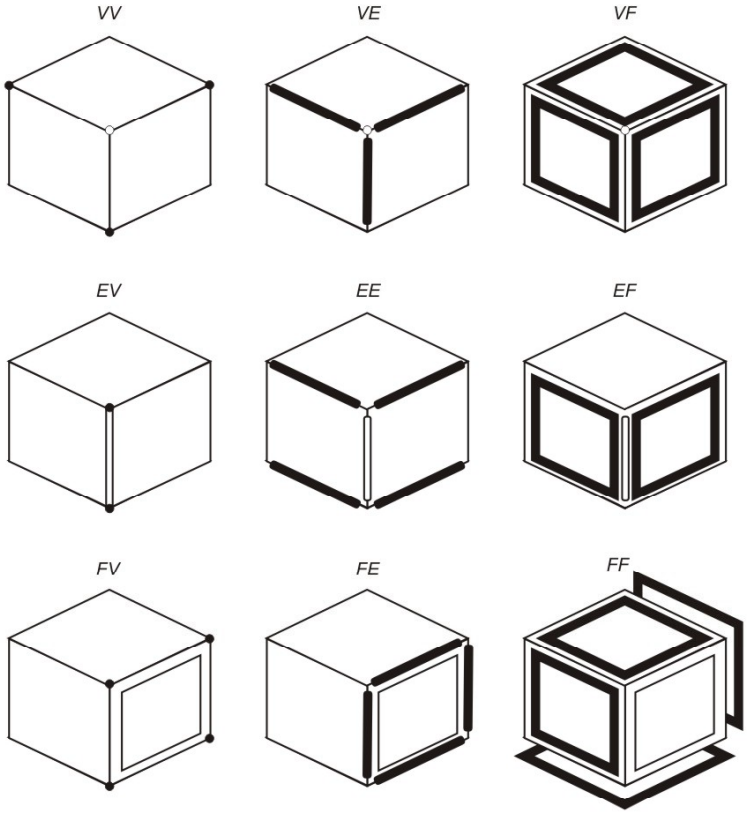


Figura A.15 – Nove relações de adjacência possíveis entre vértices, arestas e faces [17].

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.

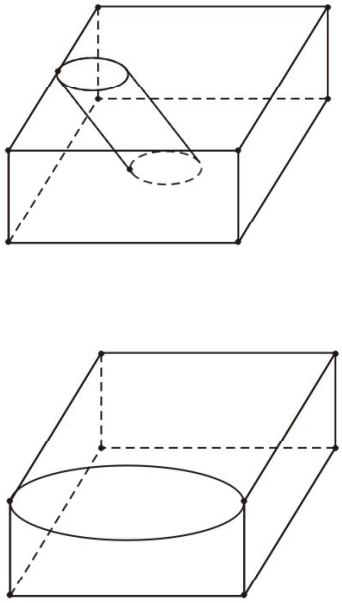


Figura A.16 – *Self-loops* e multigrafos gerados pela aplicação das operações booleanas a sólidos: a) *self-loop* criado pela subtração de um cilindro de um paralelepípedo; b) multigrafo criado pela subtração de uma esfera de um paralelepípedo [17].

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.

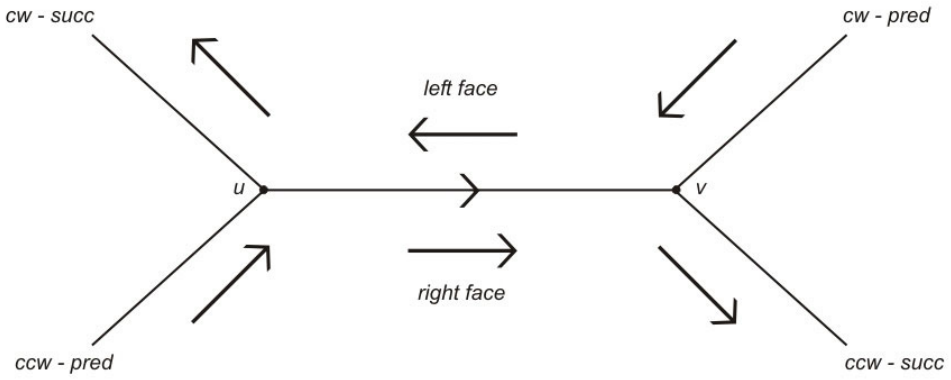


Figura A.17 – Estrutura de dados *winged-edge* [9].

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.

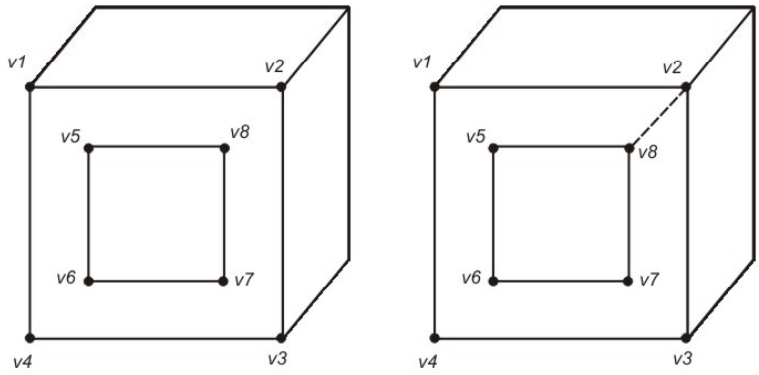


Figura A.18 – Aresta auxiliar para conectar contornos separados de uma face [17].

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.

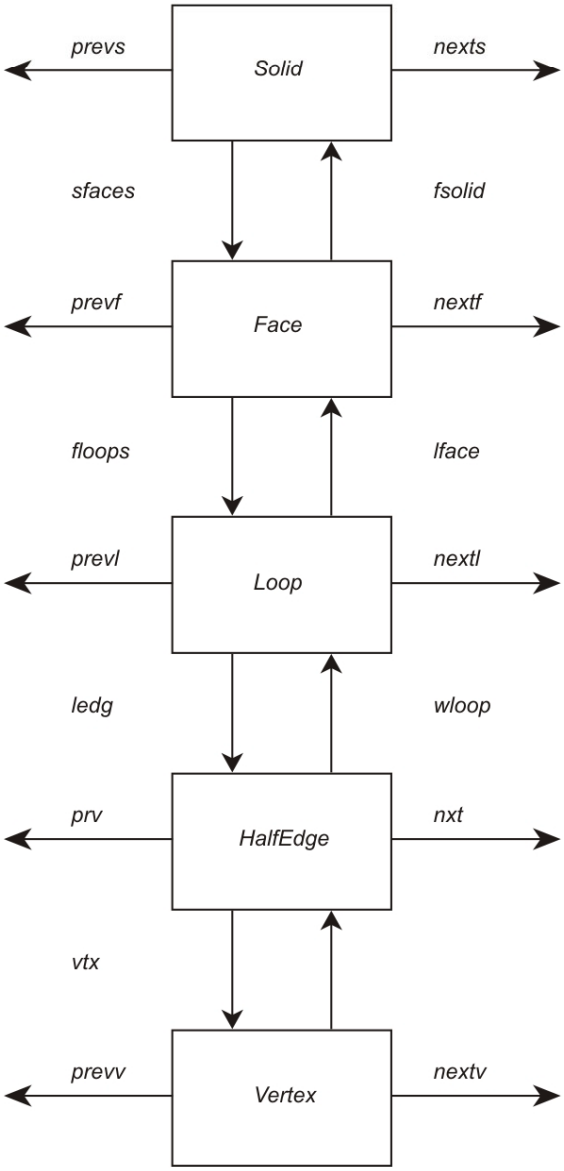


Figura A.19 – Estrutura de dados *half-edge* [10].

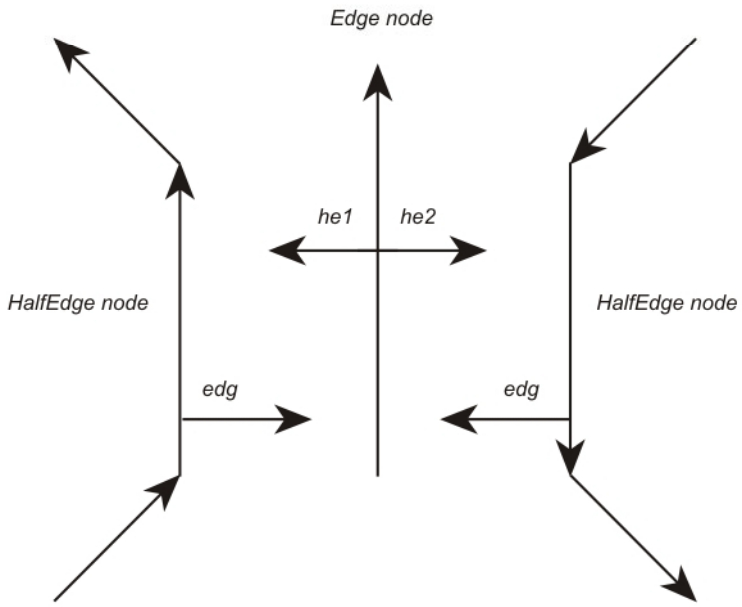


Figura A.20 – Relação entre aresta e semi-aresta na estrutura de dados *half-edge* [10].

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.

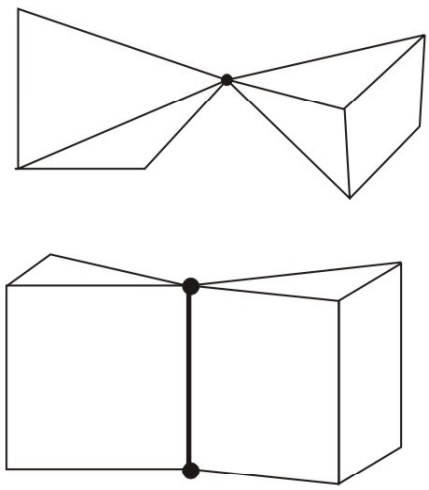


Figura A.21 – Condições *non-manifold* em um vértice e em uma aresta [17].

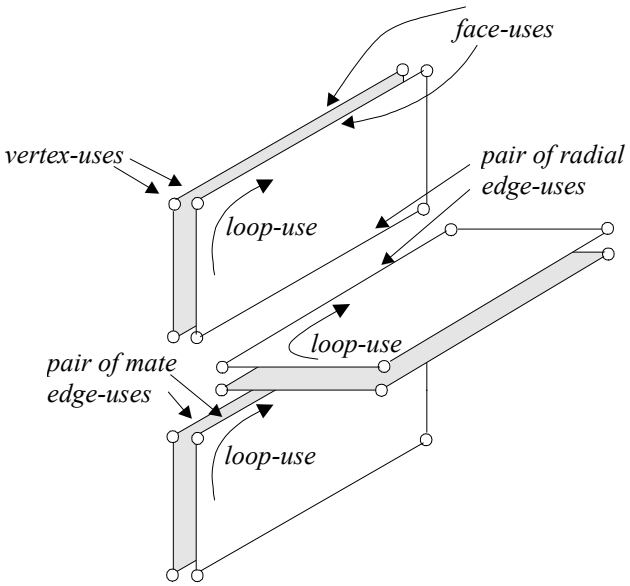


Figura A.22 – Elementos topológicos na RED [6].

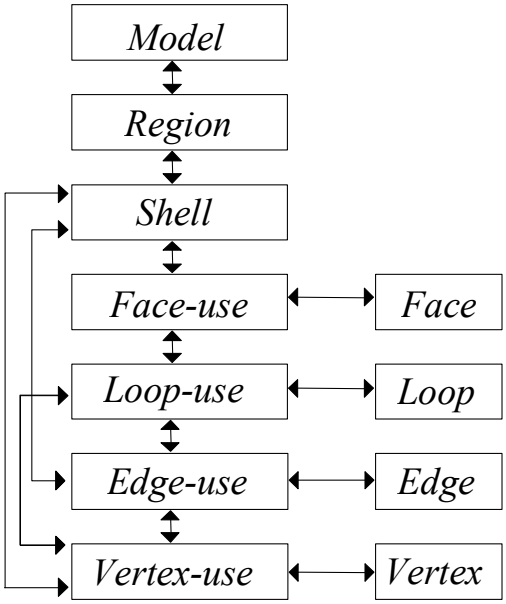


Figura A.23 – Descrição hierárquica da RED [6].

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.

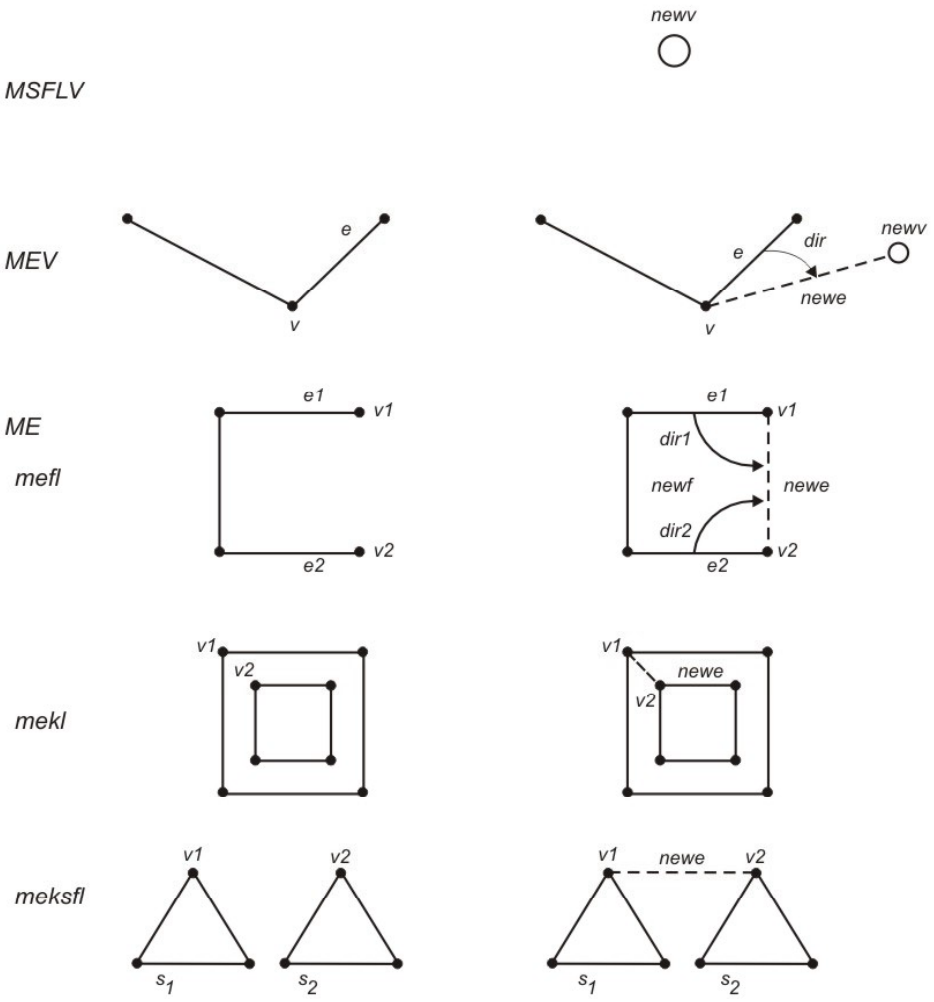


Figura A.24 – Aplicação dos operadores de Euler [17].

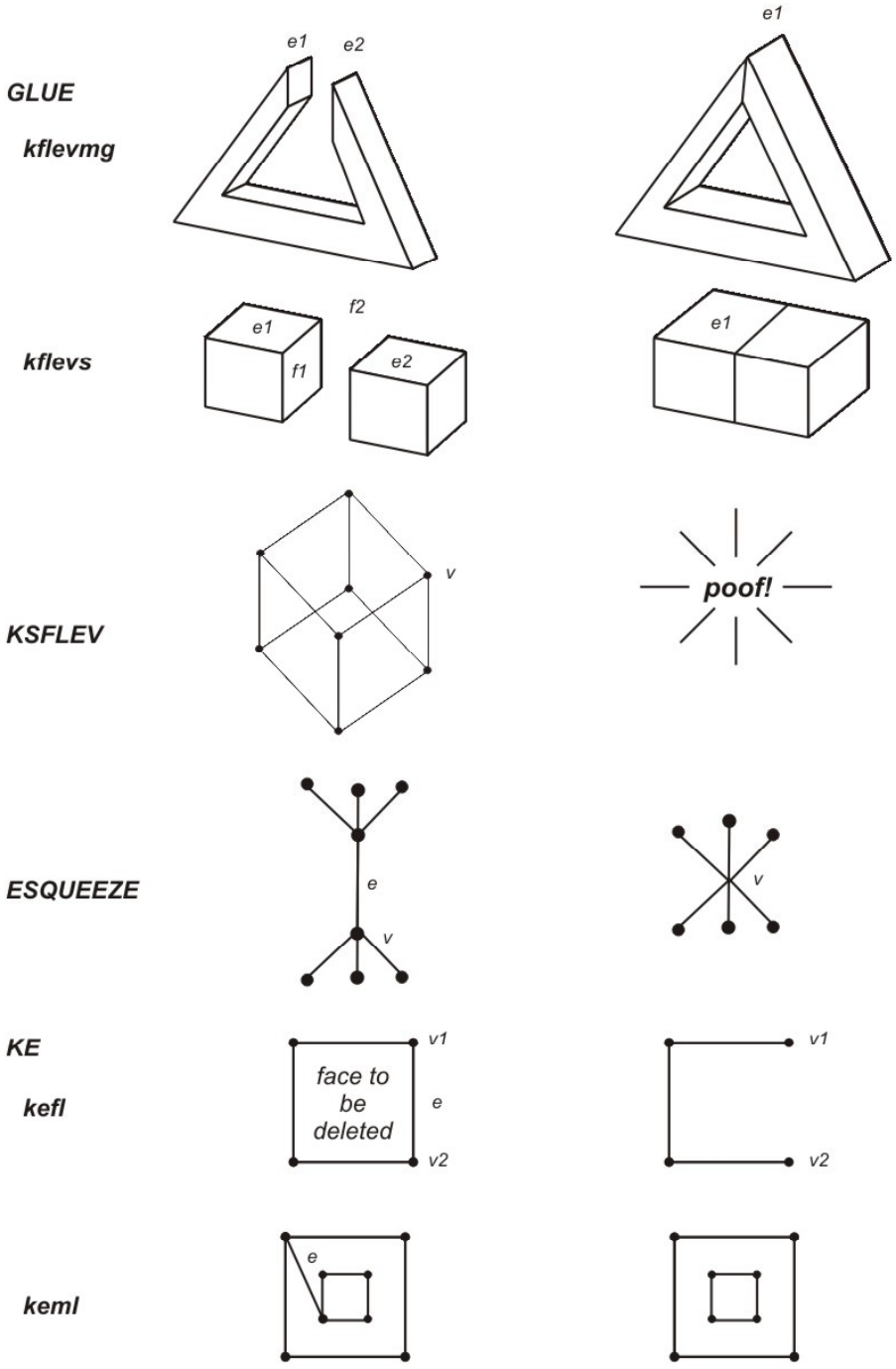


Figura A.25 – Aplicação dos operadores de Euler [17].

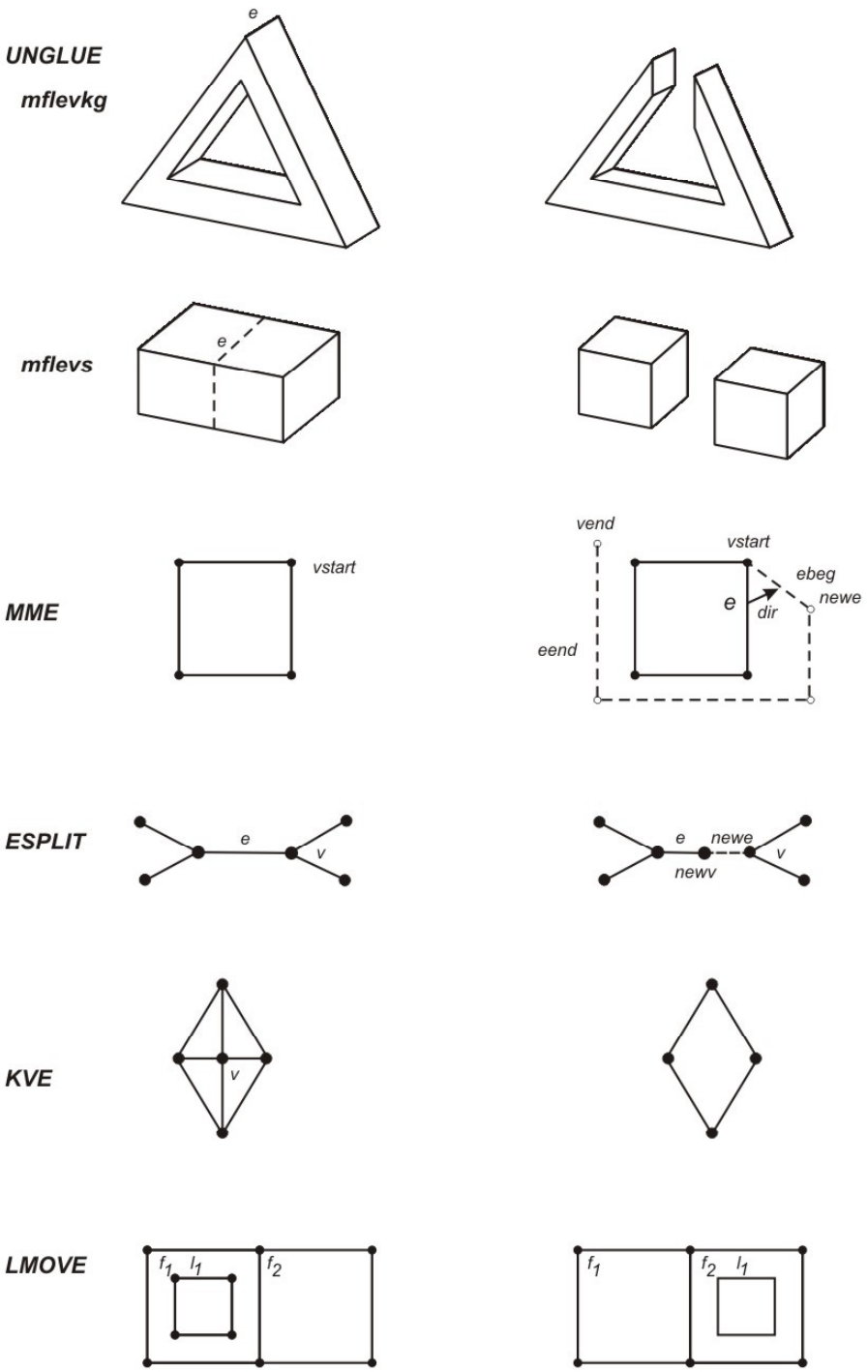


Figura A.26 – Aplicação dos operadores de Euler [17].

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		

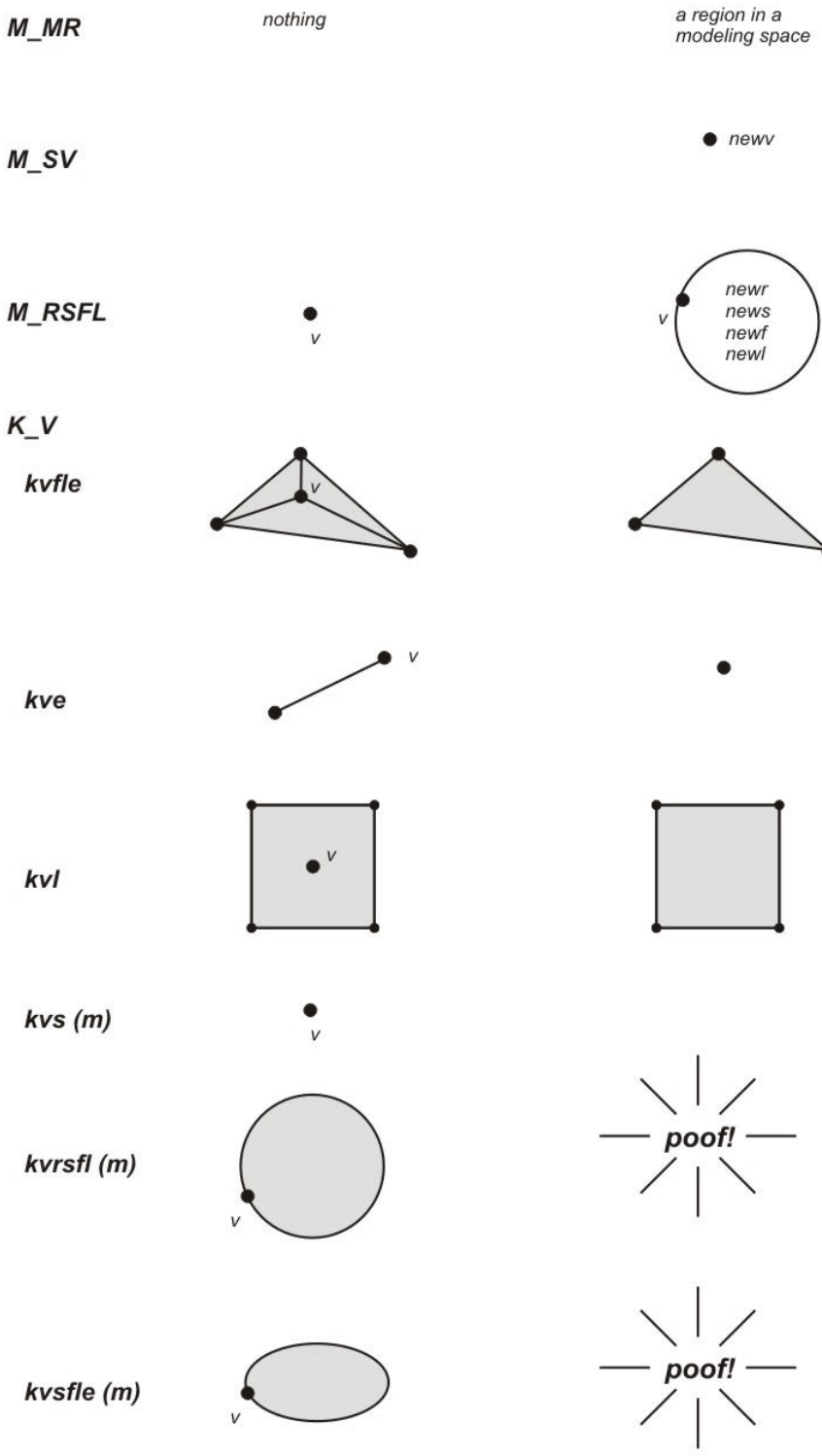


Figura A.27 – Aplicação dos operadores *non-manifold* de Weiler [17].

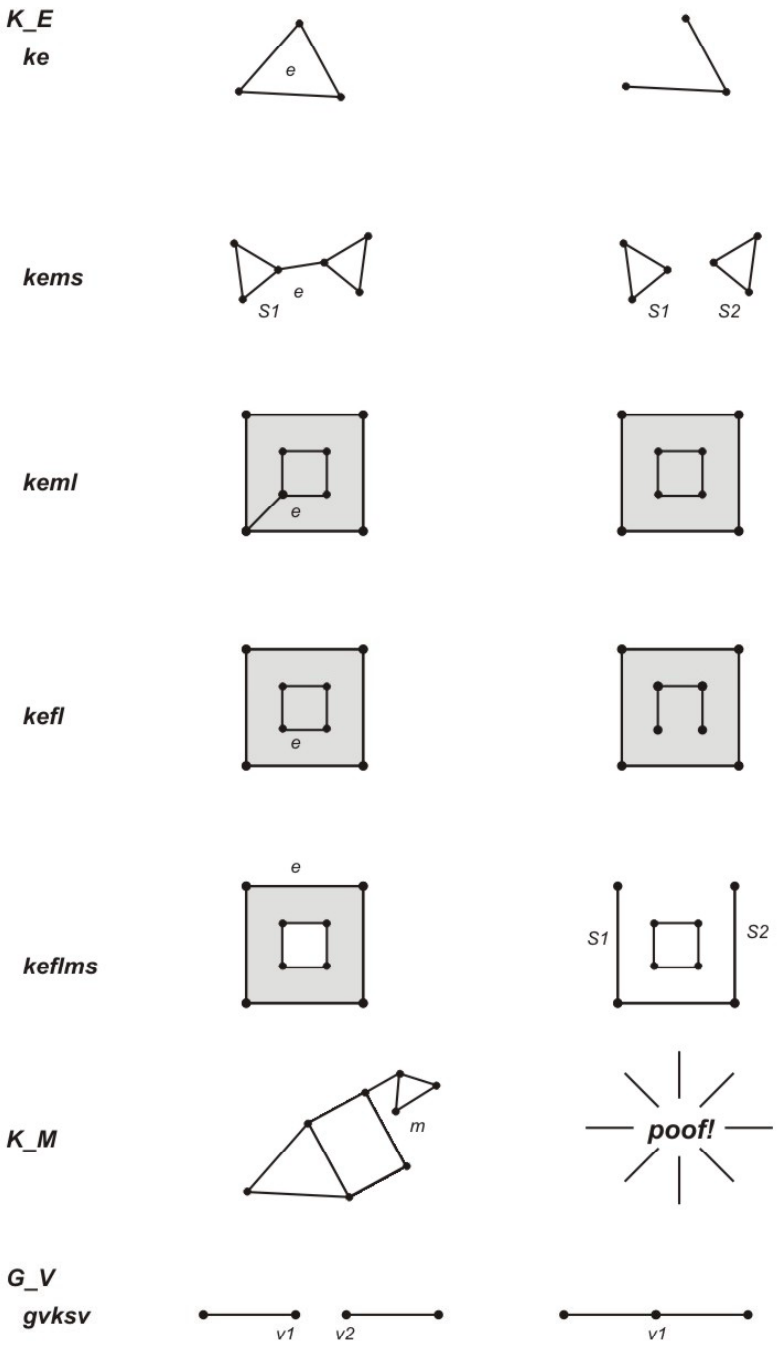


Figura A.28 – Aplicação dos operadores *non-manifold* de Weiler [17].

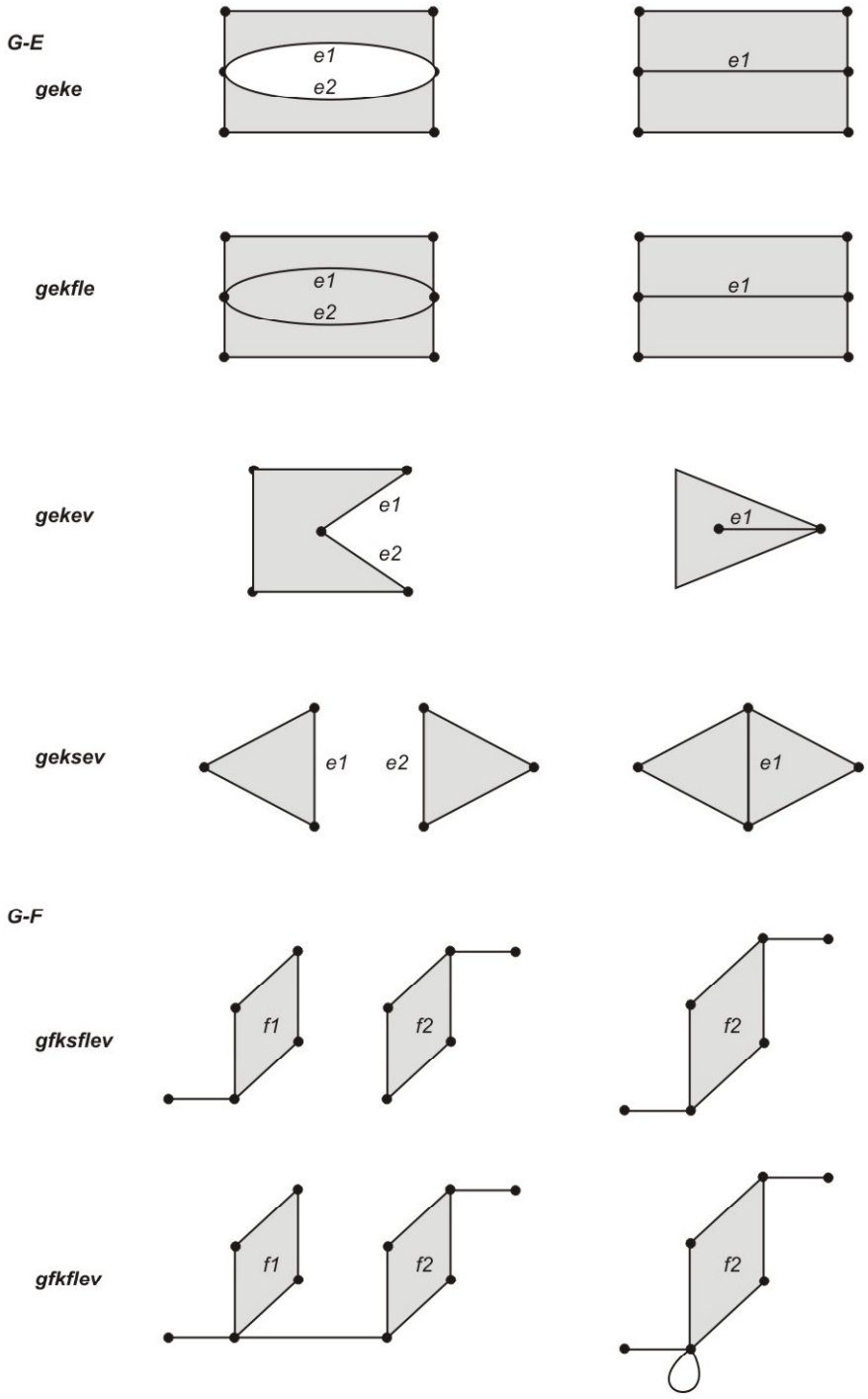


Figura A.29 – Aplicação dos operadores *non-manifold* de Weiler [17].

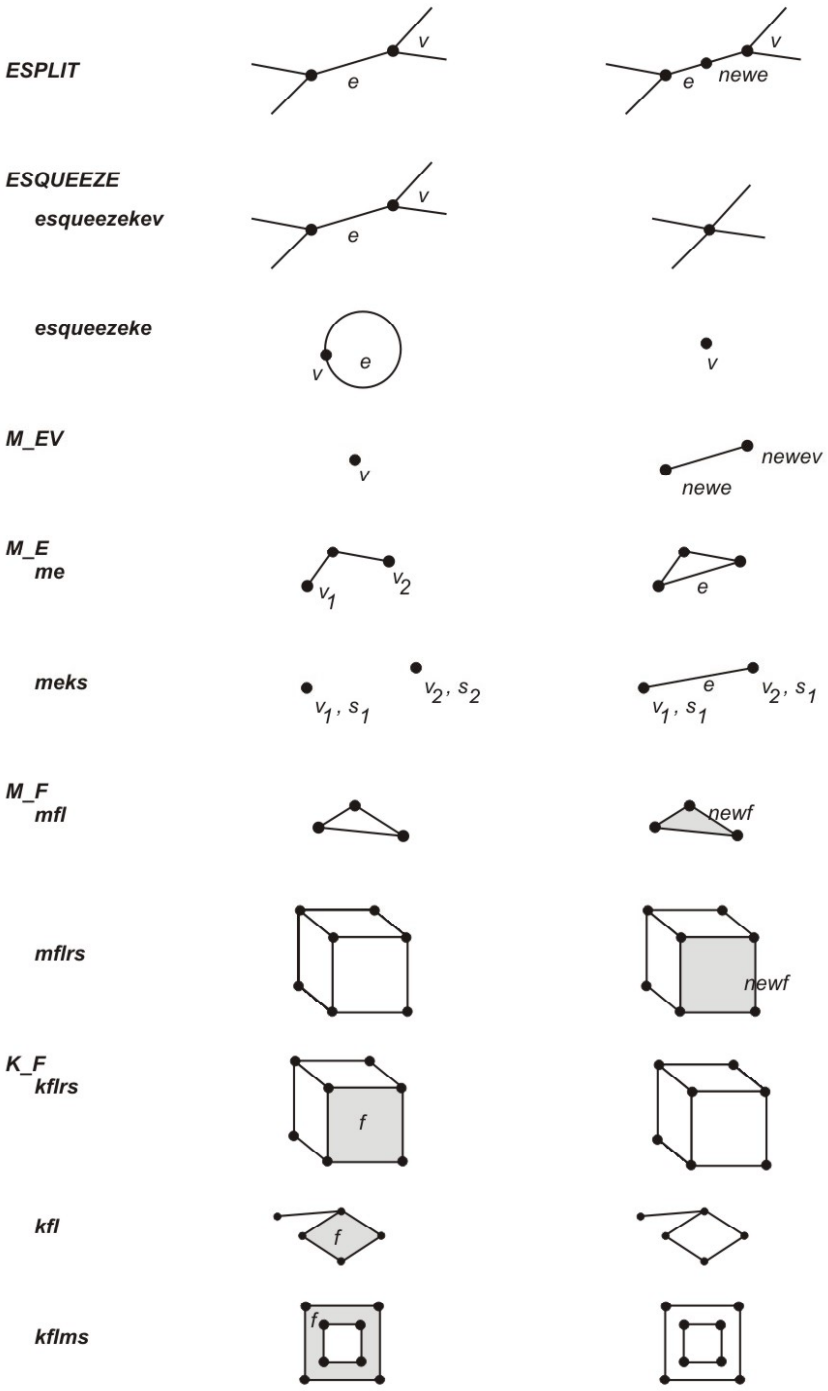


Figura A.30 – Aplicação dos operadores *non-manifold* de Weiler [17].

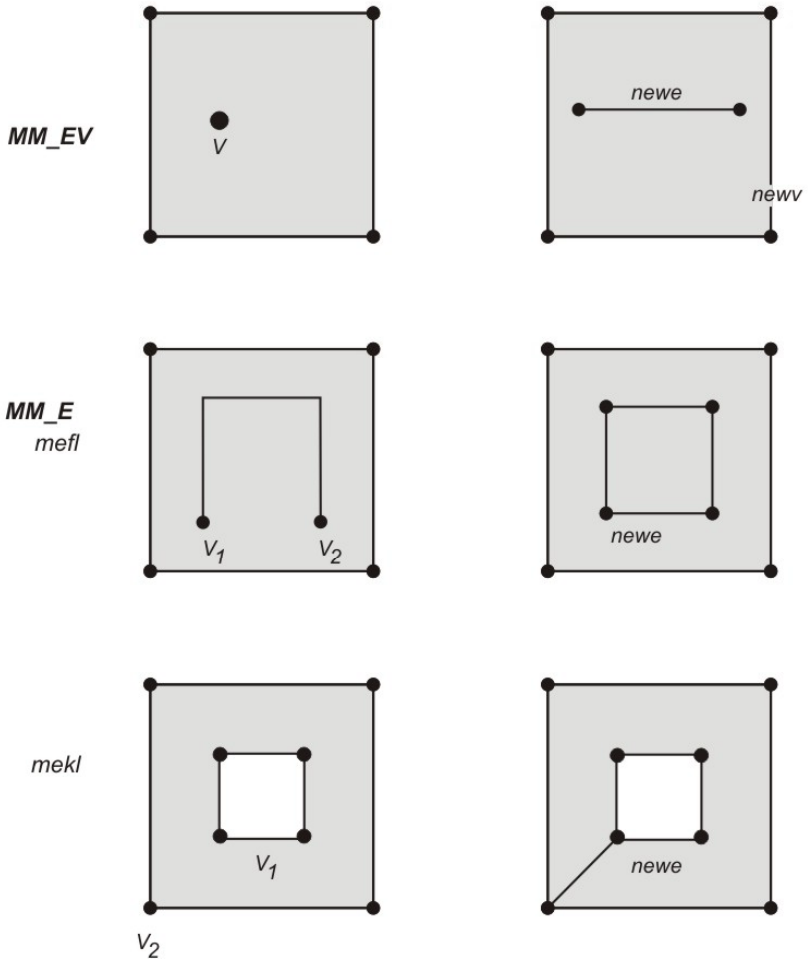


Figura A.31 – Aplicação dos operadores *non-manifold* de Weiler [17].

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.

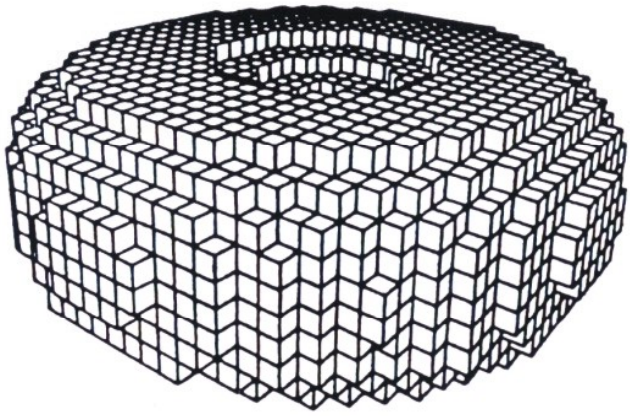


Figura A.32 – Subdivisão uniforme do espaço [10].

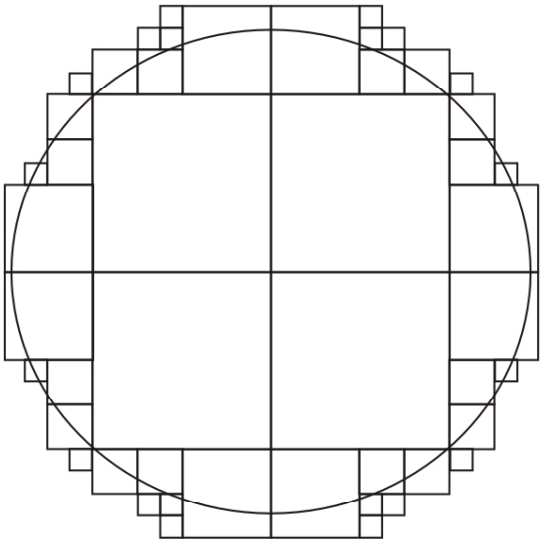


Figura A.33 – Subdivisão do espaço por Octree [9].

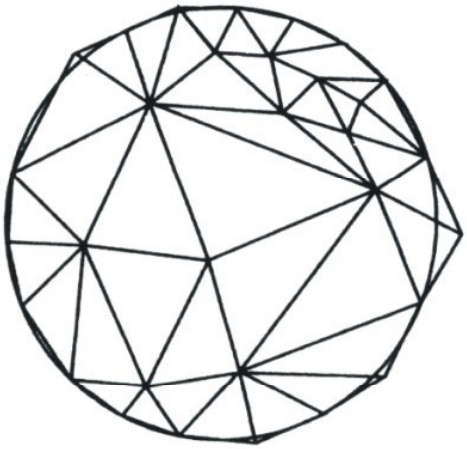


Figura A.34 – Subdivisão irregular do espaço [9].

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).

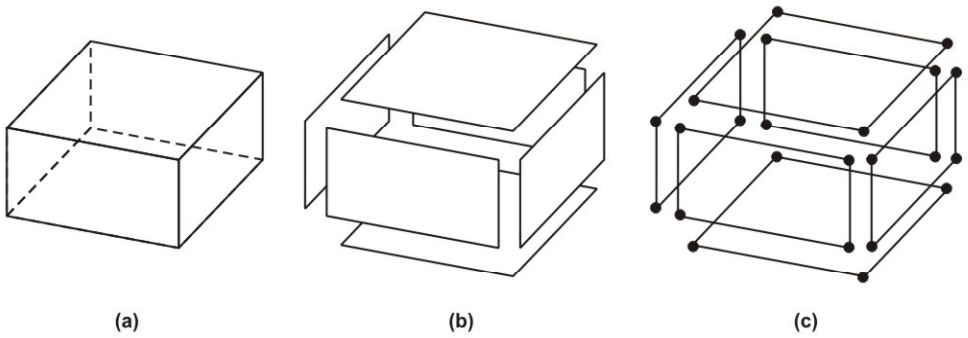


Figura A.35 – Componentes básicos de um modelo de fronteira [10].

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.

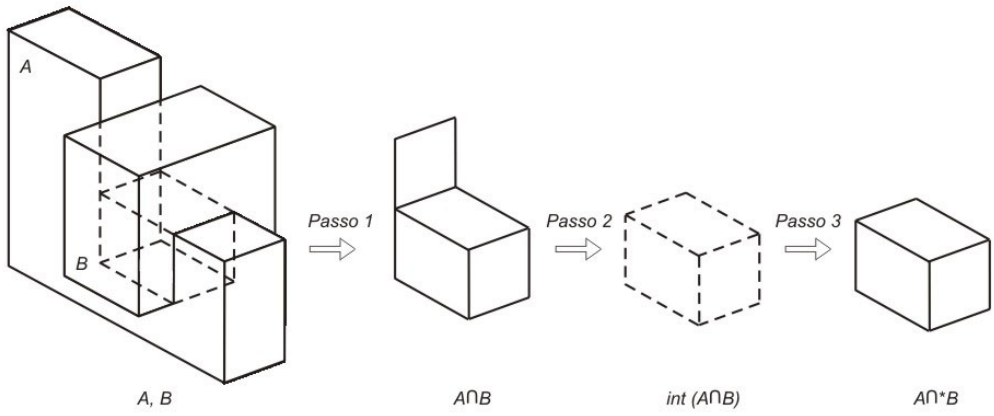


Figura A.36 – Interseção regularizada entre sólidos [9].

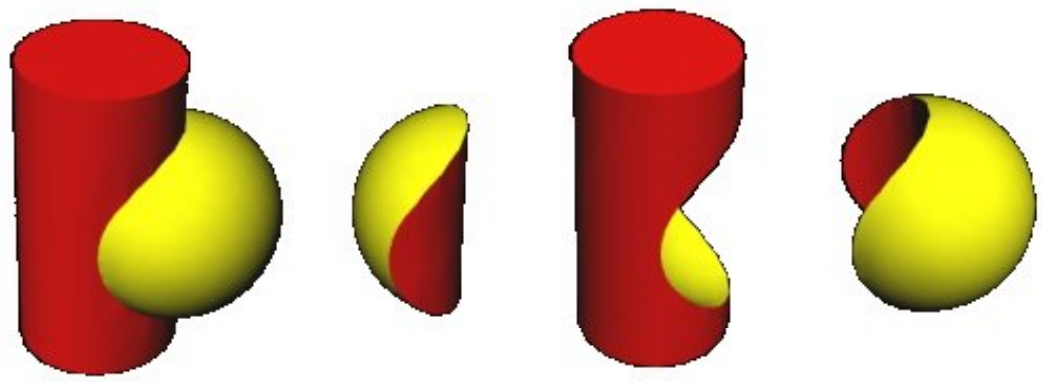


Figura A.37 – Operações booleanas aplicadas a dois sólidos: união, interseção e diferenças [56].

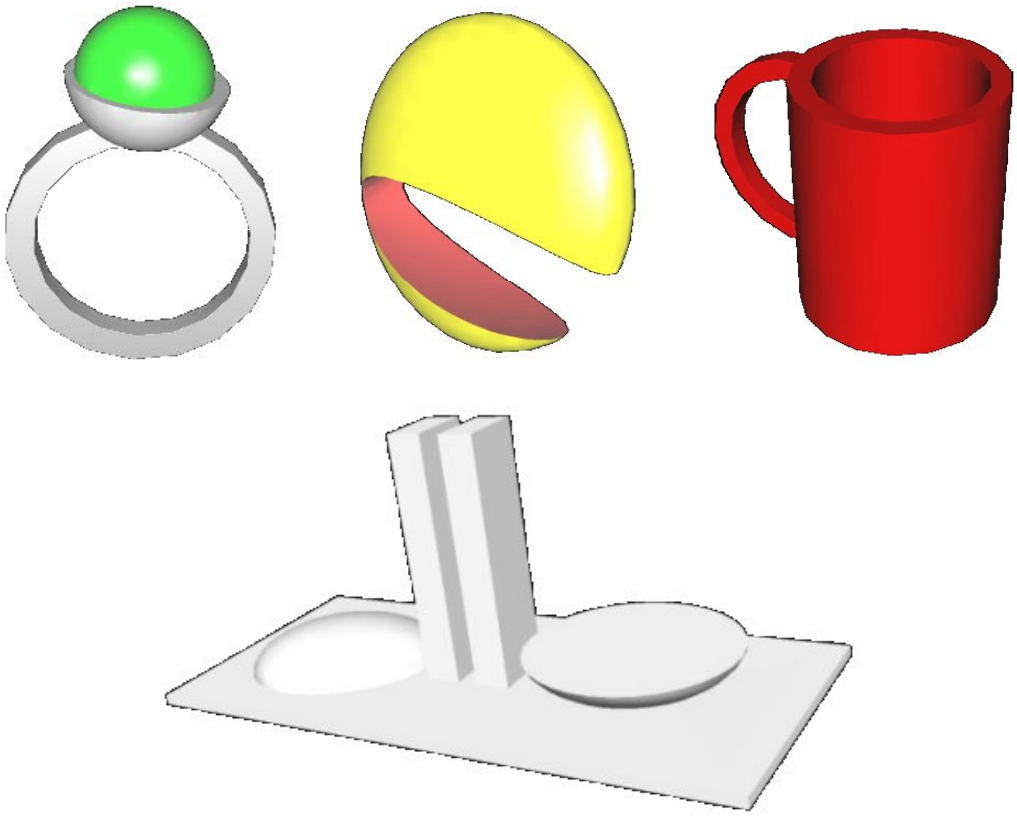


Figura A.38 - Sólidos gerados por meio de operações booleanas em primitivas básicas [56].

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)

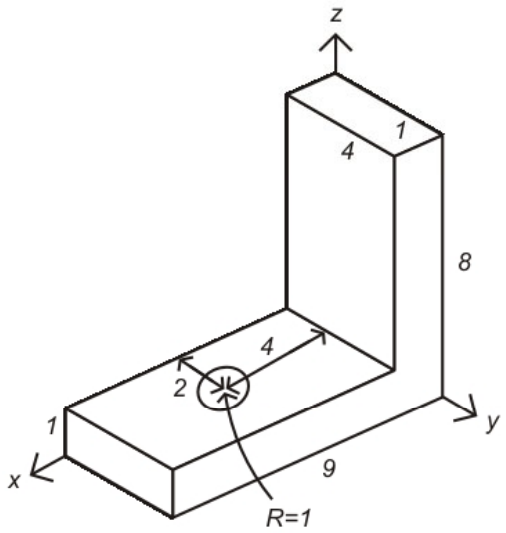


Figura A.39 – Suporte [9].

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.

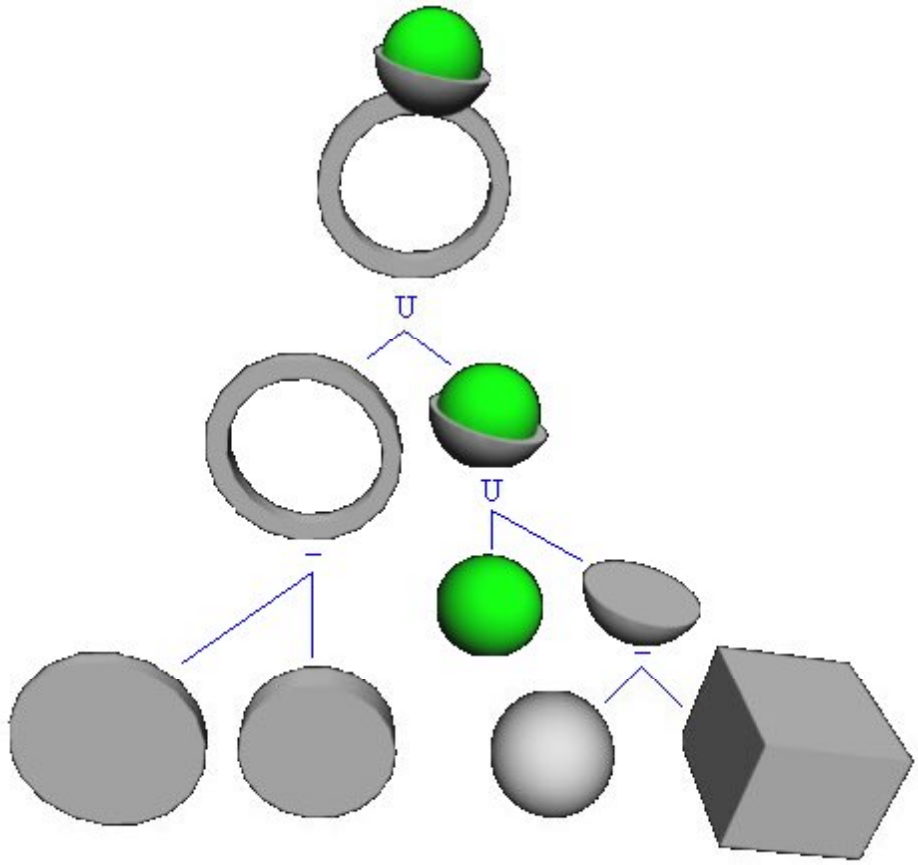


Figura A.40 – Árvore CSG [56].

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.

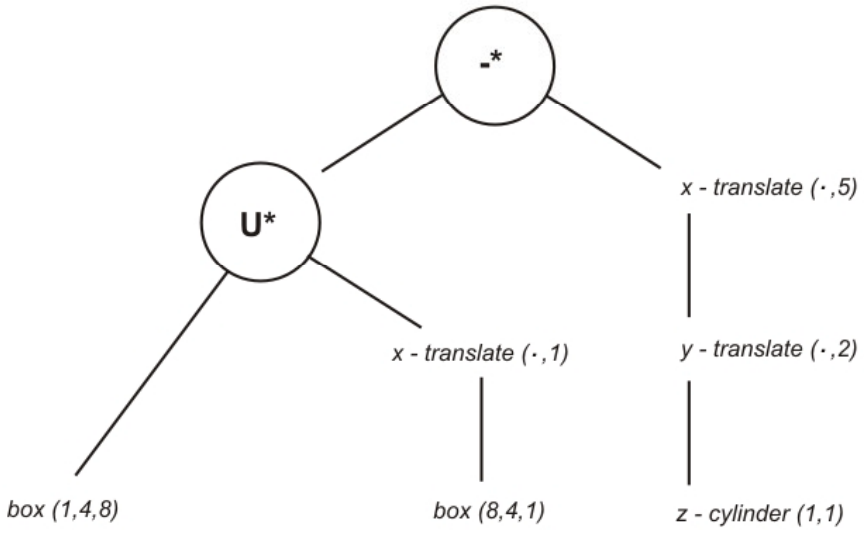


Figura A.41 – Árvore representando a expressão CSG [9].

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.

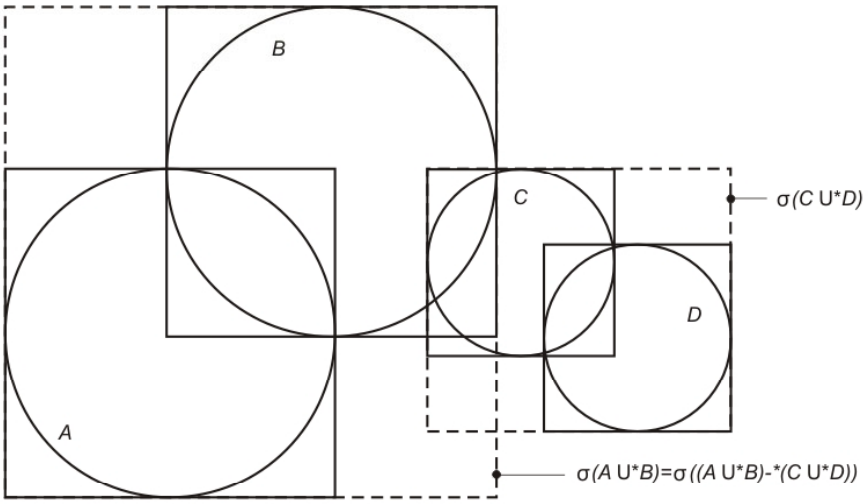


Figura A.42 – Aproximação de ($A \cup^* B$) – ($C \cup^* D$) [9].

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.

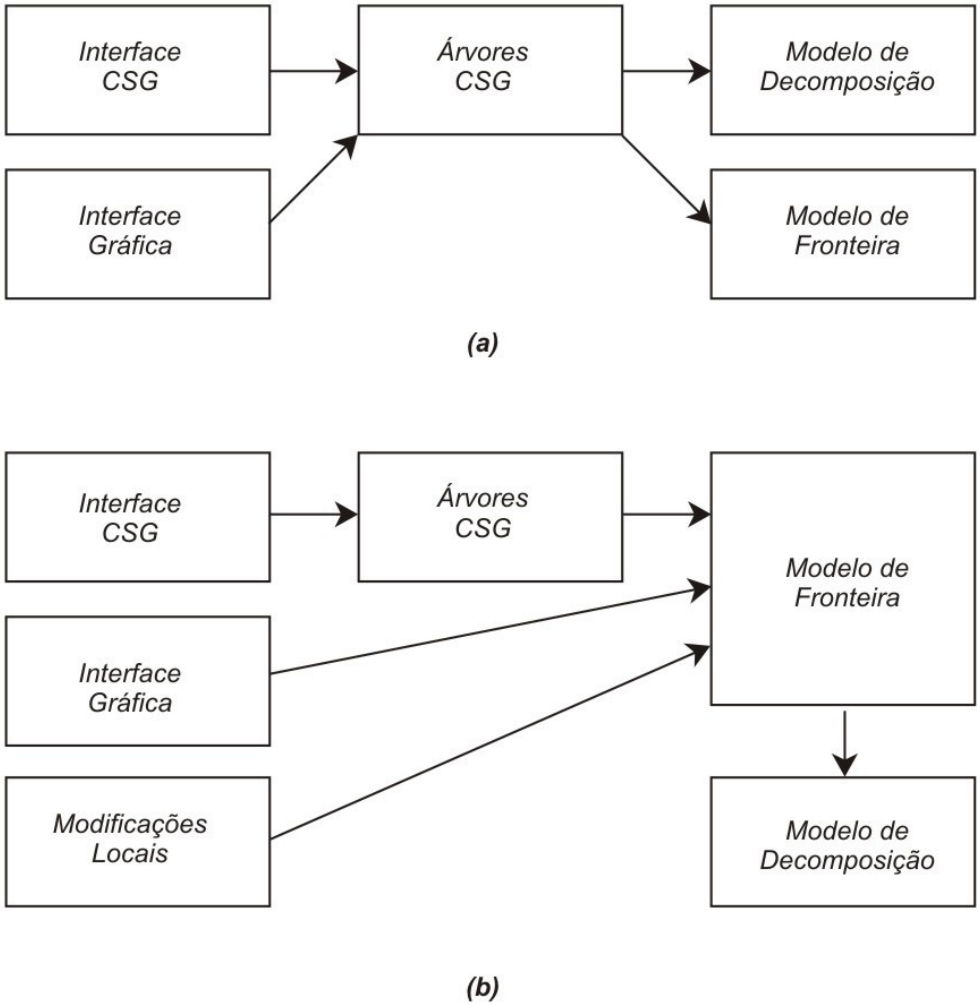


Figura A.43 – Principais arquiteturas dos modeladores híbridos: a) CSG como representação primária; b) B-Rep como representação primária [10].