3
Algoritmo para Operações Booleanas

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

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

3.1.
Domínio representacional *non-manifold*

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

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

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

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

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

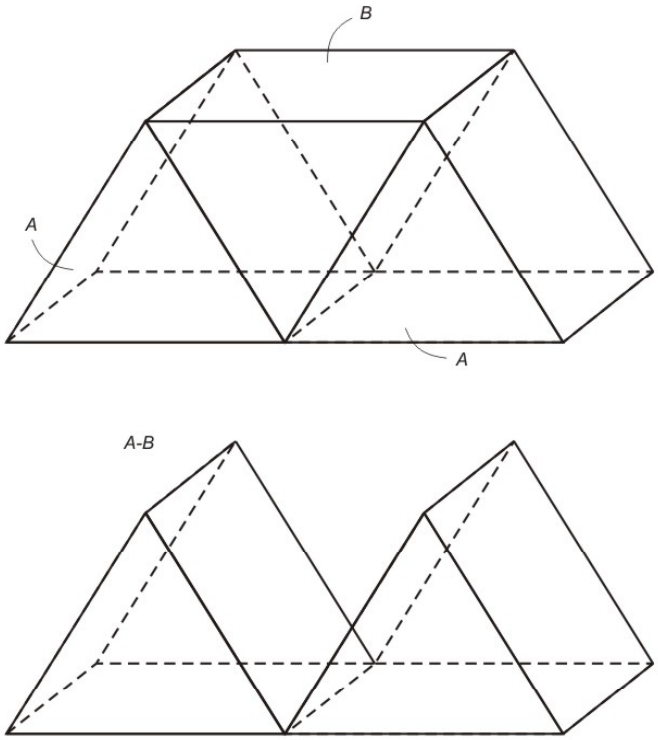


Figura 3.1 – Operação de diferença levando a um resultado *non-manifold*.

3.2. Conceito de fronteira

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

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

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

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

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

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

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

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

3.3. Grupos como parâmetros de entrada

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

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

3.4. Condições de aplicabilidade do algoritmo

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

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

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

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

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

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

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

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

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

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

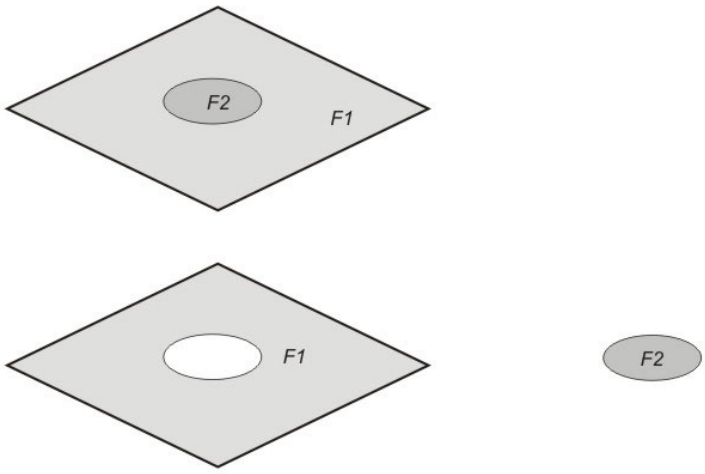


Figura 3.2 – Superposição de faces: na verdade, a face F_1 possui um *loop* interno formando uma cavidade onde se encaixa a face F_2.

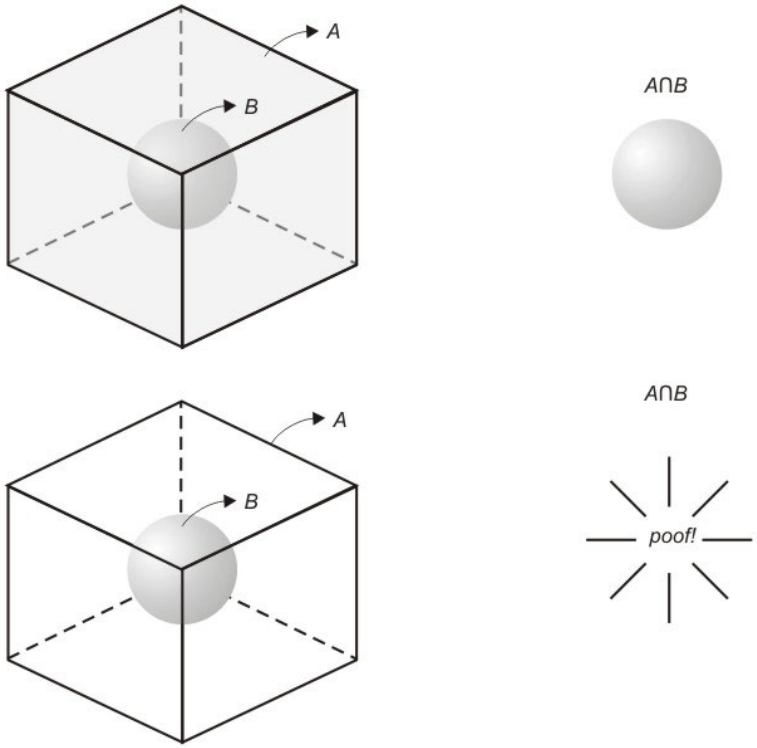


Figura 3.3 – Interseção entre dois objetos considerando um deles como uma região ou somente como uma casca.

3.5. Descrição do algoritmo

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

3.5.1. Parâmetros de entrada

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

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

3.5.2. Tratamento dos parâmetros de entrada

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

1) Armazenamento das entidades topológicas de cada grupo

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

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

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

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

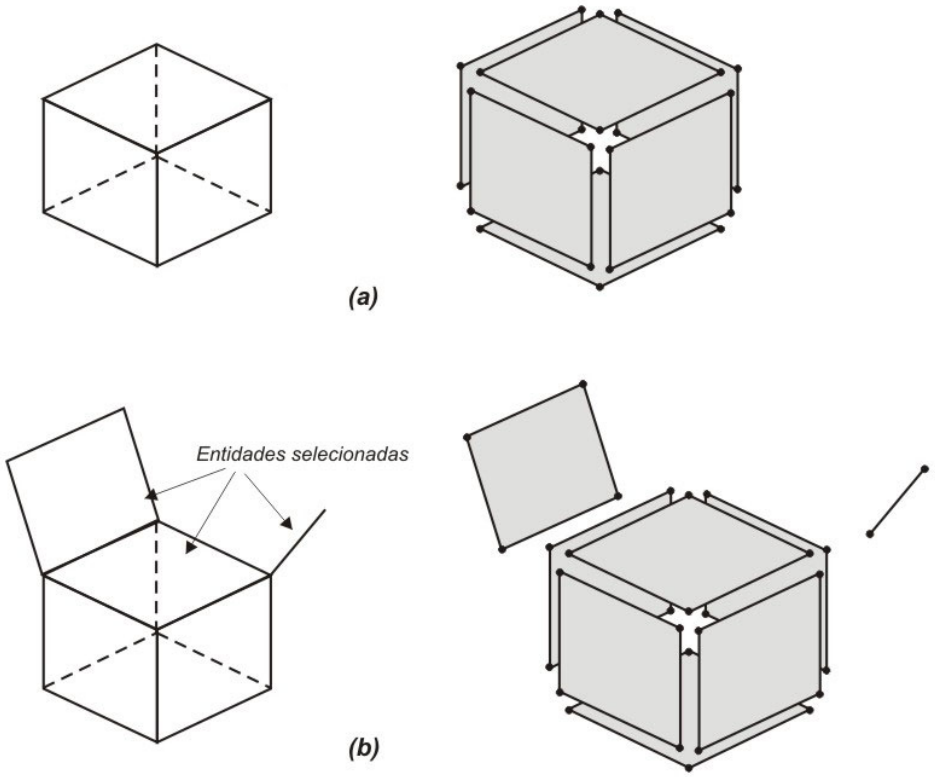


Figura 3.4 – Armazenamento das entidades topológicas explicitamente selecionadas e daquelas obtidas a partir destas por relações de adjacência: a) cubo selecionado; b) cubo, aresta pendente e face pendente selecionados.

2) Determinação das entidades comuns aos grupos

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

3) Classificação dos vértices

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

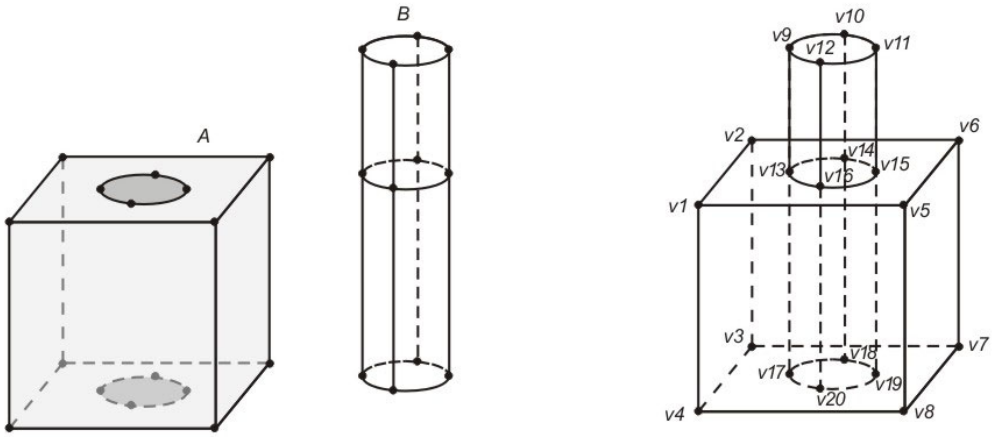
É criado um registro (estrutura de dados) para cada vértice, que contém a referência para o vértice e uma informação sobre a localização do vértice no espaço em relação às entidades do outro grupo. O critério de classificação dos vértices depende de algoritmos geométricos de *detecção de ponto em volume*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

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

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

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

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



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

Figura 3.5 – Exemplo de classificação de vértices.

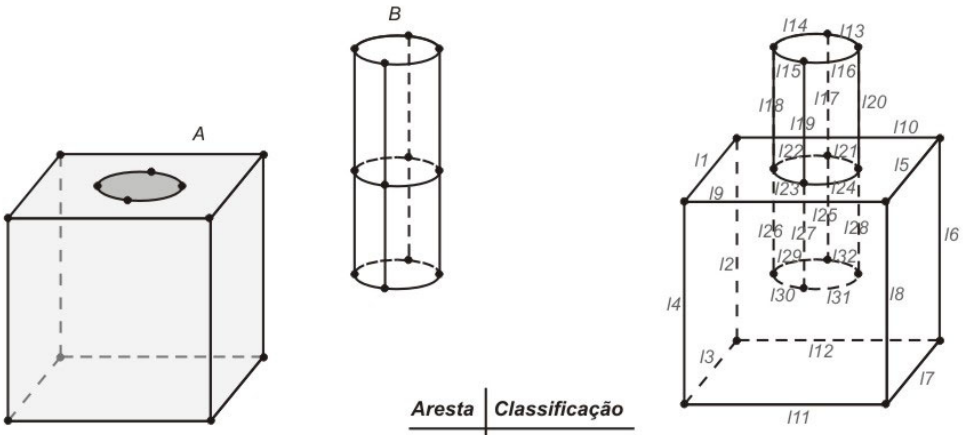
4) Classificação das arestas

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

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

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

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



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

Figura 3.6 – Exemplo de classificação de arestas.

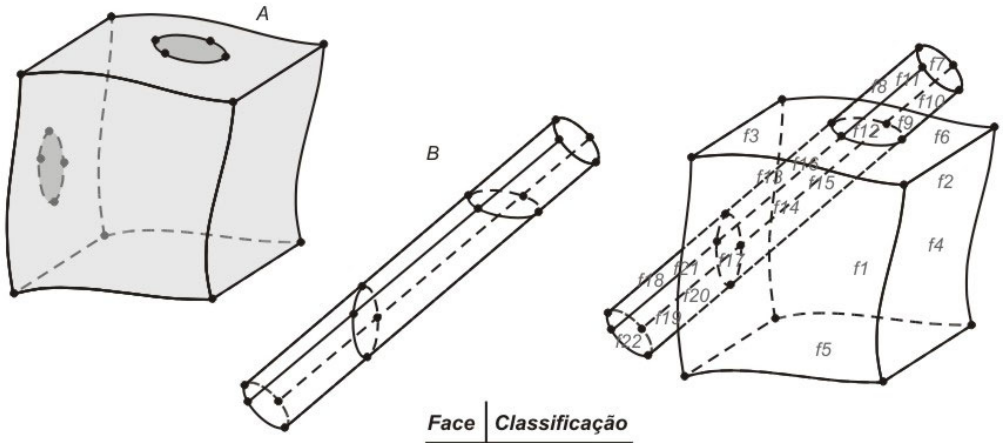
5) Classificação das faces

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

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

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

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



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

Figura 3.7 – Exemplo de classificação de faces.

3.5.3. Aplicação das Operações Booleanas sobre os Grupos

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

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

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

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

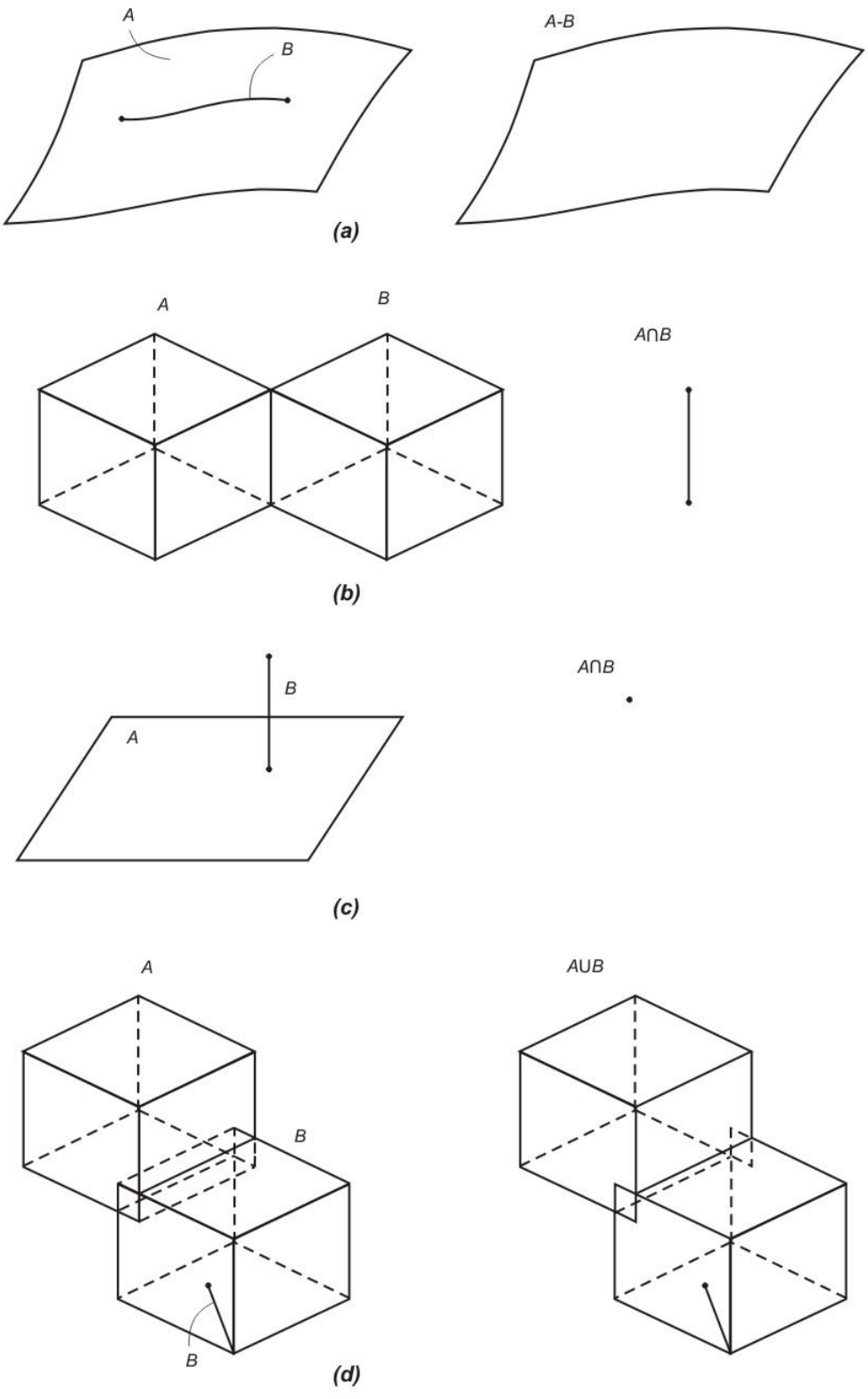


Figura 3.8 – Operações booleanas entre grupos de entidades.

3.5.3.1. Operação Booleana de União

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

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

A união entre dois grupos se divide em três partes:

1) Detecção das faces a serem removidas

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

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

2) Detecção das arestas a serem removidas

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

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

3) Detecção dos vértices a serem removidos

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

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

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

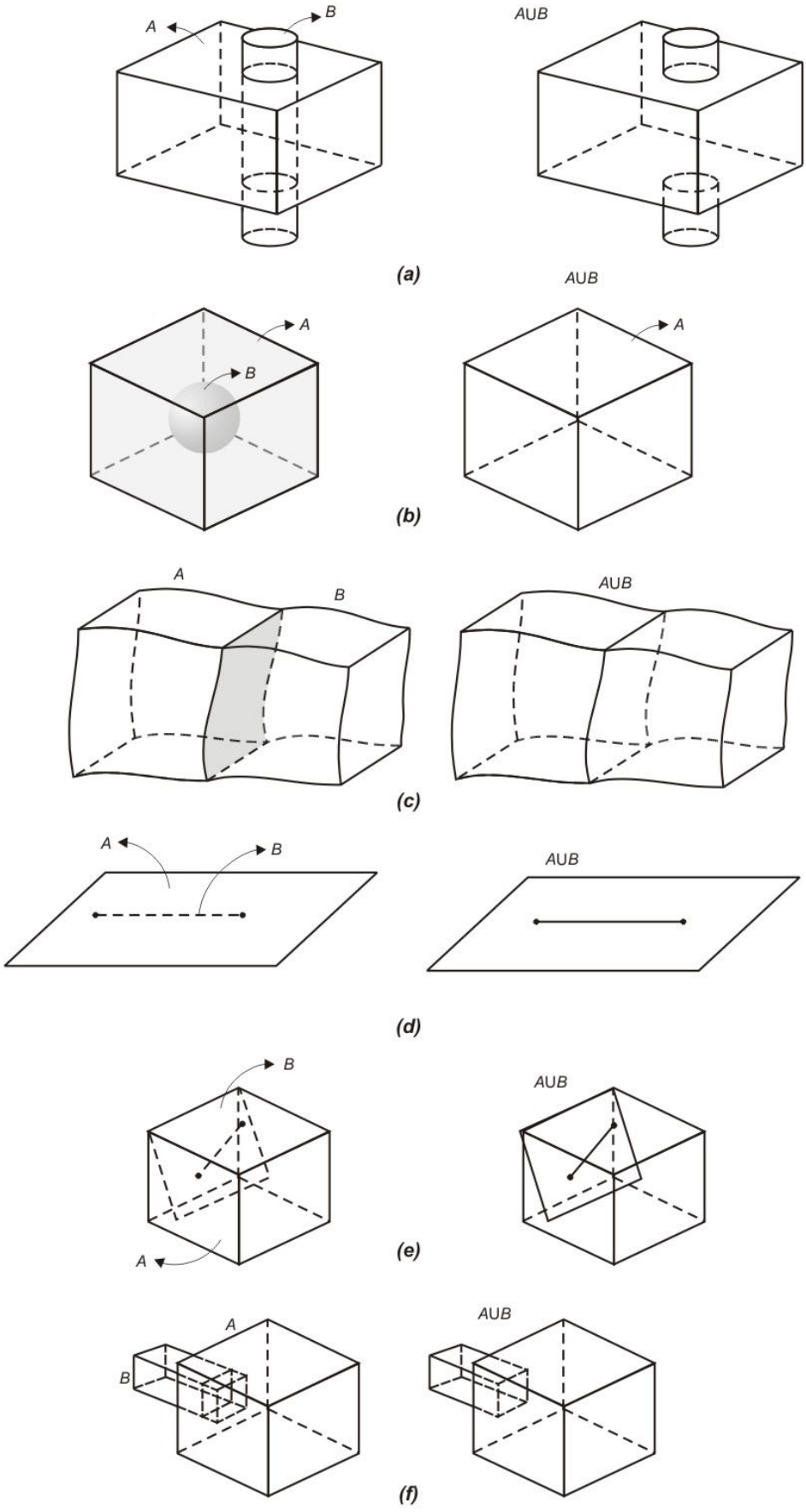


Figura 3.9 – Operação booleana de união.

3.5.3.2. Operação Booleana de Interseção

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

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

3) Detecção das faces a serem removidas

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

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

2) Detecção das arestas a serem removidas

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

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

3) Detecção dos vértices a serem removidos

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

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

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

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

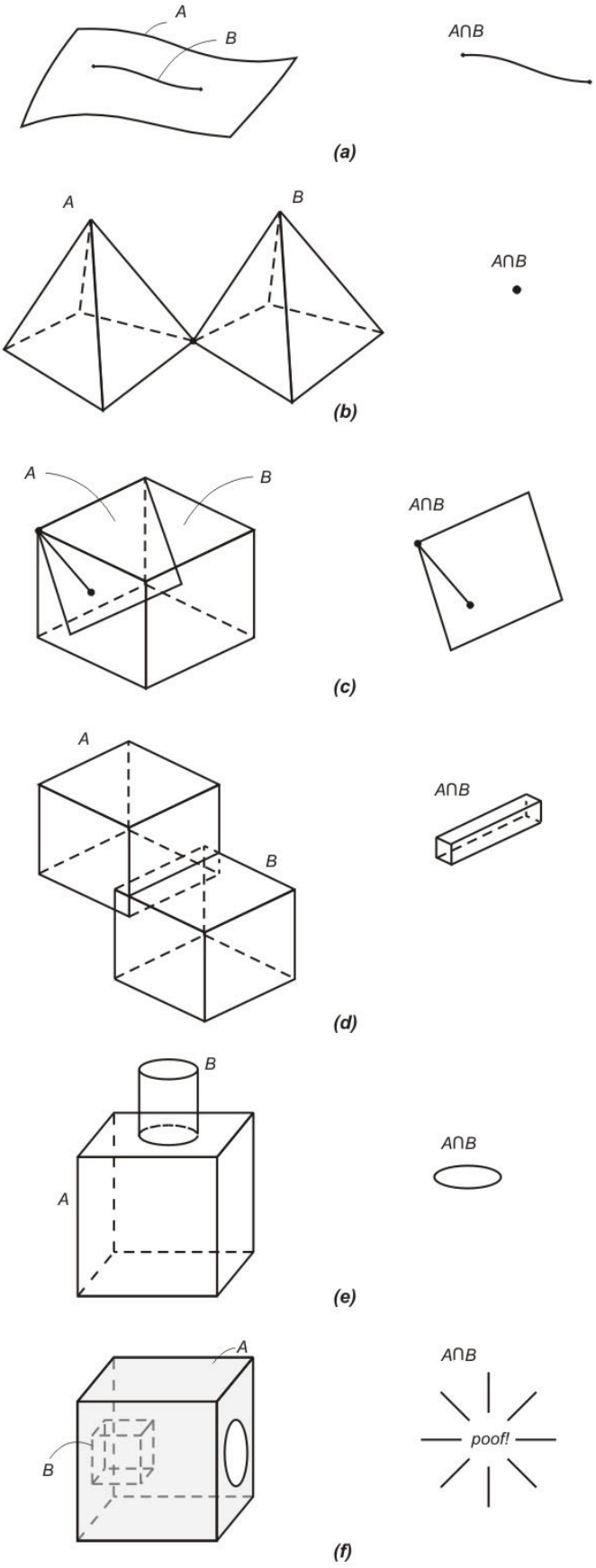


Figura 3.10 – Operação booleana de interseção.

3.5.3.3. Operação Booleana de Diferença

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

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

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

3) Detecção das faces a serem removidas

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

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

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

2) Detecção das arestas a serem removidas

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

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

3) Detecção dos vértices a serem removidos

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

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

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

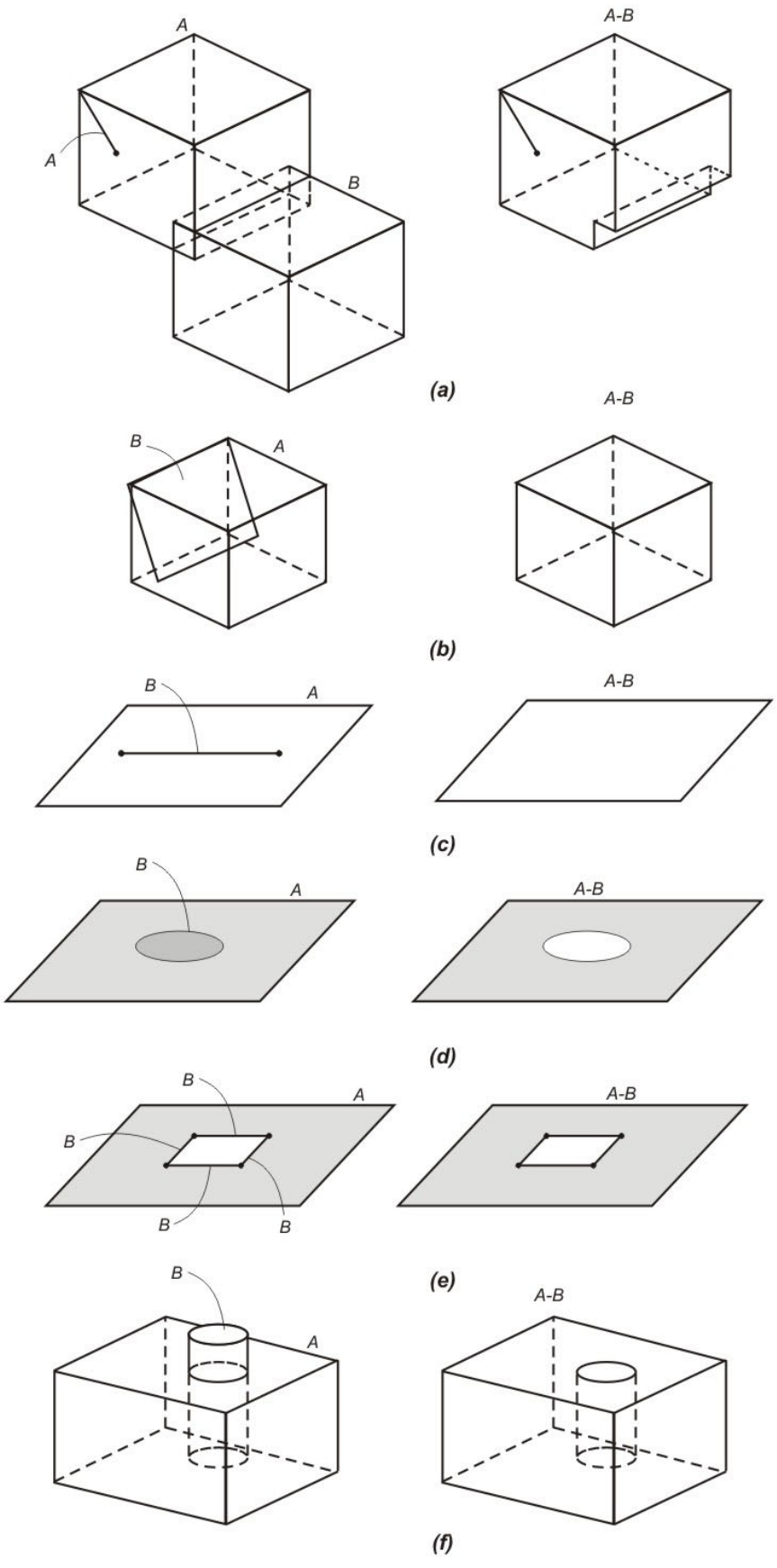


Figura 3.11 – Operação booleana de diferença.

3.5.3.4. Regularização do resultado

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

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

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

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

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

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

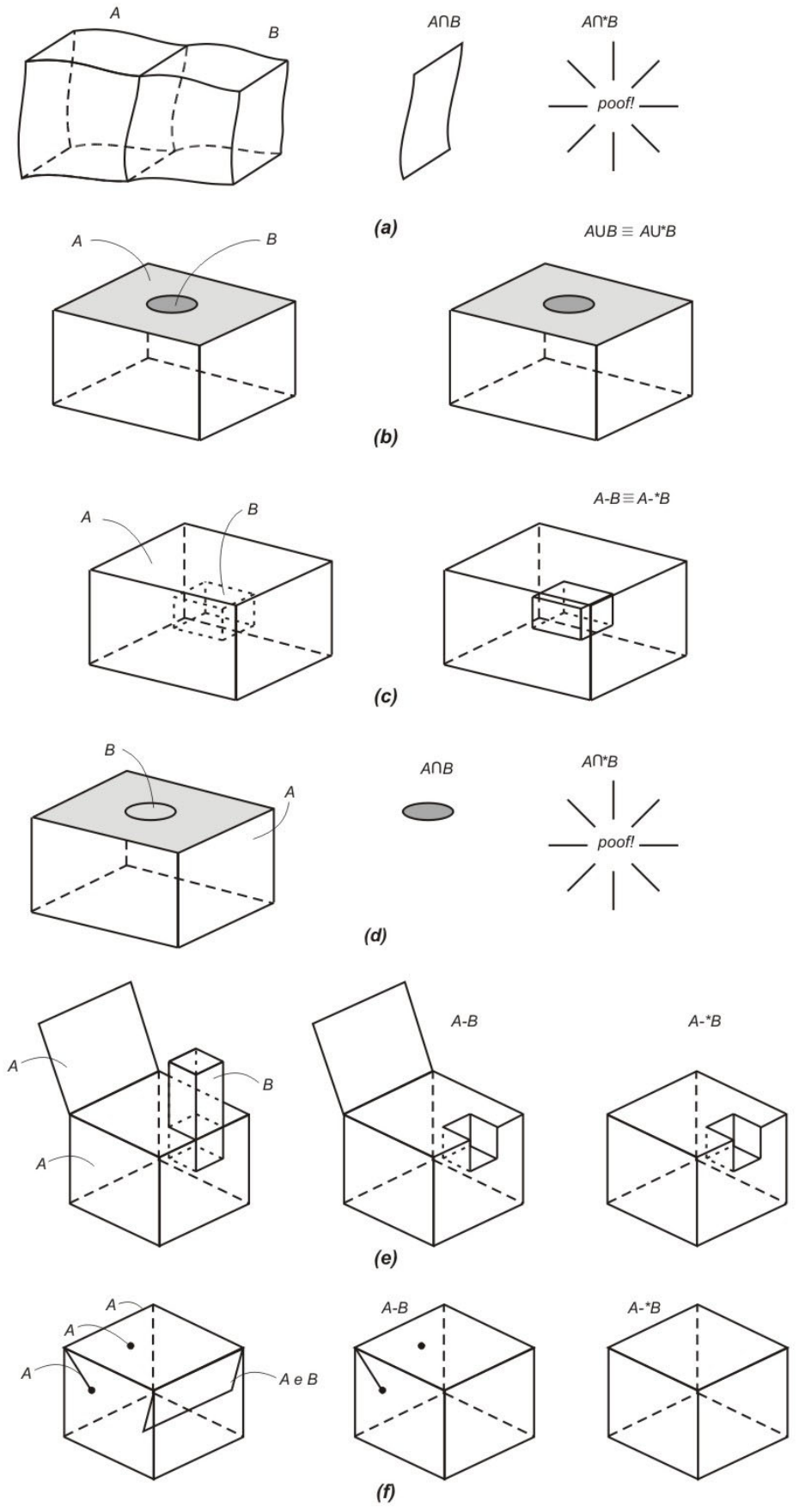


Figura 3.12 – Exemplos de regularização do resultado.

3.6. Considerações sobre o algoritmo

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

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

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

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

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