MARCOS DE SALES GUERRA TSUZUKI

CONTRIBUIÇÃO PARA A REPRESENTAÇÃO DE "FEATURES" PARAMÉTRICAS APLICADA A SISTEMAS CAD/CAE/CAM

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia.

Área de Concentração:
Engenharia Mecânica

Orientador:
Paulo Eigi Miyagi

São Paulo
1994

Aos meus pais, a minha esposa Liping e ao meu filho Akira.

Agradecimentos

Agradeço ao Prof. Dr. Paulo Eigi Miyagi pelo apoio pessoal, pelo suporte profissional e pela orientação; pelos momentos como amigo, conselheiro e orientador. Também agradeço aos amigos e professores da Mecatrônica que propiciaram um ambiente adequado para o progresso deste trabalho.

Agradeço ao CNPq pelo suporte durante o período em que fiquei no Laboratório do Prof. Fumihiko Kimura, Departamento de Engenharia de Precisão da Universidade de Tóquio.

Agradeço a Jose Tiberio Hernandez pelas frutíferas discussões durante a formalização da técnica proposta no trabalho.

Agradeço também aos meus pais que me educaram e sempre me incentivaram em todos os sentidos. Por último, mas tão pouco em menor escala, agradeço a minha esposa e companheira Liping todo o apoio material e espiritual que também muito ajudou para a conclusão deste trabalho.

Conteúdo

Mudanças de Forma do Texto

Agradecimentos, linha 5: Agradeço ao CNPq pelo suporte durante ==> Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro durante

Agradecimentos, linha 7: Tóquio. ==> Tóquio, desenvolvendo parte das pesquisas relacionadas à tese de doutorado.

Lista de símbolos, linha 4: ao nó A para o nó A ==> para o nó A

pg. 3, fig.1.1: Eixo 5.0 ==> Eixo 50.0

pg. 3, fig.1.1: (Canal :in ==> (Rebaixo :in

pg. 4, linha 8: desenvolver ==> procuram desenvolver

pg. 4, linha10: fornecer ==> procuram fornecer

pg. 4, linha 18: A definição mais completa e exata de uma "feature" foi feita por Shah [29]: ==> Shah [29] propôs a seguinte definição para "features":

pg. 5, linha 6: próprias. peças ==> próprias. Peças

pg. 5, linha 20: o comportamento das "features" pode ser estimado com relação a aplicações concretas. ==> deve haver um controle preciso sobre as "features" e os elementos que as compõem para que quando ocorrer a insersão de uma nova "feature" ainda tenhamos controle sobre as "features" já existentes, principalmente nos casos em que ocorre interferência entre "features".

pg. 8, fig.1.4: comprimento a ==> dimensão a

pg. 8, fig.1.4: comprimento b ==> dimensão b

pg. 9, linha 11: projetista possui a sua disposição um conjunto ==> projetista possui, a sua disposição, um conjunto

pg. 10, linha 27: explicitas ==> explícitas

pg. 11, fig.1.5: Comprimento d ==> Dimensão d

pg. 11, linha 7: plícita, no primeiro nível ==> plícita; no primeiro nível,

pg. 11, linha 10: devem existir pois "features" ==> devem existir, pois, "features"

pg. 12, linha 2: várias "features" definidas previamente passaram ==> várias "features", definidas previamente, passaram

pg. 12, linha 8: "features" intencionais acrescentadas ao Modelo do Produto é ==> "features" intencionais, acrescentadas ao Modelo do Produto, é

pg. 12, linha 14: conselhos importantes ==> orientações importantes

pg. 13, linha 8: edição interativa e oferecem um volume ==> edição interativa e um volume

pg. 14, linha 16: volumétricas foram propostas ==> volumétricas foram apresentadas

pg. 14, linha 18: *closing faces* que ao ser ==> *closing faces* que, ao ser

pg. 15, linha 1: superfície definirá uma "feature" ==> superfície, definirá uma "feature"

pg. 16, linha 3: que caso ==> que, caso

pg. 16, linha 4: B-Rep ==> B-Rep,

pg. 16, linha 21: "features" relacionadas com produtos mecânicos podem ==> "features", relacionadas com produtos mecânicos, podem

pg. 17, linha 17: de abstração deve ser ==> de abstração que deve ser

pg. 17, linha 19: tabalho ==> trabalho

pg. 18, linha 5: No capítulo 3 ==> No capítulo 3,

pg. 18, linha 11: No capítulo 4 ==> No capítulo 4,

pg. 18, linha 16: No capítulo 5 ==> No capítulo 5,

pg. 18, linha 19: No capítulo final ==> No capítulo final,
pg. 19, linha 1: Neste capítulo ==> Neste capítulo,
pg. 21, linha 8: pelo usuário. De tal maneira ==> pelo usuário, de tal maneira
pg. 21, linha 14: Nas próximas seções ==> Nas próximas seções,
pg. 21, linha 15: Através destas seções iniciais ==> Através destas seções iniciais,
pg. 22, linha 12: igual a 2 então ==> igual a 2, então
pg. 22, linha 19: Δ então ==> Δ, então
pg. 25, linha 7: coincidente com o seu sentido, ou transladá-la na direção de seu sentido que o ==> coincidente com a sua direção, ou transladá-la mantendo sua direção que o
pg. 26, linha 10: liberdado ==> liberdade
pg. 30, linha 4: exemplode ==> exemplo de
pg. 31, linha 18: Neste trabalho ==> Neste trabalho,
pg. 32, linha 4: Neste passo ==> Neste passo,
pg. 33, linha 7: algorimo ==> algoritmo
pg. 36, linha 8: aproximação pois ==> aproximação, pois
pg. 36, linha 11: Desta maneira é possível compará-las e selecionar ==> Desta maneira comparamo-las e selecionamos
pg. 36, linha 12: É possível observar que a dimensão ==> A dimensão
pg. 37, linha 3: Neste capítulo ==> Neste capítulo,
pg. 37, linha 14: Entretanto, algumas dificuldades não conseguiram ser superadas. Em nosso entender, as dificuldades não foram superadas pois cada ==> Entretanto, as dificuldades envolvidas no controle dos atributos associados aos elementos primitivos, não conseguiram ser superadas; pois, em nosso entender, cada
pg. 38, fig.3.1: Operadores Locais ==> Operadores locais
pg. 39, linha 2: controlar de maneira exata a ==> controlar, de maneira exata, a
pg. 39, linha 13: observe que apesar ==> observe que, apesar
pg. 39, linha 14: originais ==> originais,
pg. 39, linha 17: então surgirá no sólido resultante ==> então surgirá, no sólido resultante
pg. 39, linha 21: Operadores de Euler [18] que ==> Operadores de Euler [18], que
pg. 39, linha 22: estrutura de dados estão ==> estrutura de dados, estão
pg. 39, linha 22: figura 3.2 ==> figura 3.2,
pg. 39, linha 28: Locais ==> locais
pg. 40, fig.3.2: Operações Complexas ==> Operações complexas
pg. 40, linha 5: Complexas ==> complexas
pg. 45, linha 22: visualizacao ==> visualização
pg. 47, linha 7: elementos constituintes associados a um nó foram ==> elementos constituintes, associados a um nó, foram
pg. 47, linha 8: constituintes associados ==> constituintes, associados
pg. 49, linha 10: Encaixes das "Feature" ==> Encaixes das "Features"
pg. 50, linha 2: definidos pois ==> definidos, pois
pg. 52, linha 1: Neste capítulo ==> Neste capítulo,
pg. 53, rodapé: acentos ==> acentos,
pg. 54, linha 8: posteriormente ==> posteriormente,
pg. 55, linha 1: cada tipos ==> cada tipo
pg. 56, linha 8: Satisfazendo um Dimensão ==> Satisfazendo uma Dimensão
pg. 61, linha 1: Neste trabalho ==> Neste trabalho,

pg. 61, linha 10: é possível implementar as ==> implementamos as
pg. 61, linha 11: então será possível suportar ==> então suportaremos
pg. 61, linha 16: *intencionais explícitas associadas ao Modelo do Produto é ==> intencionais explícitas, associadas ao Modelo do Produto, é*
pg. 61, linha 18: B-Rep é possível representar ==> B-Rep representamos
pg. 62, linha 5: Desta maneira ==> Desta maneira,
pg. 62, linha 7: permitindo que vários outros conceitos se encaixem harmoniosamente. ==> permitindo que os conceitos de "feature" intencional, "feature" implícita, "feature" de volume e "feature" de superfície se encaixem harmoniosamente.
pg. 64, linha 11: de projetos ==> de projetos,
pg. 67, linha 4: ocorre múltiplas ==> ocorrem múltiplas
pg. 67, linha 7: que métodos ==> que, métodos
pg. 70, linha 1: algoritmo que ==> algoritmo, que
pg. 70, linha 2: relativas será ==> relativas, será
pg. 76, linha 1: capítulo 5 ==> capítulo 5,
pg. 78, linha 6: capíulo ==> capítulo
pg. 80, linha 1: capítulo 5 ==> capítulo 5,
pg. 80, linha 3: protótipo ==> protótipo,

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Resumo

Este trabalho foi baseado em duas crenças a respeito da próxima geração de sistemas de CAD/CAE/CAM: primeiro, será necessário armazenar explicitamente "features" e dimensões no Modelo do Produto; segundo, é necessário suportar "features" definidas pelo usuário. O Modelo do Produto é aqui introduzido como uma estrutura hierárquica onde é possível definir dois tipos de dimensões: *dimensões locais* e *dimensões relativas*. As dimensões relativas são restrições que associam dois nós diferentes da estrutura hierárquica. Os parâmetros de encaixe definidos neste trabalho permitem que "features" sejam facilmente acrescentadas ao Modelo do Produto. É proposta também a adição da representação explícita de novos elementos geométricos à representação B-Rep ("Boundary Representation"), podendo inclusive existir elementos geométricos que não estão sobre o contorno do Modelo do Produto. Esta característica permitirá representar planos e eixos de simetria. Concluíremos que as dimensões relativas e a adição de novos elementos geométricos à representação B-Rep são efetivos para a representação de "features".

Abstract

This work is based on two believes about the next generation of CAD/CAE/CAM: first, it is necessary to explicitly store features and dimensions in a Product Model; second, it is necessary to support user defined features. The Product Model is introduced as a hierarchical structure where it is possible to define two kinds of dimensions: *local dimensions* and *relative dimensions*. Relative dimensions are constraints that associate two different nodes in the hierarchical structure. The attaching parameters defined in this work, allow to easily fix features to the Product Model. It is also proposed to add the explicity representation of new geometrical elements to the B-Rep representation. It is possible that some of the geometrical elements are not in the boundary of the Product Model. This characteristic will allow to represent planes and axis of symmetry. We will conclude that the relative dimensions and the addition of explicitly represented new geometrical elements to the B-Rep (Boundary Representation) are very useful for feature representation.

Capítulo 1

Introdução

A área de Projeto, Engenharia e Manufatura Auxiliados por Computador (CAD/CAE/CAM), sofreu um grande avanço na última década; entretanto, o seu uso ainda permanece muito limitado. Uma razão para isto é que os sistemas atuais de CAD/CAE/CAM são de difícil manipulação e gasta-se muito tempo para aprender a utilizá-lo eficientemente. Isto é conseqüência de que os sistemas atuais não suportam uma interação amigável com o usuário, por meio de uma linguagem de engenharia afim à aplicação [35]. O projetista necessita expressar o seu projeto através de uma linguagem específica suportada pelo sistema de CAD/CAE/CAM, ao invés de utilizar uma linguagem adequada à aplicação. Para definir um sólido razoavelmente complexo é necessário utilizar funções que o manipulam segundo um grande nível de detalhe, portanto, o projetista necessita "traduzir" o seu modelo mental expressado em termos específicos da aplicação para termos de representação utilizados pelo sistema de CAD/CAE/CAM. Seria preferível que o projetista pudesse definir o **Modelo do Produto** [1] utilizando uma terminologia familiar a sua aplicação (vide figura 1.1) [2].

[1] Neste trabalho estaremos considerando que **Modelo do Produto** corresponde a uma representação computacional do **Produto** que possui todas as informações tecnológicas necessárias para projetar e manufaturar o produto [14, 21].

[2] As figuras não possuem acentos pois foi utilizado um editor gráfico que não o permite.

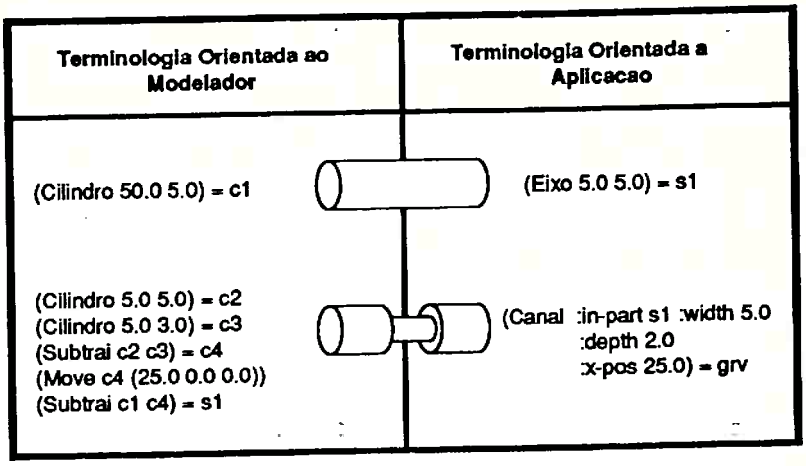


Figura 1.1: Exemplo de modelagem de um eixo com um canal utilizando uma terminologia orientada ao modelador e uma terminologia orientada à aplicação.

O atual estágio de pesquisas na área de CAD/CAE/CAM pode ser identificado como um estado de transição do tradicional *projeto e manufatura baseados na geometria* para um *projeto e manufatura baseados em "features"* [3] que denominaremos por *Modelador DbF* [4].

Grande parte dos trabalhos atuais é concentrado na solução de problemas fundamentais como: O que são "features"? Como elas são classificadas? Como elas podem ser representadas? Existe um conjunto canônico de "features" ou elas são dependentes da aplicação? Como dados de "features" podem ser intercambiados entre Modeladores DbF diferentes?

[3] Tradução do termo em inglês "Feature Based Modeling" que também é denominado por alguns autores por "Design by Feature". É importante observar que preferimos evitar a tradução do termo "feature", pois consideramos que a tradução por "características" como tem sido realizada por alguns autores não reflete de modo satisfatório o sentido atribuído ao termo original. O termo "form feature" tem sido utilizado para se referir a "features" que possuem alguma forma ou geometria associada. O termo "feature" tem sido utilizado para representar um elemento que pode ou não ter uma geometria associada. Entretanto, como neste trabalho estamos interessados apenas em "form features", por facilidade utilizaremos o termo "feature" para se referir a "features" que possuem geometria associada.

[4] Iniciais do termo "Design by Feature".

Apesar desta tecnologia estar apenas surgindo e existirem várias diferenças filosóficas entre os conceitos no qual os Modeladores DbF estão baseados, é possível encontrar alguns pontos comuns nas diversas propostas [23]:

- procuram estabelecer uma definição genérica de "feature";
- procuram desenvolver uma especificação uniforme de Modeladores DbF para fornecer aos usuários a liberdade de escolher as ferramentas e métodos necessários para suas tarefas específicas;
- desenvolver um mecanismo de mapeamento de "features" a partir do ponto de vista do projetista para o ponto de vista da aplicação;
- fornecer padronizações para intercâmbio de dados de "features".

1.1 Definição de "Feature"

A existência de uma definição comum de "feature" é muito importante para a comunidade que atua na área de projeto com "features" devido ao papel primário que ela executa nas especificações funcionais dos Modeladores DbF.

Apesar de várias tentativas diferentes de definir "feature" já terem sido publicadas na literatura (vários exemplos de tentativas podem ser encontrados na referência [34]), a maioria dos autores definem que "features" representam uma interpretação técnica da geometria do produto, peça ou montagem. A definição mais completa e exata de uma "feature" foi feita por Shah [29]: "Uma "feature" é definida como uma entidade que satisfaz as seguintes condições: ser um constituinte físico de uma peça, ser mapeável para uma forma genérica, possuir um significado técnico e possuir propriedades previsíveis". Portanto, existem quatro pontos principais nesta definição:

- "features" são constituintes físicas de peças, onde peças são elementos que possuem toda a informação necessária para analisá-los e interpretá-los. Neste

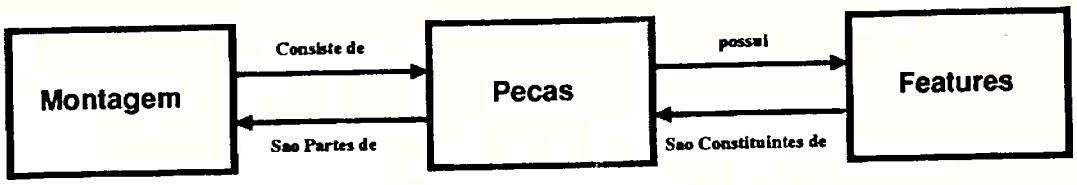


Figura 1.2: Relacionamento entre "Features", Peças e Montagens.

espectro, existe uma diferença essencial entre peças e "features": peças possuem sua existência associada a si próprias, enquanto "features" podem existir apenas segundo uma aplicação específica; em outras palavras, a "feature" não possui um significado isolado sem a peça associada, pois a "feature" corresponde a uma possível interpretação da forma da peça. Montagens também possuem sua existência associada a si próprias. peças são agrupadas para definirem uma montagem. O relacionamento entre montagens, peças e "features" deve ser conforme ilustramos na figura 1.2.

- "features" podem ser desenvolvidas geometricamente utilizando definições de forma genérica que devem ser representadas por meio de técnicas de modelagem convencionais;

- o significado técnico das "features" está relacionado às funções a que a "feature" se destina, às regras tecnológicas que devem ser satisfeitas ou como ela deve ser produzida. Por exemplo, um canal em um eixo pode ser interpretado como uma superfície de apoio; entretanto, para a área de planejamento de processos, um canal em um eixo pode ser interpretado como o ato de realizar o sangramento em um eixo;

- o comportamento das "features" pode ser estimado com relação a aplicações concretas.

Conforme discutiremos no capítulo 4, a nossa proposta para representar "features" pode ser estendida para representar uniformemente tanto o Modelo do Produto, como a

Montagem ou Peças.

1.2 Classes de "Features"

A definição de "feature" apresentada anteriormente afirma que uma *"feature" deve ser mapeável para uma forma genérica.* O fato da forma ser genérica implica que "features" podem ser caracterizadas como pertencendo a uma certa classe de acordo com as suas propriedades de forma. Também deve ser possível classificar "features" de acordo com as suas propriedades de significado técnico. O problema de como classificar "features" está relacionado ao domínio da aplicação e foge ao escopo deste trabalho.

Consideramos aqui que "features" podem ser classificadas como membros de classes de acordo com critérios variados de seleção, e dependendo da aplicação, critérios distintos devem ser utilizados. Nós não desejamos impor nenhuma restrição quanto às possíveis classes de "features". A solução deve permitir que o usuário defina as suas classes de "features" específicas necessárias para sua aplicação. Para este fim, é desejável que o projetista disponha de uma linguagem através do qual ele poderá definir a forma, o significado tecnológico e os métodos aplicáveis a uma classe de "features". Entretanto, a criação de tal linguagem envolve uma quantidade enorme de definições e não é tratada exaustivamente neste trabalho.

Shah [28] implementou um "Feature Modeling Shell" que permite ao usuário definir as suas próprias classes de "features". De acordo com os exemplos ilustrados por Shah, classes de "features" definidas pelo usuário podem ser associadas apenas a sólidos primitivos ("sweep" rotacional, "sweep" linear e lâmina); entretanto, não está claro como classes de "features" complexas podem ser definidas. Dong e Wozny [6] fizeram algumas considerações sobre o uso das representações B-Rep [5] e CSG [6] para o desenvolvimento

[5] Iniciais do termo "Boundary Representation". É uma técnica de representação de Modelagem de Sólidos.

[6] Iniciais do termo "Constructive Solid Geometry". É uma técnica de representação de Modelagem de Sólidos.

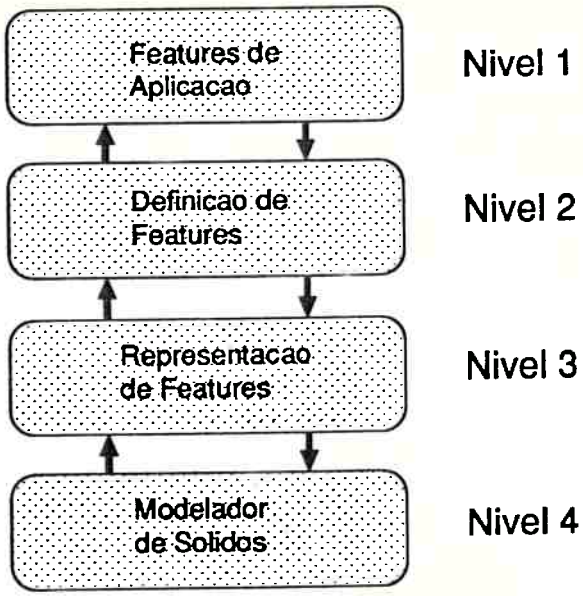


Figura 1.3: Os vários níveis de modelagem com "features".

de um Modelador DbF que fosse capaz de suportar classes de "features" definidas pelo usuário.

1.3 Níveis de Abstração Associados à "Feature"

Ovtcharova [23] propôs uma arquitetura para Modeladores DbF em que quatro níveis de abstração estão presentes: "features" de aplicação, definição de "features", representação de "features" e modelo geométrico (vide figura 1.3). Geralmente, o nível de aplicação inclui a especificação de "features" de aplicação que possuem uma semântica associada ao aplicativo específico. Neste nível possuímos dados não geométricos e dados geométricos (por exemplo, alguma dimensão relevante). No segundo nível de modelagem, as "features" devem ser definidas segundo propriedades genéricas da forma. Neste nível, os tipos de "features" e seus principais parâmetros são especificados. O terceiro nível consiste na definição da representação da "feature". No quarto nível o modelo geométrico do produto está representado com todos os detalhes desenvolvidos.

De acordo com este esquema, a nossa proposta pode ser considerada como uma

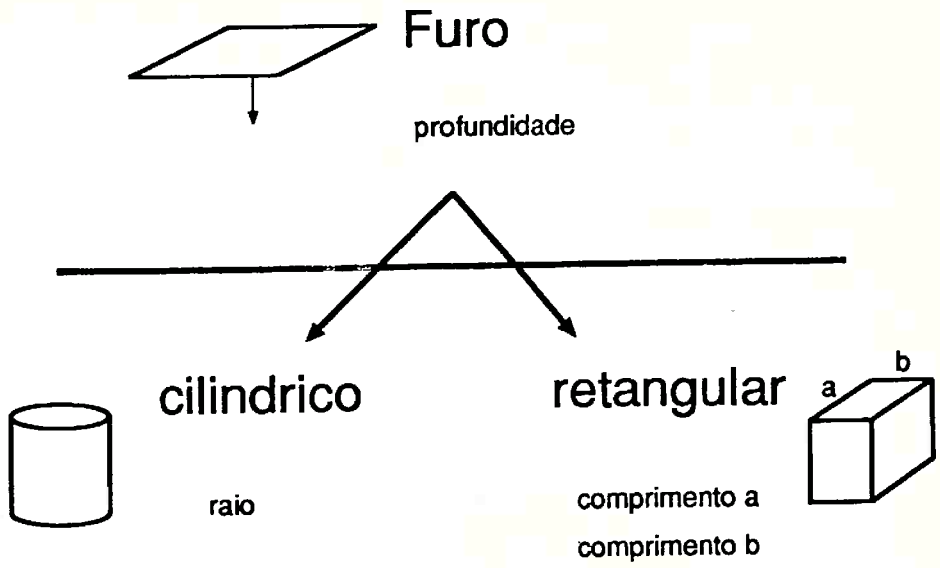


Figura 1.4: Taxonomia de "Features".

implementação do terceiro nível. Esta arquitetura possui a grande vantagem de suportar a especificação incompleta de "features". Por exemplo, a "feature" de tipo furo pode ser definida e sua profundidade ser especificada e não será necessária nenhuma outra informação adicional (vide figura 1.4). Em um próximo passo, o projetista poderá escolher que tipo de furo ele deseja e, desta vez, parâmetros específicos serão requisitados. Neste exemplo, a taxonomia de "feature" está definida no segundo nível, e a representação da classe de "features" está definida no terceiro nível. Caso o projetista decida trocar o tipo de "feature" a ser utilizado deverá haver um mecanismo para intercambiar no projeto as "features" envolvidas de maneira automática.

No terceiro nível, de acordo com a nossa opinião, devem ocorrer várias considerações sobre como a semântica de uma classe de "features" deve ser representada. O principal aspecto é relativo a como a "feature" deverá ser parametrizada. Após especificar a definição da "feature", isto é, após definir a sua semântica, será necessário implementar estas considerações. Neste caso, o projetista deverá analisar as considerações e representar a "feature" com as ferramentas disponíveis no segundo nível. O usuário da aplicação não necessita saber como a "feature" foi implementada; entretanto, ele deve saber quais são

as considerações que levaram à definição da "feature", isto é, ele deve saber a semântica das respectivas "features" que está utilizando.

Antes de continuarmos com o trabalho, convém realizarmos uma revisão bibliográfica sobre "features" para familiarizarmos com vários termos que são utilizados neste trabalho.

1.4 Revisão Bibliográfica sobre "Features"

Até o momento, grande parte das pesquisas relacionadas com "features" foram realizadas na área de planejamento de processos e muito pouco foi apresentado com relação à área de *Modeladores DbF*. Entretanto, muitos vendedores de sistemas CAD/CAE/CAM dizem que já possuem ou que já são capazes de desenvolver um sistema que auxilie o projetista oferecendo "features" como opções do menu. Geralmente, isto significa que o projetista possui a sua disposição um conjunto fixo e limitado de "features" que podem ser utilizadas para criar um modelo paramétrico.

Baseando-se no conceito de que uma "feature" pode ser associada a um conjunto de operações de usinagem, na área de planejamento de processos, vários métodos de reconhecimento de "features" foram pesquisados [11, 2]. A maioria destes métodos procuram por padrões de associações de entidades geométricas presentes no Modelo do Produto Desenvolvido. Como usinagem significa que algum material deverá ser removido, a correspondência entre uma "feature" e uma operação de usinagem nem sempre pode ser definida de maneira exata. Por exemplo, uma saliência não se refere a nenhum material que deva ser removido. Ao contrário, é o material ao redor da saliência que deve ser removido e este fato é que deve ser associado a um conjunto de operações de usinagem.

Vários trabalhos na área de *projeto com "features"* também se orientaram para as áreas de planejamento de processos e usinagem, pois seus autores imaginavam que o projeto poderia ser realizado utilizando-se exclusivamente de "features" voltadas à usinagem [1]. Entretanto, raramente as "features" utilizadas pelo projetista são as mesmas "features" utilizadas em outros processos, por exemplo, manufatura ou planejamento de

processos. Um exemplo deste fato foi citado por Wingard [35], a mesma peça foi fornecida a quatro engenheiros que atuavam nas áreas de planejamento de usinagem, planejamento para inspeção, análise por elementos finitos e projeto. Todos os engenheiros determinaram conjuntos diferentes de "features" e os conjuntos não apresentavam muitos pontos em comum. A conclusão lógica é que cada etapa do desenvolvimento de um produto deve possuir as suas "features" próprias. Cada *modelo baseado em "features"* representa uma interpretação específica do produto associada a algum processo. Na próxima seção discutiremos inicialmente as várias propostas para se criar "features" e apresentaremos alguns conceitos básicos importantes.

Modeladores DbF

Para entendermos os vários trabalhos que foram realizados nesta área será necessário discutirmos alguns conceitos básicos importantes. Estes conceitos compreendem "features" intencionais, "features" implícitas, "features" explícitas, "features" de volume e "features" de superfície [7].

Uma **"feature" intencional** é uma abstração para acessar grupos de entidades geométricas e para associar a estes grupos um tipo e conseqüentemente algumas propriedades definidas para todas as "features" deste tipo em particular [27]. Por exemplo, uma "feature" do tipo rasgo pode ser acrescentada a uma peça, isto significa que o projetista deseja que um rasgo esteja presente na peça que está sendo projetada, isto é, um espaço vazio delimitado por três faces - uma face de fundo e duas faces laterais. A "feature" intencional deve possuir referências para as entidades geométricas que a compõem. No caso do exemplo, as entidades geométricas são as três faces que delimitam o espaço vazio. Entretanto, devido a manipulações sobre o modelo, as entidades geométricas associadas à "feature" podem ser modificadas ou até mesmo eliminadas do Modelo do Produto [8].

[7] Respectivamente, tradução dos termos: "Intentional Features", "Implicit Features", "Explicit Features", "Volumetric Features" e "Surface Features".

[8] Da mesma maneira que existem "features" intencionais implícitas e explicitas, também temos o Mo-

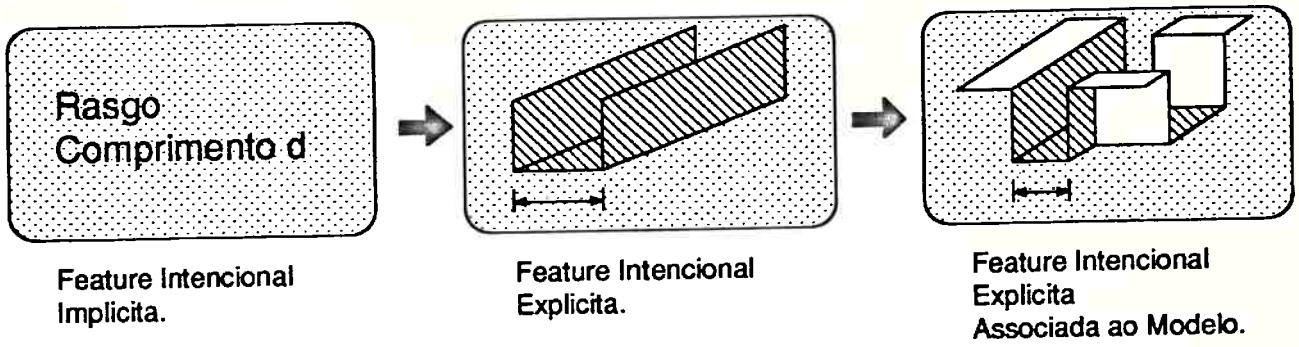


Figura 1.5: "Feature" intencional e sua representação.

O que restar das entidades geométricas e do espaço vazio pode constituir algo que não apresenta mais nenhuma propriedade relacionada com o rasgo.

"Features" intencionais podem ser representadas explicitamente ou implicitamente. Geralmente, a representação implícita está associada a uma linguagem que será utilizada pelo usuário e a representação explícita está associada ao método de representação adotado para o sistema [27]. É possível considerar dois níveis de representação explícita, no primeiro nível as entidades geométricas que definem a "feature" implícita estão presentes e no segundo nível estão presentes as entidades geométricas que resultaram da associação da "feature" ao Modelo do Produto. A figura 1.5 ilustra estes dois níveis. Estes dois níveis devem existir pois "features" intencionais devem ser consistentes com a real geometria da peça, isto pode ser realizado relacionando-se as entidades geométricas entre os dois níveis de representação explícita. É possível que em algum particular estágio do processo de projeto algumas destas referências não correspondam a nenhuma entidade geométrica no contorno do Modelo do Produto. Por exemplo, uma "feature" intencional do tipo furo cilíndrico pode estar associada a entidades geométricas que foram removidas do Modelo do Produto e portanto esta "feature", neste momento, não corresponde a nenhum furo válido. Esta "feature" intencional está em um estado inválido, entretanto

delo do Produto implícito e Modelo do Produto desenvolvido. O Modelo do Produto implícito representa a associação das "features" ao Modelo do Produto. O Modelo do Produto desenvolvido contém todas as "features" intencionais explícitas associadas ao Modelo do Produto.

não podemos considerar esta possibilidade como indesejável, pois o projetista pode ter criado (temporariamente ou não) uma situação em que várias "features" definidas previamente passaram para um estado inválido. A validade global poderá ser restabelecida ajustando-se algum parâmetro ou reposicionando algumas das "features".

A validade de uma "feature" é algo muito subjetivo e de fato depende do papel associado à respectiva aplicação [28, 27]. Critérios de validade para "features" podem ser expressos em termos de referências a entidades geométricas e a sua topologia associada. Para verificar a validade das "features" intencionais acrescentadas ao Modelo do Produto é necessário que elas sejam desenvolvidas, isto é, que a representação explícita da "feature" associada ao Modelo do Produto seja determinada. Conseqüentemente, a identificação de "features" em estados inválidos é algo muito importante para realizar correções no projeto. Usualmente, a presença de uma "feature" intencional, implica no desejo do projetista em acrescentar uma certa funcionalidade a alguma párte do Modelo do Produto. Portanto, "features" intencionais fornecem conselhos importantes quando alterações no modelo forem necessárias e auxiliam os usuários envolvidos em outros processos para que entendam as considerações realizadas pelo projetista original.

"Feature" implícita é uma "feature" não desenvolvida que possui todas as informações suficientes para que não ocorram interpretações duvidosas durante o seu desenvolvimento. Esta é uma forma compacta de representação [28]. Um exemplo típico é um furo cilíndrico sendo representado pela sua linha de simetria, pelo seu raio e pela sua altura. A representação implícita pode possuir muitas outras aplicações, por exemplo, uma peça pode exibir um grande número de "features", neste caso o produto pode ficar incompreensível devido ao volume de detalhes presente na tela. Neste caso seria conveniente retirar algumas "features" da tela, ou simplesmente substituí-las por algo que apenas indique a sua presença (por exemplo, o paralelepípedo envoltório da "feature"). Entretanto, isto exigirá um nível maior de sofisticação, pois possíveis inconsistências podem surgir devido à múltipla representação das "features". Todas as "features" devem ser desenvolvidas apenas quando houver necessidade.

Outra importante distinção foi indicada por Pratt [24], que diferencia **"features" de superfície** que são coleções de faces de uma peça como as paredes laterais e o fundo de um rasgo, de **"features" volumétricas** que são *conjuntos de pontos* [9] de uma peça ou o seu complemento como no caso de saliências e furos. Raramente estes dois tipos de "features" são suportados simultaneamente pelo mesmo sistema. Pratt fornece uma discussão detalhada sobre as motivações históricas, os méritos atuais e as desvantagens de ambas as representações e conclui que as **"features" de superfície** oferecem uma maior flexibilidade para edição interativa e oferecem um volume maior de informações que facilitam a análise do Modelo do Produto e o desenvolvimento de programas aplicativos. Entretanto, a manutenção da consistência dos elementos que compõem a "feature" de superfície é mais complexa quando comparada com a manutenção das "features" volumétricas [34].

"Features" de volume são freqüentemente criadas de maneira incremental, inicialmente definindo o corpo global da peça e, posteriormente, acrescentando ou modificando detalhes pela criação ou edição de "features". A criação de uma "feature" é necessariamente acompanhada pela modificação do volume ocupado pela peça e na maioria dos casos corresponde ao ato de acrescentar ou remover material. Esta transformação pode ser expressa como uma união ou diferença entre a peça e o volume da "feature". Como o volume da "feature" pode ser interpretado como uma forma paramétrica complexa, esta proposta deverá ser muito eficiente no caso de modeladores híbridos onde a representação B-Rep do Modelo do Produto é o resultado do desenvolvimento da árvore CSG [13].

As **"features" de volume** resultantes de operações que modificaram a forma, nem sempre fornecem um conjunto suficiente de ferramentas de abstração para interagir com o Modelo do Produto. Por exemplo, uma "feature" do tipo rasgo que é de interesse para aplicações de manufatura, pode ser criada como o efeito colateral de duas saliências paralelas (figura 1.6). A "feature" de tipo rasgo pode ser uma abstração conveniente para expressar modificações de engenharia (por exemplo, modificação da largura), portanto,

[9] Tradução de "Set of Points".

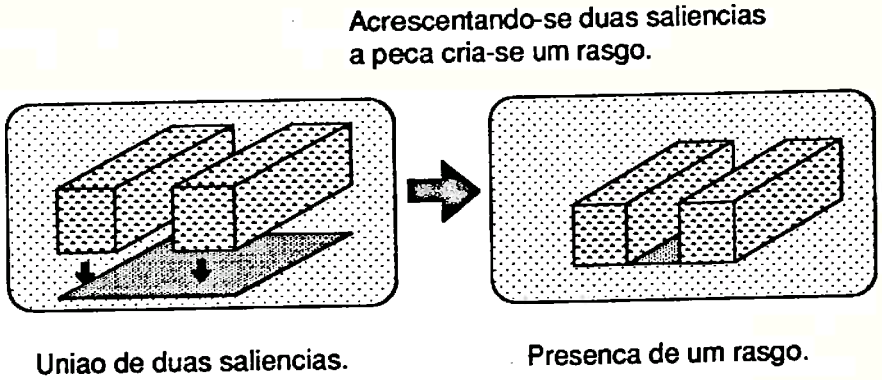


Figura 1.6: Identificação da "feature" rasgo *a posteriori*.

esta possível ferramenta para converter ou identificar "features" também deve estar acessível ao usuário. O uso destas "features" identificadas *a posteriori*, requer a associação de "features" intencionais com um subconjunto da geometria já existente. Geralmente, esta associação é realizada interativamente selecionando uma coleção de faces do Modelo do Produto e tratando-o como uma "feature" de superfície [27].

1.4.1 Métodos para a Representação de "Features"

As pesquisas na área de Modeladores DbF ainda estão em seu estágio inicial. Na seção anterior vários conceitos importantes foram apresentados. Entretanto, ainda não existe um consenso sobre como eles devem ser utilizados no desenvolvimento de Modeladores DbF. Qual nível de abstração deve ser utilizado (implícito ou explícito)? Qual dimensionalidade de "feature" deve ser adotada (volumétrica ou de superfície)? Como as "features" devem ser representadas (CSG, B-Rep ou Híbrido CSG/B-Rep)?

Vários autores consideram "features" como possuindo três dimensões [4, 16, 22, 28, 33]. Em alguns trabalhos, "features" foram consideradas como possuindo duas dimensões [3, 8, 11]. Várias propostas para converter "features" de superfície em "features" volumétricas foram propostas [20]; entretanto, a correspondência entre "features" de superfície e "features" de volume não é única. Alguns autores propõem a manipulação de um conjunto de faces chamado por *closing faces* que ao ser acrescentado à "feature" de

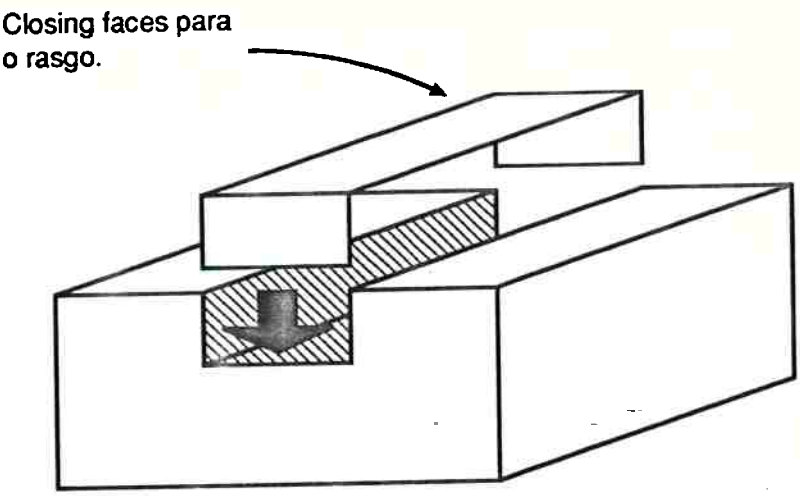


Figura 1.7: Exemplo de *closing faces.*

superfície definirá uma "feature" de volume única. A figura 1.7 ilustra um exemplo de *closing faces.* Infelizmente, as *closing faces* não podem ser obtidas pela simples extensão das faces adjacentes. Atualmente, métodos e heurísticas para criar automaticamente as *closing faces* são limitados a situações simples, e freqüentemente é necessária a intervenção do usuário para que soluções aceitáveis sejam criadas. Gossard [8] utilizou um método em que "features" volumétricas são derivadas a partir de "features" de superfície. Gomes [7] propôs uma representação pseudo CSG/B-Rep para permitir que "features" volumétricas e de superfície pudessem ser representadas simultaneamente.

Neste contexto, vários tipos diferentes de representação de Modelagem de Sólidos foram propostos e um bom resumo sobre o assunto foi escrito por Requicha e Voelcker [25]. Atualmente, apenas as representações B-Rep e CSG são consideradas viáveis para representar "features". Os pesquisadores favoráveis à representação CSG citam a facilidade de edição pelo usuário como principal motivação. Entretanto, algumas aplicações como Dimensionamento e Tolerância exigem que elementos primitivos como faces, arestas e vértices estejam explicitamente acessíveis. Ostrowski [22] desenvolveu um Modelador DbF baseado apenas em Operações Booleanas. Nesta proposta "features" volumétricas

são definidas por meio da árvore CSG e primitivos padrões [10]. Outros tipos de "features" como arestas e vértices são definidos por meio de uma extensão dos Operadores Booleanos. Apesar do sistema ter sido desenvolvido com sucesso, Ostrowski concluiu que caso fosse utilizada a representação B-Rep o esforço empreendido seria bem menor e haveria maior flexibilidade de projeto.

A representação B-Rep é a mais aceita pelos pesquisadores para o desenvolvimento de Modeladores DbF. A grande vantagem desta representação é a possibilidade de manipularmos simultaneamente "features" volumétricas e "features" de superfície. Entretanto são raros os sistemas que suportam as duas dimensionalidades. Caso "features" volumétricas sejam utilizadas, interações entre "features" serão manipuladas com maior facilidade e a função que remove uma "feature" será implementada de forma mais simples [34].

1.5 "Feature" Paramétrica

Conforme citado na seção 1.1, "features" possuem *formas genéricas*. Neste caso, o termo *genéricas* pode ser interpretado como uma condição para que "features" possam ser caracterizadas por um conjunto de parâmetros que definam uma família de "features". Por exemplo, a forma de qualquer furo cilíndrico pode ser caracterizado por seu diâmetro ou raio e sua profundidade. Nós nos referimos a este conjunto como forma genérica paramétrica. Muitos termos específicos da aplicação não referenciam "features" como uma forma que pode ser representada através de um conjunto de parâmetros. Entretanto, a grande maioria das "features" relacionadas com produtos mecânicos podem ser associadas a um conjunto de parâmetros a partir do qual a sua forma pode ser derivada. Estes parâmetros podem ser classificados em dois tipos principais:

[10] Primitivos padrões são: cubo, cone, cilindro, esfera, toróide, entre outos.

- **Parâmetros de dimensão**, são parâmetros que definem o tamanho da "feature". Sempre existe um conjunto mínimo de parâmetros necessário para definir o conjunto de dimensões completo da "feature", apesar de que isto pode ocorrer de diversas maneiras.

- **Parâmetros de encaixe**, são parâmetros que definem o encaixe da "feature" no produto. Sempre existe um conjunto mínimo de parâmetros necessários para encaixar a "feature" ao produto, apesar de que isto pode ocorrer de diversas maneiras. Estes parâmetros permitirão que o usuário associe uma "feature" ao Modelo do Produto através da seleção dos encaixes do Modelo do Produto.

Estes parâmetros fornecem uma descrição implícita da forma de uma "feature", que, segundo a nossa proposta, deverá possuir uma definição e uma representação associada. No apêndice A apresentaremos uma revisão bibliográfica sobre a área de projeto paramétrico.

1.6 Estrutura do Trabalho

Como representar "features"? Qual a dimensionalidade de "feature" que deve ser utilizada? Qual o nível de abstração deve ser utilizado? Como relacionar coerentemente todos os conceitos apresentados na seção 1.4? São perguntas que ainda não foram devidamente respondidas. Este tabalho não responderá a todas as perguntas, mas forneceremos um meio que permitirá respondê-las. Estamos interessados em:

- suportar "features" complexas definidas pelo usuário;
- suportar "features" volumétricas e de superfície simultaneamente;
- suportar exames de validação do estado das "features";
- suportar a correta representação de dimensões;

- suportar uma correta representação para os encaixes de uma "feature";
- suportar uma representação coerente para planos e eixos de simetria.

Estes objetivos são atingidos utilizando-se do conceito de **dimensões relativas** que será apresentado no capítulo 2.

No capítulo 3 apresentaremos uma extensão para a estrutura B-Rep. Discutiremos rapidamente sobre como vários pesquisadores tratam o problema de representação de "features" e concluíremos que acrescentar novos elementos geométricos que não necessitam estar sobre o contorno do Modelo do Produto é uma solução. Isto nos permitirá representar inclusive planos e eixos de simetria, além de permitir uma coerente implementação das dimensões relativas.

No capítulo 4 detalharemos a representação do **Modelo do Produto**. Apresentaremos uma possível implementação para os **encaixes de uma "feature"**. Desta maneira o usuário poderá definir as suas próprias "features" e poderá selecioná-las a partir de um menu e posicioná-las corretamente segundo a sua própria vontade. Esta representação é uma aplicação do conceito de dimensões relativas.

No capítulo 5 detalharemos a implementação do protótipo que permitiu testar a aplicabilidade do conceito de dimensões relativas ao projeto mecânico paramétrico. Também apresentamos neste capítulo alguns exemplos executados pelo protótipo.

No capítulo final apresentaremos nossas conclusões sobre este trabalho e indicaremos várias linhas de atuação que permitirão dar-lhe continuidade. Algumas delas já estão, inclusive, em andamento.

Capítulo 2

Dimensões Relativas

Neste capítulo apresentamos como o conceito de **dimensões relativas** é utilizado na estrutura que representa o **Modelo do Produto**. Este conceito de **dimensões relativas** permite uma representação flexível para "features".

2.1 Modelo do Produto

O **Modelo do Produto** [1] é representado por uma estrutura hierárquica com vários níveis de complexidade (vide figura 2.1). O Nó do Modelo do Produto representa o nó raiz. Todos os nós imediatamente inferiores ao Modelo do Produto são **montagens**. Da mesma maneira, todos os nós imediatamente inferiores às montagens são **peças**. Cada nó possui acesso apenas aos nós do nível de complexidade imediatamente inferior. Desta maneira a **montagem** possui acesso a todas as informações das **peças** imediatamente abaixo a ele. Nesta estrutura existe uma diferença entre o nó mais à esquerda e os outros nós. Todas as informações do nó mais à esquerda são transferidas diretamente para o nó superior. As informações dos outros nós são transferidas para o nó superior por meio de transformações. Devido a esta propriedade, dizemos que o nó mais à esquerda é o **nó fixo** e todos os outros nós são os **nós móveis**. Todo nó possui um conjunto de elementos

[1] Modelo do Produto é criado pela composição de montagens, peças e "features".

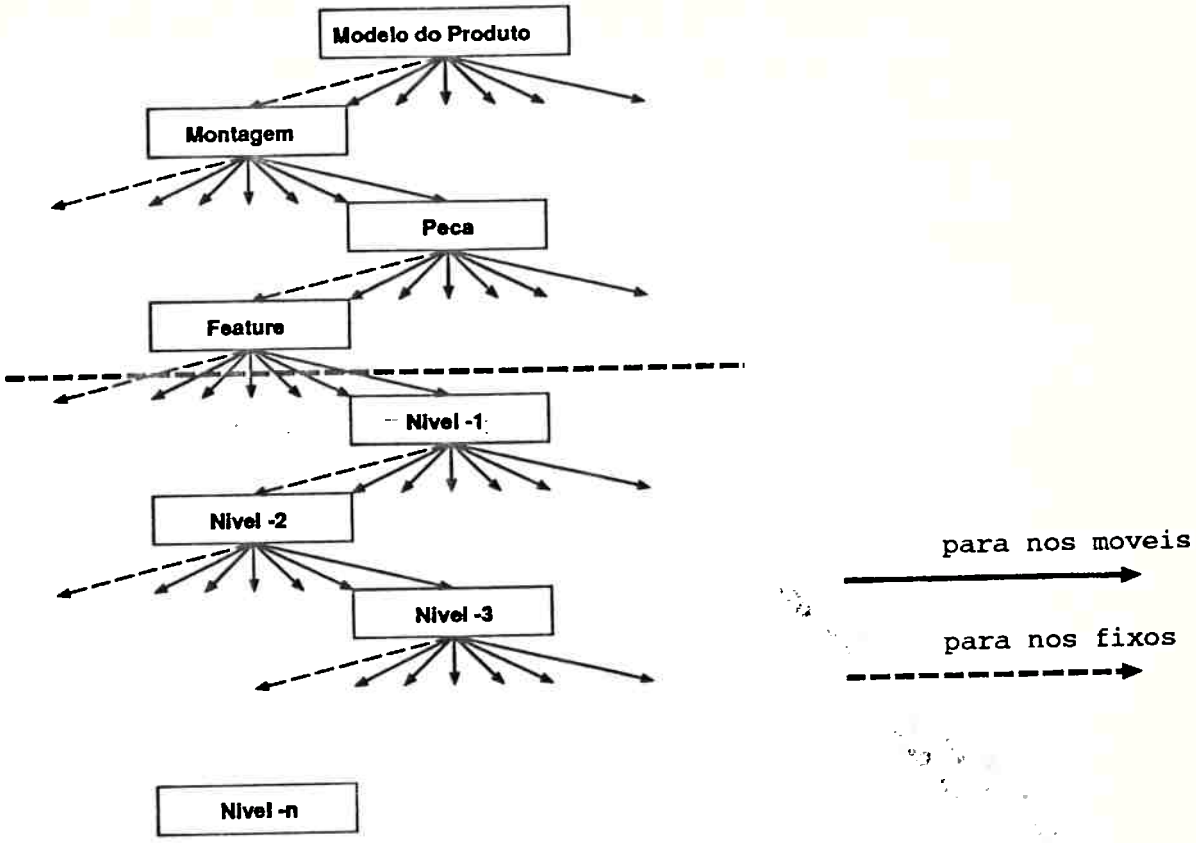


Figura 2.1: Estrutura hierárquica com vários níveis de complexidade representando o Modelo do Produto.

constituintes primitivos que podem ser pontos, linhas e planos.

De acordo com esta estrutura, é possível definir dois tipos de dimensões: **dimensões locais** e **dimensões relativas**. Dimensões associadas a um único nó são dimensões locais. Dimensões relativas restringem elementos constituintes primitivos de dois nós de níveis diferentes, um nó imediatamente superior e o outro imediatamente inferior entre si. Em outras palavras, uma **montagem** possui as suas dimensões locais e seu posicionamento em relação ao **Modelo do Produto** é restringido por um conjunto de dimensões relativas. As dimensões relativas são restrições que associam o nó móvel inferior ao nó superior. O nó fixo não possui nenhuma dimensão relativa restringindo ao nó imediatamente superior, pois suas informações são transferidas sem nenhuma transformação.

Um nó pode ser simbolicamente representado por:

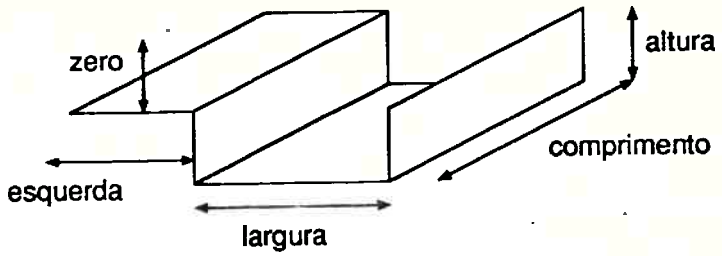


Figura 2.2: Exemplo de definição de uma "feature". *Zero* e *Esquerda* são dimensões relativas.

$$A = f_A(B_1, t_2 \cdot B_2, t_3 \cdot B_3, \cdots, t_n \cdot B_n) \tag{2.1}$$

onde, A é o nó superior, B_i é um nó inferior ao nó A e t_i é uma transformação que está sendo aplicada ao nó B_i. B_1 é o nó fixo, desta maneira ele não possui nenhuma transformação associada a si. A função f_A representa a transferência de informações para o nó superior a partir dos nós imediatamente inferiores.

2.2 Representando "Features"

O nosso objetivo é definir um mecanismo que facilite a manipulação das "features" pelo usuário. De tal maneira que ele selecione uma "feature" a partir de um menu e apenas forneça alguns elementos constituintes primitivos para anexar a "feature" ao Modelo do Produto. Este efeito pode ser implementado pela associação de algumas dimensões relativas às "features". A figura 2.2 ilustra um exemplo de definição de "feature".

Observe que apenas o elemento constituinte primitivo associado à "feature" está definido e o usuário deverá fornecer o elemento constituinte primitivo associado ao Modelo do Produto. Nas próximas seções detalharemos melhor o que vem a ser uma dimensão relativa. Através destas seções iniciais esperamos ter explicado rapidamente como pretendemos utilizar este novo conceito.

2.3 Conjuntos Especiais de Transformações

2.3.1 Dimensões Relativas

Um conjunto de dimensões relativas é representado por:

$$\Delta = \{d_1, d_2, \cdots, d_n\} \tag{2.2}$$

onde, Δ é o conjunto de dimensões relativas e d_i é uma dimensão relativa. Uma dimensão relativa possui um **tipo**, um **valor** e relaciona dois elementos constituintes. Todos os quinze tipos de dimensões relativas estão listadas na tabela 2.1. Uma dimensão relativa deve estar em um de dois estados diferentes: **satisfeita** e **não satisfeita**. Consideremos uma dimensão relativa de tipo distância plano-plano com normais coincidentes com um valor de 2. Esta dimensão relativa restringe um plano do nó móvel imediatamente inferior e um plano do nó imediatamente superior. Se os dois planos possuírem normais coincidentes e a distância entre eles for igual a 2 então esta dimensão relativa está satisfeita. Em caso contrário, a dimensão relativa não está satisfeita e é possível tansformar o nó móvel imediatamente inferior de maneira a satisfazer esta dimensão relativa. O estado do conjunto de dimensões relativas é representado por:

$$A < \Delta > B = \delta \tag{2.3}$$

onde, A representa o nó superior, B representa um nó inferior ao nó A e δ representa o conjunto de dimensões relativas que estão satisfeitas. Desta maneira, se $\delta = \{\}$ nenhuma dimensão relativa está satisfeita, e se $\delta = \Delta$ então todas as dimensões relativas estão satisfeitas.

2.3.2 Conjunto de Transformações

A transformação de um nó pode ser representada por:

Tabela 2.1: Todos os quinze tipos de dimensões relativas

tipo de restrição	código
distância plano-plano com normais coincidentes	DisNC
distância plano-plano com normais opostas	DisNO
ângulo plano-plano	AngPP
distância plano-linha (e vice-versa)	DisPR (DisRP)
ângulo plano-linha (e vice-versa)	AngPR (AngRP)
distância plano-ponto (e vice-versa)	DisPp (DispP)
distância linha-linha com direções coincidentes	DisOC
distância linha-linha com direções opostas	DisOO
ângulo linha-linha	AngRR
distância linha-ponto (e vice-versa)	DisRp (DispR)
distância ponto-ponto	Dispp

$$E' = t \cdot E \tag{2.4}$$

onde, t é uma transformação, E é um nó e E' é o nó transformado. Nós definiremos duas operações sobre conjuntos de transformações: **intersecção e distribuição**. A operação de intersecção é definida por:

$$t \in \theta_1 \text{ e } t \in \theta_2 \Rightarrow t \in \theta_1 \cap \theta_2 \tag{2.5}$$

onde, θ_1 e θ_2 são dois conjuntos de transformações, t é uma transformação e $\cap$ é o operador de intersecção. A operação de distribuição é definida por:

$$t_1 \in \theta_1 \text{ e } t_2 \in \theta_2 \Rightarrow t_1 \cdot t_2 \in \theta_1 * \theta_2 \tag{2.6}$$

onde, θ_1 e θ_2 são dois conjuntos de transformações, t_1 e t_2 são transformações e $*$ é o operador de distribuição. Observe que esta operação não é comutativa.

2.3.3 Coordenada Arbitrária

Dois nós A e B são **equivalentes**, se todos os pontos internos de A também são internos de B, se todos os pontos externos de A também são externos de B e se todos os pontos do contorno de A também são pontos do contorno de B. Dada uma transformação t_{TS} tal que:

$$E' = t_{TS} \cdot E \tag{2.7}$$

Se o nó E' é **equivalente** ao nó E e t_{TS} não é a matriz identidade, então dizemos que E possui uma ou mais coordenadas arbitrárias. O conjunto de todas as transformações t_{TS} é denominado por conjunto de **transformações de simetria**. Isto pode ser melhor compreendido através de um exemplo: o cilindro pode ser rotacionado ao redor de seu eixo de simetria que o resultado da rotação é equivalente a ele próprio. Em particular, os

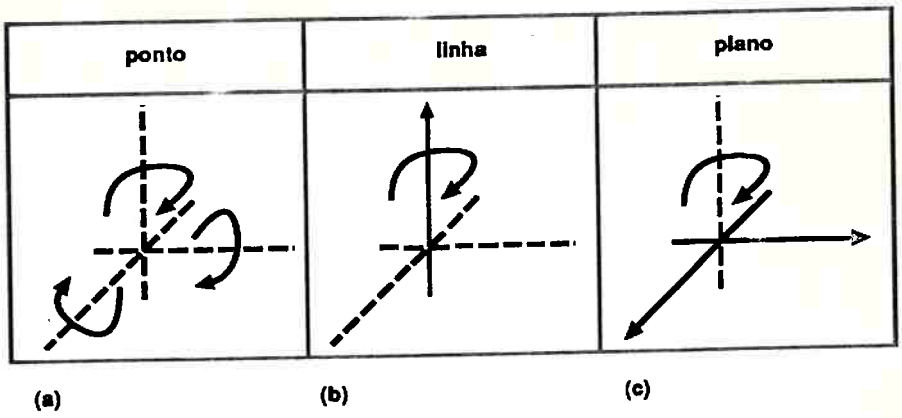


Figura 2.3: Conjuntos de transformações de simetria associados aos elementos primitivos.

elementos constituintes primitivos plano, linha e ponto possuem coordenadas arbitrárias que serão de grande valia para o desenvolvimento de nosso algoritmo.

- **ponto:** é possível rotacionar o ponto ao redor de qualquer eixo que passe por ele próprio que o resultado da transformação será equivalente ao ponto original. Representaremos o conjunto de transformações de simetria do ponto por PNT (vide figura 2.3(a)).

- **linha:** é possível rotacionar a linha ao redor do eixo coincidente com o seu sentido, ou transladá-la na direção de seu sentido que o resultado da transformação será equivalente à linha original. Representaremos o conjunto de transformações de simetria da linha por RET (vide figura 2.3(b));

- **plano:** é possível rotacionar o plano ao redor de qualquer eixo paralelo a sua normal, ou transladá-lo em qualquer direção perpendicular a sua normal que o resultado da transformação será equivalente ao plano original. Representaremos o conjunto de transformações de simetria do plano por PLN (vide figura 2.3(c));

2.3.4 Graus de Liberdade Associados a Dimensões Relativas

As dimensões relativas possuem alguns graus de liberdade no sentido de que o nó inferior pode ser transformado e as dimensões relativas que estão satisfeitas antes do nó ser transformado permanecerão satisfeitas após a transformação. Considere que a transformação $t^{\{d_1\}}$ satisfaz a dimensão relativa d_1. Isto pode ser representado simbolicamente por:

$$A < \Delta > (t^{\{d_1\}} \cdot B) = \{d_1\} \tag{2.8}$$

O conjunto de todas as transformações t_{GL} que podem ser aplicadas ao nó inferior B e que mantém a dimensão relativa d_1 satisfeita é chamado por conjunto de graus de liberdado associado à dimensão relativa d_1. Isto é representado simbolicamente por:

$$A < \Delta > (t_{GL} \cdot t^{\{d_1\}} \cdot B) = \{d_1\} \tag{2.9}$$

Utilizaremos a letra grega τ para representar o conjunto de graus de liberdade. O conjunto de graus de liberdade associado a uma dimensão relativa está associado aos conjuntos de transformações de simetria dos elementos constituintes primitivos restringidos pela dimensão relativa.

Como uma dimensão relativa d_i restringe dois elementos constituintes primitivos, um está associado ao nó superior e o outro ao nó inferior. Assim sendo, existem dois subconjuntos de graus de liberdade associados à dimensão relativa. Representaremos simbolicamente os subconjuntos de graus de liberdade por $\tau_S^{d_i}$ e $\tau_I^{d_i}$, onde o primeiro está associado ao nó superior e o segundo está associado ao nó inferior. Representaremos o conjunto de graus de liberdade associado à dimensão relativa d_i, por:

$$GL(d_i) = \{\tau_S^{d_i}, \tau_I^{d_i}\} \tag{2.10}$$

Desta maneira uma dimensão relativa apresenta dois subconjuntos de graus de liberdade. Assim, se a dimensão relativa d_i já está satisfeita, representada simbolicamente

por:

$$A < \Delta > B = \{d_i\} \tag{2.11}$$

ao transformarmos o nó inferior por qualquer transformação $t_I \in \tau_I^{d_i}$ a dimensão relativa d_i ainda permanecerá satisfeita. Isto pode ser simbolicamente representado por:

$$A < \Delta > (t_I \cdot B) = \{d_i\} \tag{2.12}$$

Se transformarmos o nó superior por qualquer transformação $t_S \in \tau_S^{d_i}$, a dimensão relativa também permanecerá satisfeita. Entretanto, é importante observar que o posicionamento relativo entre o nó superior e o nó inferior não é alterado se transformarmos o nó superior por t_S ou se transformarmos o nó inferior por t_S^{-1} [2]. É interessante observar que se:

$$t_S \in \tau_S^{d_i} \Longrightarrow t_S^{-1} \in \tau_S^{d_i} \tag{2.13}$$

Desta maneira, é possível representar a transformação do nó superior simbolicamente por:

$$A < \Delta > (t_S^{-1} \cdot t_I \cdot B) = \{d_i\} \tag{2.14}$$

Esta propriedade cria uma hierarquia entre os graus de liberdade no sentido de que ao aplicarmos uma transformação pertencente ao *conjunto de graus de liberdade do nó superior* ao nó inferior, estaremos alterando também o conjunto de transformações de simetria do nó inferior. Suponha que uma dimensão relativa do tipo distância plano-linha foi satisfeita (vide figura 2.4). O significado da equação 2.9 é que é possível transformar o plano segundo suas transformações de simetria que a dimensão relativa se manterá

[2] Uma propriedade importante das coordenadas arbitrárias é que se t_{TS} pertence a um conjunto de transformações de simetria, então t_{TS}^{-1} também pertence. Isto pode ser provado por, $E' = t_{TS} \cdot E \Rightarrow E = t_{TS}^{-1} \cdot E' = t_{TS}^{-1} \cdot t_{TS} \cdot E = I \cdot E = E$

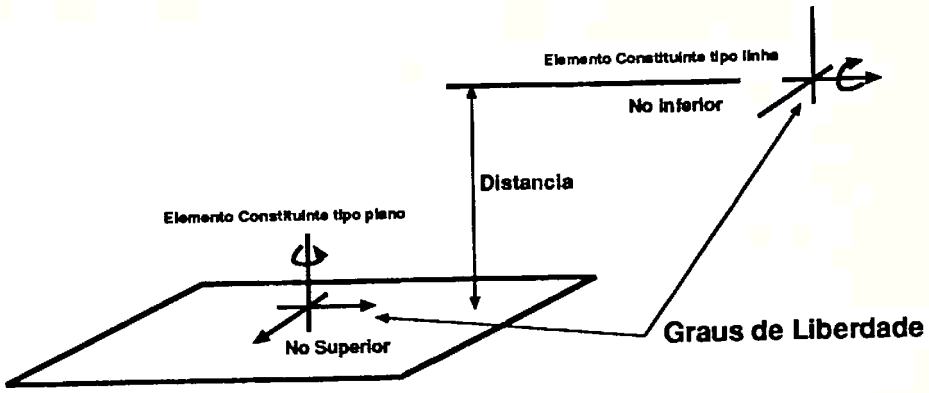


Figura 2.4: Representação dos níveis de hierarquia para uma restrição do tipo plano-linha.

válida. A definição do conjunto de graus de liberdade associados a uma dimensão relativa depende do seu tipo e dos elementos primitivos que ela restringe.

Antes de listarmos todos os conjuntos de graus de liberdade associados às dimensões relativas é necessário definirmos um conjunto especial de transformações que representa todas as possíveis translações no espaço tridimensional. Representaremos o conjunto de todas as possíveis translações por $3DT$. A tabela 2.2 define todos os conjuntos de graus de liberdade associados às dimensões relativas.

2.3.5 Grau de Simetria

Ambos os nós superior e inferior devem possuir um conjunto de transformações de simetria. Chamaremos este conjunto de transformações por conjunto de graus de simetria. Ele é representado por,

$$\chi = \{\varsigma_S, \varsigma_I\} \tag{2.15}$$

onde, ς_S é o conjunto de graus de simetria do nó superior, ς_I é o conjunto de graus de simetria do nó inferior e χ é o conjunto de graus de simetria do conjunto. Ao satisfazer a i-ésima dimensão relativa, determinaremos o conjunto de graus de liberdade $\varphi_{i+1} = \{\tau_S, \tau_I\}$. O conjunto de graus de simetria deve ser atualizado a cada passo, de modo que no i-ésimo passo o conjunto de graus de simetria pode ser representado por $\chi_{i+1} = \{\varsigma_S, \varsigma_I'\}$. Agora,

Tabela 2.2: Conjunto de graus de liberdade associados às dimensões relativas.

dimensão relativa	grau de liberdade
DisNC	$\{PLN, PLN\}$
DisNO	$\{PLN, PLN\}$
AngPP	$\{PLN * 3DT, PLN * 3DT\}$
DisRP	$\{RET, PLN\}$
DisPR	$\{PLN, RET\}$
AngRP	$\{RET * 3DT, PLN * 3DT\}$
AngPR	$\{PLN * 3DT, RET * 3DT\}$
DisPp	$\{PLN, PNT\}$
DispP	$\{PNT, PLN\}$
DisOC	$\{RET, RET\}$
DisOO	$\{RET, RET\}$
AngRR	$\{RET * 3DT, RET * 3DT\}$
DispR	$\{PNT, RET\}$
DisRp	$\{RET, PNT\}$
Dispp	$\{PNT, PNT\}$

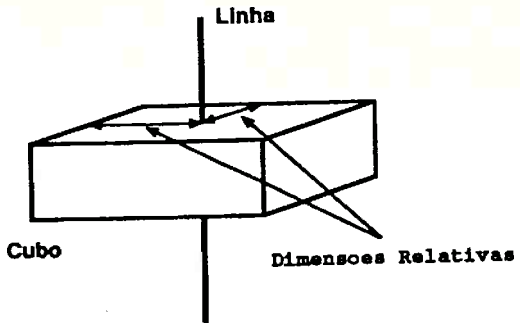


Figura 2.5: Exemplo de uma linha fixa a um cubo.

se a expressão

$$\tau_S * \tau_I \subset \varsigma_S * \varsigma_I' \tag{2.16}$$

for verdadeira, então o nó inferior está fixo ao nó superior.

Considere o exemplode uma linha que possui duas dimensões relativas de tipo plano-linha restringindo-o a um cubo (vide figura 2.5). Quando ambas as dimensões relativas estiverem satisfeitas, e o conjunto de graus de liberdade φ_2 deve ser

$$\varphi_2 = \{RET, TRN\} \tag{2.17}$$

onde, TRN é o conjunto de todas as possíveis translações em um determinado sentido. O conjunto de graus de simetria χ deve ser

$$\chi = \{RET, EMP\} \tag{2.18}$$

onde, EMP é o conjunto vazio de transformações. Como o sentido de translação que define o conjunto de transformações TRN é idêntico ao sentido da linha que define o conjunto de transformações RET, então $TRN * RET = RET \rightarrow TRN * RET \subset RET$. Portanto, a linha está fixa ao cubo.

2.3.6 Hierarquia de Transformações

Uma dimensão relativa possui dois níveis hierárquicos de conjuntos de transformações de simetria. Desta maneira, uma transformação também deve ser representada por dois componentes, o primeiro componente representa uma transformação associada ao nó superior e o segundo componente representa uma transformação associada ao nó inferior. Desta maneira, uma transformação t é em realidade a combinação de duas transformações:

$$t = \{t_S, t_I\} \tag{2.19}$$

onde, t_S é a transformação associada ao nó superior e t_I é a transformação associada ao nó inferior. É necessário aplicar primeiro a transformação associada ao nó inferior e depois a transformação associada ao nó superior. Ou seja,

$$E' = t_S \cdot t_I \cdot E \tag{2.20}$$

Esta ordem deve ser mantida, pois a transformação associada ao nó superior interfere no conjunto de transformações de simetria do nó inferior.

2.4 Satisfazendo uma Dimensão Relativa

Estamos interessados em encontrar uma transformação t^{Δ} tal que:

$$A < \Delta > (t^{\Delta} \cdot B) = \Delta \tag{2.21}$$

O nó inferior B está posicionado relativamente ao nó superior A de modo a satisfazer todas as dimensões relativas. Neste trabalho definiremos um algoritmo que permita encontrar todas as transformações que satisfazem um conjunto de dimensões relativas, caso elas existam. Este algoritmo encontrará as transformações passo a passo, satisfazen-

do uma dimensão relativa por vez. Ou seja, encontraremos primeiro uma transformação $t^{\{d_1\}}$[3]; em seguida encontraremos uma transformação $t^{\{d_2\}}_{\{d_1\}}$ [4] e assim por diante.

As dimensões relativas serão satisfeitas passo a passo. No i-ésimo passo a dimensão relativa d_i deverá ser satisfeita. Neste passo todas as dimensões relativas $d_1, d_2, \cdots, d_{i-1}$ já foram satisfeitas. Associado ao i-ésimo passo existe um conjunto de graus de liberdade φ_i que contém todas as transformações que mantém todas as dimensões relativas que estavam satisfeitas antes de sua aplicação, satisfeitas após a sua aplicação. Isto pode ser representado simbolicamente por:

$$A < \Delta > B' = \{d_1, d_2, \cdots, d_{i-1}\} \tag{2.22}$$

e,

$$\forall t \in \varphi_i \Rightarrow A < \Delta > (t \cdot B') = \{d_1, d_2, \cdots, d_{i-1}\} \tag{2.23}$$

Supondo que a dimensão relativa d_i seja satisfeita por uma transformação

$$t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\}} = \{t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\},S}, t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\},I}\} \in \varphi_i \tag{2.24}$$

onde, as transformações associadas aos nós superior e inferior estão respectivamente representados pelos índices S e I. Desta maneira,

$$A < \Delta > (t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\},S} \cdot t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\},I} \cdot B') = \{d_1, d_2, \cdots, d_{i-1}, d_i\} \tag{2.25}$$

O conjunto de graus de liberdade associado ao nó inferior deve ser atualizado pela transformação associada ao nó superior $t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\},S}$. Assim,

$$\varphi'_i = \{\tau_S, t^{\{d_i\}}_{\{d_{i-1},\cdots,d_1\},S} \cdot \tau_I\} \tag{2.26}$$

---

[3]O índice indica que a dimensão relativa d_1 foi satisfeita.

[4]Os índices indicam que a dimensão relativa d_2 foi satisfeita e que a dimensão relativa d_1 foi mantida satisfeita.

O conjunto de graus de liberdade φ_{i+1} é a intersecção entre φ'_i e $GL(d_i)$ [5]. Ou seja,

$$\varphi_{i+1} = \varphi'_i \cap GL(d_i) \tag{2.27}$$

Neste momento é necessário verificar se a expressão 2.16 é verdadeira ou não. Caso ela seja verdadeira então o algoritmo terminou. Se a expressão 2.16 for verdadeira e ainda existam dimensões relativas que não foram analisadas então os nós estavam super-restritos [6].

De acordo com a equação 2.27 a intersecção entre conjuntos de transformações de simetria possui um papel muito importante no algorimo. Todos os possíveis casos de intersecção entre conjuntos de transformações de simetria estão representados nas tabelas 2.3 e 2.4 [7]. TOT, PLN, RET, PNT, ROT, TRN, $R3D$, $T2D$ e EMP são conjuntos de transformações de simetria. Abaixo descrevemos o significado de cada conjunto:

- TOT, nenhuma restrição, representa total liberdade;
- PLN, conjunto de transformações de simetria associado ao elemento primitivo plano;
- RET, conjunto de transformações de simetria associado ao elemento primitivo linha;
- PNT, conjunto de transformações de simetria associado ao elemento primitivo ponto;
- ROT, é possível rotacionar o nó associado ao redor de um eixo de rotação que o resultado da transformação será equivalente ao original;
- TRN, é possível transladar o nó associado em uma direção que o resultado da transformação será equivalente ao original;

[5] $GL(d_i)$ é o conjunto de graus de liberdade associado à dimensão relativa d_i.

[6] Tradução de "overconstrained".

[7] Dividimos em duas tabelas, pois não há espaço suficiente para juntá-las em uma única tabela.

Tabela 2.3: Intersecção entre conjuntos de transformações de simetria - parte I.

	TOT	*PLN*	*RET*	*PNT*	*ROT*
TOT	*TOT*	*PLN*	*RET*	*PNT*	*ROT*
PLN		*PLN* *TRN*	*ROT* *TRN*	*ROT*	*ROT* *EMP*
RET			*RET* *TRN* *EMP*	*ROT* *EMP*	*ROT* *EMP*
PNT				*PNT* *ROT*	*ROT* *EMP*
ROT					*ROT* *EMP*

- *R3D*, é possível rotacionar o nó associado ao redor de um eixo de rotação e transladá-lo em qualquer sentido que o resultado da transformação será equivalente ao original;
- *T2D*, é possível transladar o nó associado em qualquer sentido perpendicular a uma dada direção que o resultado da transformação será equivalente ao original;
- *EMP*, não é possível realizar nenhuma transformação, pois o nó associado já está fixo.

2.5 Árvore de Solução

Para encontrarmos as transformações que satisfazem um dado conjunto de dimensões relativas, definiremos uma estrutura hierárquica em árvore. Nesta árvore existem

Tabela 2.4: Intersecção entre conjuntos de transformações de simetria - parte II.

	TRN	*R3D*	*T3D*	*T2D*	*EMP*
TOT	*TRN*	*R3D*	*T3D*	*T2D*	*EMP*
PLN	*TRN*, *EMP*	*PLN*, *T2D*	*T2D*	*T2D*, *TRN*	*EMP*
RET	*TRN*, *EMP*	*RET*, *TRN*	*TRN*	*TRN*, *EMP*	*EMP*
PNT	*EMP*	*ROT*	*EMP*	*EMP*	*EMP*
ROT	*EMP*	*ROT*, *EMP*	*EMP*	*EMP*	*EMP*
TRN	*TRN*, *EMP*	*TRN*	*TRN*	*TRN*, *EMP*	*EMP*
R3D		*R3D*, *T3D*	*T3D*	*T2D*	*EMP*
T3D			*T3D*	*T2D*	*EMP*
T2D				*T2D*, *TRN*	*EMP*
EMP					*EMP*

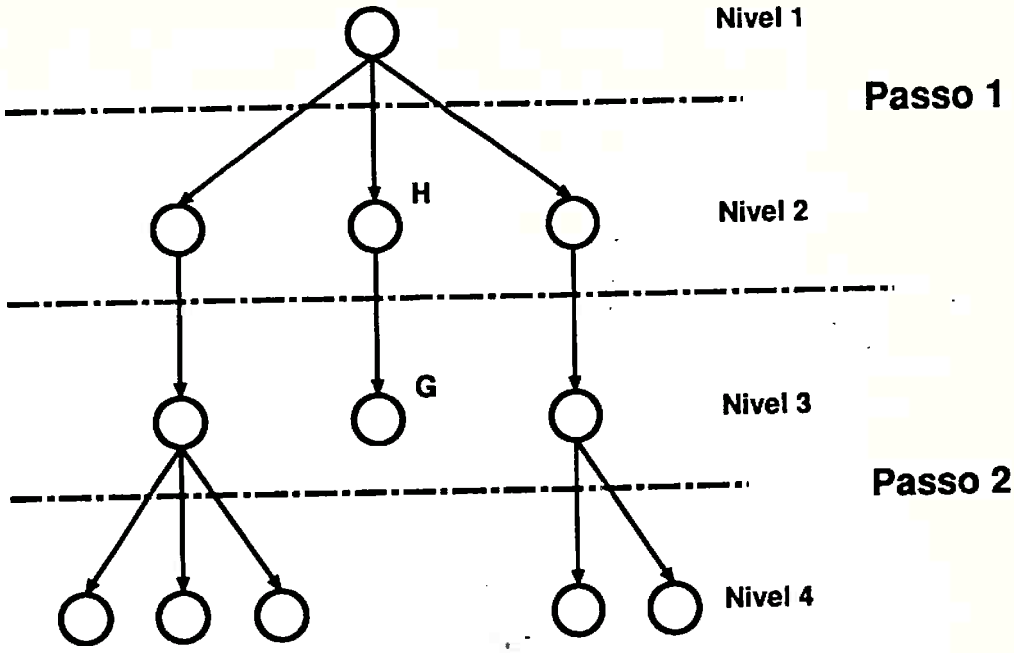


Figura 2.6: Árvore de Solução.

dois tipos de níveis (vide figura 2.6). Os nós associados aos níveis ímpares representam dimensões relativas a serem solucionadas e os nós associados aos níveis pares representam transformações que satisfazem as dimensões relativas associadas ao nó superior.

A i-ésima dimensão relativa a ser solucionada está representada no nível $2\cdot(i-1)+1$. As transformações que satisfazem a dimensão relativa d_i estão representadas no nível $2\cdot i$. Esta estrutura nos permitirá encontrar todas as transformações que satisfazem um dado conjunto de dimensões relativas.

Esta estrutura minimizará os erros de aproximação pois não existe a necessidade de calcular as transformações inversas. A figura 2.6 ilustra um exemplo de uma árvore de solução com um conjunto de duas dimensões relativas onde o algoritmo encontrou cinco transformações que satisfazem o conjunto. Desta maneira é possível compará-las e selecionar apenas as transformações diferentes. É possível observar que a dimensão relativa associada ao nó G não foi possível de ser satisfeita, desta maneira a transformação associada ao nó H é descartada. O algoritmo que busca por todas as possíveis transformações que satisfazem um conjunto de dimensões relativas encontra-se detalhado no apêndice B.

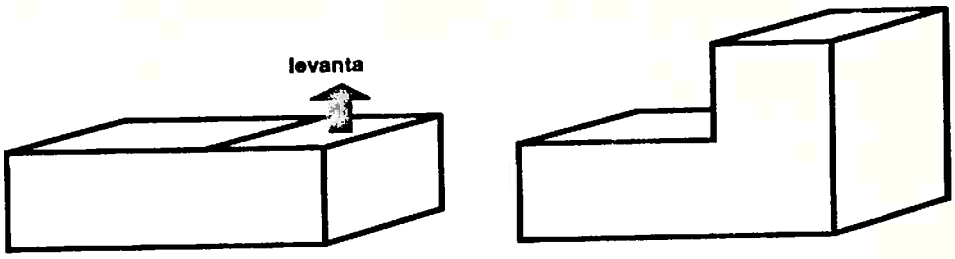
Capítulo 3

Estrutura B-Rep Estendida

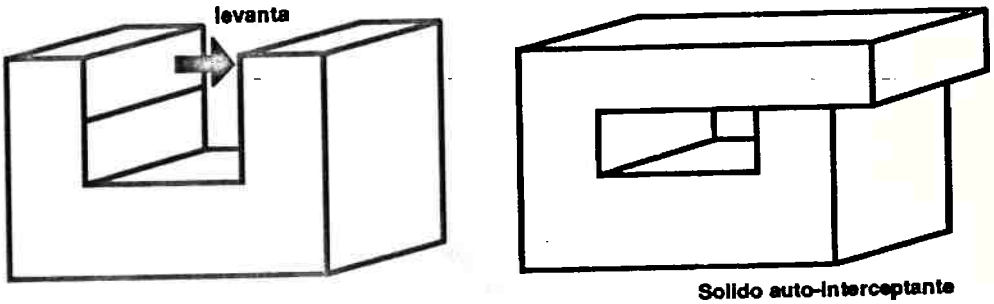
Conforme apresentamos no capítulo anterior, os elementos constituintes primitivos plano, linha e ponto possuem um papel muito importante na definição das dimensões relativas. Neste capítulo discutiremos como estes elementos constituintes primitivos podem ser representados e concluíremos que uma das possíveis soluções é acrescentar novos elementos à estrutura B-Rep. Veremos porque a proposta de associar pontos, linhas e planos aos vértices, arestas e faces, respectivamente, da representação B-Rep não nos satisfaz. Como conseqüência da proposta de acrescentar novos elementos à representação B-Rep veremos que é possível representar eixos e planos de simetria explicitamente.

3.1 Acrescentando Elementos Primitivos

A grande maioria dos pesquisadores adota a representação B-Rep para o desenvolvimento de Modeladores DbF [34]. A grande vantagem da representação B-Rep é que ela, teoricamente, suporta a representação de "features" volumétricas e "features" de superfície simultaneamente. Vários autores tentaram representar "features" por meio de atributos associados a elementos primitivos da representação B-Rep. Entretanto, algumas dificuldades não conseguiram ser superadas. Em nosso entender, as dificuldades não foram superadas pois cada elemento primitivo possui **duas funções:**



(a) Operacao Local Consistente



(b) Operacao Local Inconsistente

Figura 3.1: Operadores Locais.

- representar um elemento primitivo do sólido (primeira funcionalidade);
- representar um elemento constituinte da "feature" (segunda funcionalidade).

Nesta seção, como referenciaremos freqüentemente as duas **funções**, nós as chamaremos, respectivamente, por primeira funcionalidade e segunda funcionalidade. Recordemos que em Modelagem de Sólidos existem dois tipos de operações: **operações locais e operações complexas.** As operações locais modificam um sólido segundo uma semântica específica e não garantem a consistência da primeira funcionalidade dos elementos primitivos. O projetista necessita conhecer em detalhes tanto o sólido que está manipulando como a semântica do operador local que ele está utilizando. Caso o projetista não se atente para estes detalhes é possível produzir resultados que não são suportados pela representação B-Rep (vide figura 3.1). Apesar da primeira funcionalidade não ser garantida, a segunda funcionalidade é garantida com certa facilidade devido à semântica específica das operações locais.

Vários autores tentaram representar "features" utilizando seqüências de operações locais [9], pois elas permitem controlar de maneira exata a funcionalidade de representar elementos constituintes de uma "feature". Entretanto, quando estivermos posicionando várias "features" em um sólido, é possível que a seqüência de operações locais definida crie um sólido inconsistente. Prevendo esta possibilidade, Inui e Kimura [10] propuseram um sistema TMS ("Truth Maintenance System") que permite determinar qual é a operação inconsistente. Entretanto, mesmo neste sistema é necessário que o usuário conheça em detalhes tanto a semântica das operações locais como a representação do sólido que ele está manipulando.

As **operações complexas** são aplicadas a sólidos e, em oposição às operações locais, garantem a consistência da primeira funcionalidade mas não garantem a consistência da segunda funcionalidade. A figura 3.2 ilustra o exemplo de uma operação complexa; observe que apesar de conhecermos em detalhes a funcionalidade dos elementos primitivos dos sólidos originais a funcionalidade dos elementos primitivos do sólido resultante é variável, em parte dependente da posição relativa dos sólidos criadores. Por exemplo, supondo que as faces laterais à esquerda de ambos os cubos da figura 3.2 sejam coplanares, então surgirá no sólido resultante, uma face lateral à esquerda única que é a composição das faces laterais à esquerda dos sólidos criadores, conforme ilustrado na figura 3.2(a).

Entretanto, os casos em sua maioria não são tão simples e o controle da segunda funcionalidade está diretamente vinculado à implementação das operações complexas, mais especificamente, a como os Operadores de Euler [18] que manipulam diretamente a estrutura de dados estão sendo utilizados. Relacionado ao exemplo da figura 3.2 as seguintes perguntas podem surgir: as faces laterais à esquerda A e B dos sólidos criadores estão presentes no sólido resultante? Qual face será a face resultante quando as faces A e B forem coplanares?

A vantagem das operações complexas é que uma verificação completa do sólido resultante é realizada, portanto, não é possível produzir resultados que não são suportados pela representação. Resumindo esta discussão, podemos concluir que as **operações Locais:**

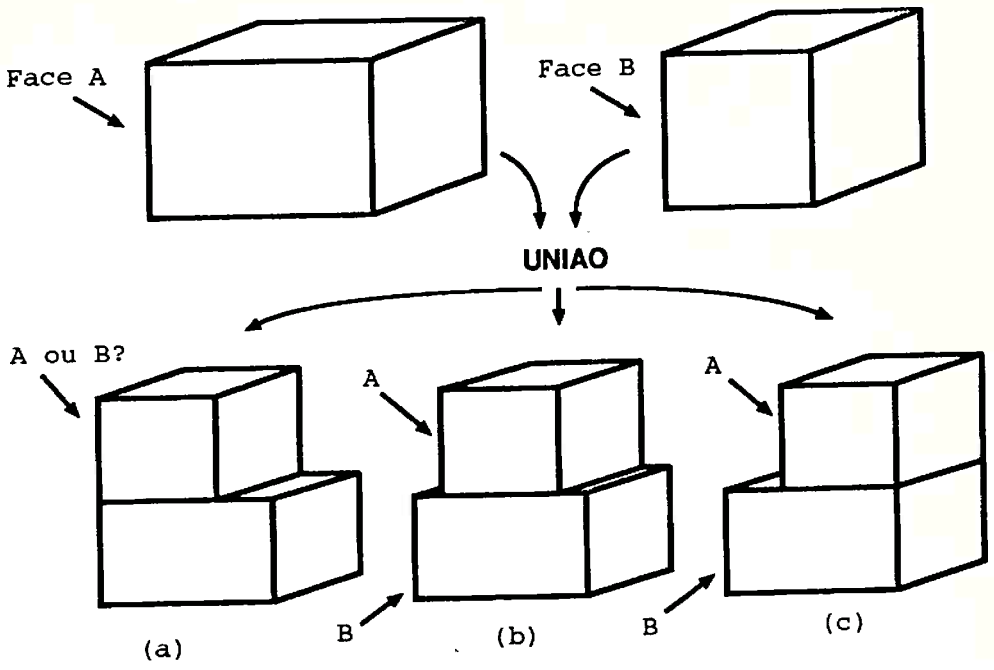


Figura 3.2: Operações Complexas.

- permitem controlar a representação de elementos constituintes de "features";
- suportam a manutenção local da topologia e da geometria do sólido;
- exigem o conhecimento de detalhes sobre o sólido e sobre a semântica das operações.

e que as **operações Complexas:**

- não permitem controlar a representação de elementos constituintes de "features";
- suportam a manutenção global da topologia e da geometria do sólido;
- não há necessidade de conhecer detalhes sobre o sólido e sobre a semântica das operações;
- exigem o conhecimento do posicionamento relativo dos sólidos criadores.

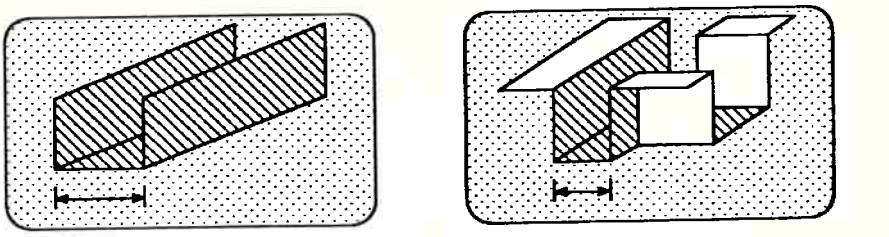


Figura 3.3: Relação entre elementos primitivos da representação B-Rep e elementos primitivos extras e a representação de "features".

Segundo o nosso ponto de vista, existe a necessidade de fazermos uso de ambos os tipos de operações (complexas e locais). Podemos adotar assim, uma nova proposta em que os elementos primitivos da representação B-Rep: vértice, aresta e face, representam elementos primitivos do sólido e que os elementos primitivos extras representam os elementos constituintes da "feature". Desta maneira, dividimos as duas funcionalidades entre dois tipos de elementos distintos. Em outras palavras, a "feature" intencional explícita associada ao Modelo do Produto é representada pelos elementos primitivos extras acrescentados à representação B-Rep (vide figura 3.3).

Definir quais são os elementos primitivos extras é um tema que foge ao escopo deste trabalho. Neste trabalho, consideraremos que existem apenas três tipos de elementos primitivos extras: ponto, linha e plano. Estes três tipos de elementos primitivos extras suportarão uma implementação do conceito de dimensões relativas introduzido no capítulo anterior.

3.2 Acrescentando Operações

Geralmente, algumas peças requerem a presença de entidades geométricas que não coincidem com nenhuma parte de seu contorno. Por exemplo, um furo pode ser definido em termos de seu eixo de simetria e um rasgo pode ser definido em termos de seu plano de

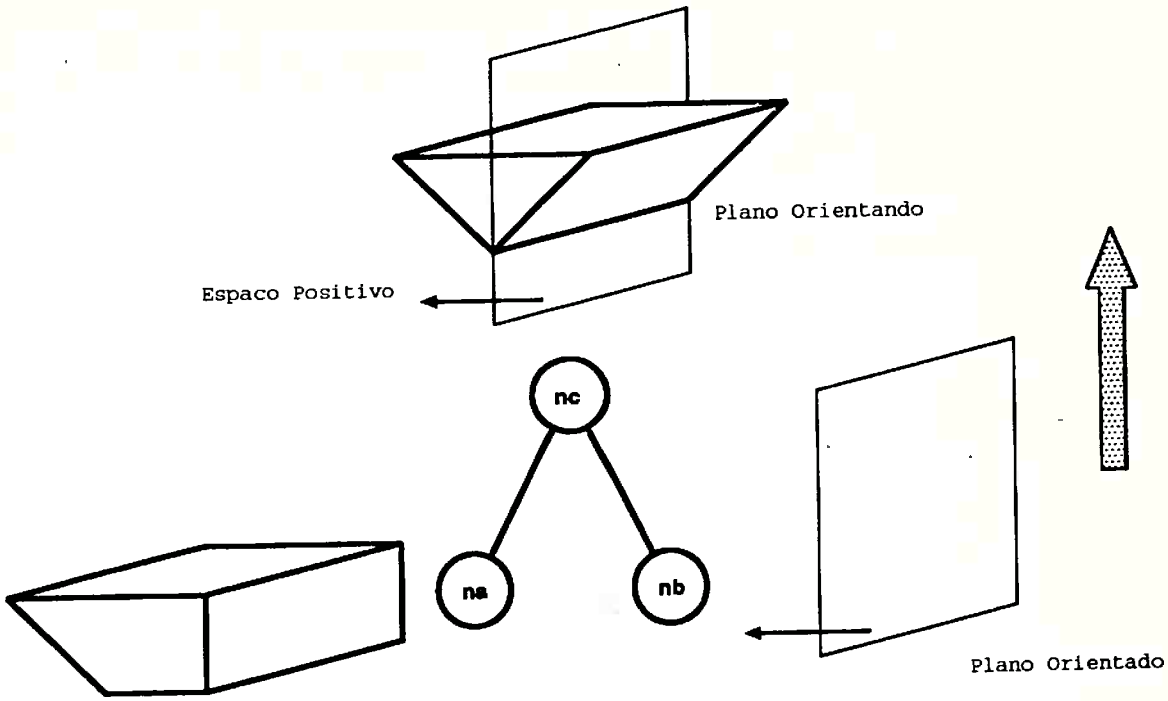


Figura 3.4: Semântica do plano de simetria.

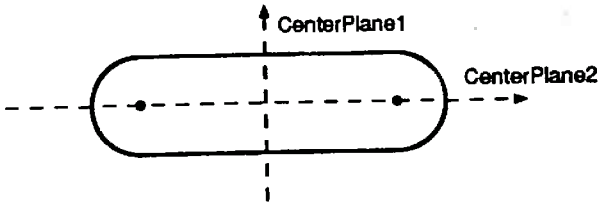


Figura 3.5: Peça com dois planos de simetria: CenterLine1 e CenterLine2.

simetria. Na maioria dos casos, o elemento de simetria não está envolvido na representação desenvolvida na peça. Veremos que é possível representar estes elementos através dos elementos constituintes primitivos.

3.2.1 Plano de Simetria

O plano de simetria pode ser representado como uma operação que combina um sólido e um plano orientado. O plano orientado divide o espaço em dois espaços: espaço positivo (concordante com o sentido da normal) e espaço negativo (oposto ao sentido

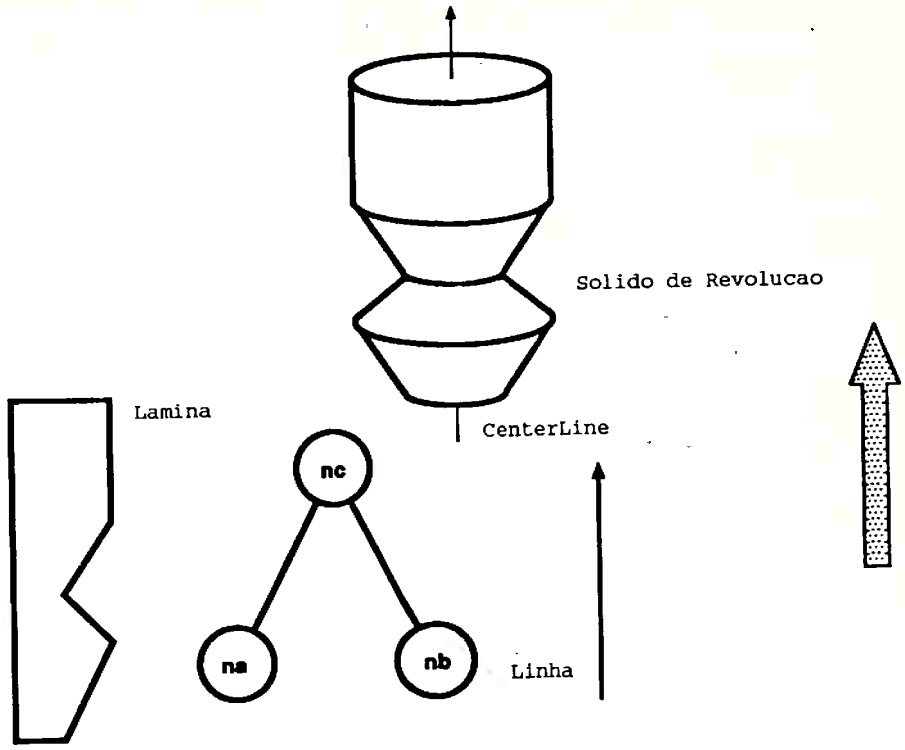


Figura 3.6: Semântica do eixo de simetria.

da normal). Para implementar esta operação, podemos adotar a seguinte semântica: remove-se tudo que estiver interno ao espaço negativo e espelha-se copiando o que estiver no espaço positivo em relação ao plano de simetria (vide figura 3.4). Utilizando esta semântica é possível representar inclusive casos especiais, como o ilustrado na figura 3.5 em que existem dois planos de simetria.

3.2.2 Eixo de Simetria

O eixo de simetria pode ser representado como uma operação que combina uma lâmina e uma linha, criando um sólido de revolução (vide figura 3.6). A linha representa o eixo de simetria do sólido de revolução.

Capítulo 4

Representação do Modelo do Produto

Conforme citado em 2.1, o Modelo do Produto é representado por uma estrutura hierárquica com vários níveis de complexidade, onde os nós intermediários são combinações dos nós imediatamente inferiores. Genericamente, todo nó possui associado a si um conjunto de parâmetros e um conjunto de elementos constituintes. Um nó é representado por

$$A(\Lambda_A, \Phi_A) \tag{4.1}$$

onde Λ_A é o conjunto de parâmetros do nó A e Φ_A é o conjunto de elementos constituintes do nó A. Λ_A é dado por:

$$\Lambda_A = \{\Lambda_{A1}, \Lambda_{A2}, \cdots, \Lambda_{An}\} \tag{4.2}$$

e Φ_A é dado por:

$$\Phi_A = \{\Phi_{A1}, \Phi_{A2}, \cdots, \Phi_{Am}\} \tag{4.3}$$

Um nó é a combinação de vários nós inferiores. Isto pode ser definido por

$$A(\Lambda_A, \Phi_A) = f_A(B_1(\Lambda_{B_1}, \Phi_{B_1}), t_2 \cdot B_2(\Lambda_{B_2}, \Phi_{B_2}), \cdots, t_n \cdot B_n(\Lambda_{B_n}, \Phi_{B_n})) \quad (4.4)$$

onde, o nó A é o nó superior e B_1, B_2, $\cdots$, B_n são os nós imediatamente inferiores ao nó A. B_1 é o nó fixo e todos os outros B_i são nós móveis. t_i são transformações aplicadas aos nós B_i. A função f_A combina os parâmetros e elementos constituintes dos nós imediatamente inferiores para definir os parâmetros e elementos constituintes do nó superior. Observe que a função que a função f_A não pode aplicar transformações aos nós que ela combina.

Nas duas próximas seções detalharemos uma possível implementação para definir como os elementos constituintes e os parâmetros podem ser combinados. Em seguida, discutiremos sobre como representar "features" segundo a estrutura hierárquica proposta.

4.1 Elementos Constituintes

Conforme ilustrado na equação 4.4, o **conjunto de elementos constituintes** associado a um nó intermediário é a combinação de elementos constituintes associados aos nós imediatamente inferiores. Entretanto, para que esta combinação ocorra, primeiro copiamos os elementos constituintes de interesse para o nó superior. Em seguida, é possível definir novos elementos constituintes através de funções que são aplicadas a conjuntos de elementos constituintes (vide figura 4.1).

Discutir sobre uma possível definição para elementos constituintes, e sobre quais elementos constituintes devem existir, são temas que fogem ao escopo deste trabalho. Estamos interessados em que existam elementos constituintes primitivos de tipo **plano**, **linha** e **ponto** para implementar o conceito de **dimensões relativas** introduzido no capítulo 2. Para facilidade de visualizacao, é interessante que elementos constituintes de tipo **volume** também possam ser definidos. Além disto, os elementos constituintes de tipo volume serão importantes para a representação das "features" conforme veremos na

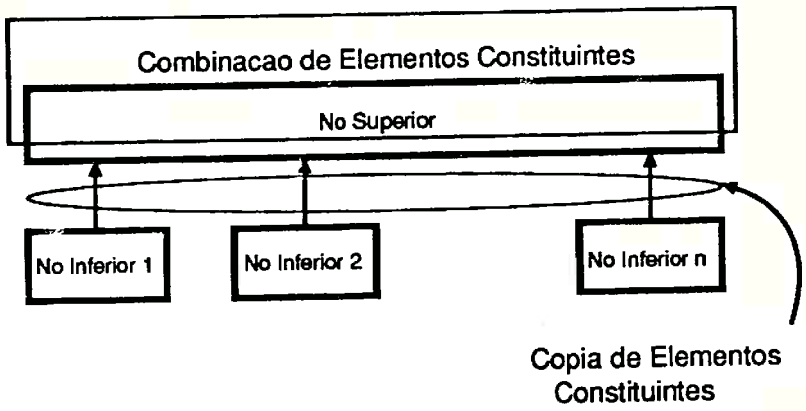


Figura 4.1: Cópia e combinação de elementos constituintes.

seção 4.3.

Alguns exemplos de funções que combinam elementos constituintes de tipo volume são:

- **Diferença;**
- **Intersecção;**
- **União.** [1]

Também é possível definir um conjunto de funções que combinem elementos constituintes de tipo volume com outros tipos de elementos constituintes. Por exemplo:

- **Espelha,** implementa a *função plano de simetria* (combina um elemento constituinte de tipo volume com um elemento constituinte de tipo plano);
- **Rotaciona uma lâmina,** implementa a *função eixo de simetria* (combina um elemento constituinte de tipo volume com um elemento constituinte de tipo linha).

[1] Estes exemplos correspondem às três operações booleanas disponíveis nos Modeladores de Sólidos convencionais [31].

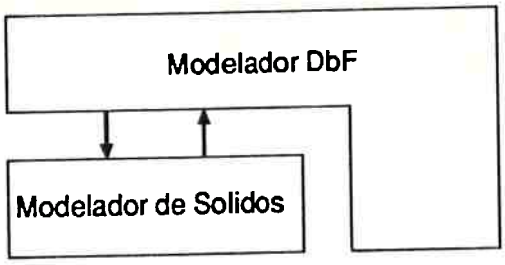


Figura 4.2: Modelador DbF como uma casca sobre um Sistema de Modelagem de Sólidos.

Esta proposta foi feita de modo que um **Núcleo de Modelagem de Sólidos** genérico possa ser utilizado. O Núcleo de Modelagem de Sólidos deverá ser capaz de manipular todos os elementos constituintes de tipo volume, ou que resultem em elementos constituintes de tipo volume. Conforme proposto por Shah [28], é possível imaginar que o Modelador DbF é uma casca onde um Sistema de Modelagem de Sólidos é o núcleo (vide figura 4.2).

A figura 4.3 ilustra como os elementos constituintes associados a um nó foram criados pela combinação de elementos constituintes associados a nós imediatamente inferiores.

4.2 Parâmetros

De maneira semelhante aos elementos constituintes, conforme ilustrado na equação 4.4, o **conjunto de parâmetros** associado a um nó intermediário é a combinação de parâmetros associados aos nós imediatamente inferiores. Entretanto, para que esta combinação ocorra, primeiro copiamos os elementos constituintes de interesse para o nó superior. A partir de um conjunto de parâmetros é possível definir novos parâmetros através de expressões matemáticas.

É possível criar novos parâmetros associados às **dimensões relativas** que relacionam o nó superior aos nós inferiores (vide figura 4.5). A figura 4.4 ilustra como os parâmetros associados a um nó foram criados pela combinação de parâmetros associados a nós inferiores.

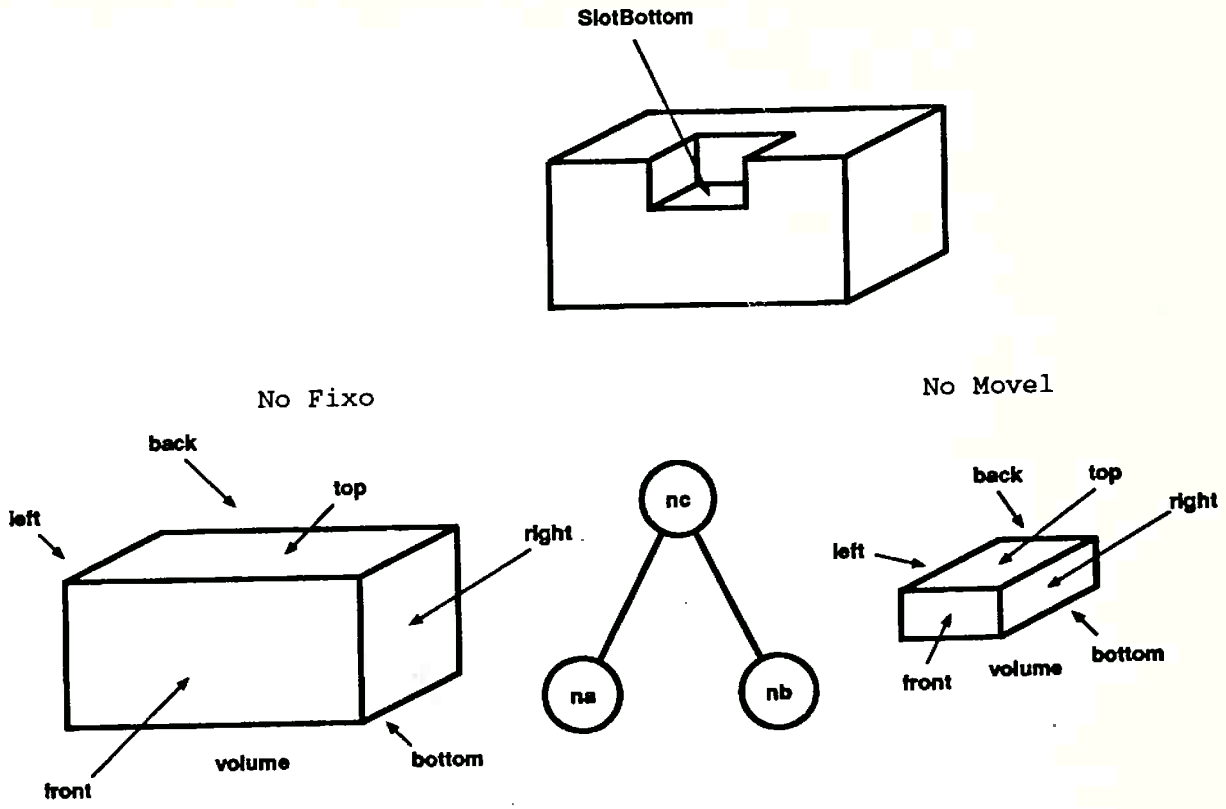


Figura 4.3: Exemplo de combinação de elementos constituintes.

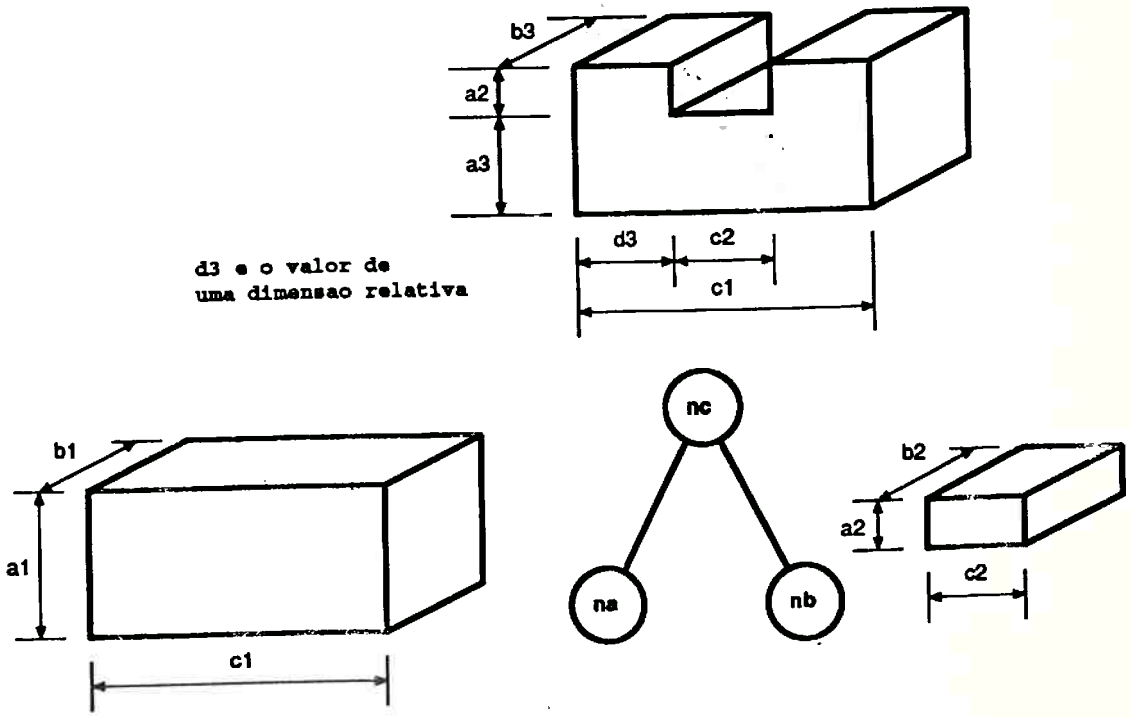


Figura 4.4: Exemplo de combinação de parâmetros.

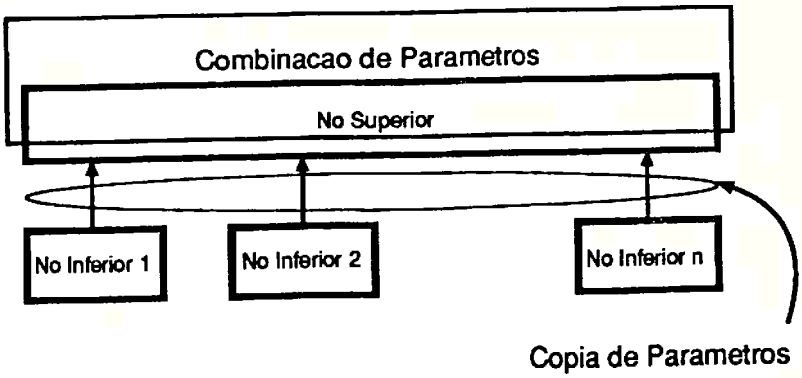


Figura 4.5: Cópia e combinação de parâmetros.

4.3 Representação de "Features"

As **montagens** são associadas ao **Modelo do Produto** através de dimensões relativas. As **peças** são associadas às **montagens** da mesma maneira. Entretanto, "features" são associadas às peças segundo um mecanismo distinto: o usuário fornece os valores dos parâmetros e informa alguns elementos constituintes primitivos presentes na peça. Estes elementos constituintes primitivos servirão de referência para posicionar corretamente a "feature". Chamaremos estes elementos constituintes por **encaixes** da "feature". Outra diferença é que algumas "features" alteram a forma da peça de modo a remover parte de seu volume ou acrescentar volume à peça.

4.3.1 Encaixes das "Feature"

O objetivo dos encaixes das "features" é facilitar o seu posicionamento em uma peça. Para implementar esta facilidade associaremos à "feature" uma dimensão relativa de tipo específico e, previamente, seleciona-se qual elemento constituinte da "feature" deve ser associado à dimensão relativa. Desta maneira, o usuário ao posicionar uma "feature" deverá informar apenas os elementos constituintes de encaixe que pertencem à peça.

4.3.2 Volumes Positivo e Negativo das "Features"

Os volumes positivo e negativo devem ser definidos pois algumas "features" alteram a forma da peça de modo a remover ou acrescentar volume à peça. Desta maneira, os volumes positivos e volumes negativos são elementos constituintes de tipo volume com atributos especiais que os qualificam, respectivamente, em positivos e negativos. Eles podem ser implementados, respectivamente, pelas funções de união e diferença.

4.4 Algoritmo para Desenvolver o Modelo do Produto

Desenvolver o Modelo do Produto implica em avaliar o valor de todos os parâmetros, satisfazer todas as dimensões relativas e determinar todos os elementos constituintes. É importante observar que, apesar das diferenças existentes entre as "features" e os outros elementos, todos os nós da árvore associada ao Modelo do Produto são idênticos. Isto nos permite definir um algoritmo unificado para toda a estrutura.

O algoritmo que desenvolve o **Modelo do Produto**, inicialmente determina o valor de todos os parâmetros da árvore no sentido de cima para baixo. Em seguida, o algoritmo satisfaz todas as dimensões relativas e determina todos os elementos constituintes no sentido inverso, de baixo para cima (vide figura 4.6).

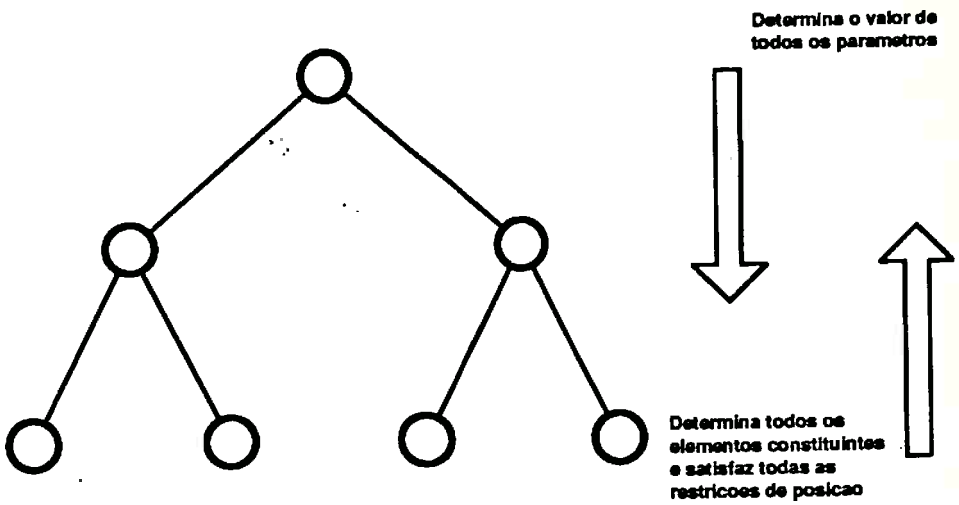


Figura 4.6: Algoritmo para desenvolver o Modelo do Produto.

Capítulo 5

Implementação do Protótipo

Neste capítulo descreveremos como foi realizada a implementação do protótipo. Dividimos em duas partes: estrutura de dados e algoritmos. Descreveremos a estrutura de dados da árvore do Modelo do Produto e da árvore de solução. Na seção de algoritmos concentraremos na determinação da intersecção entre conjuntos de transformação de simetria e na determinação das transformações que satisfazem um conjunto de dimensões relativas. Terminaremos este capítulo ilustrando alguns exemplos executados pelo protótipo. Para a implementação do protótipo, utilizamos a linguagem C.

5.1 Estrutura de Dados

5.1.1 Modelo do Produto

O **Modelo do Produto** é representado por uma estrutura em árvore. Cada nó da árvore possui os ponteiros para implementar a estrutura da árvore, lista de parâmetros, lista de elementos constituintes, lista das operações de cópia de parâmetros, lista das operações de cópia de elementos constituintes, lista de operações de combinação de parâmetros, lista de operações de combinação de elementos constituintes, lista de dimensões relativas, ponteiro para a árvore de solução, conjunto de transformações de simetria e

transformação que satisfaz o conjunto de dimensões relativas (vide listagem abaixo) [1].

```
struct CFGtree {
   CFGType *pai ;                    /* Ponteiro para a celula superior */
   CFGType *esq ;          /* Ponteiro para a celula inferior mais a esquerda */
   CFGType *dir ;                 /* Ponteiro para a celula lateral a direita */
   ELCType *elc ;        /* Ponteiro para a lista de elementos constituintes */
   PARType *par ;                    /* Ponteiro para a lista de parametros */
   CPHType *cph ;                 /* Ponteiro para a lista de associacoes com */
                                        /* parametros de nos inferiores */
   CEHType *ceh ;                 /* Ponteiro para a lista de associacoes com */
                                 /* elementos constituintes de nos inferiores */
   CEEType *cee ;                      /* Ponteiro para lista de combinacao de */
                                          /* elementos constituintes do no */
   CPPType *cpp ;                       /* ponteiro para lista de combinacao */
                                                /* de parametros do no */
   RPSType *rps ;              /* ponteiro para a lista de dimensoes relativas */
   NTPType *ntp ;                       /* ponteiro para a arvore de solucao */
   SIMType  sim ;                    /* conjunto de transformacoes de simetria */
   MATType  mat ;                              /* matriz de transformacao */
} ;
```

Cada elemento (**Modelo do Produto, Montagem, Peças e "Features"**) além da estrutura acima possuem as suas respectivas listas ligadas. As "features", entretanto, possuem algumas informações adicionais: ponteiro para a lista de volumes positivos, ponteiro para a lista de volumes negativos, ponteiro para a lista de encaixes e ponteiro para a lista de parâmetros.

```
struct FEATYPE {
   FEAType *nxt ;          /* Ponteiro para a proxima feature na lista ligada */
   FEAType *prv ;         /* Ponteiro para a feature anterior na lista ligada */
   CFGType  cfg ;                 /* Informacoes comuns a todos os elementos */
   VNGType *vng ;              /* Ponteiro para a lista de volumes negativos */
   VPSType *vps ;              /* Ponteiro para a lista de volumes positivos */
   EENType *een ;          /* Ponteiro para a lista de elementos contituintes */
                                                        /* de encaixe */
   PENType *pen ;           /* Ponteiro para a lista de parametros de encaixe */
} ;
```

5.1.2 Árvore de Solução

Para implementar a árvore de solução é necessário especificar dois tipos de nós: os nós associados aos **níveis pares** e os nós associados aos **níveis ímpares** (conforme figura 2.6). Chamaremos os nós associados aos níveis ímpares por ***nós de dimensão***

[1] As listagens não possuem acentos pois foram obtidas diretamente dos programas em C.

relativa e os nós associados aos níveis pares por *nós de solução.* Os nós associados aos níveis ímpares devem armazenar o conjunto de transformações que satisfazem a dimensão relativa associada ao nó e que pertença ao conjunto de graus de liberdade associado ao nó.

```
struct NodeRPSType {
   NSLType *esq ;          /* ponteiro para o no solucao mais a esquerda */
   NSLType *pai ;                /* ponteiro para o no solucao superior */
   NRType *rps ;   /* ponteiro para a dimensao relativa a ser satisfeita */
   MATType *mat ;          /* ponteiro para a lista de transformacoes que */
                    /* satisfazem a dimensao relativa associada ao no. */
} ;
```

Os nós de solução devem armazenar todas as informações associadas ao passo atual do algoritmo.

```
struct NodeSOLType {
   NRPType *pai ;        /* ponteiro para o no dimensao relativa superior */
   NRPType *esq ;          /* ponteiro para o no dimensao relativa abaixo */
   NSLType *dir ;         /* ponteiro para o no solucao lateral a direita */
   NRType *rps ;           /* ponteiro para lista de dimensoes relativas */
   NELType *elc ;       /* ponteiro para lista de elementos constituintes */
   NGRType  grlmova ;     /* conjunto de graus de liberdade associado ao  */
                            /* no inferior relacionado ao passo anterior. */
   NGRType  grlfixa ;     /* conjunto de graus de liberdade associado ao  */
                            /* no superior relacionado ao passo anterior. */
   NGRType  grlmovn ;      /* conjunto de graus de liberdade associado ao */
                               /* no inferior relacionado ao passo atual. */
   NGRType  grlfixn ;      /* conjunto de graus de liberdade associado ao */
                               /* no superior relacionado ao passo atual. */
   NGRType  simmov ;            /* conjunto de transformacoes de simetria */
                                          /* associado ao no inferior. */
   NGRType  simfix ;            /* conjunto de transformacoes de simetria */
                                          /* associado ao no superior. */
   matrix mat ;         /* matriz de transformacao que satisfaz a dimensao */
                      /* relativa associada ao no dimensao relativa acima. */
} ;
```

5.2 Satisfazendo uma Dimensão Relativa

5.2.1 Determinando a Intersecção entre Conjuntos de Graus de Liberdade

Para a determinação da intersecção entre os conjuntos de liberdade, inicialmente, analisa-se separadamente cada tipo de dimensão relativa e, posteriormente analisa-se se-

paradamente cada tipos de conjunto de graus de liberdade. Como existem quatro tipos de conjuntos de graus de liberdade associados às dimensões relativas (vide tabela 2.2 [2]) e existem dez tipos de conjuntos de graus de liberdade que podem surgir devido à intersecção entre os vários conjuntos de graus de liberdade (vide tabelas 2.3 e 2.4), obtemos um total de $4 \cdot 10 = 40$ casos particulares. Este número pode ser reduzido à metade, pois existe uma simetria. A rotina RPS determinaGrauLiberdadeAtual analisa separadamente cada tipo de dimensão relativa e a rotina RPS verificaPlanoGrauLiberdade analisa separadamente cada tipo de conjunto de graus de liberdade para o caso de um elemento constituinte de tipo plano. O algoritmo está apresentado abaixo resumidamente.

```
int RPS_determinaGrauLiberdadeAtual(NSLType *nsl)
{
   switch (nsl->up->rps->tipo) {
      case DISNC:
      case DISNO:
         /* verifica as tansformacoes de simetria tipo plano fixo */
         if (RPS_verificaPlanoGrauLiberdade(nsl, 1) == GR_ERR)
            return(FALSE) ;
         /* verifica as tansformacoes de simetria tipo plano movel */
         if (RPS_verificaPlanoGrauLiberdade(nsl, 0) == GR_ERR)
            return(FALSE) ;
         break ;
      case DISRP:
         /* verifica as tansformacoes de simetria tipo linha fixa */
         if (RPS_verificaRetaGrauLiberdade(nsl, 1) == GR_ERR)
            return(FALSE) ;
         /* verifica as tansformacoes de simetria tipo plano movel */
         if (RPS_verificaPlanoGrauLiberdade(nsl, 0) == GR_ERR)
            return(FALSE) ;
         break ;
      case ... :
      default:
         return(FALSE) ;
   }
   return(TRUE) ;
}
TGRType RPS_verificaPlanoGrauLiberdade(NSLType *nsl, int flag)
{
   switch (grl_old->tipo) {
      case GR_TOT:
         return(GR_PLN) ;
      case GR_TRN:
         /* verifica se a normal do plano e perpendicular a direcao de   */
         /*  translacao. Neste caso acrescenta a translacao.             */
         if (RPS_verificaReto(pln, grl_old->p1))
            return(GR_TRN) ;
         return(GR_EMP) ;
      case GR_PLN:
         /* verifica se ambos os planos possuem as mesmas equacoes, em   */
         /*  case positivo, acrescentar o grau de liberdade de tipo plano*/
```

[2]Observe que $PLN * 3DT$ e $RET * 3DT$ representam o mesmo conjunto de graus de liberdade.

```
        if (RPS_verificaDirecao(pln, grl_old->p1))
           return(GR_PLN) ;
        return(GR_TRN) ;
     case ... :
  }
  return(GR_ERR) ;
}
```

5.2.2 Satisfazendo um Dimensão Relativa Específica

Dada uma dimensão específica, analisamos separadamente como satisfazê-la segundo os vários conjuntos de graus de liberdade possíveis. Como possuímos os conjuntos de graus de liberdade associados aos níveis superior e inferior então existem um total de $10 \cdot 10 = 100$ casos possíveis para serem analisados para cada dimensão relativa. Abaixo está a listagem resumida para determinar a transformação que satisfaz a dimensão relativa distância plano-plano.

```
int RPS_solucionaDISNC(NSLType *nsl)
{
   assert(nsl->down->rps->elcmov->tipo == PLANO) ;  // verifica se e um plano
   assert(nsl->down->rps->elcfix->tipo == PLANO) ;  // verifica se e um plano
   switch(nsl->grlmovn.tipo) {
      case GR_TOT:
         switch (nsl->grlfixn.tipo) {
            case GR_TOT:       // conjunto de graus de liberdade { TOT , TOT }
               ...             // soluciona a dimensao relativa DISNC
            default:           // conjunto de graus de liberdade { TOT , *** }
               break ;         // nao existe
         }
         break;
      case GR_PLN:
         switch (nsl->grlfixn.tipo) {
            case GR_PLN:       // conjunto de graus de liberdade { PLN , PLN }
               ...             // soluciona a dimensao relativa DISNC
            case ... :         // outros conjuntos de graus de liberdade
                               // soluciona a dimensao relativa DISNC
            default:           // conjunto de graus de liberdade { PLN , *** }
               break ;         // nao existe
         }
         break ;
      case ...:
      case GR_TRN:
         switch (nsl->grlfixn.tipo) {
            case GR_TRN:       // conjunto de graus de liberdade { TRN , TRN }
                               // soluciona a dimensao relativa DISNC
            case ... :         // outros conjuntos de graus de liberdade
                               // soluciona a dimensao relativa DISNC
            default:           // conjunto de graus de liberdade { TRN , *** }
               break ;         // nao existe
         }
         break ;
      case GR_EMP:
         break ;
```

```
    }
    return(TRUE) ;
}
```

5.3 Exemplo Utilizando o Protótipo

Para testar o protótipo foi desenvolvida uma linguagem que permite manipular a estrutura de dados do **Modelo do Produto** e determinar o Modelo do Produto desenvolvido a partir de um conjunto de parâmetros. Como resultado do teste, gera-se um arquivo de comandos para o Modelador de Sólidos MSD que também foi desenvolvido [31]. Desta maneira foi possível criar exemplos e, posteriormente, exibi-los graficamente através do Modelador de Sólidos MSD.

Para ilustrarmos este trabalho foi feito um exemplo de uma peça com um rasgo e um furo cilíndrico passante. O rasgo foi associado à peça por dimensões relativas do tipo distância plano-plano e o furo passante foi associado à peça por dimensões relativas do tipo distância linha-plano e distância plano-plano. O anexo C contém os comandos utilizados para criar o exemplo. O anexo D contém os comandos do Modelador MSD que foram gerados a partir de um conjunto de parâmetros fornecidos ao exemplo. As figuras 5.1, 5.2 e 5.3 ilustram o exemplo com diferentes valores atribuídos a algumas dimensões relativas.

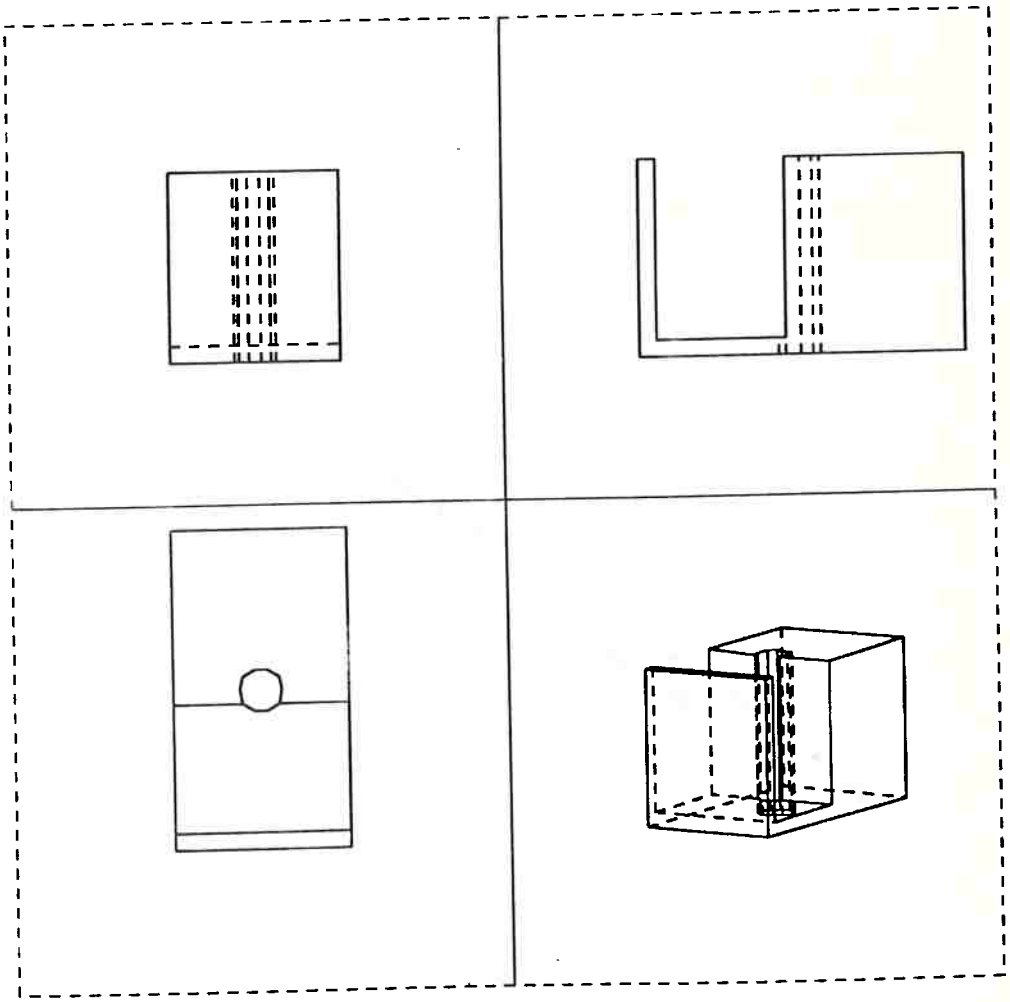


Figura 5.1: Exemplo de utilização do protótipo - I.

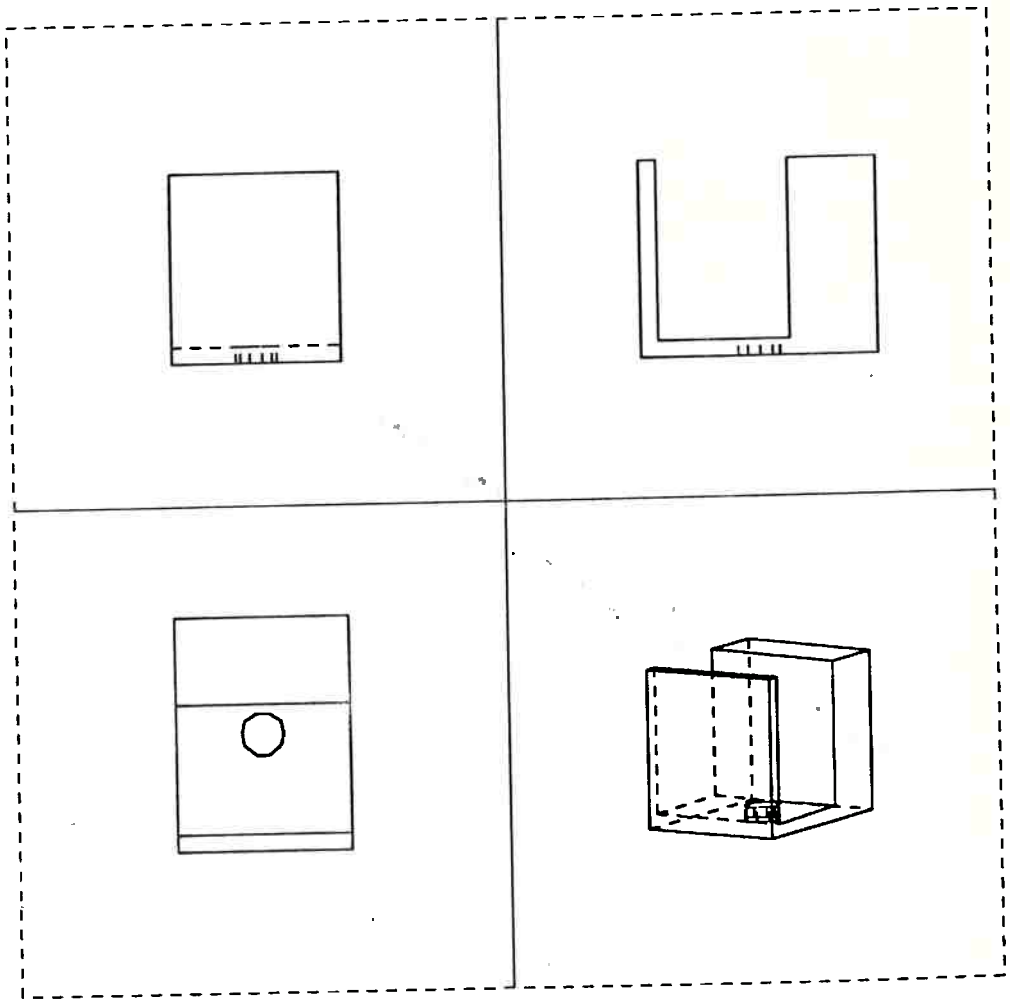


Figura 5.2: Exemplo de utilização do protótipo - II.

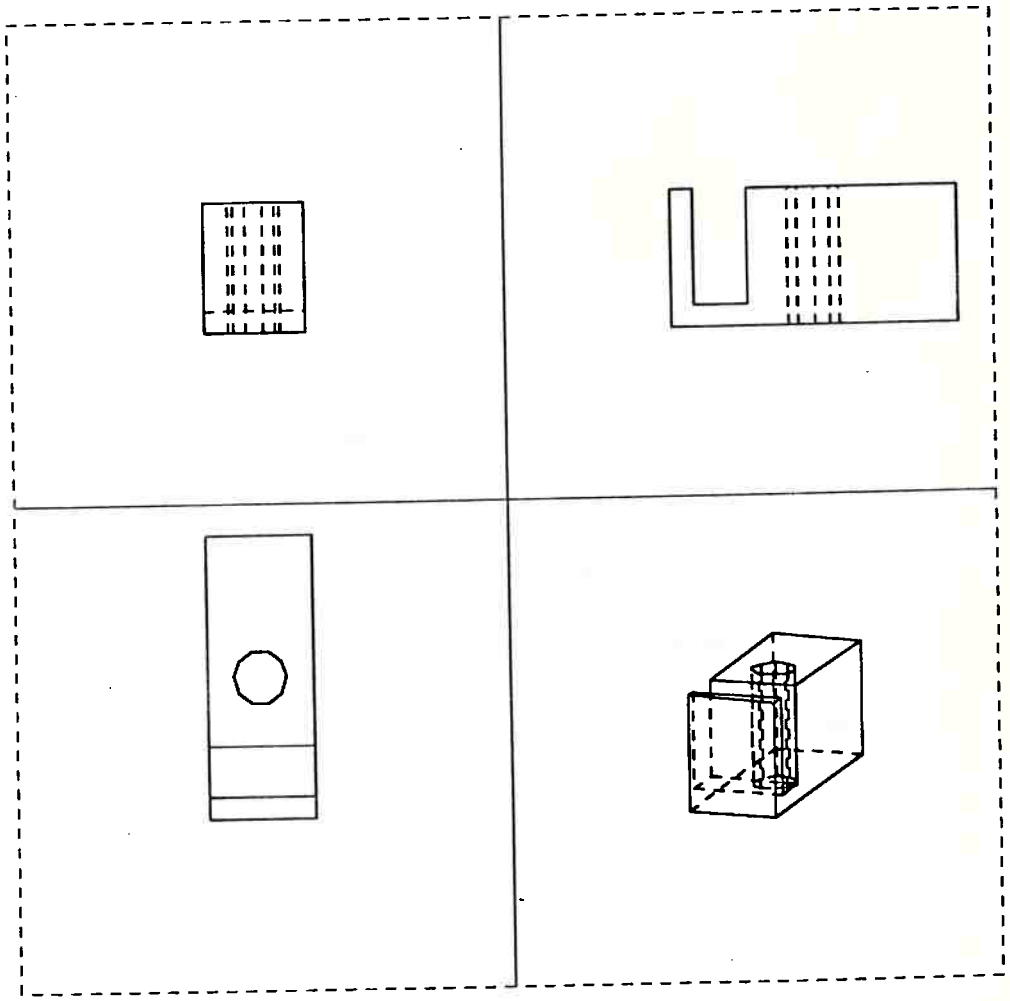


Figura 5.3: Exemplo de utilização do protótipo - III.

Capítulo 6

Conclusões

Neste trabalho introduzimos um conceito que consideramos fundamental: **dimensões relativas**. Para implementarmos as dimensões relativas, conforme discutido no capítulo 3, é importante que elementos geométricos extras (ponto, linha e plano) sejam acrescentados à representação B-Rep. Estes elementos geométricos extras podem inclusive não estar sobre o contorno do Modelo do Produto. Independente disso, foi concebida uma estrutura hierárquica uniforme para a representação do Modelo do Produto que é uma aplicação direta do conceito de dimensões relativas. O conceito de dimensões relativas também permite representar corretamente **dimensões** e os **encaixes das "features"**.

Ao acrescentarmos elementos geométricos (ponto, linha e plano) extras à representação B-Rep é possível implementar as dimensões relativas. Entretanto, se considerarmos a possibilidade de criarmos elementos geométricos mais complexos, então será possível suportar simultaneamente a representação de **"features" de superfície** e de **"features" volumétricas**. A "feature" representada desta maneira é a "feature" intencional explícita. Além disto, suportando a *representação de "features" intencionais explícitas* e suportando a *representação de "features" intencionais explícitas associadas ao Modelo do Produto* é possível suportar **exames de validação do estado da "feature"**. Além disso, ao acrescentarmos elementos geométricos (linha e plano) extras à representação B-Rep é possível representar também **planos e eixos de simetria**.

Nesta técnica é possível criar tanto "features" simples como "features" complexas. Portanto o usuário poderá definir as suas "features" voltadas a sua aplicação específica. Esta é uma estrutura elegante que permite representar uniformemente o Modelo do Produto.

Desta maneira concluímos que o conceito de dimensões relativas e a idéia de acrescentar elementos geométricos extras à representação B-Rep é efetiva para a representação de "features" permitindo que vários outros conceitos se encaixem harmoniosamente. A seguir apresentamos alguns possíveis trabalhos futuros.

6.1 Trabalhos Futuros

6.1.1 Estudo de Inferências para Busca de Transformações que Satisfaçam um Conjunto de Dimensões Relativas

Um sistema com uma implementação completa do algoritmo que busca um conjunto de transformações que satisfaçam um conjunto de dimensões relativas levaria ao estudo de no máximo $10 \cdot 10 = 100$ casos por cada tipo de dimensão relativa. Entretanto, este número de casos pode ser reduzido pela eliminação de casos impossíveis e por agrupamento em casos semelhantes. Um estudo deste tipo foge ao escopo deste trabalho, apesar de ser uma conseqüência imediata dele.

6.1.2 Condições de Validade

As condições de validade possuem a função de verificar se os parâmetros especificados para uma "feature" estão corretos e verificar se uma "feature" está sendo encaixada corretamente. Devido à capacidade de representarmos consistentemente os elementos constituintes das "features" é possível especificar um Modelo Condicional genérico.

Por exemplo, uma característica das "features" é que cada elemento constituinte está associado a elementos constituintes que estão presentes na representação desenvolvida do

Modelo do Produto. Entretanto, alguns elementos constituintes podem não ter nenhum correspondente, pois o elemento correspondente na representação desenvolvida pode ter sido removido por alguma operação. Por outro lado, um elemento constituinte pode possuir vários correspondentes no Modelo do Produto, pois o elemento constituinte pode ter sido dividido por alguma operação (vide figura 1.5).

6.1.3 Definição de uma Representação para os Elementos Constituintes

É necessário definir uma representação completa para os elementos constituintes das "features". Segundo esta análise, esta representação deverá suportar pelo menos a representação de pontos, linhas, planos e volumes.

Apêndice A

Projeto Paramétrico

Um projeto paramétrico representa uma família de projetos que possuem as mesmas restrições topológicas e geométricas. Vários métodos foram propostos para modelagem paramétrica em CAD. Os métodos podem ser classificados em quatro grupos: programação variante [1], sistema de equações de restrição [2], avaliação baseada em regras [3] e métodos construtivos [4].

Todos os métodos possuem alguns princípios comuns. O usuário deve, inicialmente, definir a família do projeto, que corresponde a fornecer a topologia e o esquema de dimensionamento associados ao objeto. A topologia descreve a conectividade dos primitivos geométricos, enquanto que o esquema de dimensionamento especifica as restrições do projeto. Estas informações devem ser fornecidas apenas no estágio inicial. Esta etapa também é chamada de projeto primário. Um elemento da família de projetos que também é chamado por **variação do projeto**, é criado por métodos de análise que se baseiam na descrição da topologia, no esquema de dimensionamento e nos valores das dimensões. Os vários métodos de projeto paramétrico se diferenciam na forma de descrição da topo-

[1]Tradução do termo em inglês "Variants Programming".

[2]Tradução do termo em inglês "Sistem of Constraint Equations".

[3]Tradução do termo em inglês "Rule Based Variation Evaluation".

[4]Tradução do termo em inglês "Constructive Approach".

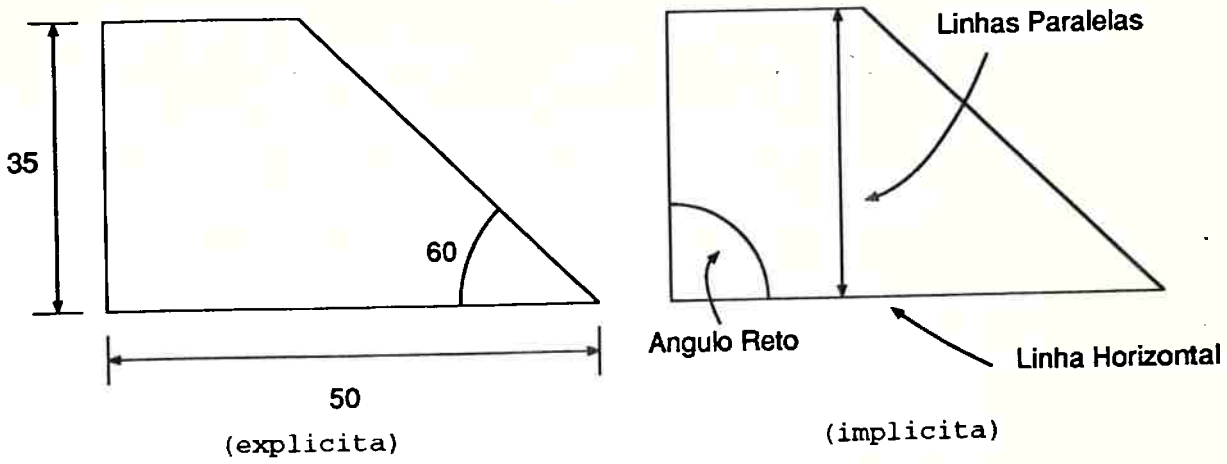


Figura A.1: Exemplo de restrições explícitas e implícitas.

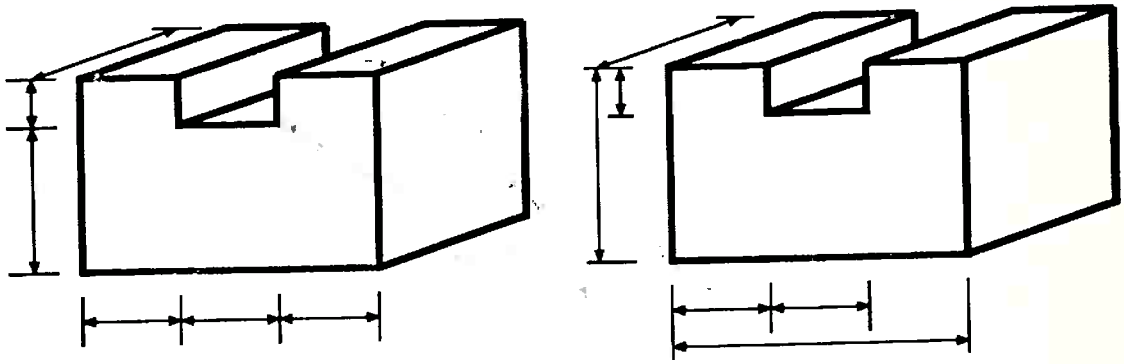


Figura A.2: A mesma forma segundo esquemas de dimensionamento distintos.

logia, no esquema de dimensionamento e nos métodos de análise utilizados para criar as variações do projeto.

Usualmente, nem todas as restrições de dimensão da geometria são especificadas explicitamente nos desenhos técnicos e projetos. Exemplos de restrições que não estão incluídas no esquema de dimensionamento do desenho técnico são: linhas paralelas, linhas tangentes, linhas colineares, linhas ortogonais, linhas horizontais, entre outros. Estas restrições são denominadas por restrições implícitas. A figura A.1 ilustra algumas restrições explícitas e algumas restrições implícitas. Ambos os tipos de restrições devem estar explicitamente representadas internamente a um sistema de CAD paramétrico.

Através de um sistema de Modelagem de Sólido convencional o projetista não trans-

mite as suas intenções, mas apenas o resultado final de suas considerações: o produto final. Isto é penalizante para outras pessoas que necessitam entender as funcionalidades de uma peça. Particularmente, a semântica das dimensões incorporam considerações importantes realizadas pelo projetista. O projetista deve decidir como a forma deverá ser dimensionada de acordo com suas análises de projeto e manufatura (vide figura A.2).

Nas próximas seções apresentaremos um método de programação e três métodos diferentes para modelagem paramétrica.

A.1 Programação Variante

Programação variante é o método de modelagem paramétrica mais difundido, e é muito utilizado para representar famílias de peças e elementos em bibliotecas de padrões. Este método é baseado em uma linguagem de programação que deve estar acoplada a algum sistema de Modelagem de Sólidos [12, 19].

Neste método, o projeto primário é criado ao se escrever o programa que define a família do projeto. Inicialmente, parâmetros de entrada devem ser fornecidos, a execução do programa acionará funções do sistema de Modelagem de Sólidos e a variação do projeto é criada segundo a representação do sistema adotado.

A.2 Sistema de Equações de Restrições

Este método é baseado em traduzir todas as restrições (implícitas e explícitas) em equações, onde as incógnitas são os pontos característicos do modelo geométrico. A variação do projeto é determinada pela solução do sistema de equações por um método numérico iterativo [17].

Este é um método genérico e poderoso que pode suportar modelos variacionais tri-dimensionais e bi-dimensionais. De fato, todas as restrições podem ser expressas por equações. Se uma nova restrição é acrescentada pelo usuário, apenas a correspondente

equação de restrição deve ser acrescentada ao sistema de equações. Situações com restrições cíclicas, onde uma restrição depende de restrições que dependem de si próprias, podem ser solucionados devido a solução simultânea das equações.

Uma limitação do método é o caso em que ocorre múltiplas soluções, pois neste caso apenas uma delas será encontrada. Além disso, não existe uma representação única para a solução desejada, pois a solução depende dos valores iniciais fornecidos ao método numérico iterativo. Também, é conveniente recordar que métodos numéricos podem ter um alto custo computacional.

A.3 Avaliação da Variação por meio de Regras

Mais recentemente, métodos baseados em inteligência artificial foram desenvolvidos para determinar a variação do projeto a partir de um conjunto de restrições (implícitas e explícitas). Esta proposta utiliza sistemas especialistas para propagar seqüencialmente as coordenadas dos pontos que podem ser determinadas a partir de restrições fornecidas e das coordenadas conhecidas. Neste caso o projeto primário é criado pela tradução do conjunto de restrições em regras. Entretanto, em situações de restrições cíclicas, que podem ocorrer na prática, o processo de dedução resultará em um laço infinito e nenhuma solução será determinada. Existem poucas experiências práticas publicadas, particularmente no caso de projetos em três dimensões [15, 30].

A.4 Proposta Construtiva

Neste caso, a variação do projeto é determinada analisando-se a semântica de comandos. Este método é capaz de capturar o procedimento pelo qual o projetista traduz o seu modelo funcional mental e cria um modelo procedural computacional equivalente. Esta equivalência entre os dois modelos (funcional mental e procedural computacional) é baseada na hipótese de que o procedimento de trabalho adotado pelo projetista não

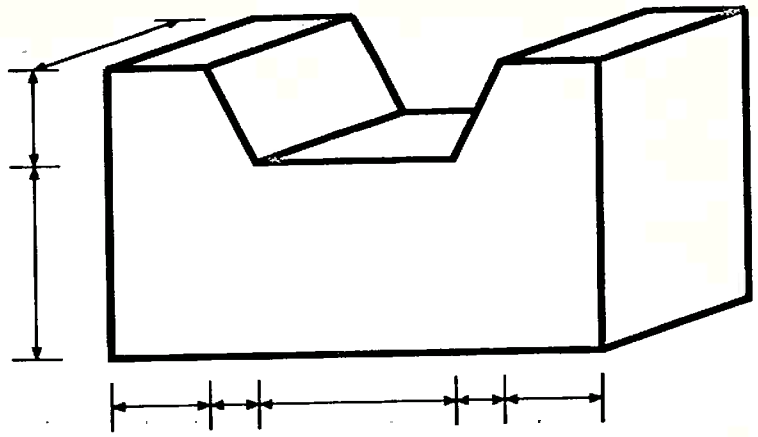


Figura A.3: Exemplo de "feature" paramétrica.

é casual, mas indiretamente reflete a conexão funcional entre as partes do modelo do produto.

Cugini [5] desenvolveu um sistema de CAD de duas dimensões que armazena os comandos aplicados pelo projetista em um grafo onde os nós correspondem a elementos primitivos como linhas, círculos e pontos; e as arestas correspondem a relações topológicas particulares relacionando dois elementos primitivos. Roller [26] propôs um método similar, também para desenhos em duas dimensões. O projetista pode aplicar comandos de acordo com três modos diferentes de operação: fixo, variável e flexível. No modo FIXO, o sistema criará um elemento com dimensões fixas. O modo VARIÁVEL associa variáveis às dimensões dos elementos criados e quando o projetista não conhece a dimensão de um elemento o modo FLEXÍVEL deve ser utilizado.

Gossard [8] desenvolveu um sistema que permite representar dimensões explicitamente em sólidos tridimensionais utilizando o operador chamado *operador de posição relativa* (operador *rpo*). Este sistema é baseado em um método que combina as representações CSG e B-Rep em um grafo que foi chamado por *object graph*. Entretanto, para representar a parametrização de um sólido como o ilustrado na figura A.3 é necessário representar explicitamente arestas em um grafo do objeto ou definir um operador *rpo* mais complexo. Eles não explicam como este tipo de parametrização deve ser representado e

segundo o nosso entender não deverá ser fácil.

Apêndice B

Satisfazendo Dimensões Relativas

Para apresentarmos o algoritmo que busca o conjunto de transformações que satisfazem um conjunto de dimensões relativas será necessário definirmos algumas funções que manipulam a árvore de soluções. Estas funções podem ser divididas em dois grupos: funções associadas à construção da árvore de solução e funções que recuperam informações da árvore de soluções.

A árvore de soluções é criada segundo um sentido de **montagem** e um sentido de **busca**.O sentido de montagem é de cima para baixo. Sempre partindo do nó mais à esquerda (vide figura B.1, ciclo representado por $p1$). As funções que representam este sentido são NewD e NewS. O sentido de busca é da esquerda para a direita, ao atingir o extremo à direita, realiza-se a mesma busca um nó acima até encontrarmos uma solução que não seja pai de nenhum nó (vide figura B.1). A função que representa este ciclo é $p2$.

A simbologia utilizada pelo algoritmo está descrita abaixo:

- Δ': conjunto de dimensões relativas satisfeitas;
- k: índice do nível que está sendo processado;
- φ_k: conjunto de graus de liberdade associado ao nível k;
- φ'_k: conjunto de graus de liberdade associado ao nível k transformado;

- χ_k: conjunto de graus de simetria associado ao nível k;
- χ'_k: conjunto de graus de simetria associado ao nível k transformado;
- **fim**: variável que indica o final do algoritmo;
- d_k: k-ésima dimensão relativa;
- ς_S^k: subconjunto de graus de simetria associado ao nó superior da árvore do Modelo do Produto do k-ésimo nível da árvore de solução;
- ς_I^k: subconjunto de graus de simetria associado ao nó inferior da árvore do Modelo do Produto do k-ésimo nível da árvore de solução;
- τ_S^k: subconjunto de graus de liberdade associado ao nível superior da árvore do Modelo do Produto do k-ésimo nível da árvore de solução;
- τ_I^k: subconjunto de graus de liberdade associado ao nível inferior da árvore do Modelo do Produto do k-ésimo nível da árvore de solução;
- nd: nó associado à dimensão relativa que está sendo processada;
- n: número de transformações que satisfazem a dimensão relativa d_k segundo o conjunto de graus de liberdade φ_k;
- $T_{\Delta',i}^{d_k}$: i-ésima transformação que satisfaz a dimensão relativa d_k e mantém satisfeitas todas as dimensões relativas do conjunto Δ';
- $\{d_1, \cdots, d_n\}$: conjunto de dimensões relativas;
- $\{d'_1, \cdots, d'_n\}$: conjunto de dimensões relativas transformadas;
- $GL(d_k)$: conjunto de graus de liberdade associado à dimensão relativa d_k;
- ns: nó associado à transformação que satisfaz a k-ésima dimensão relativa.

A função Solve encontra todas as transformações que satisfazem uma dada dimensão relativa obedecendo a um conjunto de graus de liberdade. As funções GetTransform, GetLevel, GetLevel, Getφ, Getχ, GetΔ, GetΔ' são funções que recuperam informações

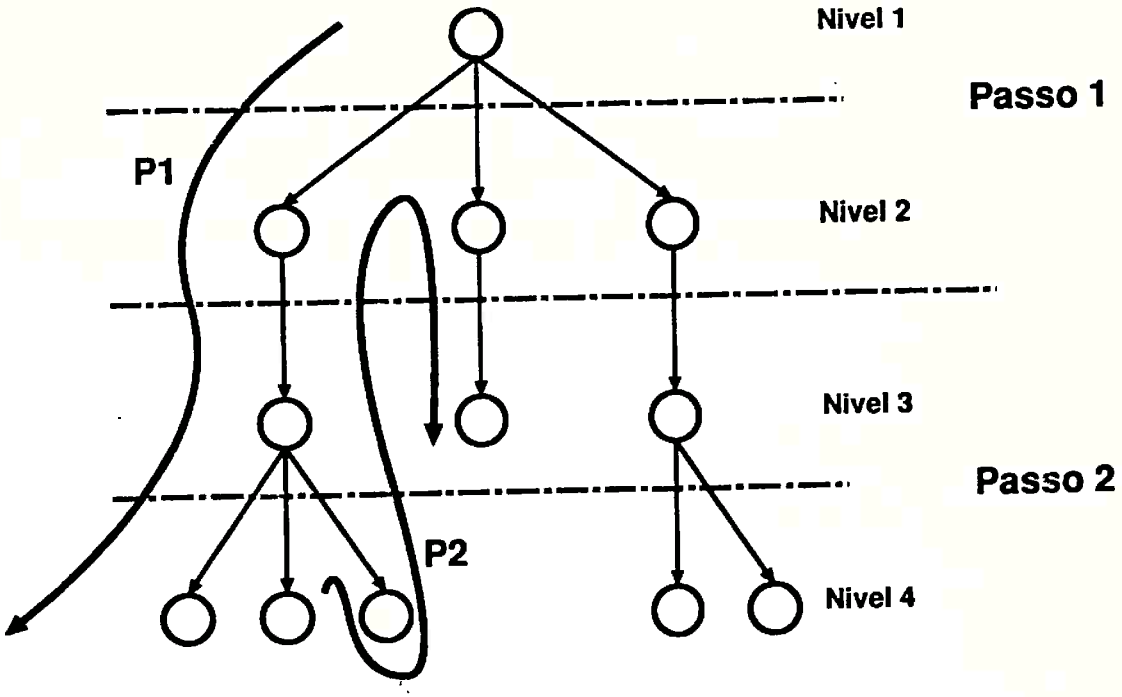


Figura B.1: Sentidos de busca e montagem na árvore de solução.

da árvore de solução. O significado detalhado de cada função está explicado no final do apêndice.

$\Delta' \leftarrow \{\}$

$\varphi_1 \leftarrow \{\tau_S^1 \leftarrow TOT, \tau_I^1 \leftarrow TOT\}$

$\chi_1 \leftarrow \{\varsigma_S^1, \varsigma_I^1\}$

$k \leftarrow 1$

$fim \leftarrow$ **FALSE**

$ns \leftarrow$ **null**

while(!fim){

if(d_k! = **nul**){

if($\varsigma_S^k \cdot \varsigma_I^k \not\subset \tau_S^k \cdot \tau_I^k$){

$nd \leftarrow NewD(ns, d_k, \varphi_k)$

if($(n \leftarrow Solve(nd, d_k, \varphi_k))! = 0$){

for($i = 0; i < n; i++$){

$T^{d_k}_{\Delta',i} \leftarrow GetTransform(nd, i)$

$\varphi'_k \leftarrow T^{d_k}_{\Delta',i} \cdot \varphi_k$

$\chi'_k \leftarrow T^{d_k}_{\Delta',i} \cdot \chi_k$

$\{d'_1, \cdots, d'_n\} \leftarrow T^{d_k}_{\Delta',i} \cdot \{d_1, \cdots, d_n\}$

$\varphi_{k+1} \leftarrow \varphi'_k \cap GL(d_k)$

if($i == 0$)

$ns \leftarrow NewS(nd, \varphi_{k+1}, \chi'_k, \{d'_1, \cdots, d'_n\}, T^{d_k}_{\Delta',i})$

else

$NewS(nd, \varphi_{k+1}, \chi'_k, \{d'_1, \cdots, d'_n\}, T^{d_k}_{\Delta',i})$

}

$k \leftarrow k + 1$

$\varphi_k \leftarrow \varphi'_k$

$\chi_k \leftarrow \chi'_k$

$\{d_1, \cdots, d_n\} \leftarrow \{d'_1, \cdots, d'_n\}$

$\Delta' \leftarrow \Delta' \cup \{d_k\}$

}

else{

if($(ns \leftarrow NextSol(ns))! =$ **nul**){

$k \leftarrow GetLevel(ns)$

$\varphi_k \leftarrow Get\varphi(ns)$

$\chi_k \leftarrow Get\chi(ns)$

```
                    {d_1, ..., d_n} ← GetΔ(ns)
                    Δ' ← GetΔ'(ns)
                }
                else
                    fim ← FALSE
            }
        }
        else{
            if((ns ← NextSol(ns))! = nul){
                k ← GetLevel(ns)
                φ_k ← Getφ(ns)
                χ_k ← Getχ(ns)
                {d_1, ..., d_n} ← GetΔ(ns)
                Δ' ← GetΔ'(ns)
            }
            else
                fim ← FALSE
        }
    }
    else{
        if((ns ← NextSol(ns))! = nul){
            k ← GetLevel(ns)
            φ_k ← Getφ(ns)
            χ_k ← Getχ(ns)
            {d_1, ..., d_n} ← GetΔ(ns)
            Δ' ← GetΔ'(ns)
        }
        else
            fim ← FALSE
    }
}
```

$nd \leftarrow \mathbf{NewD}(ns, d_k, \varphi_k)$, esta função cria um nó associado à dimensão relativa d_k com

conjunto de graus de liberdade φ_k. O novo nó nd é filho do nó ns.

$ns \leftarrow \mathbf{NewS}(nd, \varphi_{k+1}, \chi_{k+1}, \{d_1, \cdots, d_n\}, T^{d_k}_{\Delta',i})$, esta função cria um nó associado à transformação que satisfez a dimensão relativa d_k com os conjuntos χ_{k+1} e $\{d_1, \cdots, d_n\}$, φ_{k+1} já devidamente transformados por $T^{d_k}_{\Delta',i}$. O novo nó ns é filho do nó nd.

$T \leftarrow \mathbf{GetTranform}(nd, i)$, esta função recupera a i-ésima transformação associada ao nó nd.

$ns \leftarrow \mathbf{NextSol}(ns)$, esta função recupera o próximo nó solução segundo uma busca à direita e acima ciclicamente.

$k \leftarrow \mathbf{GetLevel}(ns)$, esta função recupera o nível associado ao nó ns.

$\varphi_i \leftarrow \mathbf{Get}\varphi(ns)$, esta função recupera o conjunto de graus de liberdade associado ao nó ns.

$\chi_i \leftarrow \mathbf{Get}\chi(ns)$, esta função recupera o conjunto de graus de simetria associado ao nó ns.

$\{d_i, \cdots, d_n\} \leftarrow \mathbf{Get}\Delta(ns)$, esta função recupera o conjunto de dimensões relativas associado ao nó ns.

$n \leftarrow \mathbf{Solve}(nd, d_k, \varphi_k)$, esta função encontra todas as transformações que pertençam a φ_k e que satisfaçam d_k. As transformações são associadas ao nó d_k.

$\Delta'_k \leftarrow \mathbf{Get}\Delta'(ns)$, esta função recupera o conjunto de dimensões relativas satisfeitas até o nó ns.

Apêndice C

Programa Exemplo do Protótipo

Para a criação do exemplo apresentado no capítulo 5 foi utilizada a linguagem do protótipo implementado . Os comandos utilizados para criar o exemplo estão explicados abaixo:

- crtcube nome: cria uma "feature" de tipo cubo com nome nome. O cubo possui três parâmetros X, Y e Z que representam, respectivamente, o comprimento, a largura e a altura do cubo. O cubo também possui seis elementos constituintes de tipo plano e um de tipo volume: plano superior, plano inferior, plano frontal, plano anterior, plano lateral direito, plano lateral esquerdo e volume do cubo. Cada elemento constituinte é representado, respectivamente, por: top, bottom, front, back, right, left e volume;
- crtcilinder nome: cria uma "feature" de tipo cilindro com nome nome. O cilindro possui dois parâmetros R e H que representam, respectivamente, o raio e a altura do cilindro. O cilindro também possui dois elementos constituintes de tipo plano, um de tipo linha e um de tipo volume: plano superior, plano inferior, eixo de simetria e volume do cilindro. Cada elemento constituinte é representado, respectivamente, por: top, botom, axis e volume;

- combina nome1 nome2 nome3: cria uma "feature" de nome nome3 que é a combinação das "features" nome2 e nome1. O nó associado à "feature" nome1 é fixa e o nó associado à "feature" nome2 é móvel;
- copyrparam nome1 nome2 nome3: copia o parâmetro nome2 que está no nó inferior à direita da "feature" nome1 e nomeia o parâmetro copiado como nome3;
- copylparam nome1 nome2 nome3: copia o parâmetro nome2 que está no nó inferior à esquerda da "feature" nome1 e nomeia o parâmetro copiado como nome3;
- copyrelem nome1 nome2 nome3: copia o elemento constituinte nome2 que está no nó inferior à direita da "feature" nome1 e nomeia o elemento constituinte copiado como nome3;
- copylelem nome1 nome2 nome3: copia o elemento constituinte nome2 que está no nó inferior à esquerda da "feature" nome1 e nomeia o elemento constituinte copiado como nome3;
- crtrestpos nome1 nome2 nome3 nome4 tipo: cria uma dimensão relativa relacionando os nós inferiores direito e esquerdo. A dimensão relativa associa o elemento constituinte nome2 do nó fixo e o elemento constituinte nome3 do nó móvel. O valor da dimensão relativa está associada ao parâmetro nome4 que é criado associado à "feature" nome1. A dimensão relativa é de tipo tipo;
- assexpmatpar nome1 nome2 exp: associa a expressão matemática exp ao parâmetro nome2 que está associado à "feature" nome1;
- manipelem nome1 nome2 nome3 nome4 op: define um novo elemento constituinte de nome nome4 a partir de dois outros elementos constituintes nome2 e nome3 que estão associados à "feature" nome1. O novo elemento constituinte é criado a partir da operação op;
- crtencaixe nome1 nome2: faz com que o parâmetro nome2 fique como parâmetro de encaixe da "feature" nome1;

- assencaixeval nome1 nome2 val: associa o valor val ao parâmetro de encaixe nome2 da "feature" nome1;
- proc nome: processa a "feature" nome e cria arquivo interface para o Modelador de Sólidos MSD [32, 31].

A listagem abaixo foi utilizada para criar o exemplo de utilização do protótipo apresentado no capíulo 5.

```
crtcube fea1
crtcube fea2
combina fea1 fea2 fea3
copyrparam fea3 X a1
copyrparam fea3 Y b1
copyrparam fea3 Z c1
copylparam fea3 X a0
copylparam fea3 Y b0
copylparam fea3 Z c0
copyrelem fea3 top top
copyrelem fea3 left left
copyrelem fea3 front front
copyrelem fea3 volume vol1
copylelem fea3 volume vol2
crtrestpos fea3 front back null1 DISPPNC
crtrestpos fea3 left right side DISPPNC
crtrestpos fea3 top top null2 DISPPNC
assexpmatpar fea3 b1 b0
assexpmatpar fea3 null1 0.0
assexpmatpar fea3 null2 0.0
assexpmatpar fea3 a1 a0+side+2.0
assexpmatpar fea3 c1 c0+2.0
manipelem fea3 vol1 vol2 vol3 diferenca
crtcilindro cil
combina fea3 cil peca
copylparam peca b0 b0
copylparam peca a0 a0
copylparam peca c0 c0
copylparam peca side side
copyrparam peca R raio
copyrparam peca H alt
crtrestpos peca top top null3 DISPPNC
crtrestpos peca left axis rb DISPR
crtrestpos peca front axis rf DISPR
assexpmatpar peca alt c0+2.0
assexpmatpar peca null3 0.0
assexpmatpar peca rb    (side+a0+2.0)*(-0.5)
assexpmatpar peca rf    b0*(-0.5)
crtencaixe peca b0
crtencaixe peca a0
crtencaixe peca c0
crtencaixe peca side
crtencaixe peca raio
assencaixeval peca b0 10.0
assencaixeval peca c0 11.0
assencaixeval peca a0 5.0
assencaixeval peca side 20.0
assencaixeval peca raio 2.5
```

```
copylelem peca vol3 vol4
copyrelem peca volume vol5
manipelem peca vol4 vol5 vol6 diferenca
proc peca
```

Apêndice D

Programa Interface para o MSD

Para a criação do exemplo apresentado no capítulo 5 foi utilizada a linguagem do protótipo implementado. Esta linguagem foi descrita no apêndice C. Ao executar o comando proc do protótipo cria-se um arquivo de comandos para o Modelador de Sólidos MSD [32] que também foi implementado. A linguagem do Modelador de Sólidos está descrita em seu Manual do Usuário [31]

A listagem abaixo é um arquivo interface para o Modelador de Sólidos MSD que foi criado a partir da execução do protótipo utilizando o exemplo listado no apêndice C com um dado conjunto de parâmetros de encaixe.

```
camera -g cam 100.0 50.0 40.0 0.0 0.0 0.0 5 3 3 1 0
Cylinder v23 10     2.5000    13.0000
CUBE v13    27.0000    10.0000    13.0000
CUBE v6     5.0000    10.0000    11.0000
TRANSFORMA v13
   -1.0000     0.0000     0.0000     0.0000
   -0.0000    -1.0000     0.0000     0.0000
    0.0000     0.0000     1.0000     0.0000
    7.0000    10.0000    -2.0000     1.0000
rename v6 v27
rename v13 v28
MINUS v28 v27 v32
TRANSFORMA v23
    1.0000     0.0000     0.0000     0.0000
    0.0000     1.0000     0.0000     0.0000
    0.0000     0.0000     1.0000     0.0000
   -6.5000     5.0000    -2.0000     1.0000
rename v23 v33
rename v32 v34
MINUS v34 v33 v35
display cam **
```

Bibliografia

[1] BERNARD, A. The feature approach for the integrated design and machining of forming dies. **Robotics & Computer Integrated Manufacturing,** v.10, n.1/2, p.71-76, 1993.

[2] CHANG, T.C. **Expert process planning for manufacturing.** Addison Wesley Publishing Company, 1990.

[3] CHUNG, J.C.H. et al. Feature-based geometry construction for geometric reasoning. In: ASME COMPUTERS IN ENGINEERING CONFERENCE, San Francisco, 1988. **Proceedings.** San Francisco, ASME, 1988. p.497-504.

[4] CUTKOSKY, M. et al. Features in process based design. In: ASME COMPUTERS IN ENGINEERING CONFERENCE, San Francisco, 1988. **Proceedings.** San Francisco, ASME, 1988. p.557-562.

[5] CUGINI, U. Capturing context dependent rules from interaction sequences: an example for mechanical design. In: SATA, T., ed. **Organization of engineering knowledge for product modeling in computer integrated manufacturing,** Amsterdam, Elsevier, 1989. p.385-410.

[6] DONG, X.; WOZNY, M. Instantiation of user defined features on a geometric model. In: Turner, J. et al. **Product modeling for computer aided design and manufacturing,** Amsterdam, Elsevier, 1991. p.183-195.

[7] GOMES, A.J.P.; TEIXEIRA, J.C.G. Form feature modelling in a hybrid CSG/BRep scheme. **Computer & Graphics,** vol.15, n.2, p.217-229, Feb. 1991.

[8] GOSSARD, D.C. et al. Representing dimensions, tolerances, and features in MCAE systems. **IEEE Computer Graphics & Applications**, v.8 , n.2 , p.51-59, Mar. 1988.

[9] INUI, M. et al. Generation and verification of process plans using dedicated models of products in computers. In: ASME SYMPOSIUM ON KNOWLEDGE-BASED EXPERT SYSTEMS FOR MANUFACTURING, 1986. **Proceedings.** ASME, 1986. v.24, p.275-286.

[10] INUI, M.; KIMURA, F. Using a truth-maintenance system to assist product-model construction for design and process planning. **Computer Aided Design**, v.25, n.1, p.59-70, Jan. 1993.

[11] JOSHI, S.; CHANG, T.C. Graph-based heuristics for recognition of machined features from a 3-D solid model. **Computer Aided Design**, v.20, n.2, p.58-64. Mar. 1988.

[12] KAWAGOE, K.; MANAGAKI, M. Parametric object model and its applications to mechanical product design. In: WARMAN, E.A., ed. **Computer applications in product design.** Amsterdam, North Holland, 1983. p.353-370.

[13] KIM, Y.S. Recognition of form features using convex decomposition. **Computer Aided Design**, v.24, n.9, p.461-476, Sep. 1992.

[14] KIMURA, F. et al. A study on product modelling for integration of CAD/CAM. In: KOSCHAN, D., ed. **Integration of CAD/CAM.** Amsterdam, Elsevier, 1984. p.227-251.

[15] KIMURA, F. et al. A uniform approach to dimensioning and tolerancing in product modelling. In: K., Bo et al. eds. **Computer aided production engineering.** Amsterdam, North Holland, 1987. p.165-178.

[16] LEE, Y.C.; FU, K.S. Machine understanding of CSG: extraction and unification of manufacturing features. **IEEE Computer Graphics & Applications**, v.7, n.1,

p.20-32, Jan. 1987.

[17] LIGHT, R.; GOSSARD, D. Modification of geometric models through variational geometry. **Computer Aided Design**, v.14, n.4, p.209-214, Jul. 1982.

[18] MÄNTYLÄ, M. **An introduction to solid modeling.** Maryland, Computer Science Press, 1988.

[19] MCMACHON, C.A. et al. Observations on the application and development of parametric-programming techniques. **Computer Aided Design**, v.24, n.10, p.541-546, Oct. 1992.

[20] MARKOWSKY, G.; WESLEY, M.A. Fleshing out wire frames. **IBM J. Research and Development**, v.24, n. 5, p.582-587, Sep. 1980.

[21] MIYAGI, P.E.; BARRETTO, M.R.P. Sistemas integrados de manufatura. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 8., Belém, 1990. **Anais.** Belém, SBA, 1990. v.1, p.TC 81-87.

[22] OSTROWSKI, M.C. Feature-based design using constructive solid geometry. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES - SIGGRAPH'90, 17., Dallas, 1990. **Course Notes.** Dallas, SIGGRAPH, 1990.

[23] OVTCHAROVA, J. et al. A proposal for feature classification in feature-based design. **Computer & Graphics**, v.16, n.2, p.187-195, Feb. 1992.

[24] PRATT, M.J. Synthesis of an optimal approach to form feature modelling. In: ASME COMPUTER AND ENGINEERING CONFERENCE, San Francisco, 1988. **Proceedings.** San Francisco, ASME, 1988. p.263-274.

[25] REQUICHA, A.A.G.; VOELCKER, H.B. Solid modeling: a historical summary and contemporary assessment. **IEEE Computer Graphics & Applications**, v.2, n.2, p.9-24, Mar. 1982.

[26] ROLLER, D. An approach to computer-aided parametric design. **Computer Aided Design**, v.23, n.5, p.385-391, Jun. 1991.

[27] ROSSIGNAC, J.R. Issues on feature-based editing and interrogation of solid models. **Computer & Graphics**, v.14, n.2, p.149-172. Feb. 1990.

[28] SHAH, J.J.; ROGERS, M.T. Expert form feature modelling shell. **Computer Aided Design**, v.20, n.9, p.515-524, Nov. 1988.

[29] SHAH, J.J. Assessment of Feature Technology. **Computer Aided Design, v.23,** n.5, p.331-343, Jun. 1991.

[30] SUZUKI, H. et al. Synthesizing product shapes with geometric design constraints and reasoning. In: YOSHIKAWA, H.; HOLDEN, T., eds. **Intelligent CAD II.** Amsterdam, Elsevier, 1990. p.309-324.

[31] TSUZUKI, M.S.G. **Manual de usuário do modelador de sólidos MSD.** São Paulo, EPUSP/Departamento de Engenharia Mecânica/Mecatrônica, 1991.

[32] TSUZUKI, M.S.G. MSD - Modelador de Sólidos Didático. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, 6., Recife, 1993. **Anais.** Recife, 1993. p.17-20.

[33] TURNER, G.P.; ANDERSON, D.C. An object oriented approach to interactive feature design for quick turnaround manufacturing. In: ASME COMPUTERS IN ENGINEERING CONFERENCE, San Francisco, 1988. **Proceedings.** San Francisco, ASME, 1988. p. 551-555.

[34] WILSON, P.R. Feature modeling overview. In: INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES - SIGGRAPH'90, 17., Dallas, 1990. **Course Notes.** Dallas, SIGGRAPH, 1990. p.XI.1-XI.56.

[35] WINGARD, L. **Introducing form features in product models: a step towards CAD/CAM with engineering terminology.** Stockholm, 1991. 87p. Licentiate

Thesis - The Royal Institute of Technology.