UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# DEFINIÇÃO E IMPLEMENTAÇÃO DE UM MODELO DE DADOS PARA REPRESENTAÇÃO DE SÓLIDOS

por

MAGALÍ TERESINHA LONGHI

Dissertação submetida como requisito parcial para a obtenção do grau de Mestre em Ciência da computação

Profa. Lia Goldstein Golendziner

Orientadora

Profa. Carla M. Dal Sasso Freitas

Co-orientadora

Porto Alegre, abril de 1987

CATALOGAÇÃO NA FONTE

LONGHI, Magalí Teresinha.
Definição e implementação de um modelo de dados para representação de sólidos. Porto Alegre, PGCC da UFRGS, 1987.
203p.
Diss.(mestr. ci. comp.) UFRGS-PGCC. Porto Alegre, BR-RS, 1987.

Dissertação: Modelagem de Sólidos:Octree:Geometria de Sólidos Construtivos:Operações de Combinação e Transformação Geométrica:Agregação:Generalização.

"Hay hombres que luchan un día y son buenos

Hay otros que luchan un año y son mejores

Hay quienes luchan muchos años y son muy buenos

Pero hay los que luchan toda la vida

Esos son los imprescindibles"

(Bertold Brecht)

Aos meus pais

## AGRADECIMENTOS

Às professoras Lia Goldstein Golendziner e Carla M. Dal Sasso Freitas pela dedicação, estímulo e ajuda constantes na orientação de todas as etapas deste trabalho.

Ao Centro de Processamento de Dados desta Universidade, representado pelo prof. Clesio Saraiva dos Santos, pelas condições de benefício recebidos durante o curso e elaboração deste trabalho. Em especial ao prof. Neron Arruda Leonel pelo estímulo e compreensão.

Ao curso de Pós-Graduação em Ciência da Computação, representado na pessoa de seu coordenador Roberto Tom Price, pelas condições de estudo e trabalho oferecidas.

À Biblioteca CPD/PGCC, em especial à Margarida, pela disponibilidade, auxílio e interesse permanentes na seleção do material utilizado na elaboração deste trabalho.

Aos meus amigos pelo apoio recebido.

À Tere pela digitação do trabalho, pela paciência e acima de tudo pela amizade.

À Helenara e à Lígia pela parte dos desenhos.

Ao Low e à Vânia pela força de sempre (Obrigada, Vânia, pelo apoio de amiga!). Ao Lampert, Lusamar e Clarisse pela força dos últimos tempos.

Ao Rudnei pelas revistas, livros e idéias.

Ao Brettas pela paciência nos estudos de matemática.

À Valéria pela acolhida destes anos.

Ao Seu Trúculo que, sem me conhecer, deu uma força enorme.

Ao Ike e à Bana pela torcida.

Aos meus pais que, mesmo longe, sempre estiveram muito presentes.

A você, Nelsinho, pela tua força e amizade fundamentais para a conclusão deste trabalho.

SUMÁRIO

## LISTA DE ABREVIATURAS

| | |
|---|---|
| B-rep | "Boundary-representation" - Representação por faces limitantes |
| CAD | "Computer Aided Design" - Projeto Auxiliado por Computador |
| CAM | "Computer Aided Manufactured" - Fabricação Auxiliada por computador |
| CSG | "Constructive Solid Geometry" - Geometria de sólidos construtivos |
| DDL | "Data Definition Language" - Linguagem de definição de dados |
| DML | "Data Manipulation Language" - Linguagem de manipulação de dados |
| N-MOS | Núcleo de Modelagem de Sólidos |
| SGBD | Sistema de Gerência de Banco de Dados |

## LISTA DE FIGURAS

# RESUMO

O trabalho tem por objetivo definir um modelo de dados adequado à representação de sólidos e implementar um núcleo de modelagem geométrica para validar este modelo. Foi adotada a abordagem semântica Abstração de Dados para a representação formal do modelo geométrico.

Os objetos são modelados utilizando operações de combinação e de transformação geométrica. Foram adotadas as formas de representação interna por OCTREE, que define explicitamente as porções do espaço ocupadas pelos objetos, e por uma árvore CSG que mantém a descrição estruturada dos objetos segundo o processo construtivo. São apresentados os algoritmos para as operações de manipulação (combinação e transformação geométrica) e de exibição através de vistas ortográficas e de seções planas, bem como operações de consulta às propriedades dos objetos.

## ABSTRACT

The goal of this work is to define a data model suitable for representing solids and to implement a solids modelling kernel. It was used the Data Abstraction semantic model for representing the geometric model.

The objects are modeled through set operations and geometric transformations. It was used two schemas of solids representations: OCTREE, which explicitly define the space occupied by objects, and CSG that is used to keep a structural description of objects according to the construction process. The algorithms for solids manipulation (set operations and geometric transformations) and the display generation through orthographic views and plane sections are presented as well as query operations over the objects properties.

# 1 INTRODUÇÃO

A viabilização das técnicas interativas gráficas, na década de 60, através do pioneiro trabalho de Sutherland /SUT 63/ representa o marco inical do desenvolvimento que levou aos primeiros sistemas de projeto auxiliado por computador (CAD, "Computer Aided Design"). Tais sistemas são dotados, fundamentalmente, de um conjunto de ferramentas de projeto que permitem a construção de um modelo do objeto, ou seja, uma descrição do objeto real.

Pelo fato de que as informações no modelo servem a uma variedade de propósitos (produção de desenhos, análise, simulação, especificação de fabricação/montagem), o modelo deve ser o mais completo possível.

Quando o modelo é constituído principalmente de informações geométricas (gráficas) do objeto, diz-se que é um modelo geométrico. Além disso, informações analíticas referentes apenas a processos de análise/simulação devem estar presentes no modelo completo de um objeto de uma determinada aplicação.

O processo de construção do modelo geométrico de um objeto real é denominado de modelagem geométrica. A informação de dados não-geométricos pode estar incluída em etapas deste processo.

Sistemas CAD são aplicados em grande escala nas áreas de engenharia, com o objetivo básico de reduzir o custo de projeto e o tem-

po de concepção do produto a que se propõem, além de melhorar a qualidade dos produtos fabricados e aumentar o potencial do projetista em termos de produtividade. Projetos de peças mecânicas, automóveis, navios, aviões, estruturas metálicas, obras de construção civil e circuitos integrados são alguns exemplos de aplicações CAD.

A figura 1.1 ilustra a arquitetura de um sistema CAD e o sistema de modelagem associado (SMG). Um sistema CAD moderno requer três importantes componentes:

1) estações de trabalho, que contenham facilidades de entrada e saída para dados e comandos, com equipamentos projetados para facilitar a interação homem - máquina;

2) processadores e "software", para operarem sobre as informações presentes no sistema de acordo com os comandos recebidos da estação de trabalho. A geração de um modelo, sua análise e modificação, através de uma estação de trabalho, envolve uso de facilidades gráficas e processos de edição, como também pacotes de análise específicos que tratam dos atributos analíticos (físicos) do modelo /ROW 85/.

3) banco de dados, para facilitar as modificações e análises feitas pelo usuário e agilizar o processamento para a obtenção de respostas rápidas.

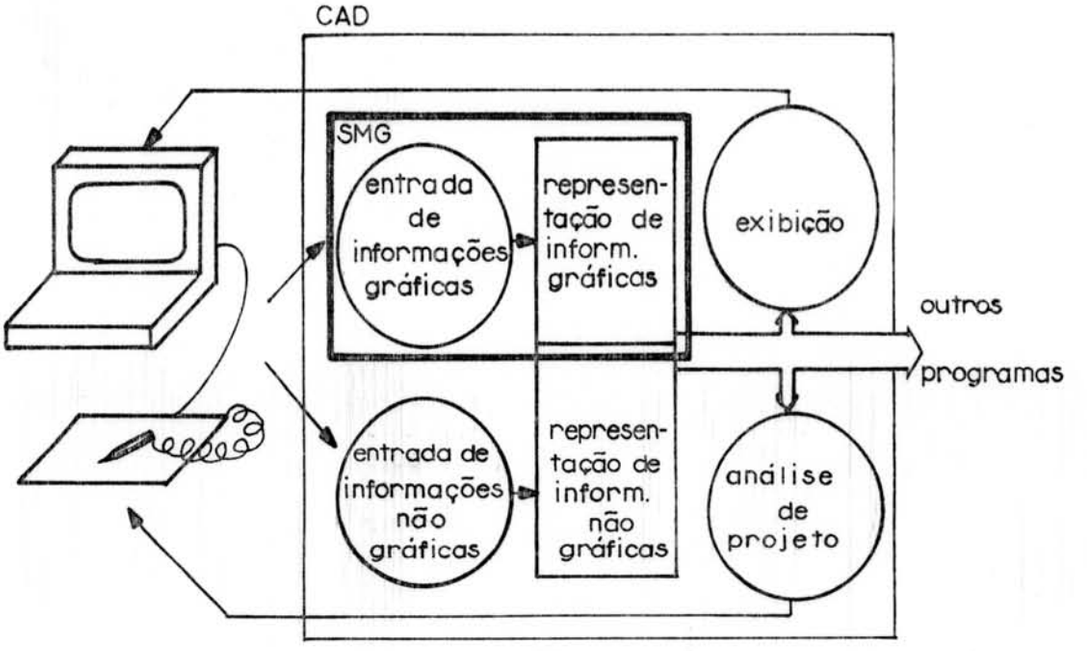



FIG. 1.1 Visão funcional de um sistema CAD

O uso da abordagem de banco de dados em sistemas CAD começou a ser discutido já em meados da década de 70. A multiplicidade de informações comuns necessárias às várias ferramentas de projeto levou naturalmente à tentativa de centralização dos dados sobre os objetos e ao uso dos sistemas de gerência de banco de dados disponíveis. As peculiaridades dos sistemas de projeto auxiliado por computador, no entanto, impediram o uso direto dos sistemas de banco de dados /SID 80/. As aplicações CAD foram então enquadradas, pelos pesquisadores de banco de dados, como mais uma das chamadas aplicações não-convencionais. O termo expressa a contraposição existente com os sistemas de informação comuns nas empresas (estoque, pessoal, etc ), classificados como aplicações convencionais.

A evolução de sistemas CAD, em termos do processo de modelagem e do uso de banco de dados, passou no mínimo por quatro fases. Na primeira, os objetos descritos eram essencialmente bidimensionais e quando se tratava de objetos tridimensionais as informações eram incompletas e redundantes (representação por vistas e seções planas). A exibição consistia da apresentação de vistas armazenadas e as descrições bidimensionais tornavam-se insuficientes para o processamento de objetos tridimensionais.

Na segunda fase, os objetos eram descritos em termos de pontos e linhas no espaço (modelagem por arestas). Vários problemas existiam como, por exemplo, a dificuldade de se identificar uma face do objeto ou seu interior.

Na terceira fase, observa-se a tendência dos modelos representarem fielmente a realidade dos objetos e garantir que suas propriedades possam ser recuperadas e calculadas quando solicitadas. Esta fase é caracterizada pelos sistemas de modelagem geométrica.

Na quarta e atual fase, a descrição dos objetos é válida não somente para a etapa de projeto mas também para a etapa de fabricação. Os dados são mantidos numa única base de dados, compartilhada entre os sistemas CAD e CAM ("Computer Aided Manufacturing"). Essas aplicações integradas são a atual geração de sistemas CAD/CAM. Nestes sistemas, o modelo do objeto precisa necessariamente ser completo. Daí a importância do sistema de modelagem geométrica.

Um sistema de modelagem geométrica fornece facilidades de entrada, armazenamento e modificação das representações dos objetos e meios para responder questões do tipo "qual é o volume do objeto", "qual é o peso do objeto", etc. Modelos geométricos fiéis à realidade devem conter dados suficientes para derivar qualquer informação sobre os objetos, bem como permitir a execução de operações diversas.

Weiler /WEI 85/, classifica em três tipos as técnicas de modelagem desenvolvidas: modelagem por arestas ("wireframe"), modelagem de superfícies e modelagem de sólidos.

A modelagem por arestas descreve os objetos em termos de suas linhas. Não suporta algoritmos para a eliminação de linhas ocultas e não mantém informações de superfícies para que faces possam ser identificadas /REQ 80/. Esse tipo de modelagem não garante que os objetos sejam descritos sem ambiguidades /MAR 80/, /FOL 82/.

A modelagem de superfícies é própria para aplicações onde os objetos apresentam superfícies curvas, esculpidas, como é o caso do projeto de aeronaves e automóveis. Este modelo, embora completo, não é eficiente para cálculos de volume e outros processos geométricos.

A modelagem de sólidos incorpora uma série de teorias, técnicas e métodos para representar sólidos de tal forma que suas propriedades possam ser calculadas automaticamente.

A figura 1.2 ilustra os diferentes tipos de técnicas de modelagem apresentadas acima.

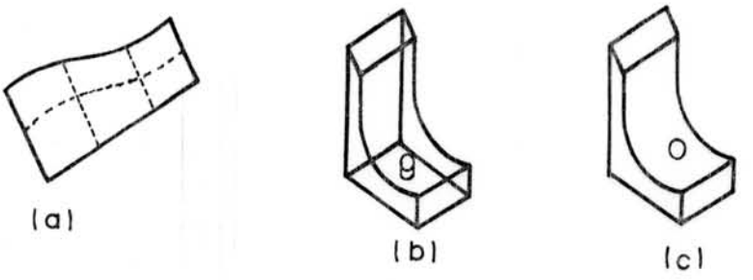



FIG. 1.2 Técnicas de modelagem: (a) por superfícies
(b) por arestas ("wireframe")
(c) por volume (sólido)

Neste trabalho será dada ênfase à técnica de modelagem de sólidos, que será melhor abordada no capítulo 2. Neste capítulo serão apresentados os modelos propostos para representar os sólidos e as operações de modelagem associadas. Serão apresentados, também, comentários sobre técnicas de interação e um resumo da aplicação dos modeladores de sólidos atualmente existentes no mercado.

O capítulo 3 introduzirá o estudo da integração de Sistemas de Gerência de Banco de Dados a sistemas de Modelagem de Sólidos, bem como a utilização dos conceitos de banco de dados para representação de dados gráficos e não-gráficos.

O capítulo 4 trata do Núcleo de Modelagem de Sólidos (N-MOS) desenvolvido no CPGCC - UFRGS, como um núcleo básico para a futura implantação de um sistema completo de modelagem de sólidos.

No capítulo 5 e 6 serão relatados os módulos implementados, a organização e acesso das estruturas de dados, as funções de modelagem

e de consulta, como também a interface com o usuário suportada pelo sistema.

No capítulo 7 será analisada a implementação e serão apresentadas as limitações e restrições do sistema, bem como algumas considerações sobre a continuidade e extensões futuras do trabalho.

# 2. MODELAGEM GEOMÉTRICA DE SÓLIDOS

## 2.1 Introdução

Dentre as técnicas de modelagem, a modelagem de sólidos é reconhecida como uma das mais importantes para sistemas CAD em áreas que tratam de objetos tridimensionais (engenharia mecânica, engenharia civil, arquitetura, etc).

O desenvolvimento das técnicas de modelagem de sólidos começou em meados de 1970 e hoje, apesar de serem cada vez mais utilizadas, estas técnicas continuam sendo limitadas por diversos fatores como a complexidade do software, a quantidade de memória necessária para armazenar código a ser executado e dados associados, e os problemas de interação usuário - sistema.

A característica mais importante dos modeladores de sólidos é a capacidade de suportar automaticamente uma grande variedade de aplicações. Contudo, isto se refere apenas às técnicas construtivas dos objetos: cada aplicação tem particularidades de interação homem - máquina e de consultas a dados associados aos objetos.

A expressão "Modelagem de Sólidos" incorpora uma série de teorias, técnicas e formas de representação de sólidos onde qualquer propriedade geométrica bem definida pode ser calculada automaticamente /REQ 82/, /REQ 83/. Desse modo, um sistema de modelagem de sólidos deve garantir a representação de um sólido de forma não ambígua e geome-

tricamente correta, para que ele possa ser manipulado posteriormente. Uma forma de representação é não-ambígua (ou completa) se, a partir de uma definição, pode-se depreender um único objeto. É única, se cada objeto admite apenas ser representado de uma única forma. Por geometricamente correta, entende-se aquela que garante a validade dos sólidos representados. Um sólido é válido se ele obedece a lei básica de Euler, ou seja, se dados V, número de vértices, A, número de arestas, e F, número de faces, V - A + F = 2.

Um sistema de modelagem de sólidos deve ser confiável, eficiente, garantir a integridade dos dados representados no modelo e executar cálculos para obter propriedades de massa como peso, volume, centro de gravidade, momentos de inércia, área de superfície, etc.

Os modeladores construídos ao longo de quase duas décadas de pesquisa na área introduziram várias formas de representação, as quais são revistas na próxima seção. Na sequência, relata-se a atual utilização dos modeladores de sólidos.

## 2.2 Formas de Representação de Sólidos

Requicha /REQ 80/ classificou as formas de representação em ambíguas e não-ambíguas.

Como representações ambíguas podemos citar:

1) representação por vistas ortográficas, onde os sólidos são especificados por projeções planares (figura 2.1(a));

2) representação por arestas ("wireframe"), onde os sólidos são especificados por coleções de arestas, com a possibilidade de definir objetos sem sentido (figura 2.1(b)).

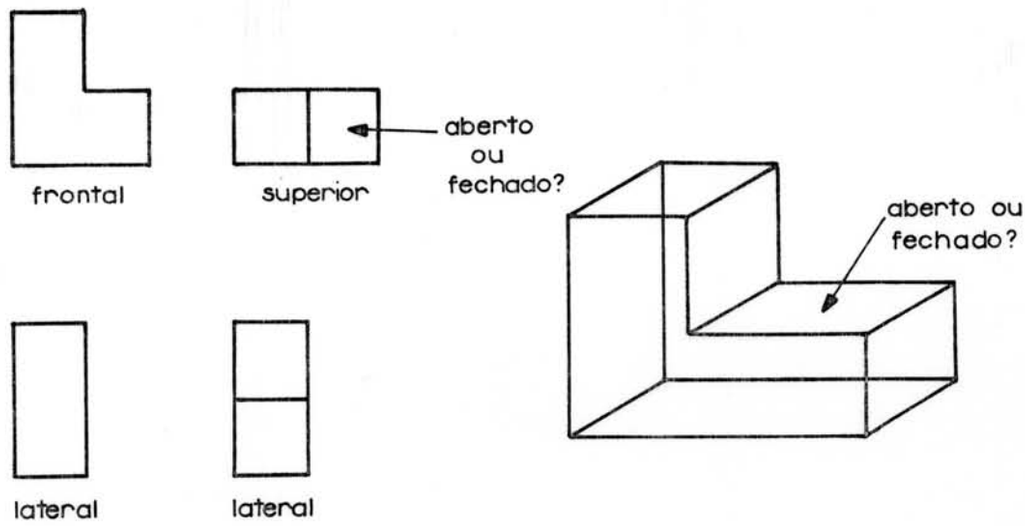



FIG. 2.1 Representações ambíguas: (a) vistas ortográficas (b) coleção de arestas

As representações não-ambíguas são caracterizadas por serem completas. A maioria delas caracteriza-se também por ser única. Entre elas pode-se distinguir:

1) Instanciamento de Primitiva: é baseada no conceito de famílias de objetos cujos componentes são diferenciados por um conjunto limitado de parâmetros (figura 2.2). A família é conhecida como primitiva genérica. Cada componente é uma instância. A princípio, esta forma de representação é úni-

ca, fácil de validar e usar, mas, na prática, tais propriedades são garantidas para pequenos domínios e cada um possuindo um conjunto pequeno de parâmetros.

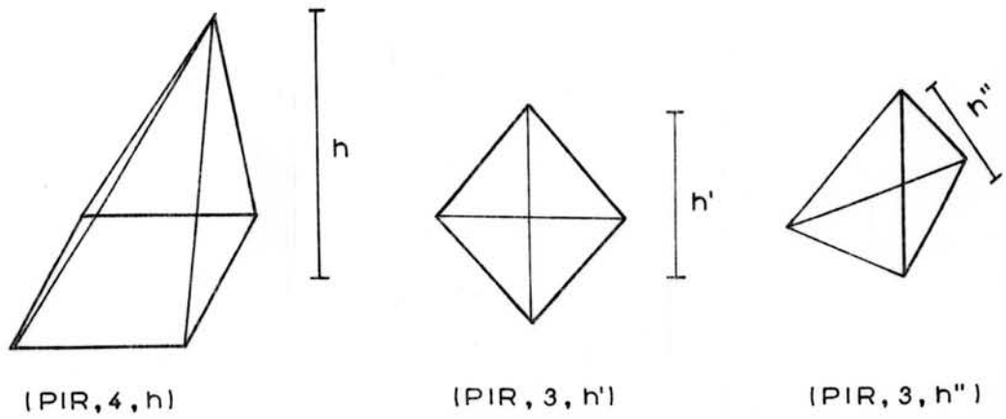



FIG. 2.2 Instâncias de pirâmides representadas por triplas ('PIR',N,H) onde 'PIR' indica o nome da família, 'N' o número de lados e 'H' a altura da pirâmide.

2) Enumeração Espacial: é única e fácil de validar. O objeto é representado por uma lista de células (voxels) que ele ocupa no espaço. Pode ser caracterizada como:

a) Regular: para voxels de mesmo tamanho (figura 2.3(a));

b) Irregular (também chamada de decomposição celular): para voxels de tamanhos diferentes (figura 2.3(b)). Fazem parte desta representação as octrees que serão tratadas no capítulo 4.

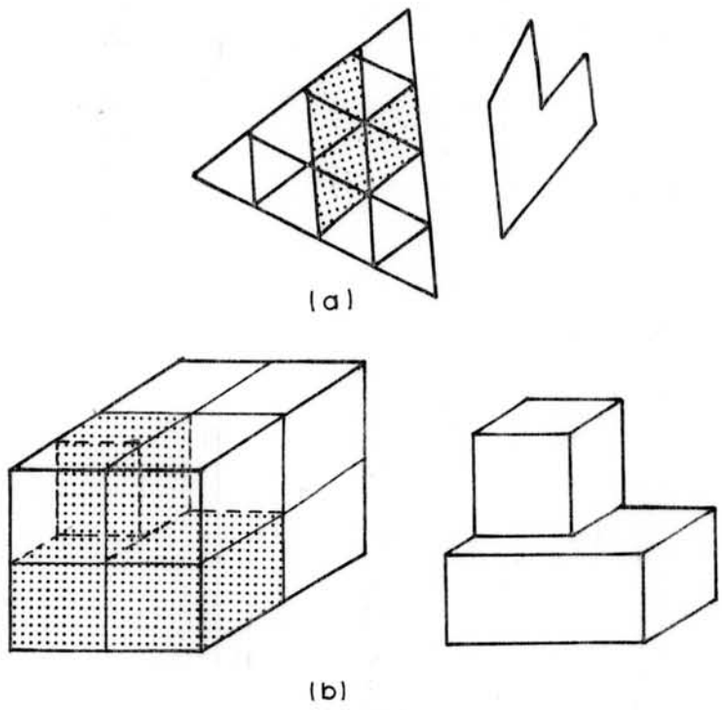



FIG. 2.3 Representação por enumeração espacial: (a) regular
(b) irregular

3) Geometria de Sólidos Construtivos("Constructive Solid Geometry", CSG): os objetos são representados pela combinação de sólidos primitivos através de operações de conjunto (de combinação, "booleanas") como união, intersecção e diferença. Cada objeto é representado pelo nodo pai de uma árvore binária onde os nodos folha representam os sólidos primitivos e os intermediários especificam operações realizadas sobre os sólidos representados pelas suas subárvores (figura 2.4). Esta forma de representação é não-ambígua, mas não-única porque representações distintas podem ser instâncias de um objeto. Seu domínio depende do conjunto de primitivas e dos operadores de combinação e transformação geométrica. A representação será válida se as primitivas forem válidas e os operadores regularizados

(ver seção 2.3).

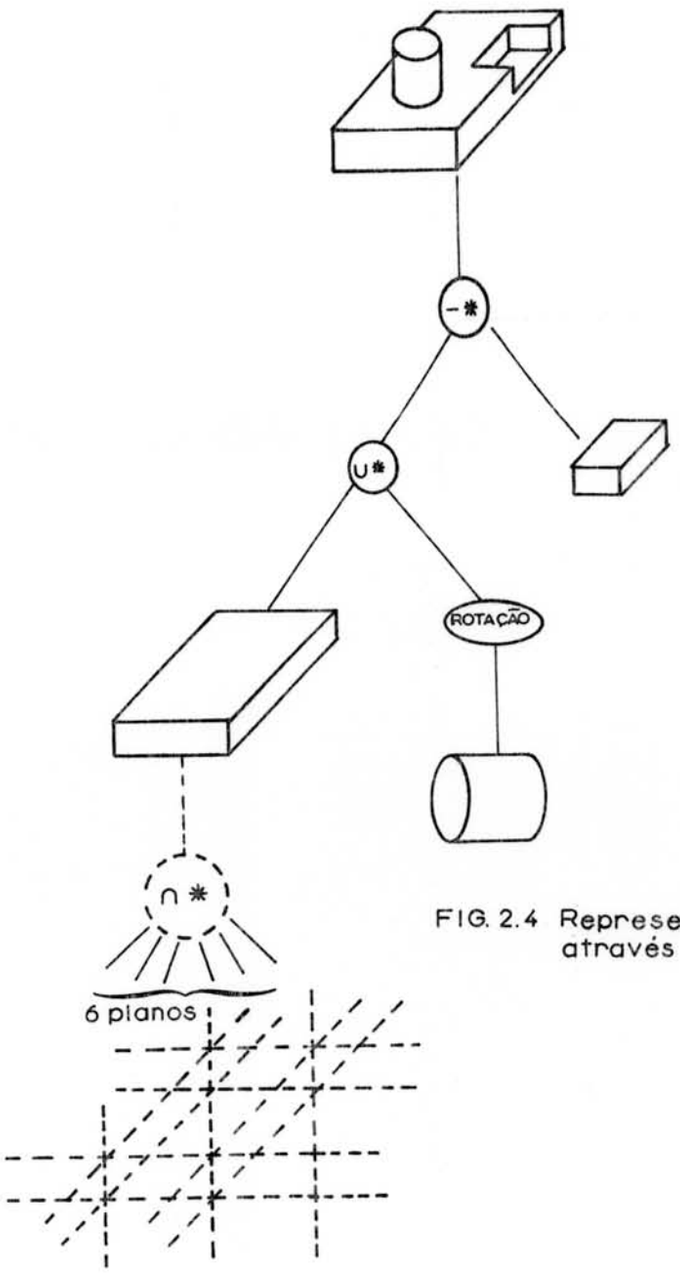



FIG. 2.4 Representação dos objetos através da CSG.

4) Representação por varredura ("Sweeping") rotacional ou translacional (sólidos 2 1/2D): os sólidos são descritos por formas planares movimentadas no espaço segundo uma trajetória rotacional ou translacional (figura 2.5). Numa forma de representação similar, pode-se utilizar um volume ao invés de uma forma planar. É não-ambígua, mas não-única e seus domínios são limitados para objetos com simetria

rotacional e translacional.

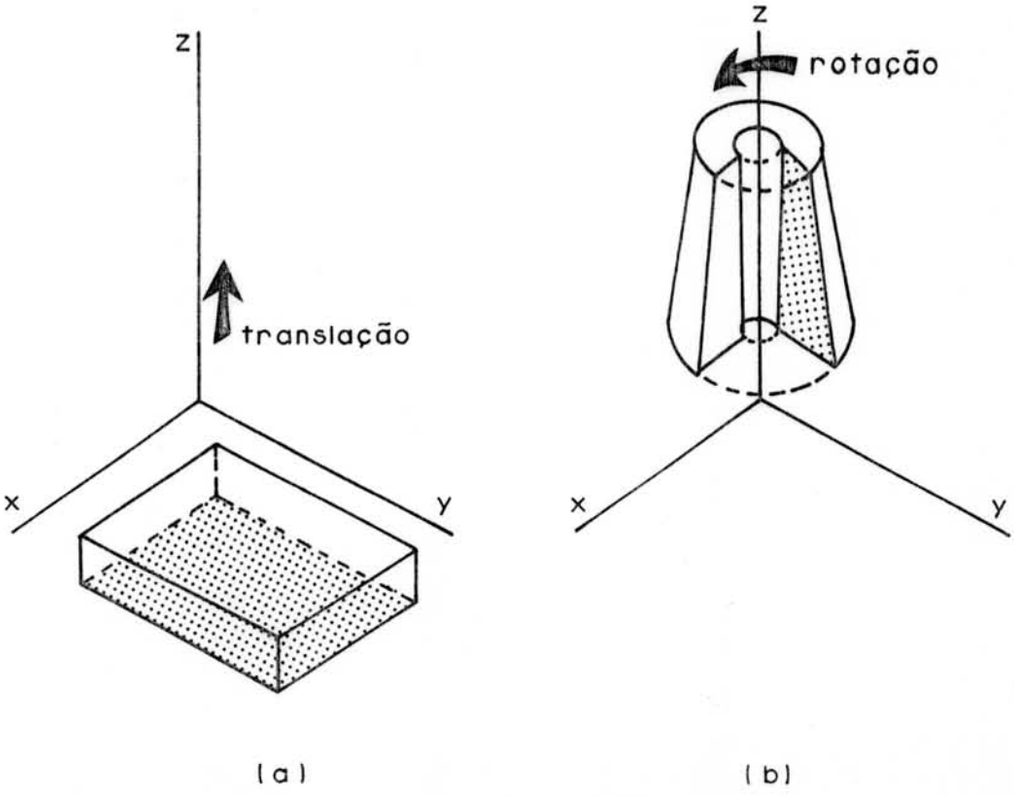



FIG. 2.5 Sólidos $2\frac{1}{2}$ D: (a) sweep translacional
(b) sweep rotacional

5) Representação por faces limitantes ("Boundary-representation", B-rep): os sólidos são representados por uma coleção de faces, arestas e vértices e os relacionamentos entre eles. Utiliza-se apenas a fronteira entre o interior e o exterior do sólido para representá-lo (figura 2.6). É não-ambígua se faces forem representadas não ambiguamente, mas geralmente é não-única.

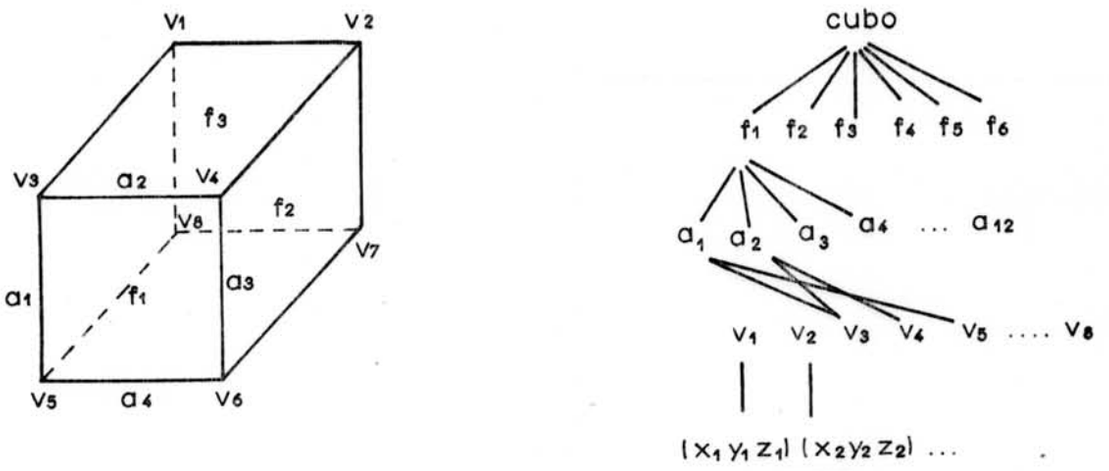



FIG. 2.6 Representação por faces limitantes de um sólido

Baer et alli /BAE 79/ fazem uma análise dos sistemas mais conhecidos, indicando qual a forma de representação usada para descrever internamente o sólido e em qual forma estes sistemas se basearam para construir a interação com o usuário.

As formas de representação mais usadas são a CSG e a B-rep. B-rep é apropriada para se obter imagens dos objetos e para cálculos de propriedades como volume, peso, etc. Entretanto, os algoritmos que tratam as operações de combinação são complexos. CSG é usada pela simplicidade de tratamento destas operações. Contudo, no cálculo de propriedades e na exibição dos objetos, a performance dos sistemas que a utilizam é mais baixa.

Muitos sistemas combinam as duas formas originando as formas híbridas onde os objetos são definidos segundo a técnica CSG e convertidos para B-rep através de uma operação de avaliação dos limites ("boundary evaluation") /REQ 85/. Para a exibição rápida, os objetos em formato B-rep são transformados em polígonos. A figura 2.7 ilustra a conversão de CSG para B-rep e de B-rep para polígonos (Ar representa

o bloco A após uma transformação de rotação).

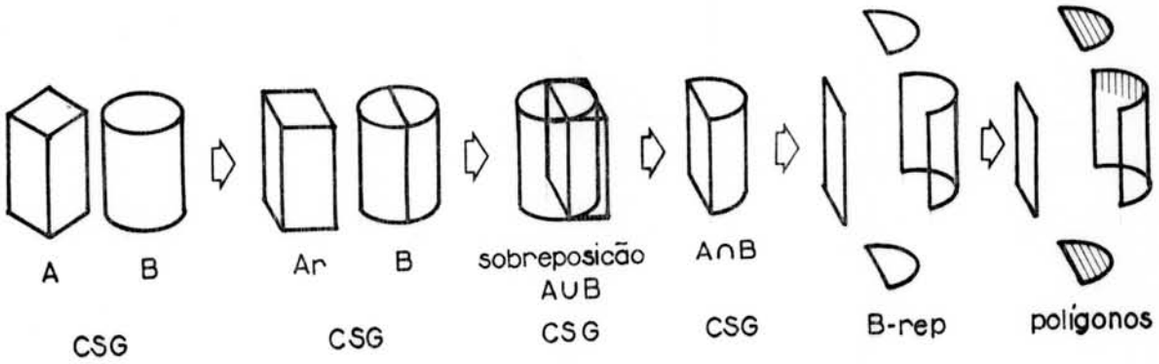



FIG. 2.7 Conversão CSG → B-rep → polígonos

Sistemas baseados nestas duas técnicas são caracterizados por um domínio restrito de objetos representáveis /MEA 82a/, pois os objetos são construídos a partir de um número limitado de superfícies matematicamente bem definidas ou por sólidos primitivos. A generalização destes sistemas acarreta grandes custos no desenvolvimento de ferramentas matemáticas juntamente com significativas alterações no software já desenvolvido.

Neste trabalho, ênfase será dada à forma de representação Enumeração Espacial Irregular conhecida por OCTREE. No capítulo 4 será apresentada a definição de OCTREE e sua representação interna no Núcleo de Modelagem de Sólidos desenvolvido. Os diversos tipos de representações internas, a origem e aplicações de tal forma de representação foram abordadas em /LON 86/.

## 2.3 Operações de Modelagem de Sólidos

As operações de manipulação de sólidos podem ser classificadas em dois grupos: operações de combinação (também conhecidas por operações de conjunto ou "booleanas") e pelas operações de transformação geométrica (transformação linear ou de movimento rígido).

Os operadores de combinação compreendem união, intersecção e diferença e necessitam ser regularizados para preservar a característica de validade do objeto resultante /REQ 77/, /TIL 80/ (figura 2.8).

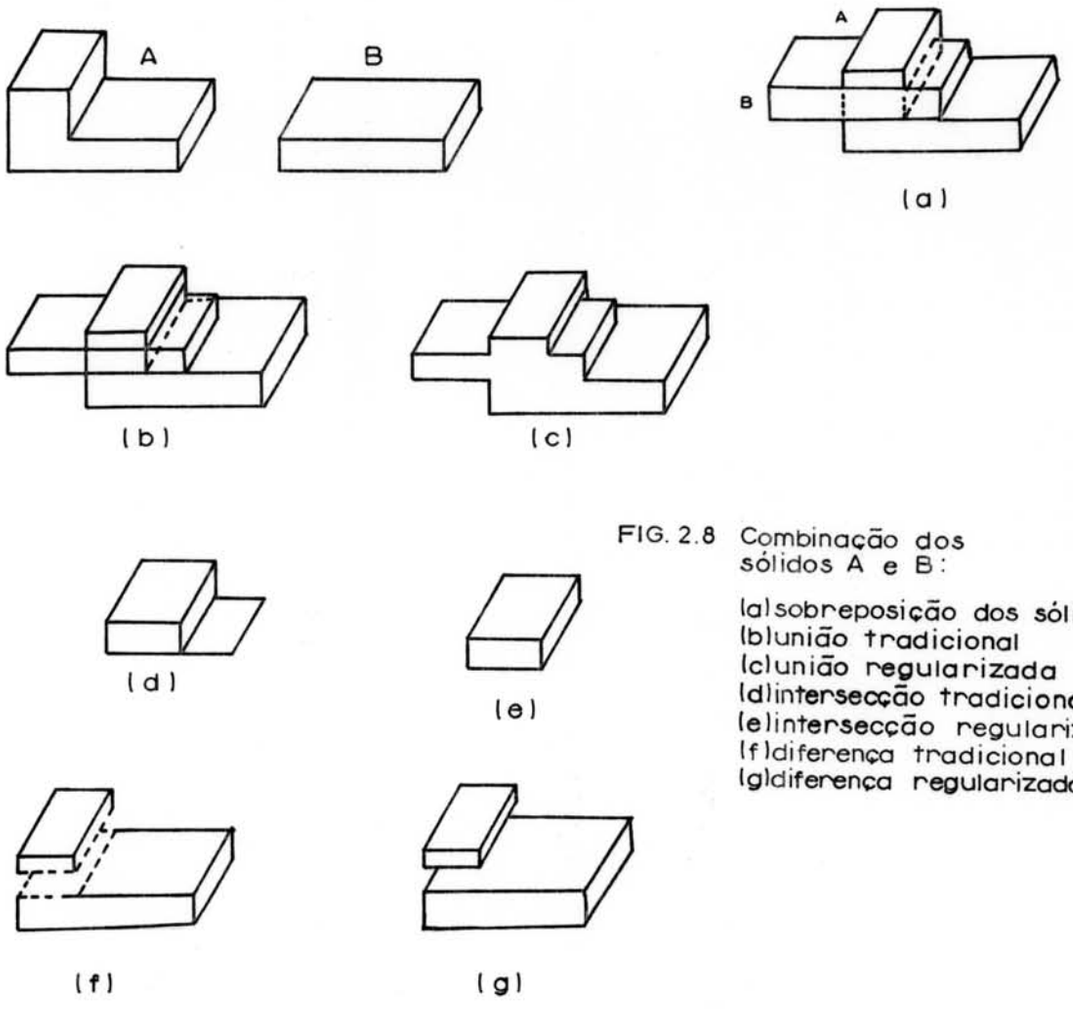



FIG. 2.8 Combinação dos sólidos A e B:
(a) sobreposição dos sólidos
(b) união tradicional
(c) união regularizada
(d) intersecção tradicional
(e) intersecção regularizada
(f) diferença tradicional
(g) diferença regularizada

As operações de translação, rotação e mudança de escala fazem parte das operações de transformação geométrica. Um sólido pode ser transladado para uma nova posição do universo, rotacionado segundo um ângulo e sofrer ampliações ou reduções conforme o fator de escala.

O módulo que realiza as operações de combinação e de transformação num modelador é o mais delicado e o mais complexo de todo o sistema de modelagem. A eficiência dos algoritmos é importante para o desempenho do sistema como um todo.

Os algoritmos dependem completamente das formas de representação adotadas. Modeladores baseados em CSG necessitam de algoritmos de avaliação de limites enquanto que modeladores B-rep operam diretamente sobre faces, arestas e vértices.

Modificações, como a apresentada na figura 2.9, são fáceis de serem executadas em modeladores CSG porque esta forma representa o objeto através da combinação de cada sólido que o compõe. Qualquer erro que possa ocorrer pode ser corrigido sem ser necessário retomar todas as operações a partir dos sólidos primitivos, isto é, pode-se atuar sobre nodos intermediários da árvore CSG /TOR 86/.

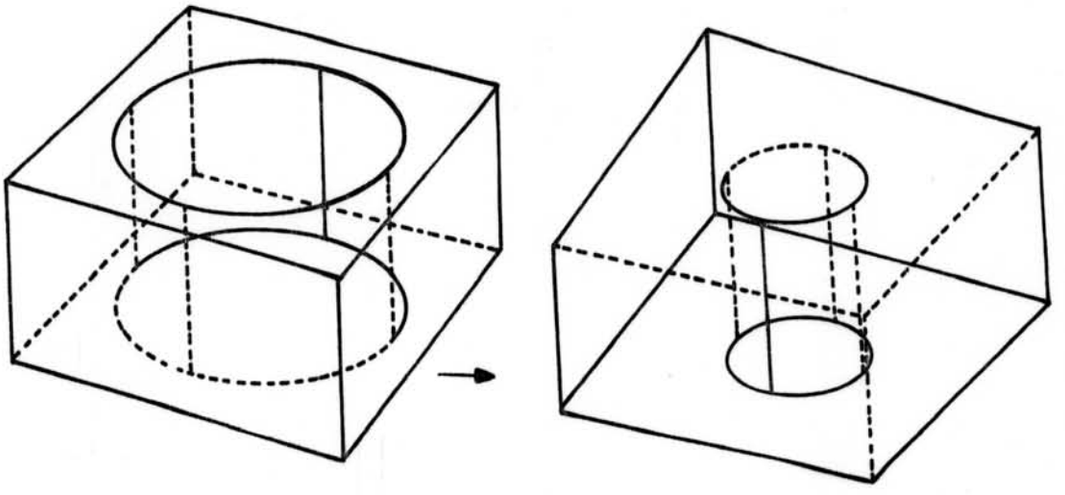


FIG. 2.9 Modificações na definição de um sólido: a perfuração central é diminuída pela redução do sólido primitivo.

Na técnica de modelagem B-rep o tipo de modificação apresentada na figura 2.9 não é tão imediata quanto na CSG. A capacidade de recuperação de erros ou retorno a situações anteriores é importante uma vez que erros de especificação das operações podem conduzir o usuário a situações indesejadas. A figura 2.10 mostra um tipo de erro onde a união de dois sólidos pode apagar partes importantes do objeto devido ao mau posicionamento.

A decisão de escolher qual a melhor forma de representação e, com isso, as operações associadas, depende da aplicação. Objetos de algumas classes de aplicações são difíceis de serem descritos construtivamente (por exemplo, superfícies de aeronaves). Já peças mecânicas, por exemplo, apresentam características que permitem a utilização dessas técnicas de modelagem de sólidos.

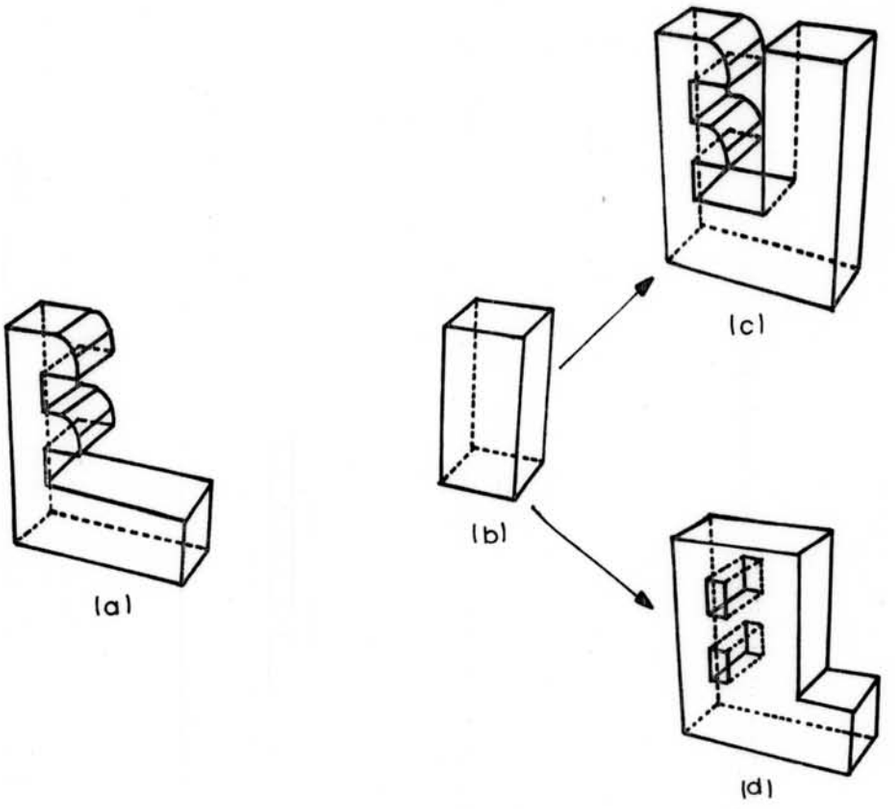



FIG. 2.10 Erros na operação de união. O sólido (c) mostra a definição correta do sólido. O sólido (d) é o resultado do mau posicionamento dos sólidos (a) e (b).

## 2.4 Técnicas Interativas

Pesquisas quanto à melhor forma de representar sólidos e quanto ao melhor método de realizar operações sobre os sólidos demonstram que é possível tornar um sistema de modelagem de sólidos mais eficiente. Entretanto, a interface homem - máquina continua sendo um dos maiores obstáculos para aumentar a produtividade dos sistemas voltados para auxílio a projetos /YOS 84/.

Para Requicha e Voelcker /REQ 82/ os usuários de um sistema de modelagem geométrica são dois: o homem e o conjunto de programas aplicativos. Tanto o homem quanto os programas são capazes de criar,

modificar, fazer acesso a representações de sólidos e produzir saídas gráficas ou não-gráficas (por exemplo, cálculo de propriedades de massa).

A interface para o usuário-homem é bem diferente daquela para o usuário-programa. O homem dispõe desde linguagens de comandos textuais, com facilidades de linguagens de programação, até interfaces gráficas interativas das mais simples às mais sofisticadas, enquanto que o programa utiliza, basicamente, chamadas de procedimentos (ver figura 2.11). A interface com o usuário-homem é um componente explícito do sistema. Já a interface com os programas constitui-se apenas de chamadas de procedimentos, com passagem de parâmetros.

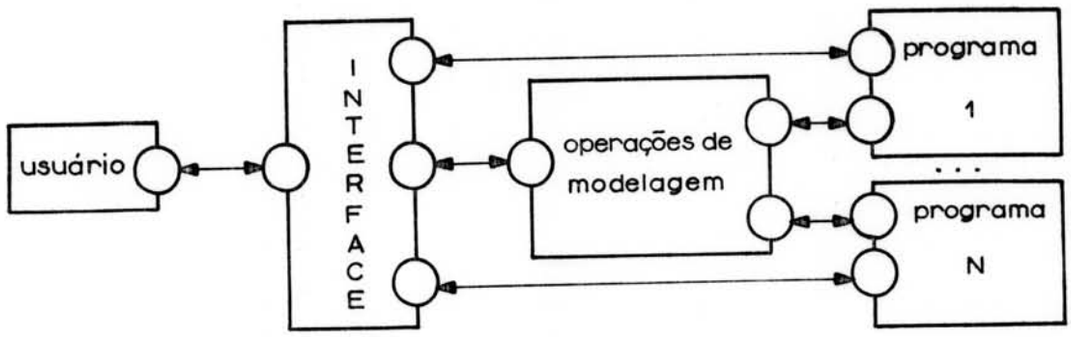



FIG. 2.11 Representação da interface com usuário e programas.

A interação gráfica é uma maneira natural de especificar os diversos procedimentos num sistema de modelagem geométrica. Por outro lado, as funções características das linguagens de programação proporcionam o acesso a construções poderosas como "macros", "loops", etc /REQ 82/, que podem ser úteis no contexto de determinadas aplicações.

Considerando o problema da interface homem - máquina, Chiyokura e Kimura /CHI 85/ desenvolveram um método de gerar e modificar sólidos através de operações primitivas e suas inversas correspondentes (para o efeito de desfazer qualquer operação) que são armazenadas numa estrutura em árvore. Assim, o sólido pode ser recriado em qualquer passo do projeto e o usuário não precisa se preocupar com erros que podem destruir seu trabalho. Todas as operações de mais alto nível são implementadas usando combinações de operações primitivas. Nesse sistema, o usuário fornece comandos para o módulo de geração e modificação de sólidos, que usa operações primitivas armazenando-as numa estrutura em árvore e exibindo os resultados no vídeo.

Representar o processo de projeto de um sólido através de uma estrutura em árvore tipo CSG permite que ele possa ser refeito a partir de qualquer estágio do projeto. Utilizando esta estrutura em árvore, Yoshira et alli /YOS 84/ apresentam uma interface em linguagem natural que permite construir objetos em aplicações de engenharia. Em /TOR 86/ foram apresentadas operações que "desfazem" e "refazem" um sólido descrito em árvore. Dessa forma, o usuário fica seguro de que seu trabalho pode ser refeito em caso de erros.

Os modeladores de sólidos existentes apresentam técnicas de interação diversificadas. O PADL-2 /BRO 82/ fornece três tipos de interface com o usuário: linguagem de comandos, programas e interação gráfica. O EUCLID /SIS 86/ permite que o usuário entre com os dados gráficos e não-gráficos através do teclado alfanumérico e mesa digitalizadora. Os dados não-gráficos fornecidos ao sistema são associados ao modelo geométrico. O usuário pode utilizar a linguagem de programa-

ção EUCLID que oferece funções para definição e modificação geométrica, acesso ao banco de dados, etc. O GMSolid /BOY 82/ interage com o usuário graficamente através de cardápios e caneta luminosa. Também oferece uma linguagem de comandos que inclui comandos de exibição. No DESIGN /HIL 82/, o sistema é operado através de cardápios e o teclado é usado para entradas ocasionais de informações alfanuméricas. No BUILD /HIL 82/, o usuário interage com o sistema através de uma série de comandos via teclado alfanumérico. No ROMULUS /HIL 82/, a interface com o usuário depende do sistema onde está sendo executado, isto porque ele foi projetado como um módulo de modelagem geométrica. Na sua forma primária é similar ao BUILD mas pode apresentar características do DESIGN. O GWB /MAN 82/ oferece operações de alto nível para o usuário, construídas através de operações primitivas transparentes ao usuário.

Os modeladores de sólidos podem ser vistos como ferramentas para permitir ao usuário criar livremente objetos independentes da aplicação ou serem restritos a uma ou poucas aplicações. A interface com o usuário deve ser, no entanto, adequada à aplicação, quaisquer que sejam as técnicas utilizadas para implementá-la.

## 2.5 Aplicação dos Modeladores de Sólidos

Modeladores de sólidos são utilizados atualmente em várias áreas. Muitos foram construídos para certas aplicações particulares, enquanto outros tiveram seu desenvolvimento baseado em características mais genéricas. A figura 2.12 compara alguns modeladores existentes,

apresentando as formas de representação interna e de entrada, bem como o nível de interação entre usuário e sistema. A maioria opera interativamente e alguns possuem uma linguagem de comandos limitada.

Modeladores de sólidos são empregados, atualmente, na engenharia para a geração automática de malhas para a análise de elementos finitos /SHE 82/, /KNO 84/, /WOO 84/, /WOR 84/, na geração automática de programas de comando numérico /BAR 84/, /HAT 84/, /BOB 85/, no projeto de peças mecânicas em geral /ELL 78/, /VOE 78/, /BAE 79/, /FIT 81/, /MYE 82/, /BOL 85/, /GUL 85/ e no projeto de estruturas na construção civil /FEN 81/, /THA 82/. Os modeladores de sólidos são utilizados, também, na determinação de propriedades de massa como cálculo de volume, peso, centro de gravidade, momentos de inércia, na determinação de interferência através da análise de movimento para examinar possíveis superposições e tolerâncias /BOY 79/, /HIL 82/, /REQ 83/, /HAT 84/.

Outras áreas estão começando a utilizar sistemas de modelagem de sólidos: na arquitetura, os modeladores de sólidos são utilizados para projetos de edificação, projeto de interiores /AIS 84/, /LAN 84/, /HER 85/ e na área biológica, para a representação de entidades biológicas e imagens 3D do corpo humano /MEA 85/.

| NOME | LOCAL DE DESENVOLVIMENTO | FORMA DE REPRESENTAÇÃO ADOTADA | ENTRADA DO SISTEMA BASEADA EM | INTERAÇÃO COM O SISTEMA |
|---|---|---|---|---|
| TIPS-1 | Hokkaido University (Japão) | CSG | CSG | batch |
| GRIN | IBM (EUA) | CSG | CSG | interativo |
| PADL | University of Rochester (EUA) | CSG | CSG | interativo |
| SYNTHAVISION | MAGI (EUA) | CSG | CSG | interativo |
| GMSolid | General Motors (EUA) | CSG + B-rep | CSG, sweep | interativo |
| CAAD | McAuto (EUA) | B-rep | Sweep | |
| EUCLID (F) | Matra/Datavision (França) | B-rep | CSG, sweep | interativo/batch |
| EUCLID (S) | Fides Co. (Suiça) | B-rep | CSG | batch |
| GEOMAP | University of Tokyo (Japão) | B-rep | CSG | batch |
| GEOMED | University of Stanford (EUA) | B-rep | CSG | interativo |
| DESIGN | MDSI (EUA) | B-rep | CSG | interativo |
| ROMULUS | Ewan & Sutherland (EUA) | B-rep | CSG, Sweep, B-rep | interativo |

FIG. 2.12 Sistemas de Modelagem Geométrica contemporâneos (continua)

| NOME | LOCAL DE DESENVOLVIMENTO | FORMA DE REPRESENTAÇÃO ADOTADA | ENTRADA DO SISTEMA BASEADA EM | INTERAÇÃO COM O SISTEMA |
|---|---|---|---|---|
| BUILD | University of Cambridge(Inglaterra) | B-rep | CSG, B-rep | interativo |
| HICAD | Lab. Hitachi (Japão) | CSG + B-rep | CSG | interativo |
| GLIDE | University of Carnigie (EUA) | B-rep | CSG, Sweep | interativo |
| SOLIDS ENGINE | Phoenix Data System (EUA) | OCTREE | CSG | interativo |

FIG. 2.12 Sistemas de Modelagem Geométrica contemporâneos

# 3 BANCO DE DADOS E SISTEMAS DE MODELAGEM DE SÓLIDOS

## 3.1 Introdução

A princípio, na década de 70, quando surgiram os primeiros sistemas de modelagem geométrica, não houve a preocupação de se utilizar os conceitos de banco de dados. Quando havia necessidade, os bancos de dados eram desenvolvidos para aplicações pequenas e específicas. Os problemas surgiram quando um grande volume de dados necessitava ser armazenado de forma duplicada, acarretando problemas de consistência no sistema como um todo ou, quando o formato dos dados era modificado, exigindo alterações nos programas de aplicação. Além disso, surgiu a preocupação de como gerenciar os tipos de dados gráficos e não-gráficos de forma eficiente, rápida, concisa e sem muito esforço.

Os motivos para a utilização de Sistemas de Gerência de Banco de Dados (SGBDs) em aplicações de projeto foram, basicamente, os mesmos que levaram ao desenvolvimento dos SGBDs comerciais. São eles:

- o aumento no volume de dados,
- a segurança dos dados,
- a consistência dos dados e
- a possibilidade de concorrência no uso dos dados.

Um SGBD pode manter e controlar os tipos de dados gráficos e não-gráficos e seus relacionamentos como também permitir uma melhor

integração dos sistemas de modelagem geométrica em sistemas CAD/CAM. Assim, os dados podem ser mantidos sem redundância (ou com redundância controlada) e totalmente independentes dos programas que os usam.

O uso de banco de dados em aplicações de projeto pode melhorar a performance de utilização dos dados no sistema. Os requisitos necessários para a gerência dos dados nas aplicações de projeto diferem das aplicações convencionais e as técnicas desenvolvidas para estes sistemas não suportam diretamente os tipos de dados não-convencionais /SID 80/, /HAR 82/, /STA 86/.

As peculiaridades de aplicações de projeto geraram a necessidade de vários estudos dirigidos ao desenvolvimento de novos sistemas de gerência de banco de dados voltados para os sistemas CAD ou a realização de mudanças em SGBDs existentes para suportar aplicações não-convencionais.

Cabe salientar que o principal problema no uso de SGBDs para CAD /STA 86/ está na inflexibilidade de se modificar e estender a visão da realidade modelada no banco de dados.

As tendências atuais sobre banco de dados para sistemas CAD foram relatadas por Buchmann /BUC 84/, Katz /KAT 85/ e Stanley /STA 86/. Muitas das características apresentadas pelos autores foram implementadas em banco de dados dedicados /FIS 79/, /ULF 81/, /PAT 82/ ou incorporadas a SGBDs existentes /DIT 85/.

Neste capítulo será dada uma introdução a banco de dados e feito um estudo sobre as vantagens e o uso atual de sistemas de banco de dados em modelagem de sólidos.

## 3.2 Conceitos Básicos de Banco de Dados

### 3.2.1 Definição de Banco de Dados

Um banco de dados é definido por Date /DAT 82/ como "uma coleção de dados operacionais armazenados e usados pelos sistemas de aplicação de uma empresa específica". Dados operacionais são referentes às operações da empresa e não incluem dados de entrada ou saída, ou qualquer informação temporária. A figura 3.1 ilustra como é composto um ambiente de banco de dados.

O SGBD representado na figura serve como interface entre o banco de dados e os programas de aplicação e/ou usuários. As principais funções de um SGBD são /DAT 82/:

1) fornecer diferentes vistas dos dados;
2) manter os dados consistentes;
3) manter a segurança dos dados;
4) possibilitar a recuperação de dados;
5) proteger os dados contra falhas do sistema;
6) permitir acesso concorrente aos dados.

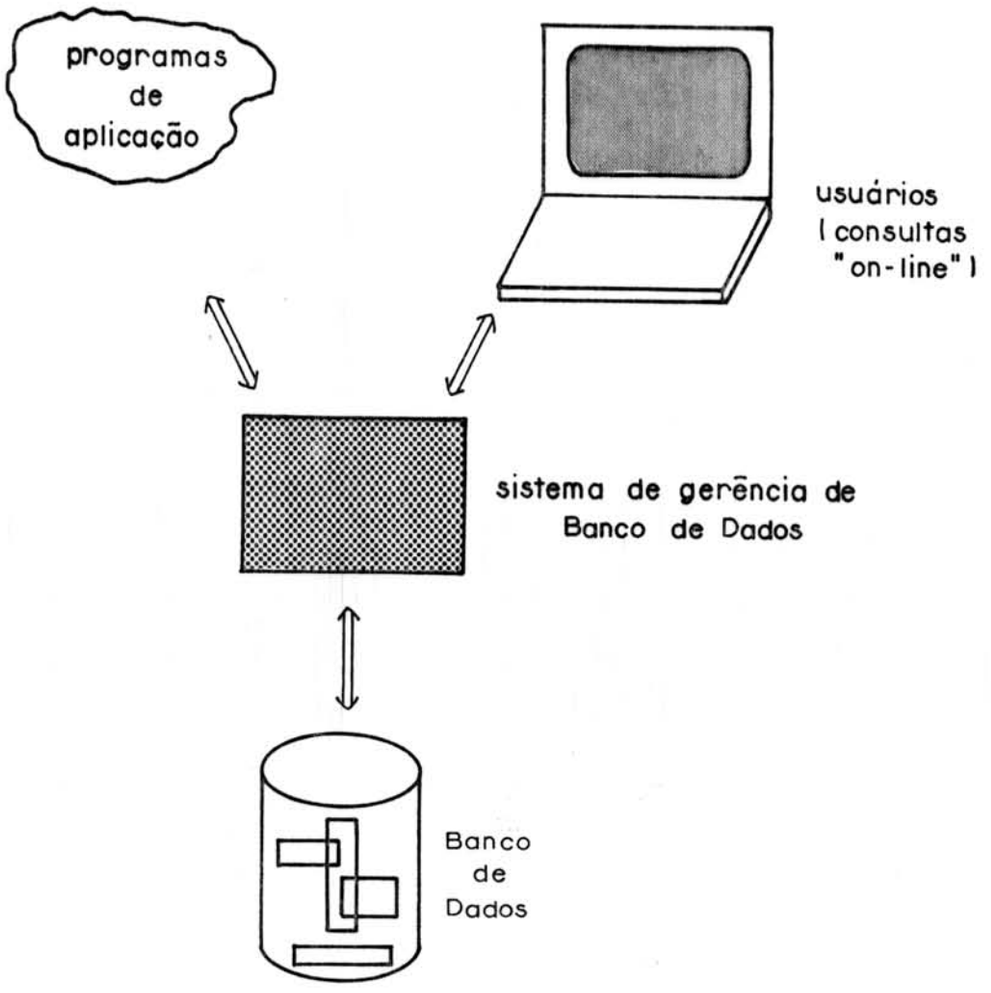



FIG. 3.1 Ambiente de Banco de Dados

Um SGBD é encarregado de gerenciar todos os dados do sistema, qualquer que seja sua natureza, e apresenta, geralmente, duas linguagens: Linguagem de Definição de Dados, conhecida como DDL ("Data Definition Language") e Linguagem de Manipulação de Dados, conhecida como DML ("Data Manipulation Language") /DAT 82/.

Um sistema de banco de dados pode ser dividido em três níveis de abstração conhecidos como esquemas: esquema conceitual, esquema lógico e esquema interno /ROB 81/.

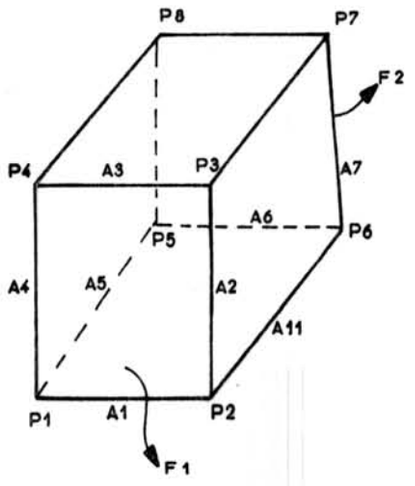
O esquema conceitual fornece uma visão da realidade a ser representada no banco de dados, através de um modelo adequado (p. ex., entidade-relacionamento), independente de SGBD. O esquema lógico traduz os dados relevantes às necessidades do usuário representando-os conforme as restrições de uma estrutura de dados particular (p. ex., relações, hierarquias). Já o esquema interno permite definir como os dados são armazenados, isto é, as técnicas de implementação de registros que serão empregadas, determinação dos caminhos de acesso aos dados, etc.

### 3.2.2 Abordagens Clássicas de Banco de Dados

Será dada uma breve introdução às três abordagens mais conhecidas:

- abordagem relacional,
- abordagem hierárquica e
- abordagem em rede.

Nas três seções seguintes serão apresentados exemplos baseados na representação de um sólido simples nas três abordagens. A figura 3.2 mostra o referido sólido.




Arestas:
A1=$\overline{P1P2}$ A7=$\overline{P6P7}$
A2=$\overline{P2P3}$ A8=$\overline{P7P8}$
A3=$\overline{P3P4}$ A9=$\overline{P8P4}$
A4=$\overline{P4P1}$ A10=$\overline{P8P5}$
A5=$\overline{P1P5}$ A11=$\overline{P2P6}$
A6=$\overline{P5P6}$ A12=$\overline{P3P7}$

Faces:
F1=A1A2A3A4
F2=A7A8A10A6
F3=A1A11A6A5
F4=A3A9A8A12
F5=A4A5A10A9
F6=A3A12A7A11

FIG. 3.2 Sólido exemplo para a representação segundo as três abordagens

3.2.2.1 Abordagem Relacional

Esta abordagem permite descrever o banco de dados através de conceitos matemáticos simples como relações entre conjuntos. Uma relação pode ser vista como uma tabela onde as linhas correspondem às tuplas e as colunas aos atributos. O conceito de domínio é importante e consiste no conjunto de valores de onde são retirados os valores que constam em uma dada coluna da tabela /DAT 82/.

A abordagem relacional foi inicialmente descrita por Codd /COD 70/. Os principais motivos que levaram ao desenvolvimento da abordagem relacional foram a necessidade de dar uma base teórica para o processo de organização dos dados e permitir aos usuários a manipulação das informações através de uma interface de alto nível simples e natural /COD 70/, /DAT 82/. A abordagem relacional tem como características:

- permitir que a tradução do esquema conceitual para a estrutura relacional seja feita minimizando perdas de informações;
- permitir operações de álgebra relacional que constituem uma linguagem relacionalmente completa;
- permitir acesso aos dados sem precisar conhecer a estrutura física pois é independente dos caminhos de acesso.

A utilização de bancos de dados relacionais é crescente em sistemas gráficos principalmente por ser uma organização facilmente compreensível, onde a disposição dos dados em forma de relações torna a sua manipulação mais flexível e diferentes visões lógicas podem ser mantidas.

A redundância neste tipo de abordagem é inerente, embora haja meios que tentam minimizar a duplicação de dados. Contudo, a principal característica desta organização é que as tuplas de duas relações diferentes são associadas somente por valores de dados de um domínio comum entre elas.

A figura 3.3 apresenta uma estrutura relacional para o sólido da figura 3.2. A relação CUBO contém, para cada face, uma identificação e a informação de visibilidade ou não desta. A relação FACES identifica as arestas que compõem cada face. A descrição de cada aresta em termos de seus pontos inicial e final consta na relação de ARESTAS. Já a relação PONTOS especifica cada ponto em termo de suas coordenadas X, Y e Z.

CUBO

| FACES | VISIBILIDADE |
|---|---|
| F1 | VIS |
| F2 | INV |
| F3 | INV |
| F4 | VIS |
| F5 | INV |
| F6 | VIS |

FACES

| #F | AR1 | AR2 | AR3 | AR4 |
|---|---|---|---|---|
| F1 | A1 | A2 | A3 | A4 |
| F2 | A7 | A8 | A10 | A6 |
| F3 | A1 | A11 | A6 | A5 |
| F4 | A3 | A9 | A8 | A12 |
| F5 | A4 | A5 | A10 | A9 |
| F6 | A3 | A12 | A7 | A11 |

ARESTAS

| #AR | PI | PF |
|---|---|---|
| A1 | P1 | P2 |
| A2 | P2 | P3 |
| A3 | P3 | P4 |
| A4 | P4 | P1 |
| A5 | P1 | P5 |
| A6 | P5 | P6 |
| A7 | P6 | P7 |
| A8 | P7 | P8 |
| A9 | P8 | P4 |
| A10 | P8 | P5 |
| A11 | P2 | P6 |
| A12 | P3 | P7 |

PONTOS

| #P | XREL | YREL | ZREL |
|---|---|---|---|
| P1 | 0 | 0 | 0 |
| P2 | +2 | 0 | 0 |
| P3 | 0 | +2 | 0 |
| P4 | -2 | 0 | 0 |
| P5 | 0 | -2 | +2 |
| P6 | +2 | 0 | 0 |
| P7 | 0 | +2 | 0 |
| P8 | -2 | 0 | 0 |

FIG. 3.3 Relações representativas da figura 3.2

### 3.2.2.2 Abordagem Hierárquica

A abordagem hierárquica permite estruturar os dados em árvores. O topo da árvore é conhecido como raiz e os nodos terminais como folhas. Cada nodo, exceto a raiz, possui um nodo pai e todos os nodos (exceto os nodos folhas) possuem um ou mais nodos filhos. É fundamental, na visão hierárquica dos dados, que qualquer nodo só tenha significado quando visto no contexto. Assim, um nodo dependente não pode existir sozinho sem estar ligado a um nodo pai.

A abordagem hierárquica se adapta perfeitamente a uma realidade que possa ser modelada hierarquicamente, isto é, onde existe uma relação 1:n entre os objetos. Fora disto existe uma falta de simetria /DAT 82/ na representação e manipulação dos dados, trazendo várias consequências como o tempo despendido e o esforço necessário para resolver problemas que são introduzidos pela estrutura e que não são intrínsecos às questões sendo formuladas.

A figura 3.4 ilustra parcialmente a representação dos dados gráficos numa estrutura hierárquica baseada na figura 3.2. A raiz representa o CUBO. Cada face é representada por um nodo cujos filhos são as arestas que a definem. Cada aresta, por sua vez, aponta para os seus pontos inicial e final representados por nodos folhas.

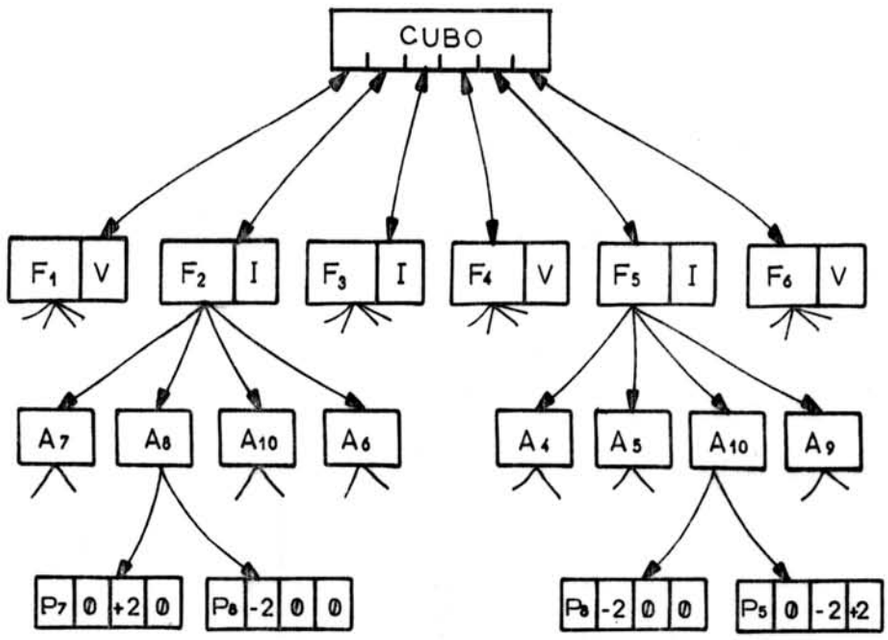



FIG. 3.4 Hierarquia representativa da figura 3.2.

3.2.2.3 Abordagem em Rede

A abordagem em rede apresenta uma estrutura de dados mais geral que a da abordagem hierárquica, onde cada nodo, exceto a raiz, pode ter mais do que um pai /DAT 82/. Os dados são representados como um grafo dirigido.

A abordagem em rede permite modelar uma realidade n:m de forma mais direta do que a hierárquica. Ela é mais simétrica que a hierárquica /DAT 82/ e mais flexível, contudo, os programas que a manipulam são complexos e as operações (inserção, remoção, atualização de registros) não são tão naturais para o usuário. O tempo de acesso às informações pode-se tornar lento devido aos ponteiros mantidos pela organização.

Quando forem mantidos registros conectores, que representam as associações, o usuário deve decidir o melhor caminho para buscar as informações, isto é, deve "navegar" sobre as estruturas e deve estar ciente de como está organizado seu banco de dados.

A figura 3.5 apresenta uma estrutura em rede para as faces F1 e F3 do sólido da figura 3.2. F1A1, F1A2, F1A3 e F1A4 são registros conectores representando as associações entre faces e arestas. É mantido um anel entre os nodos da rede com seus conectores. Estes nodos representam as faces, arestas e pontos.

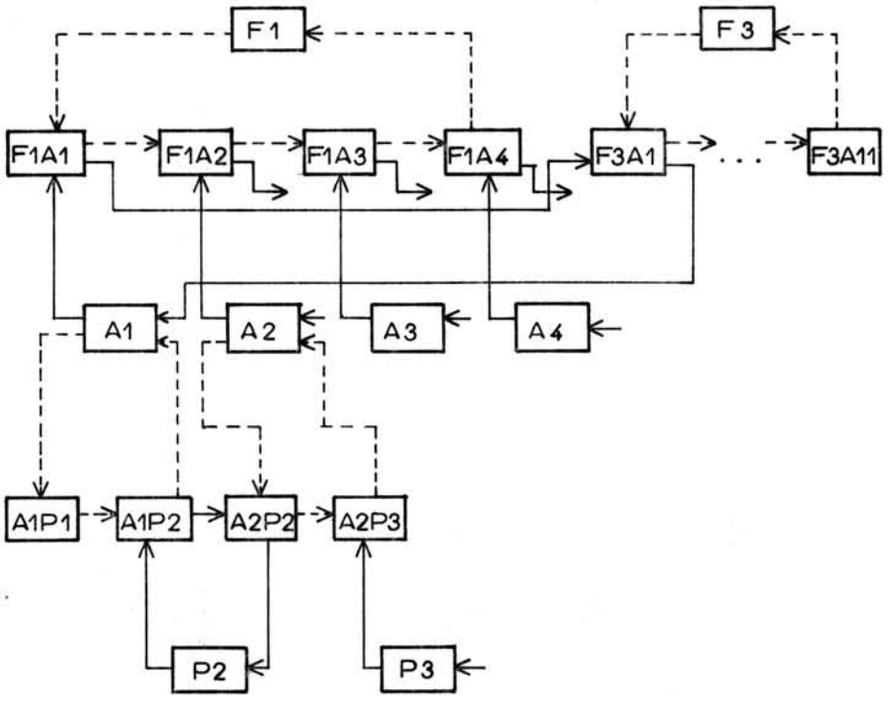



FIG. 3.5 Representação em rede da figura 3.2.

### 3.2.3 Abordagem Semântica

Além das abordagens já descritas, outras formas de modelagem de dados têm sido desenvolvidas. Esses modelos, chamados "semânticos", constituem um nível mais alto de abstração, auxiliando o processo de tradução da visão que o usuário tem do mundo real para a perspectiva de banco de dados, tentando diminuir a perda de informações nesta tradução.

Os componentes de um modelo semântico são /GRA 79/:

- um conjunto de elementos sintáticos básicos e um repertório de regras para a construção de uma estrutura;
- uma descrição do esquema com regras semânticas para a representação das informações através dos elementos sintáticos básicos;
- um repertório de operações aplicáves aos elementos sintáticos básicos (p. ex., insere, remove, modifica, busca,...);
- restrições de integridade com regras intrinsecamente estabelecidas no modelo utilizado;
- restrições de integridade com regras explicitamente descritas pelo usuário.

Serão apresentados os seguintes modelos semânticos:

- Modelo Semântico Básico /SCH 75/,
- Entidade-Relacionamento /CHE 77/,
- Abstração de Dados /SMI 77a/, /SMI 77b/,

. Modelo Semântico de Grabowski e Eigner /GRA 79/.

O Modelo Semântico Básico e o de Abstração de Dados baseiam-se em extensões da abordagem relacional, porque, em ambos, objetos são representados através de relações. O modelo Entidade-Relacionamento é centrado em três elementos básicos: entidade, atributo e relacionamento. Grabowski e Eigner apresentam uma proposta de modelo semântico, voltado para aplicações CAD, baseado em elementos dos três modelos citados acima.

3.2.3.1 Modelo Semântico Básico

Proposta por Schmid e Swenson /SCH 75/, esta abordagem considera que o mundo real seja constituído de objetos (entidades) e associações. Os objetos podem ser dependentes ou independentes, onde os objetos independentes existem por si só. Uma desvantagem desta proposta é que objetos e associações são distintos: associações, não sendo objetos, não podem ter propriedades e nem participar em outras associações. A Figura 3.6 mostra os elementos sintáticos básicos do modelo proposto por Schmid e Swenson, conforme apresentado em /GRA 79/.

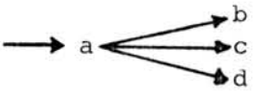
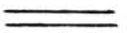
| denominação dos elementos | representação gráfica |
|---|---|
| objeto independente |   |
| característica simples |   |
| característica de repetição |   |
| característica complexa |   |
| associação |  |

FIG. 3.6 Elementos sintáticos básicos da abordagem semântica básica

A figura 3.7 mostra a representação no modelo semântico básico do exemplo onde um sólido é uma instância de um primitivo (bloco, cilindro ou cone).

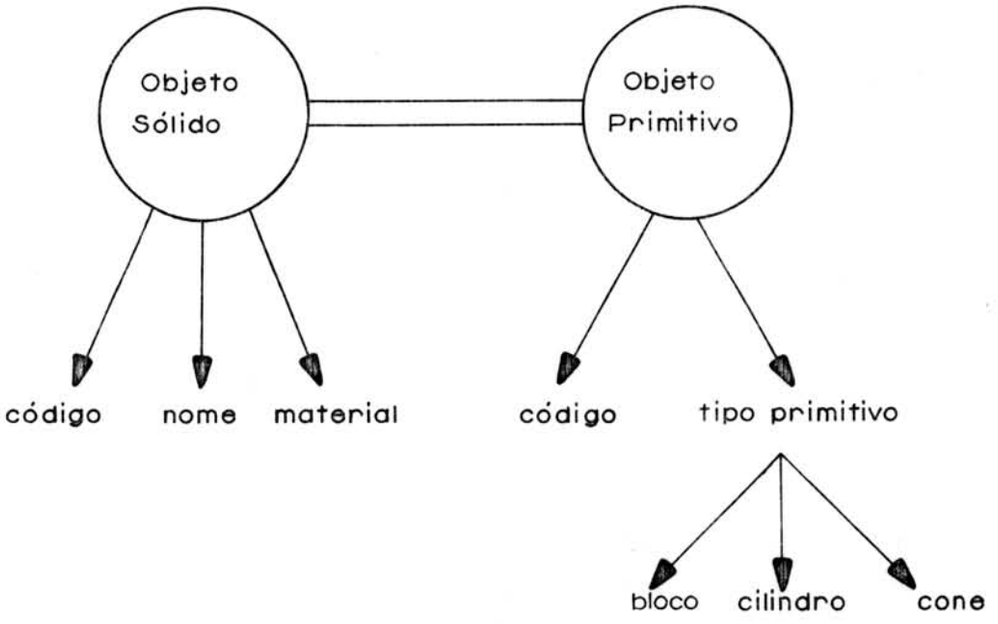



FIG. 3.7 Exemplo utilizando semântica básica.

### 3.2.3.2 Entidade-Relacionamento

Proposta por Chen /CHE 77/, onde os termos entidade e relacionamento substituem os termos objeto independente e associação definidos por Schmid e Swenson, a abordagem Entidade-Relacionamento não difere muito do modelo semântico básico a não ser na terminologia /DAT 82/. Contudo, é a que tem gerado maior interesse entre os pesquisadores de banco de dados. A figura 3.8 mostra os elementos sintáticos básicos da abordagem Entidade-Relacionamento.

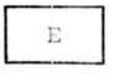
| denominação dos elementos | representação gráfica |
|---|---|
| entidade |   |
| relacionamento |   |
| atributo |   |
| conjunto de valores |  |

FIG. 3.8 Elementos sintáticos básicos do modelo E/R

A figura 3.9 mostra a representação na abordagem Entidade-Relacionamento do exemplo apresentado na seção anterior.

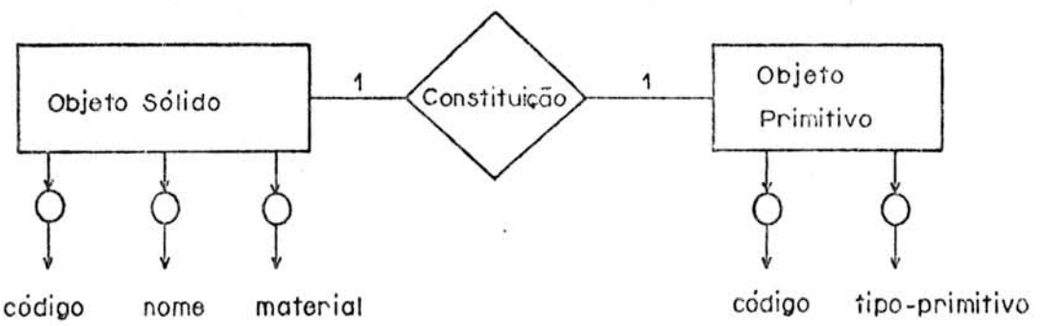



FIG. 3.9. Exemplo utilizando o modelo E/R.

3.2.3.3 Abstração de Dados

Proposta por Smith e Smith /SMI 77a/, /SMI 77b/, esta estrutura semântica suporta dois tipos de abstrações que auxiliam o usuário na transformação das informações vistas no mundo real para o banco de dados: agregação e generalização. Abstrair significa eliminar certos detalhes não relevantes à aplicação e se concentrar naqueles detalhes relevantes à aplicação.

Os conceitos de agregação e generalização são importantes para entender o mecanismo de definição dos dados do banco de dados. O tipo de abstração dito agregação mantém o relacionamento entre vários objetos num único objeto de mais alto nível. Este objeto único é conhecido como objeto agregado. O tipo generalização é uma abstração onde um conjunto de objetos com propriedades semelhantes pode ser visto como um objeto de mais alto nível com detalhes suprimidos.

A abordagem relacional básica não suporta o tipo generalização. No entanto, um banco de dados relacional pode ser projetado para

suportar estes dois tipos de dados e a estrutura genérica /SMI 77b/ onde é feito um tratamento hierárquico de relações. A idéia de agregação e generalização é uma ferramenta poderosa para a definição dos dados. A abstração generalização, capaz de representar as informações em forma de relações, é formalizada por um conjunto de propriedades invariantes que devem ser satisfeitas por todas as relações do banco de dados. A definição formal da estrutura genérica apresentada em /SMI 77b/ é apropriada para as relações definidas por Codd /COD 70/.

Os tipos agregação e generalização podem ser representados graficamente de forma ortogonal onde o plano da página representa a agregação e o plano perpendicular à página representa a generalização.

As Figuras 3.10 e 3.11 mostram a representação das abstrações agregação e generalização usando o exemplo das seções anteriores.

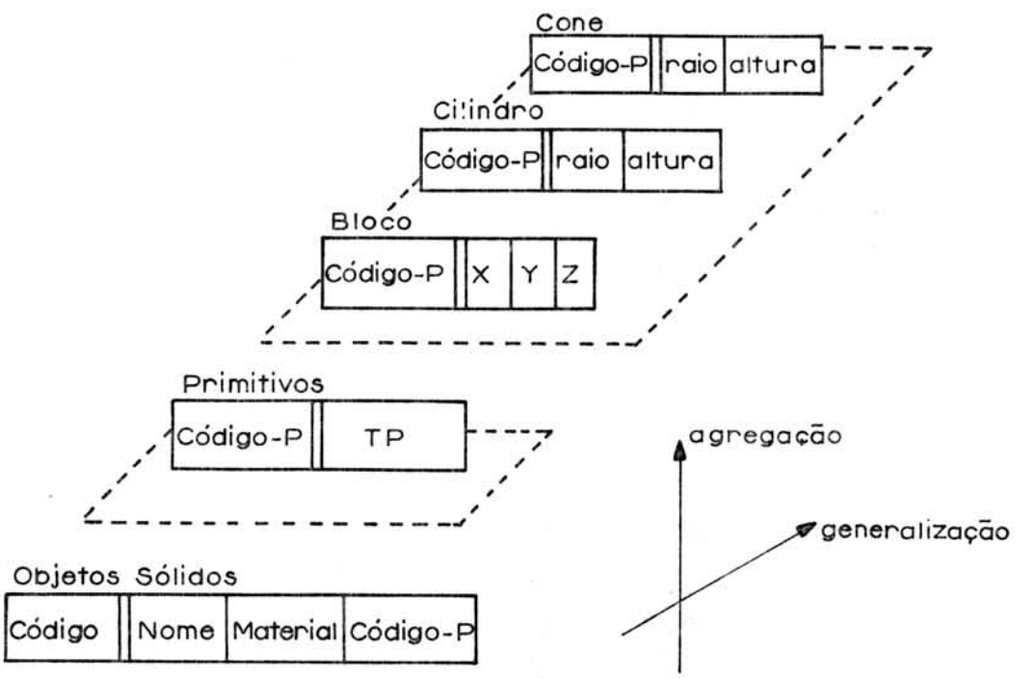



FIG. 3.10 Representação Gráfica.

```
var Objetos-Sólidos:
    generic
    of
    aggregate  [Código]
          Código: identificador do sólido;
            nome: nome do sólido;
        material: material do sólido;
        Código-P: Key Primitivo;

var Primitivos:
    generic
        TP=(Bloco, Cilindro, Cone)
    of
    aggregate  [Código-P]
        Código-P: identificador de primitivo;
              TP: tipo de primitivo;

var Bloco:
    generic
    of
    aggregate  [Código-P]
        Código-P: identificador de primitivo;
               x: largura;
               y: altura;
               z: profundidade;
```

```
var Cilindro:
    generic
    of
    aggregate [Código-P]
        Código-P: identificador de primitivo;
            raio: raio da base;
          altura: altura;

var Cone:
    generic
    of
    aggregate [Código-P]
        Código-P: identificador de primitivo;
            raio: raio da base;
          altura: altura;
```

FIG. 3.11 Utilização da Estrutura Genérica para o exemplo citado nas seções anteriores

3.2.3.4 Modelo Semântico de Grabowski e Eigner

O modelo está baseado na teoria proposta por Chen /CHE 77/, Abrial /ABR 74/, Smith e Smith /SMI 77a/ e Schmid e Swenson /SCH 75/. Um sistema de banco de dados relacional denominado RELCAD foi implementado para suportar esta proposta, que é voltada para aplicações CAD. Os elementos sintáticos básicos do modelo são apresentados na figura 3.12.

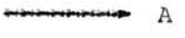
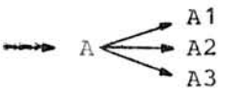
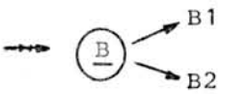
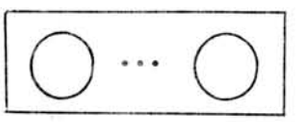
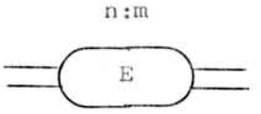
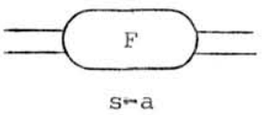
| Denominação dos Elementos | Representação Gráfica |
|---|---|
| definição de domínio |   |
| tipo atributo simples |   |
| tipo atributo chave primária |   |
| tipo atributo chave candidata |   |
| tipo atributo chave estrangeira |   |
| tipo atributo composto |   |
| tipo atributo repetição |   |
| tipo atributo complexo |   |
| tipo entidade |   |
| tipo classe de entidades diferentes |  |
| tipo associação independente |   |
| tipo associação dependente |   |

FIG. 3.12 Elementos sintáticos básicos do modelo proposto /GRA 79/

A quantidade de elementos sintáticos torna o projeto de banco de dados mais difícil. Em compensação, o usuário tem uma ferramenta voltada para a realidade das aplicações de projeto.

A figura 3.13 mostra a representação no modelo semântico proposto por Grabowski e Eigner do exemplo utilizado nas seções anteriores.

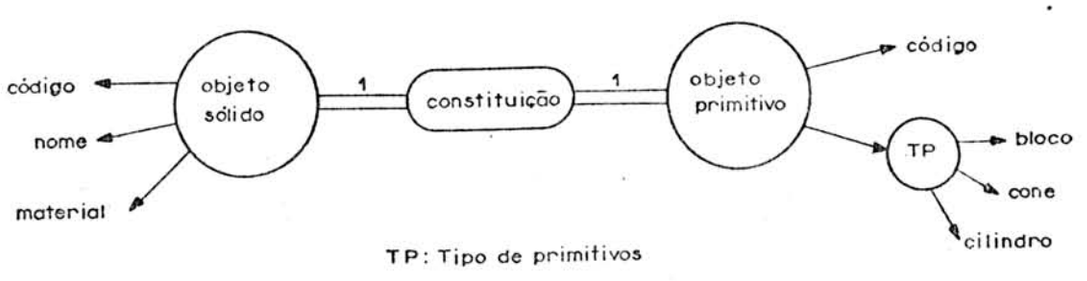



FIG. 3.13 Exemplo utilizando o modelo proposto.

## 3.3 Uso de Banco de Dados em Modelagem de Sólidos

### 3.3.1 Vantagens da Utilização de Banco de Dados

A função de um banco de dados em sistemas CAD/CAM é manter as informações gráficas e não-gráficas usadas pelos diversos programas de aplicação.

Sistemas CAD/CAM integrados com SGBDs permitem, como vantagens básicas, que:

- o caminho de acesso aos dados seja determinado automaticamente pelo SGBD;
- o tempo de desenvolvimento de novos programas seja reduzido pois o SGBD provê independência de dados;
- as informações se mantenham íntegras de tal forma que modificações em registros e campos possam ser realizadas com segurança;
- a manutenção dos programas de aplicação seja relativamente fácil.

A comunidade de CAD/CAM vem reconhecendo os bancos de dados como um meio ideal para integrar uma variedade de aplicações CAD com dados comuns a elas, permitindo uma comunicação mais rápida entre as mesmas, além de garantir a consistência dos dados e evitar erros de conversão /BUC 84/. Entretanto, o desenvolvimento deste tipo de banco de dados está-se processando de forma lenta justamente porque as aplicações CAD/CAM continuam sendo desenvolvidas independentes umas das outras utilizando arquivos convencionais próprios.

Entender como as técnicas existentes podem ser aplicadas, refinadas ou generalizadas para os novos ambientes de projeto tem-se tornado um dos maiores desafios para os pesquisadores de banco de dados. Novas facilidades estão sendo embutidas em sistemas de banco de dados /KAT 85/ para melhor servir as aplicações de projeto. Estas incluem:

a) manipulação de objetos complexos (objetos complexos são um conjunto de dados relacionados logicamente e manipulados

como uma entidade única /HAS 82/ /DIT 85/), incluindo controle de versões e diferentes representações do mesmo objeto;

b) suporte para dados não estruturados, ou seja, o sistema deve ser capaz de tratar ítens de dados de tamanho variável /HAS 82/;

c) tratamento de transações longas através de novos mecanismos para verificação de estados consistentes do banco de dados /NEU 82/ /DIT 85/.

Como visto, sistemas de banco de dados disponíveis comercialmente não são suficientes, na sua forma original, para tratar aplicações de projeto. A figura 3.14, baseada em /SID 80/ e /HAR 82/ mostra algumas questões e o tratamento que cada uma recebe conforme sejam aplicações ditas convencionais ou não-convencionais.

| QUESTÕES | TRATAMENTO | |
|---|---|---|
| | APLICAÇÕES CONVENCIONAIS | APLICAÇÕES NÃO-CONVENCIONAIS |
| 1) Realidade a ser modelada | A realidade é estática: entidades e relacionamentos são conhecidos de forma clara | Existem dois tipos de informação:<br>a) dados do objeto a ser projetado - dados gráficos: não são conhecidos no início do projeto, mas ao longo dele<br>b) dados sobre o ambiente do projeto - dados não-gráficos: são conhecidos desde o início |
| 2) Natureza do esquema lógico | É estático e compilativo | É dinâmico |
| 3) Frequência de alterações | Sobre valores dos dados: constante e moderada<br>Sobre estruturas de dados: esporádica e baixa | Sobre valores e estruturas dos dados: alta na fase de projeto |
| 4) Quem realiza as alterações | Sobre os valores dos dados: usuário;<br>Sobre as estruturas de dados: administrador de banco de dados | Projetista |
| 5) Tipos de ítens | Ítens atômicos representados por tipos numéricos e caracteres | Ítens atômicos e estruturados considerados como tipos particures |

FIG.3.14 Questões e o tratamento dado pelas aplicações convencionais e não-convencionais (continua)

| QUESTÕES | TRATAMENTO | |
|---|---|---|
| | APLICAÇÕES CONVENCIONAIS | APLICAÇÕES NAO-CONVENCIONAIS |
| 6) Complexidade na modelagem dos dados | Grande número de entidades. As estruturas são simples | Grande número de entidades e relacionamentos pode levar a uma estrutura mais complexa |
| 7) Tipos de atividades | Essencialmente constantes | Constantes, dependendo do ciclo de projeto |
| 8) Volume de informações | Grande | Aumenta ao longo do projeto |
| 9) Restrições de integridade | Podem ser verificadas automaticamente pelo sistema | Dificuldade de verificação automática. |
| 10) Estrutura do banco de dados | Com poucos tipos de registros e relacionamentos simples | Com muitos tipos de registros e relacionamentos complexos |
| 11) Segurança | Proteção contra falhas do sistema ou acessos inadequados | Proteção contra falhas do sistema ou acessos inadequados |
| 12) Múltiplas vistas | Correspondem a porções do esquema conceitual e são definidas como privadas | São necessárias para separar atividades de projeto |

FIG. 3.14 Questões e o tratamento dado pelas aplicações convencionais e não-convencionais

As abordagens clássicas de banco de dados estão sofrendo alterações para permitir as facilidades descritas acima. As mais empregadas são a abordagem relacional e em rede. Já na abordagem hierárquica, os dados se associam uns aos outros de uma maneira "vertical" justamente porque este tipo de abordagem representa um relacionamento 1:n, não permitindo que registros de um mesmo nível se relacionem. A modelagem da informação dentro de uma estrutura hierárquica pode levar a uma alta redundância de dados como pode ser visto na figura 3.4 da seção anterior. Considerando-se que o volume de informações aumenta durante a vida do projeto, isto faz com que a abordagem hierárquica não seja considerada como ideal, apesar de ser apropriada para sistemas onde a realidade é de natureza hierárquica.

### 3.3.2 O Uso das Abordagens

#### 3.3.2.1 Utilização da Abordagem em Rede

TORNADO /ULF 81/, /ULF 82/ é um SGBD em rede para aplicações de projeto em engenharia mecânica. É usado integrado a um sistema de modelagem de sólidos e de superfícies esculpidas. O sistema tem como características gerais:

- tratar objetos e registros de tamanho variável;
- tratar relacionamentos n:m, 1:n ou n:1;
- permitir acesso por nome aos objetos;

- possuir funções especiais para percorrer estruturas em árvore;
- tratar classes de objetos complexos;
- ser monousuário.

A figura 3.15 ilustra a arquitetura do sistema.

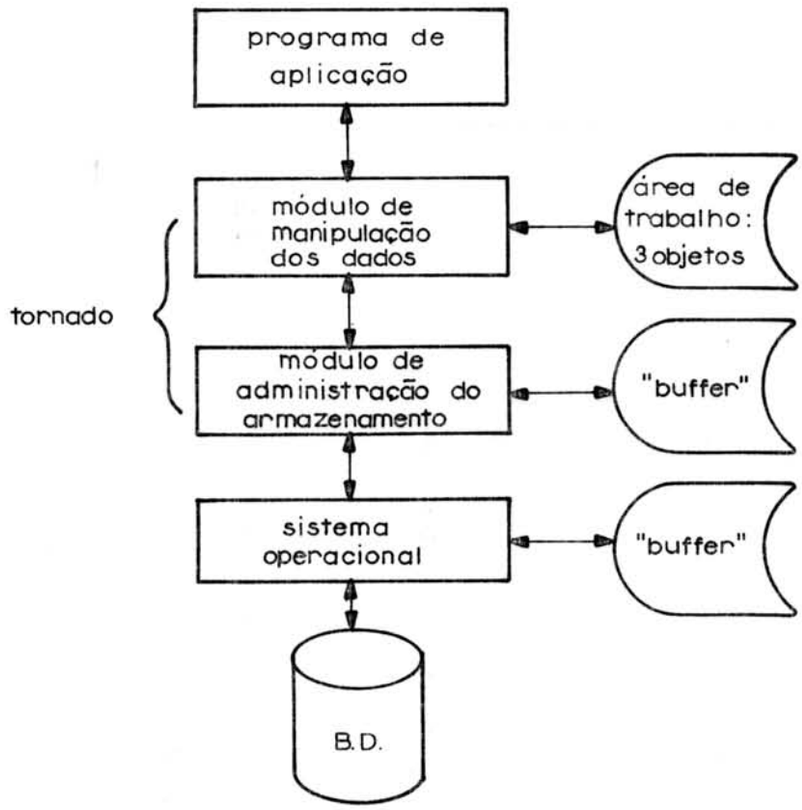



FIG. 3.15 Arquitetura do sistema TORNADO

O módulo de manipulação de dados trata dos objetos em conjunto e o módulo de administração do armazenamento administra os dados no banco de dados.

Outro sistema baseado na abordagem em rede é o PHIDAS /FIS 79/ que é usado junto com o sistema CAD/CAM conhecido por PHILIKON. O banco de dados age como centralizador das informações. A figura 3.16 apresenta a arquitetura geral do sistema que segue basicamente a proposta CODASYL /DBT 71/.

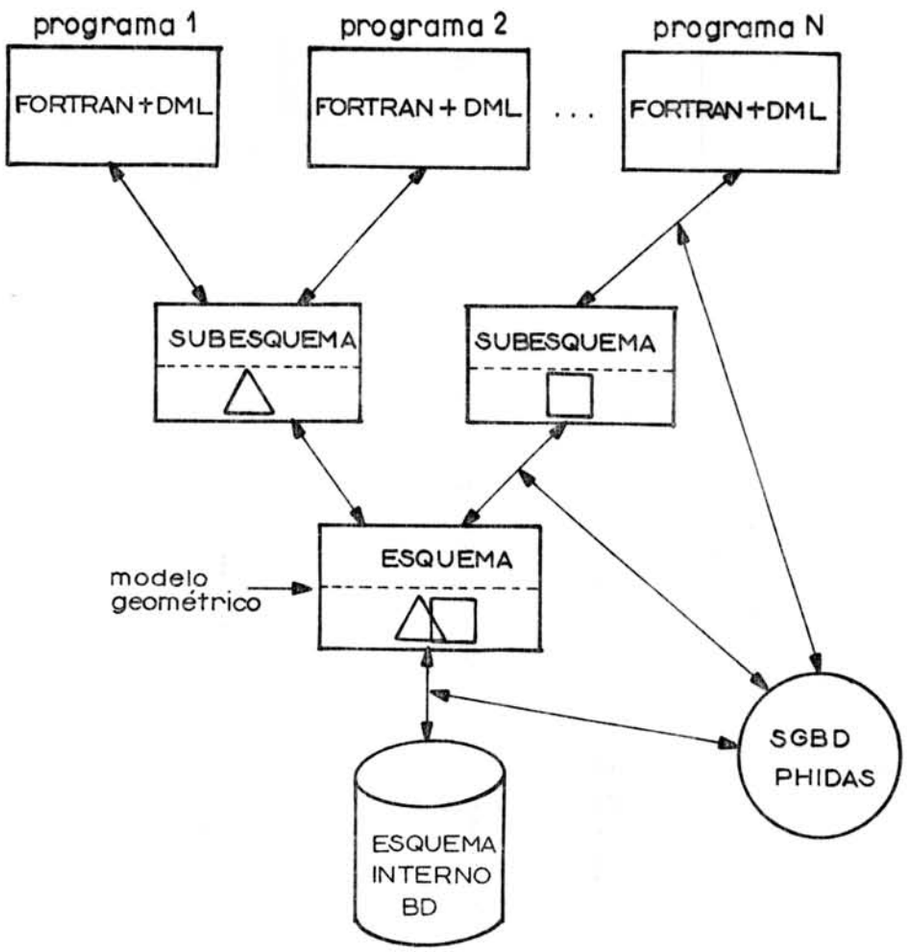



FIG. 3.16 Arquitetura do PHIDAS

3.3.2.2 Utilização da Abordagem Relacional

Haskin e Lorie /HAS 82/ consideram a abordagem relacional apropriada para armazenar e recuperar dados de aplicações complexas,

como é o caso da modelagem de sólidos, onde é impossível se saber quais tipos de dados e relacionamentos deverão ser utilizados. Através de uma organização relacional é possível definir dinamicamente novas relações ou adicionar novos atributos aos já existentes. Isto traz grandes vantagens quando se está tratando de uma realidade que não é estática.

A maioria das formas de representação de sólidos /REQ 80/ (cf. capítulo 2) foram definidas para permitir uma manipulação eficiente dos dados com propósitos de combinar objetos primitivos gerando outros mais complexos através de operações adequadas. O preço pago para manter esta eficiência é a redundância de dados e a proliferação de ponteiros necessários para manter os relacionamentos. A abordagem relacional pode facilitar o armazenamento dos dados e permitir um acesso concorrente por diversas aplicações como, também, pode suportar extensões (para incluir novos domínios) sem invalidar as aplicações que a utilizam.

Extensões na abordagem relacional foram propostas por Haskin e Lorie /HAS 82/. Foi introduzida a vantagem de uma vista hierárquica dos dados enquanto mantêm a simplicidade e a flexibilidade da estrutura relacional. O custo da extensão de bancos de dados relacionais, com a implementação de estruturas de dados gráficos e não-gráficos, é bem menor do que se essas mesmas estruturas fossem implementadas em sistemas de bancos de dados dedicados. Entretanto, deve-se ter em mente as limitações de um SGBD convencional.

O ARDBID ("A Relational Database for Interactive Design") /SHE 86/ é um SGBD relacional que pode ser usado em várias aplicações de projeto porque mantêm três tipos de dados:

- dados gráficos 2D,
- dados gráficos 3D e
- dados não-gráficos,

decompostos em três bancos de dados como mostra a figura 3.17. As vantagens de manter uma decomposição estrutural do banco de dados /CHA 80/ são a eficiência de recuperação e armazenamento dos dados, a segurança do banco de dados e a possibilidade de realizar processamento paralelo das várias estruturas mantidas no banco de dados.

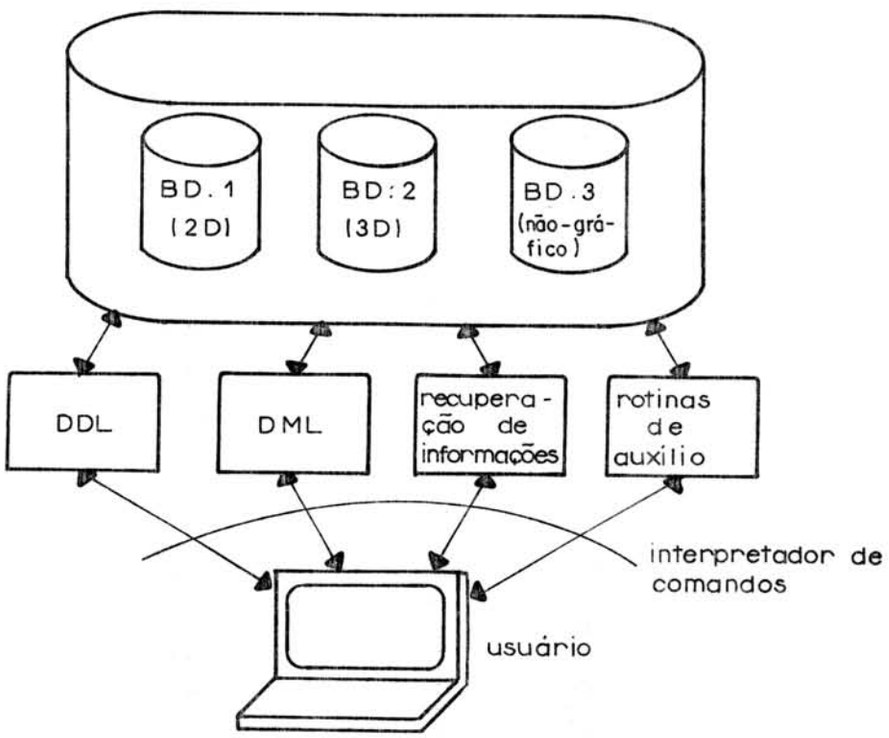



FIG. 3.17 Arquitetura do ARDBID

O interpretador de comandos chama o módulo desejado (DDL, DML, tratamento de questões, rotinas de auxílio) retomando o controle após a tarefa ser realizada. O processo de definição (através do módulo DDL) para tipos de dados gráficos e não-gráficos é realizado separadamente. O módulo DML permite ao usuário atualizar e editar suas relações. O módulo de recuperação de informações simplesmente busca os dados nas relações sem que eles sejam manipulados. O módulo rotinas de auxílio contém procedimentos específicos ou de otimização dependentes da aplicação.

3.3.2.3 Estrutura Genérica

O trabalho proposto por Lee e Fu /LEE 83/ baseia-se na idéia de agregação e generalização e foi por eles denominado de Abordagem Relacional Genérica. A abordagem implementa esquemas altamente estruturados em relações (estrutura genérica /SMI 77a/, /SMI 77b/) com um tratamento uniforme para todos os tipos de objetos: individuais, agregados ou genéricos. A forma de representação de sólidos adotada foi a CSG.

Os autores apresentam um método sistemático de converter a forma CSG em uma estrutura genérica. A conversão é conceitualmente fácil de se realizar, embora a estrutura genérica exija definições rigorosas para conter informações estruturais. O método de Lee e Fu baseia-se num procedimento que converte automaticamente uma gramática BNF, que especifica a árvore CSG, em uma estrutura genérica.

No trabalho /LEE 83/ é apresentada uma estrutura genérica para peças mecânicas, onde cada peça é constituída por um objeto genérico de um dentre três tipos: primitivo, combinado ou transformado. Cada primitivo, por sua vez, é um objeto genérico de um dentre três tipos: cubo, cilindro, cone. A estrutura genérica é parcialmente mostrada a seguir.

```
var peça-mecânica:
    generic
    of
    aggregate [PM#]
        PM# : número de identificação;
        ...
        O#  : key Objeto;
    end

var Objeto:
    generic
        CC = (Primitivo, Obj-combinado, Obj-transformado)
    of
    aggregate [O#]
        O# : número de identificação;
        ...
        CC : categoria construtiva;
    end
```

```
var Primitivo:
    generic
        CG = (Cubo, cilindro, Cone)
    of
    aggregate [O#]
        O# : número de identificação;
        ...
        CG : categoria geométrica;
    end

    ...
```

Para definir, manipular e controlar as relações foi usada a linguagem SQL. A linguagem teve que ser modificada e incluídas facilidades para o tratamento das relações que representam a estrutura genérica.

### 3.3.3 Requisitos de Banco de Dados para um Modelador de Sólidos

Os requisitos para aplicações não-convencionais são diferentes daqueles para as aplicações convencionais, principalmente porque aquelas são caracterizadas por um grande número de relações complexas entre os dados e por possuirem tipos diferentes de dados. Vários requisitos de banco de dados para CAD foram levantados e relatados em diversas publicações. Pode-se listar os requisitos sob quatro aspectos:

a) Quanto à complexidade dos objetos manipulados, o banco de dados deve permitir:

- representação hierárquica dos objetos, segundo o processo construtivo;
- representação de múltiplos relacionamentos entre objetos;
- representação de dados gráficos e não-gráficos;
- representação de versões diferentes de projeto do mesmo objeto;
- representação de um objeto em diferentes níveis de detalhe.

b) Quanto ao processo de projeto dos objetos, o banco de dados deve permitir:

- um processo de evolução natural e interativo, sem impor restrições sobre a sequência de atividades;
- operações explícitas de retorno a situações anteriores do projeto;
- armazenamento de informações semânticas para fins de documentação.

c) Quanto à segurança e consistência dos dados, o banco de dados deve:

- possibilitar a especificação de restrições de integridade de certas características dos objetos durante o processo de projeto;
- prever a restauração dos dados no caso de falhas do sistema;

- manter mecanismos de controle de acesso aos dados;
- garantir a realização correta de atualizações quando do uso concorrente dos dados.

d) Quanto à linguagem de manipulação dos dados, o banco de dados deve manter:

- uma interface com linguagem hospedeira;
- possibilidades de consultas "ad hoc".

Stanley e Anderson /STA 86/ apresentam dois tipos de arquitetura para sistemas CAD com banco de dados: o banco de dados como um módulo centralizador ou um módulo executivo centralizador. No primeiro (figura 3.18(a)), todos os aplicativos utilizam um banco de dados compartilhado e, no segundo (figura 3.18(b)), um módulo executivo se encarrega de fazer a comunicação entre programas, usuários e banco de dados.

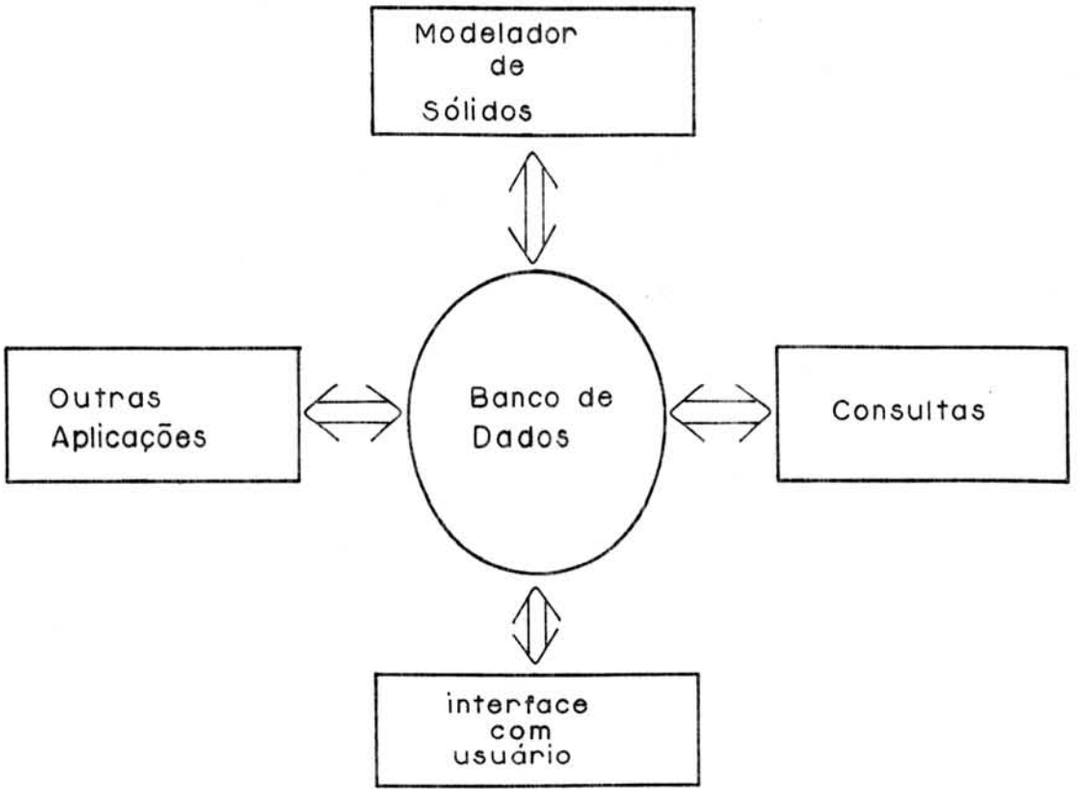



FIG. 3.18 Localização do banco de dados em sistemas CAD:
(a) como módulo centralizador
(b) gerenciado por um módulo executivo

# 4 NÚCLEO DE MODELAGEM DE SÓLIDOS

"Que os nossos esforços desafiem as impossibilidades" (Charles Chaplin)

## 4.1 Introdução

No capítulo 2 foram apresentados os fundamentos de modelagem de sólidos, mais especificamente as formas de representação presentes na maioria dos modeladores desenvolvidos até o momento, e as operações de modelagem necessárias no processo de construção dos objetos. As formas de representacão CSG e B-rep foram comparadas quanto à facilidade de realização das operações de modelagem e exibição. Naquele capítulo foi também introduzida a forma de representação conhecida como octree, que apenas recentemente passou a ser utilizada em modelagem de sólidos /MEA 82a/. Além do fato desta forma ser utilizada em outras áreas (Robótica, Medicina, ...), certas operações de modelagem de sólidos são realizadas de modo simplificado.

O Núcleo de MOdelagem de Sólidos (N-MOS), aqui descrito, utiliza uma abordagem híbrida para representar os objetos: CSG e octree. A forma de representação CSG permite manter uma descrição estruturada (hierárquica) dos objetos, segundo o processo construtivo. A octree é utilizada para representar explicitamente as porções do espaço ocupadas pelos objetos. Desta forma pode-se tratar isoladamente componentes de um objeto, ou operar com ele como um todo.

O N-MOS está inserido no contexto de um sistema de modelagem de sólidos cujas cinco funções principais podem ser observadas na figura 4.1.

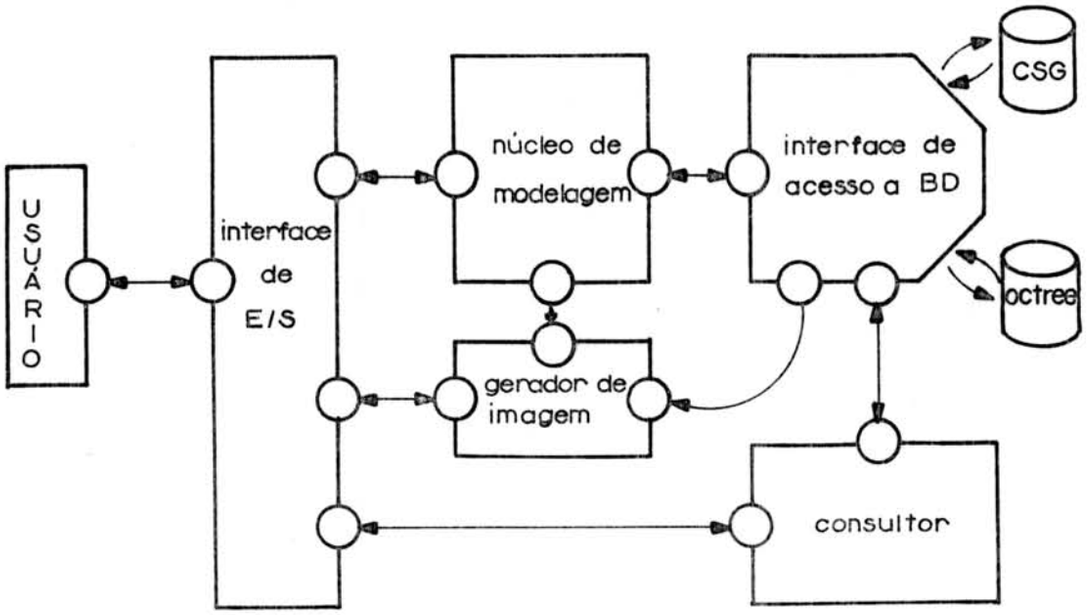



FIG. 4.1 Estrutura funcional do Sistema de Modelagem de Sólidos

A Interface de Entrada e Saída serve de comunicação entre o usuário e o sistema. Permite a exibição das saídas gráficas e de resultados obtidos na execução do sistema. O usuário interage com o sistema informando os dados exigidos e escolhendo através de cardápios. O teclado alfanumérico é utilizado como:

- seletor: para indicação de opção escolhida nos cardápios;
- entrada de texto: para a entrada de informações alfanuméricas;
- quantificador: para especificação de valores necessários às operações;

. posicionador: para indicação de posição dos sólidos no universo.

A Interface de Acesso ao Banco de Dados é o elemento que possibilita o acesso aos dados gráficos e não-gráficos que são armazenados na forma de relações. As informações gráficas e não-gráficas estão separadas na base de dados. Esta metodologia de decomposição estrutural foi descrita em /CHA 80/ e tem a vantagem de reduzir o tempo de acesso e de resposta (ver seção 3.3.2.2., exemplo do ARDBIB). A interface deve ser capaz de interpretar os dois tipos de dados e fazer acesso à porção apropriada do banco de dados.

O Núcleo de Modelagem é encarregado das funções de modelagem de sólidos. As funções são caracterizadas pelas operações de manipulação dos sólidos (combinações e transformações realizadas sobre octrees) e geração de primitivos. A figura 4.2 mostra os módulos pertencentes ao N-MOS. Cada módulo será melhor descrito na seção 4.4.

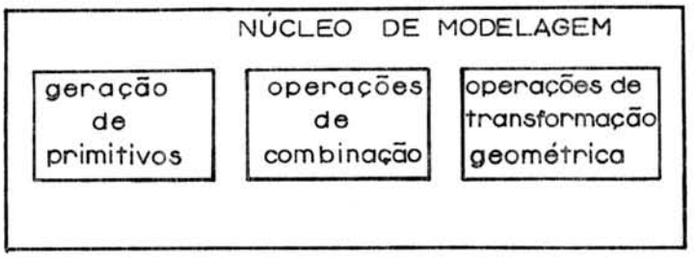



FIG. 4.2 Módulos funcionais do N-MOS

O Gerador de Imagem tem por função exibir imagens dos objetos. Estas imagens são tomadas diretamente da octree que representa o objeto a ser exibido.

O Consultor permite a realização de dois tipos de busca de informações a respeito dos objetos presentes no banco de dados. O primeiro deles se refere a informações já armazenadas no banco de dados, cuja busca é realizada diretamente pela interface de acesso ao banco de dados. O segundo tipo de consultas se refere a propriedades ou verificações que exigem processamento a parte (por exemplo: cálculo de volume, verificação de interferência espacial). Os dois módulos encarregados de realizar os cálculos no consultor: Cálculo de Propriedades e Verificação de Interferência serão descritos na seção 4.5.

## 4.2 Modelo Geométrico

O modelo geométrico é estruturado de forma hierárquica onde objetos complexos podem ser definidos a partir de objetos mais simples pré-definidos ou gerados pelo sistema. A hierarquia imposta na definição do modelo geométrico do N-MOS é descrita por:

- Objetos Primitivos (ou sólidos básicos);
- Objetos Transformados;
- Objetos Combinados;
- Entidades.

### 4.2.1 Objeto Primitivo

Objetos primitivos ou sólidos básicos são formas tridimensionais simples, pré-definidas ou geradas pelo sistema, e utilizadas como

base na definição de formas tridimensionais mais complexas.

Os objetos primitivos fazem parte de um subconjunto limitado do espaço euclidiano: os poliedros. Requicha e Voelcker /REQ 80/, /REQ 77/ estabelecem certas características que, em termos computacionais, os sólidos devem possuir. As características essenciais são:

a) Rigidez: o sólido deve possuir forma invariante, independente da sua posição ou orientação no espaço. Uma instância da forma pode ser vista como uma forma resultante de uma operação (p. ex., rotacionar a forma) sobre a forma original;

b) Solidez (ou Regularidade): o sólido deve possuir interior claramente delimitado e nenhum de seus limites pode ter partes isoladas ou "penduradas" (figura 4.3);

c) Finitude: o sólido deve ocupar uma parte finita do espaço;

d) Fechamento: esta propriedade deve ser mantida para certas operações de transformação geométrica (translação, rotação) e de combinação (união, intersecção, diferença) pois quando aplicadas a sólidos devem produzir outros sólidos;

e) Representabilidade: deve ser possível utilizar certos aspectos do sólido (p. ex., um número finito de faces) para representá-lo em computadores;

f) Determinismo de limites: os limites do sólido devem determinar, de forma não-ambígua, o sólido como um todo.

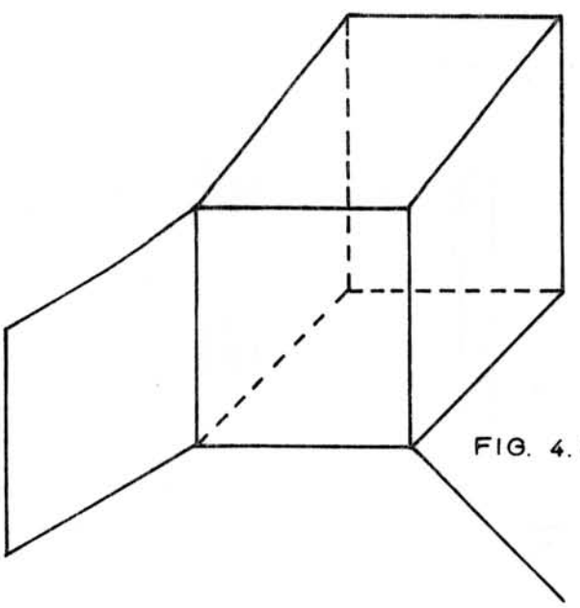


FIG. 4.3 Um subconjunto não considerado sólido regular

A seleção de quais conjuntos de primitivos uma aplicação deve possuir difere de uma para outra. A principal abordagem /ELL 78/, /BRA 75/ trabalha com uma "biblioteca" de volumes básicos - cilindros, cubóides, etc - que podem ser parametrizados, e operações de combinação podem ser aplicadas sobre eles para criar formas complexas. Assim, para aplicações diferentes existiriam bibliotecas diferentes. A maioria dos sistemas /VOE 77/, /BRO 82/, /BOY 82/, /VOE 78/, /FIT 81/, /TOK 83/, /LEE 83/ utiliza como primitivos básicos: o cubo, cilindro, esfera, prisma e cone, sendo todos baseados na forma de representação CSG. Já o SISGRIM /GUL 85/, sistema gráfico interativo para o projeto de peças mecânicas desenvolvido no PGCC, dispõe como primitivos básicos: o paralelepípedo, prisma com base triangular, prisma com base sextavada, cilindro facetado, pirâmide com base triangular e cunha.

O N-MOS pretende ser um modelador interativo de propósito geral. Considerando, então, os resultados encontrados na literatura, definiu-se o conjunto de objetos primitivos como segue:

- o cubo,
- o prisma,
- a pirâmide,
- o cone,
- a esfera e
- o cilindro.

Todos os objetos primitivos serão gerados, de acordo com os parâmetros especificados pelo usuário, e armazenados num sistema normalizado $[-1, 1]$, para X, Y e Z, e englobados num universo cúbico (ver seção 4.3).Formalmente, o objeto primitivo parametrizado pode ser descrito através de uma BNF como na figura 4.4.

<objeto primitivo parametrizado> ::= <cubo> | <prisma> |
<pirâmide> | <cone> |
<esfera> | <cilindro>

FIG. 4.4 Representação do objeto primitivo parametrizado

### 4.2.2 Objeto Transformado

Um objeto transformado é definido a partir de um objeto qualquer que sofra transformações geométricas. As transformações geométricas são operações de rotação, translação e mudanças de escala (redução e ampliação). O objeto transformado pode ser um objeto primitivo, um objeto combinado (ver seção 4.2.3), ou outro objeto transformado (figura 4.5).

```
<obj. transformado> ::= < transformação>  < obj. transformadc > |
                        < transformação>  < obj. combinado> |
                        < transformação>  < obj. primitivo>
```

FIG. 4.5 Representação do objeto transformado

### 4.2.3 Objeto Combinado

O objeto combinado é definido por quaisquer dois objetos quando combinados através das operações de conjunto (união, intersecção e diferença). Pode ser composto de três formas: combinação de dois sólidos combinados, combinação de dois sólidos transformados ou combinação de um sólido transformado e um sólido combinado (figura 4.6).

<obj. combinado>::=<obj. transformado><combinação><obj. combinado> |
<obj. combinado><combinação><obj. combinado> |
<obj. transformado><combinação><obj. transformado> |
<obj. combinado><combinação><obj. transformado>

FIG. 4.6 Representação do objeto combinado

4.2.4 Entidade

A entidade corresponde ao objeto final modelado. É definida como sendo um objeto transformado, combinado ou primitivo parametrizado. Possui um nome externo que permite acesso direto pelo usuário (figura 4.7).

<Entidade>::=<obj. transformado> |
<obj. combinado> |
<obj. primitivo parametrizado>

FIG. 4.7 Representação da entidade

4.2.5 Relacionamento entre os Elementos do Modelo Geométrico

O modelo geométrico pode ser representado através de um grafo, conforme visto na figura 4.8, onde as arestas representam a rela-

ção de composição.

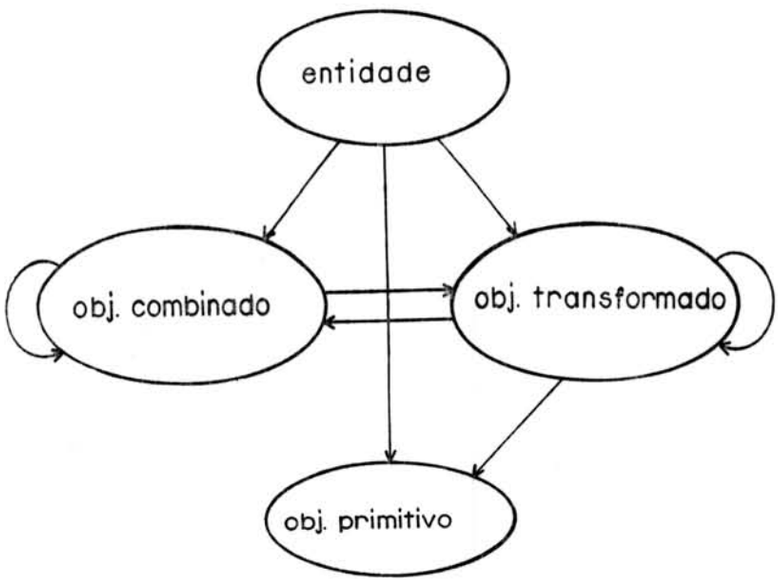



FIG. 4.8 Representação geral do modelo geométrico

Todos os componentes de qualquer elemento do modelo geométrico (entidade, objeto combinado, objeto transformado e objeto primitivo) estão definidos no seu próprio sistema de coordenadas e são transportados para o sistema de coordenadas do universo, podendo ser manipulados e identificados isoladamente pelo usuário.

A definição completa do modelo geométrico é descrita na figura 4.9.

<entidade> ::= <obj. transformado> |
<obj. combinado> |
<obj. primitivo parametrizado>

<obj. transformado>::=<transformação><obj. transformado> |
<transformação><obj. combinado> |
<transformaçao><obj. primitivo parametrizado>

<obj. combinado>::=<obj. transformado><combinação><obj. combinado> |
<obj. combinado><combinação><obj. transformado> |
<obj. transformado><combinação><obj. transformado> |
<obj. combinado><combinação><obj. combinado>

<obj. primitivo parametrizado> ::= <cubo> | <prisma> | <pirâmide> |
<cilindro> | <esfera> | <cone>

<transformação> ::= <translação> |
<rotação> |
<mudança de escala>

<combinação>::= <união regularizada> |
<intersecção regularizada> |
<diferença regularizada>

FIG. 4.9 Definição do modelo geométrico do N-MOS.

Conforme se pode observar, o processo de construção de uma entidade pode ser constituído de várias operações tanto de transformação como de combinação. Para tornar mais eficiente a realização destas operações, mantém-se associada a cada objeto a representação geométrica completa na forma de uma octree. Estas informações geométricas são tratadas como atributos dos objetos, não aparecendo, portanto, na descrição da figura 4.9.

### 4.2.6 O Modelo Geométrico Associado à Estrutura Genérica

A utilização de um sistema de gerência de banco de dados servindo de base para um modelador de sólidos é altamente desejável quando o objetivo deste modelador é satisfazer uma variedade de aplicações, pois vários tipos de consultas podem ser obtidas concorrentemente através dos SGBDs.

A integração do N-MOS a um banco de dados visa possibilitar a expansão futura do sistema de modelagem onde ambos estão inseridos, bem como facilitar a implementação do módulo Consultor. Devido à estrutura hierárquica do modelo geométrico do N-MOS, o conceito de estrutura genérica é conveniente para permitir um tratamento uniforme dos elementos de banco de dados (cf. seção 3.3.2.3). Esta abordagem foi considerada mais adequada na caracterização da abstração de dados de estruturas hierárquicas enquanto representadas como relações /LEE 83/. Como consequência, a estrutura genérica introduzida por Smith e Smith /SMI 77a/, /SMI 77b/ pode ser adotada para suportar os tipos de objetos do N-MOS.

A decomposição de ENTIDADE, OBJETO e PRIMITIVO no plano de generalização é mostrada na figura 4.10.

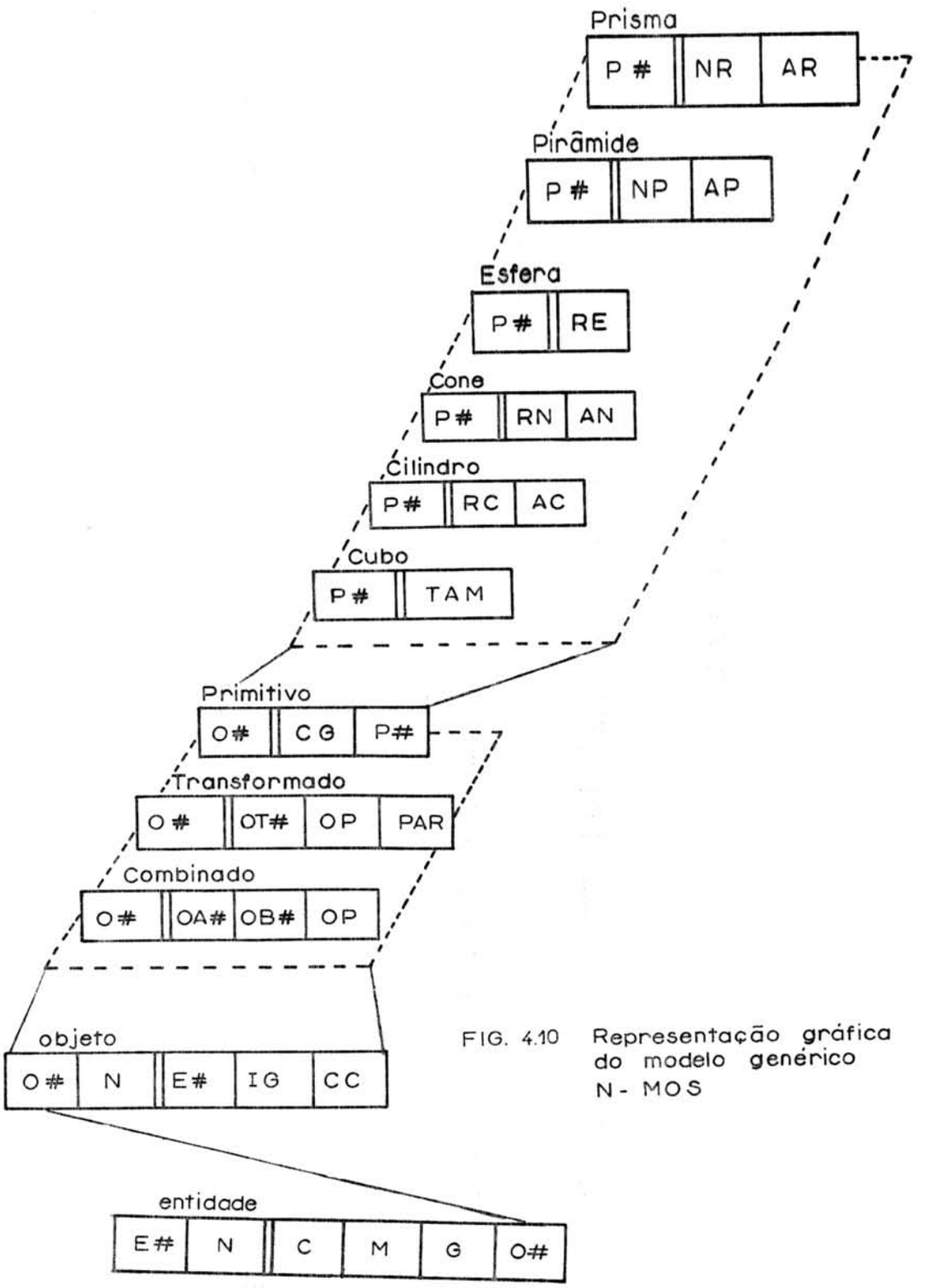



FIG. 4.10 Representação gráfica do modelo genérico N-MOS

Existem seis atributos que devem ser registrados, no plano de agregação, na composição do tipo genérico ENTIDADE: número de identi-

ficação da entidade (E#), nome da entidade (N), cor (C), material (M), grau de acabamento (G) e o número de identificação do objeto (O#) que a entidade representa. O tipo genérico OBJETO é composto pelo número de identificação do objeto (O#) e da entidade (E#), pelas suas informações geométricas (IG) e pela sua categoria construtiva (CC). A categoria construtiva será composta pelos objetos genéricos COMBINADO, TRANSFORMADO e PRIMITIVO. O tipo genérico PRIMITIVO é composto dos atributos: número de identificação do objeto (O#), tipo de categoria geométrica (CG) e identificação da instância de primitiva (P#) que ele representa. A categoria geométrica será composta pelos objetos genéricos referentes ao tipo de primitivo: cubo, cilindro, cone, esfera, pirâmide e prisma. Os objetos Combinado e Transformado apresentam atributos que são chave no objeto OBJETO. Estes atributos representam o código dos objetos que fazem parte da combinação ou transformação. A definição completa dos objetos do modelo é mostrada a seguir.

```
var Entidade:
    generic
    of
    aggregate [E#, N ]
        E# : número de identificação;
        N  : nome;
        C  : cor;
        M  : material;
        G  : grau de acabamento;
        O# : key Objeto;
    end
```

```
var Objeto:
    generic
        CC = (Combinado, Transformado, Primitivo);
    of
    aggregate [O#, N]
        O# : número de identificação;
         N :  nome
        E# : key Entidade;
        IG : representação espacial;
        CC : categoria construtiva;
    end

var Combinado:
    generic
    of
    aggregate [O#]
        O# : número de identificação;
       OA# : Key Objeto;
       OB# : Key Objeto;
       OP  :  união, intersecção, diferença;
    end

var Transformado:
    generic
    of
    aggregate [O#]
        O# : número de identificação;
       OT# : Key Objeto;
```

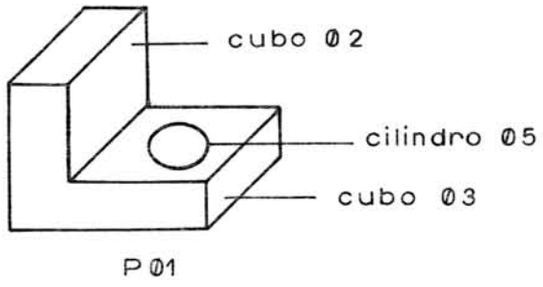
```
        OP :  translação, rotação, escala;
       PAR :  parâmetros da transformação;
    end

var Primitivo;
    generic
        CG:  (Cubo, Cilindro, Cone, Esfera, Pirâmide,  Pris-
             ma);
    of
    aggregate [O#]
        O# : número de identificação;
        CG : categoria geométrica;
        P# : número de identificação;
    end

var Cubo:
    generic
    of
    aggregate [P#]
        P# : número de identificação;
       TAM : tamanho;
    end

var Cilindro:
    generic
    of
    aggregate [P#]
        P# : número de identificação;
```

```
        RC : raio;
        AC : altura;
    end

var Cone:
    generic
    of
    aggregate [ P# ]
        P# : número de identificação;
        RN : raio;
        AN : altura;
    end

var Esfera:
    generic
    of
    aggregate [P#]
        P# : número de identificação;
        RE : raio;
    end

var Pirâmide:
    generic
    of
    aggregate [P#]
        P# : número de identificação;
        NP : número de lados da base;
        AP : altura;
```

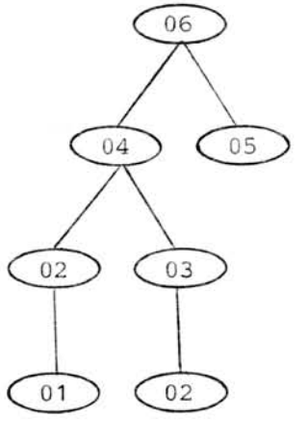
```
    end

var Prisma:
    generic
    of
    aggregate [P#]
        P# : número de identificação;
        NR : número de lados da base;
        AR : altura;
end
```

A figura 4.11 mostra como um sólido pode ser armazenado no banco de dados. Para criar um sólido no banco de dados basta realizar inserções na relação Entidade do mesmo modo que a árvore CSG é construída. As descrições destas entidades são armazenadas em outras relações conforme suas categorias. A linguagem SQL /DAT 82/ é usada como exemplo de linguagem relacional que pode oferecer facilidades na definição de procedimentos e assim reduzir os esforços de programação e projeto (modificações na linguagem SQL devem ser realizadas para permitir definição, controle e manipulação de modelos genéricos altamente estruturados). Um exemplo típico de como listar todos os primitivos que compõem o sólido P01:

```
SELECT CG
FROM Primitivo
WHERE O# IN SELECT O#
            FROM Objeto
            WHERE E# = 'P01'
```




(a) entidade P01




(b) árvore CSG de P01

Entidade

| E# | O# |
|---|---|
| P01 | 06 |

Objeto

| O# | E# | CC |
|---|---|---|
| 01 | P01 | primitivo |
| 02 | P01 | transformado |
| 03 | P01 | transformado |
| 04 | P01 | combinado |
| 05 | P01 | primitivo |
| 06 | P01 | combinado |

Combinado

| O# | OA# | OB# | OP |
|---|---|---|---|
| 04 | 02 | 03 | união |
| 06 | 04 | 05 | dif. |

Primitivo

| O# | CG | P# |
|---|---|---|
| 01 | cubo | 85 |
| 05 | cilindro | 90 |

Transformado

| O# | OT# | OP | PAR |
|---|---|---|---|
| 02 | 01 | rot. | ax,ay,az |
| 03 | 02 | esc. | kx,ky,kz |

Cubo

| P# | TAM |
|---|---|
| 85 | 5.0 |

Esfera

| P# | RE |
|---|---|

Cilindro

| P# | RC | AC |
|---|---|---|
| 90 | 3.0 | 7.0 |

Cone

| P# | RN | AN |
|---|---|---|

Pirâmide

| P# | NP | AP |
|---|---|---|

Prisma

| P# | NR | AR |
|---|---|---|

(c) banco de dados simplificado para o sólido P01:
alguns atributos foram omitidos

FIG. 4.11 Representação de um sólido no banco de dados

## 4.3 Representação do Modelo Geométrico

O modelo geométrico adotado no N-MOS é uma árvore de sólidos construtivos refletindo a sua estrutura hierárquica. Paralelamente, cada objeto referido na árvore é descrito espacialmente através de uma octree. Por esta razão, diz-se que o N-MOS é dotado de uma abordagem híbrida de representação dos objetos. Conforme foi visto na seção anterior, a árvore de sólidos construtivos pode ser modelada segundo a estrutura genérica /SMI 77a/, /SMI 77b/. A descrição espacial de cada objeto é considerada como atributo.

### 4.3.1 Representação da Árvore de Sólidos Construtivos

Para representar a árvore de sólidos construtivos, o N-MOS mantém um conjunto de arquivos convencionais organizados de maneira a resolver satisfatoriamente a forma de representação CSG. O ideal seria gerenciar estes arquivos através de um SGBD que suportasse a estrutura genérica apresentada anteriormente. Porém, como tal SGBD não está disponível, o N-MOS utiliza sete arquivos (todos descritos no capítulo 5) de acesso randômico. Os arquivos criados para implementar o modelo geométrico do N-MOS são: Entidades, Objetos, Combinados, Transformados, Primitivos, Geometria de Objetos e Geometria de Primitivos.

O acesso e recuperação de objetos nestas estruturas são procedimentos simples envolvendo, no entanto, um certo tempo de processamento perceptível pelo usuário.

### 4.3.2. Representação dos Objetos por Octrees

#### 4.3.2.1 Octree : Origem e Definição

Klinger e Alexandridis /KLI 78/, propuseram uma estrutura de dados em árvore mais conhecida como quadtree para as aplicações que exigiam decomposição de figuras. (Estruturas análogas à quadtree foram empregadas por outros autores em problemas diversos daquele de decomposição das figuras /SAM 84/.) As quadtrees foram desenvolvidas para representar regiões numa imagem (exemplo, figura 4.12(a)), e são baseadas na subdivisão sucessiva de uma região em quadrantes, subquadrantes, etc, até que o quadrante esteja inteiramente contido na região ou fora dela (figura 4.12(b)). Uma quadtree pode ser definida como uma representação hierárquica (figura 4.12(c)) de uma estrutura bidimensional onde o quadrante pode estar no estado de preenchido ou não preenchido. Dizendo de outra forma, os pixels pertencentes a um bloco preenchido são "ligados", caso contrário, permanecem "desligados".

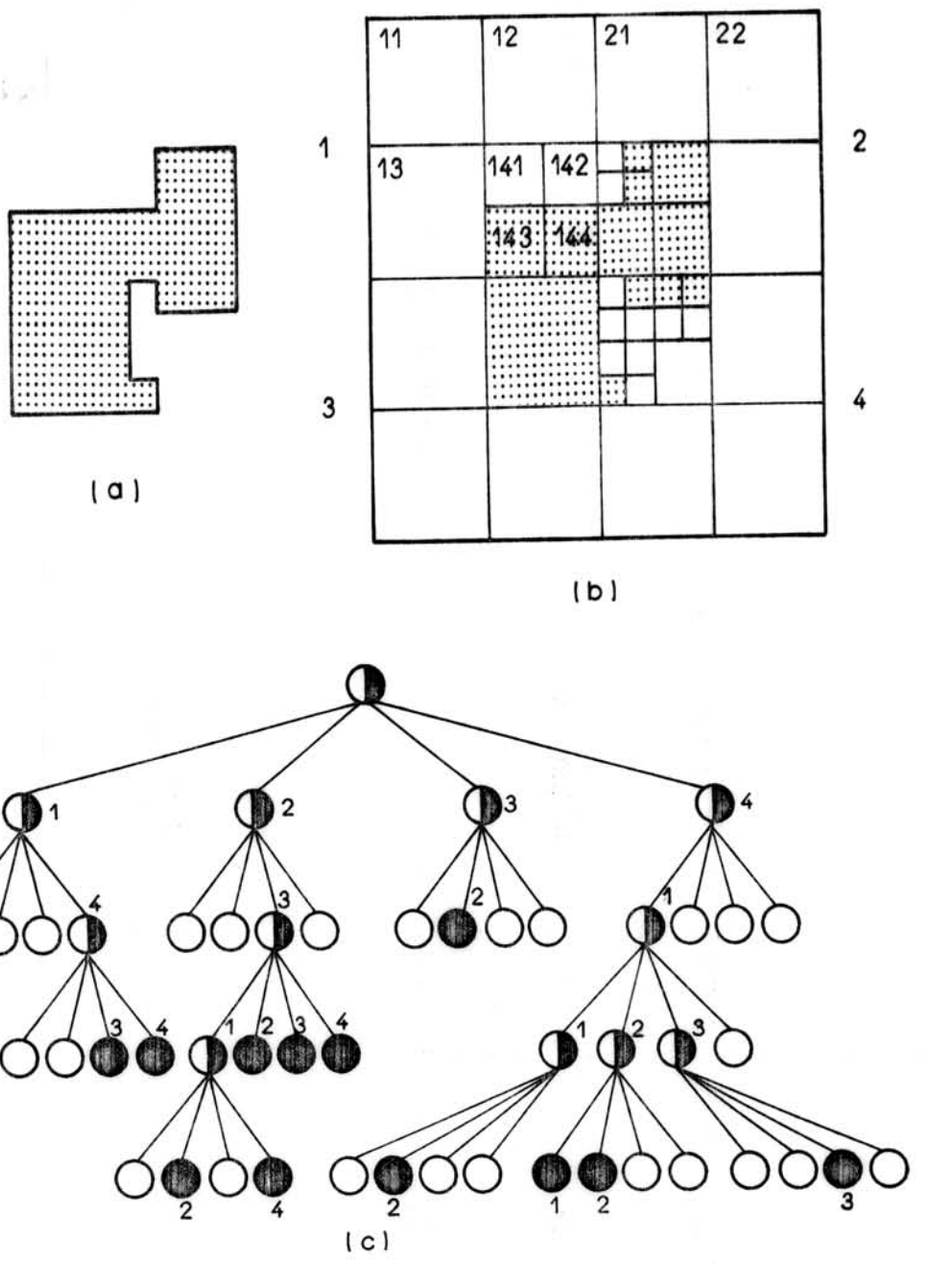



FIG. 4.12 Representação por quadtree:
(a) região a ser representada
(b) decomposição em quadrantes
(c) representação dos quadrantes em árvores

Samet /SAM 84/ faz um estudo geral sobre os quadtrees dando maior ênfase em como representar as regiões, isto é, como especificar os seus limites e como organizar o seu interior. Samet, em outras publicações, descreve algoritmos para realizar remoção de nodos da quadtree /SAM 80c/, conversão de uma região representada por quadtree para

uma representação de limites conhecida por "código em cadeia" /SAM 80a/ e vice-versa /SAM 80b/. Hunter e Steiglitz /HUN 79/ descrevem um algoritmo que permite uma transformação linear genérica (ampliação, redução, translação, rotação) de uma figura qualquer.

Levando-se em consideração o fato de que objetos bidimensionais podem ser representados por meio de quadtrees e que as operações de conjunto e de transformações geométricas (ou transformações lineares) são mais simples, objetos tridimensionais também poderiam ser representados por uma estrutura semelhante: uma estrutura que facilitasse o desenvolvimento de algoritmos eficientes para a geração, manipulação, análise e exibição de objetos tridimensionais. Tal estrutura foi denominada OCTREE /MEA 80/, /MEA 82a/, /JAC 80/: uma extensão da quadtree para objetos tridimensionais.

O espaço de modelagem dos objetos tridimensionais (ou, o universo) é subdividido sucessivamente em oito subregiões conhecidas como octantes. Cada octante, sub-octante, etc, é definido por cubos consistentes (sólidos) ou não consistentes (vazios). Para representar um objeto tridimensional o método se utiliza de uma estrutura em árvore onde cada nodo representa um cubo ou a região do espaço ocupada ou não pelo objeto. Se for nodo folha, a região estará completamente descrita. Caso contrário, o nodo será pai de outros oito filhos formando outra octree. A figura 4.13 mostra a subdivisão do universo em octantes, a representação de um objeto no espaço de modelagem e a estrutura em árvore (octree) do objeto.

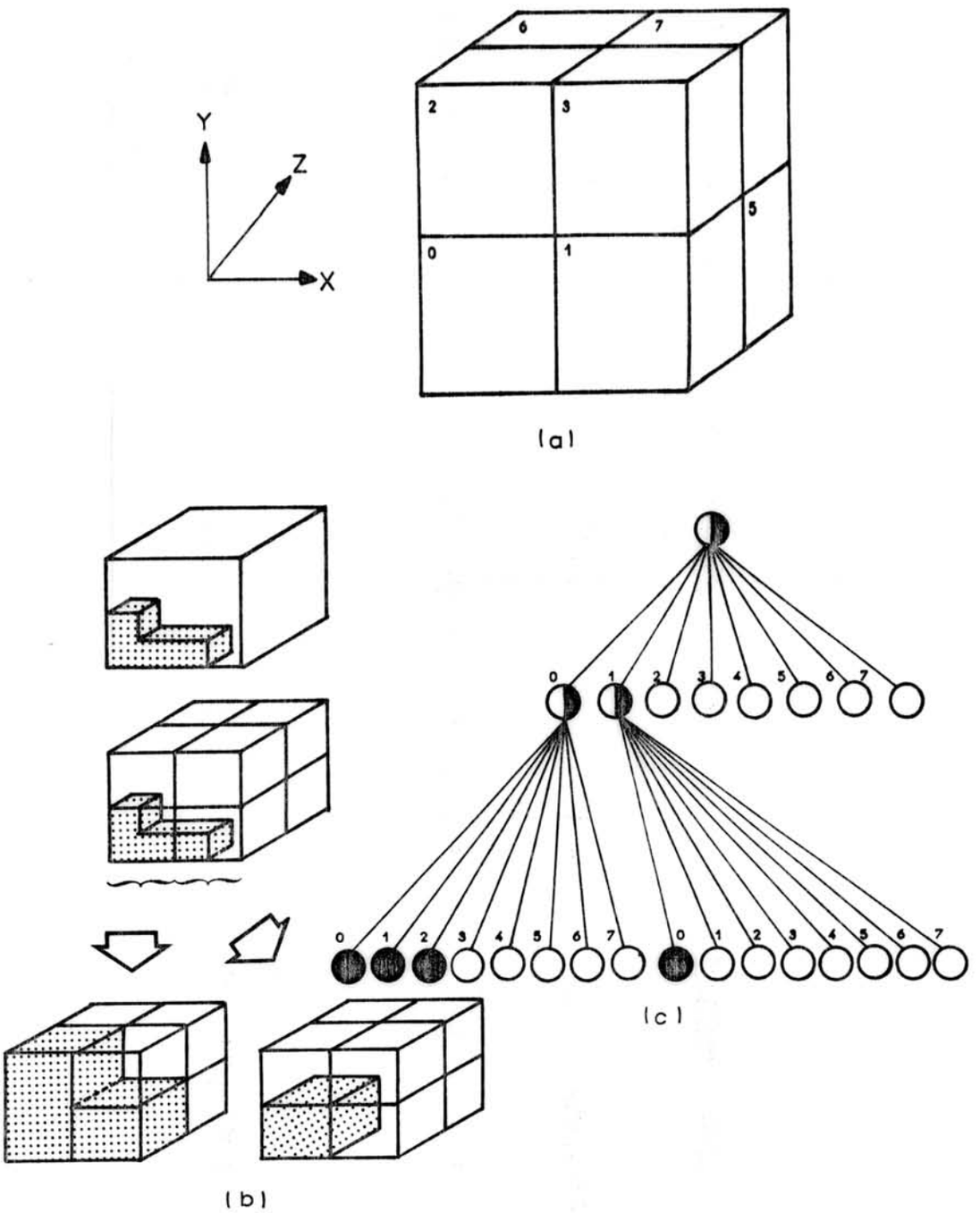



FIG. 4.13 Representação por octree :

(a) subdivisão do universo em octantes
(b) representação do objeto 3D no espaço de modelagem
(c) estrutura em árvore: octree do objeto 3D

A representação por octree possui diversas vantagens /MEA 80/, /MEA 82a/, /CAR 85/:

a) qualquer objeto, côncavo ou convexo, com ou sem espaços vazios no seu interior, pode ser representado pela aproximação de pequenos cubos;

b) as propriedades geométricas tais como área de superfície, volume, centro de gravidade e interferência são facilmente calculadas;

c) apenas um conjunto pequeno de algoritmos para manipulação e análise é necessário para os diversos objetos;

d) não há necessidade de técnicas matemáticas sofisticadas para tratar objetos mais complexos;

e) operações de ponto flutuante, multiplicação e divisão de inteiros não são usadas nos algoritmos que implementam as operações de conjunto e de transformações geométricas;

f) os algoritmos podem ser implementados em hardware, com vários processadores executando as operações em paralelo;

g) os objetos representados são exibidos simplesmente percorrendo-se as árvores em uma ordem específica, dependendo do ponto de vista;

h) algoritmos para a remoção de superfícies ocultas não necessitam de rotinas de classificação ou pesquisa, pois os objetos são pré-classificados espacialmente, isto é, percorre-se a árvore num sentido predeterminado;

i) pode-se manter informações de cor e intensidade de ilumination das superfícies (e interior, se necessário) do objeto.

Contudo, esta representação possui algumas limitações, como:

a) para representar objetos com detalhes precisos, é necessária uma quantidade muito grande de nodos, o que equivale a muita memória;

b) dependendo da resolução, é uma aproximação dos objetos. Assim, somente propriedades aproximadas podem ser calculadas;

c) há dificuldades na sua incorporação a sistemas CAD/CAM já existentes.

Meagher /MEA 80/ provou que a quantidade de memória necessária e o tempo de processamento gasto para representar um objeto são diretamente proporcionais à área de sua superfície.

4.3.2.2 Representação Interna da Octree

Uma estrutura hierárquica pode ser implementada através de ponteiros ou de forma sequencial.

A maior vantagem de uma estrutura sequencial é a redução de memória por não usar ponteiros. Entretanto, a maior desvantagem é o tempo gasto para buscar um nodo qualquer, pois todos os nodos anteriores devem ser visitados antes dele. Pode-se ainda buscá-los através de cálculo de endereços, o que minimizaria este problema. Já uma estrutura com ponteiros permite uma gerência dinâmica de memória e facilita a busca de nodos.

Para o uso eficiente do espaço de armazenamento, Gargantini /GAR 82a/, /GAR 82b/ propôs um método de representar octrees em uma estrutura linear, mantendo nela um código para cada nodo folha ocupado. O código consiste de dígitos octais que expressam o caminho da raiz até um nodo qualquer (figura 4.14).

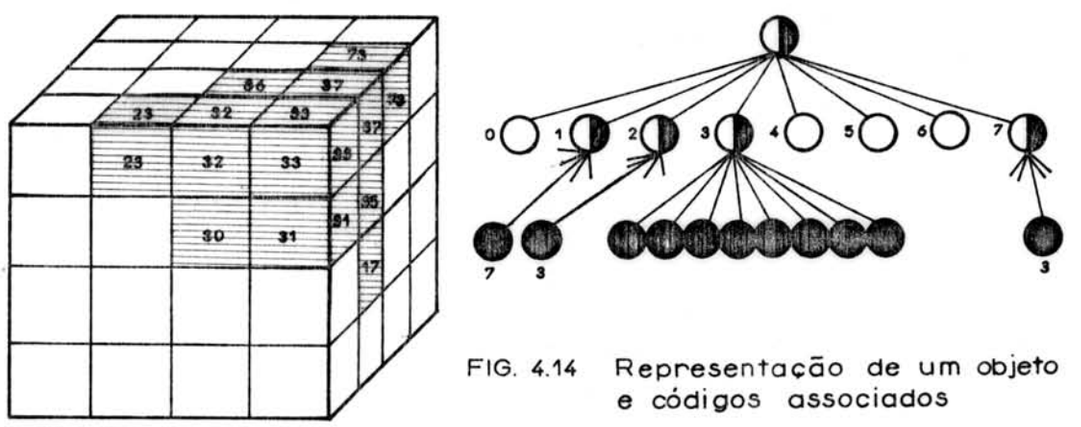



FIG. 4.14 Representação de um objeto e códigos associados

Outros métodos de armazenamento para estrutura linear e encadeada dos endereços dos nodos foram apresentados em /LON 86/. Nestes métodos, os valores dos nodos (ocupado, semi-ocupado e vazio) são representados em dois e até três bits /OLI 83/ e alguns deles armazenam os endereços dos blocos que contenham nodos filhos.

Considerar-se-á como esquema de representação interna das octrees aquela proposta por Gargantini /GAR 82b/ e o sistema de coordenadas do universo limitado em $\left[-2^N,\ 2^N\right]$ para X, Y e Z, onde N é o grau de refinamento do objeto. O grau de refinamento do objeto é limitado pelo número máximo de níveis da octree, que corresponde ao nível de resolução máximo do universo (RU). Esta representação foi escolhida principalmente por economia de memória e porque os algoritmos que implementam as operações de manipulação tratam cada nodo como um conjunto de valores octais que representa o endereço do nodo.

## 4.4 O Núcleo de Modelagem

O núcleo é composto de três módulos essenciais para a função de modelagem: Geração de Primitivos, Operações de Combinação e Operações de Transformação. Cada um desses módulos será abordado nas seções seguintes, e as operações envolvidas em cada um deles serão apresentadas detalhadamente no capítulo que trata da implementação do núcleo (capítulo 5).

### 4.4.1 Geração de Primitivos

O módulo de geração de primitivos tem por função permitir ao usuário gerar os primitivos que melhor lhe convêm. Há duas classes de primitivos, segundo a forma de geração: os pré-definidos e os gerados automaticamente. Cada primitivo pré-definido ou gerado automaticamente é representado espacialmente por uma octree.

Os primitivos gerados automaticamente fazem parte da família dos prismas e das pirâmides. O usuário informa o número de lados da base e o módulo gera a octree referente ao primitivo. O número de lados da base deve ser maior ou igual a três e menor ou igual a quinze.

O polígono é circunscrito por uma circunferência onde o centro é a posição (X=0, Y=0). Por sua vez, a circunferência pode estar inscrita na base do universo ou, como na figura 4.15, na face frontal.

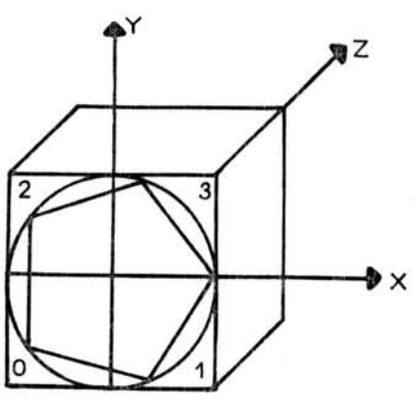



FIG. 4.15 Construção do polígono base

Para determinar o estado dos octantes (preenchido, vazio e semi-preenchido) o universo deverá ser subdividido sucessivamente até o seu nível de resolução. Quanto maior o nível de resolução do universo (RU) maior será a resolução do sólido primitivo.

Os primitivos gerados serão encapsulados por um universo normalizado onde o centro do universo será localizado na posição (0, 0, 0). E, no caso das pirâmides, o ponto ápice estará na posição (0, 0, 1).

Os primitivos pré-definidos no sistema são cubo, cilindro, cone e esfera. São sólidos aproximados já presentes no sistema. O cilindro é um prisma com número máximo de lados igual a 20 e o cone é uma pirâmide cuja base tem 20 lados.

### 4.4.2 Operações de Combinação

O módulo tem por função permitir ao usuário manipular os sólidos através de operações de conjunto como união, intersecção e diferença. O módulo recebe duas octrees e gera uma nova com o resultado da operação. Não há necessidade de algoritmos para verificar se todas as superfícies devem ou não permanecer na composição final do sólido, pois uma das características da octree é preservar a regularidade do sólido resultante.

As operações de combinação são ferramentas poderosas para gerar objetos complexos através da combinação de objetos mais simples. No mínimo, duas octrees são usadas para gerar uma terceira, a octree resultante da operação aplicada.

A eficiência destes algoritmos é um dos problemas enfrentados pelos sistemas de modelagem. Com as octrees, os algoritmos garantem a eficiência e rapidez independente da complexidade do modelo a ser gerado /AYA 85/, /HER 85/, /BRU 85/.

### 4.4.3 Operações de Transformação

O módulo de operações de transformação geométrica tem por função permitir ao usuário aplicar transformações de rotação, translação e mudança de escala a um sólido. O módulo produz uma octree representando o resultado da transformação geométrica especificada e realizada sobre uma octree presente na memória. Dependendo da operação, a nova árvore terá o número de níveis aumentado.

Em sistemas B-rep a aplicação das operações de transformação geométrica pode ser resolvida aplicando-se matrizes de transformação. Embora os algoritmos apresentem cálculos matemáticos, estes são menos complexos se forem implementados por octrees. As transformações de translação para deslocamentos expressos como potência de 2, rotação para os ângulos de 90°, 180° e 270° e mudança de escala pelo fator de 2, quando implementadas sobre octrees, são algoritmos simples e utilizam apenas funções aritméticas como comparação, adição e subtração de

inteiros. Operações de translação, rotação e mudança de escala para valores arbitrários exigem, no entanto, algoritmos não triviais.

## 4.5 O Consultor

É o módulo responsável pelo acesso a informações diversas sobre as entidades. Propriedades como material, cor e grau de acabamento são mantidas na base de dados como parte dos registros descritores das entidades. Sua consulta é feita diretamente através da interface de acesso aos arquivos.

Os módulos que permitem a consulta a propriedades derivadas, Cálculo de Propriedades e Verificação de Interferências, serão comentados a seguir.

### 4.5.1 Cálculo de Propriedades

Este módulo permite que medidas como altura, comprimento, largura, área, volume, peso e outras possam ser obtidas. As medidas são calculadas a partir de uma octree residente em memória.

Para o cálculo de altura, comprimento e largura, são utilizados os endereços dos nodos da octree. Cada endereço é desmembrado em seus códigos constituintes; a cada código é atribuído um valor, para efeitos de medida, de modo que a soma dos valores associados aos códigos de um endereço especificam o valor daquele nodo na medida sendo

calculada. Considerando-se os nodos convenientes, calcula-se as medidas desejadas. No caso do volume e peso, um valor é associado aos nodos folha de nível mais baixo, que passam a ser considerados unidades de volume. A soma dos volumes dos nodos (considerando a unidade padrão) deriva o volume (ou peso) total do objeto.

### 4.5.2 Verificação de Interferências

O módulo que realiza a verificação de interferências permite ao usuário ter certeza de que dois objetos que entrarão na composição de um terceiro não se sobrepõem. Duas octrees são necessárias para efetivar a pesquisa. Somente os nodos que possam conter interferências são examinados. Subárvores que não se conflituam são ignoradas. O algoritmo é bastante simples, bastando percorrer as duas árvores ao mesmo tempo, buscando endereços similares de nodos.

# 5 IMPLEMENTAÇÃO DO NÚCLEO DE MODELAGEM DE SÓLIDOS

## 5.1 Introdução

O N-MOS foi desenvolvido para validar o modelo geométrico escolhido para representação de sólidos.

Atualmente, ele está implementado em Turbo Pascal, em microcomputadores compatíveis com IBM-PCxt no Centro de Processamento de Dados e no Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Rio Grande do Sul. Os sistemas estão configurados com memória de 256 KB, 512 KB ou 640 KB, vídeo com resolução máxima de até 640 x 400 pixels a 4 cores (ou tons), impressora paralela, um controlador de disquetes de 5 1/4 polegadas, winchester com capacidade de 5 a 10 MBytes e expansões como comunicação coaxial com um "mainframe".

As seguintes funções foram implementadas:

1) Geração de sólidos primitivos;

2) Construção de entidades através de transformações geométricas (translação, escala e rotação) e de combinações (união, intersecção e diferença);

3) Exibição através de vistas ortográficas e seções planas.

## 5.2 Estruturas de Armazenamento

O N-MOS manipula, através do módulo de interface com a base de dados, sete arquivos em disco, a saber:

- Entidades,
- Objetos,
- Combinados,
- Transformados,
- Primitivos,
- Geometria de Primitivos e
- Geometria de Objetos.

O arquivo Entidades mantém os registros descritores das entidades construídas (objetos projetados). Cada registro corresponde à raiz da árvore que descreve a entidade. O arquivo Objetos contém a especificação da categoria a que pertencem os objetos componentes de uma entidade, ou seja, cada registro descreve um objeto como combinado, transformado ou primitivo. Estes registros correspondem aos nodos intermediários da árvore CSG. Os arquivos Combinados, Transformados e Primitivos especificam os parâmetros dos objetos destes três tipos.

A capacidade dos arquivos depende da periodicidade de atualização não existindo limitação quanto ao tamanho, exceto a limitação do espaço físico em disco.

Atualmente, todos os arquivos são mantidos em disquete e organizados de forma sequencial com possibilidade de acessos randômicos devido à estrutura do sistema Pascal.

A organização destes arquivos foi baseada na estrutura genérica apresentada no capítulo 4 e os dados foram estruturados na forma de relações.

### 5.2.1 Entidades

O arquivo Entidades mantém informações que descrevem cada entidade produzida no processo de projeto. A construção da entidade, executada através de um conjunto de atividades, pode levar à geração de um modelo de produto aproximando-se tanto quanto possível do ideal e que possa ser fabricado conforme os parâmetros previamente estabelecidos. Uma entidade é um objeto único composto por um ou vários sólidos.

Cada registro (figura 5.1) contém campos como segue:

- Código da entidade;
- Nome da entidade;
- Material;
- Cor;
- Grau de acabamento;
- Código do objeto representado (no arquivo Objetos).

| Cód. Ent. | Nome da Entidade | Mater. | Cor | Grau de Acabam. | Cód. Arq. Objetos |
|---|---|---|---|---|---|
| 999 | XXXXXXXX | XXXXX | 99 | 999 | 999999 |
| : | : | : | : | : | : |
| : | : | : | : | : | : |
| 999 | XXXXXXXX | XXXXX | 99 | 999 | 999999 |

FIG. 5.1 Organização dos dados no arquivo Entidades

5.2.2 Objetos

O arquivo Objetos mantém informações sobre cada etapa da composição de uma entidade através de uma árvore CSG. Dessa forma, em qualquer estágio do projeto, o objeto pode ser desfeito ou refeito dependendo das necessidades do usuário. Os campos do registro possuem dados sobre:

- Código do objeto;
- Nome do objeto;
- Código da entidade;
- Categoria construtiva (combinado, transformado, primitivo;
- Tipo de geração:
  - Pré-definida;
  - Automática;
- Informações geométricas:
  - Número de níveis da octree;
  - Número de registros da octree;
  - Endereço inicial para a representação interna da octree;

. Remoção lógica.

A figura 5.2 ilustra a disposição dos campos no arquivo Objetos.

| Cód. Obj. | Nome Obj. | Cód. Ent. | Cat. Cons. | Tipo Ger. | OCTREE | | | Rem. Lóg. |
|---|---|---|---|---|---|---|---|---|
| | | | | | Nív. | Nro Reg. | Ender. | |
| 999 | XXXXXX | 999 | 9 | 9 | 99 | 99999 | 999999 | 9 |
| : | : | : | : | : | : | : | : | : |
| 999 | XXXXXX | 999 | 9 | 9 | 99 | 99999 | 999999 | 9 |

FIG. 5.2 Organização dos dados do arquivo Objetos

5.2.3 Combinados

O arquivo Combinados mantém os objetos resultantes das operações de combinação. Cada registro refere-se a um objeto combinado, possuindo um campo que indica o tipo de operação executada (união, intersecção e diferença) e os objetos operandos (figura 5.3).

| Código Obj. | Código Obj. A | Código Obj. B | Operação |
|---|---|---|---|
| 999 | 999 | 999 | 9 |
| : | : | : | : |
| 999 | 999 | 999 | 9 |

FIG. 5.3 Organização dos dados no arquivo Combinados.

### 5.2.4 Transformados

O arquivo Transformados mantém um registro para cada objeto transformado (figura 5.4) com campos para especificar a operação e seus parâmetros, e o objeto que deve sofrer a transformação.

As operações de transformação (rotação, translação, mudança de escala) podem ser realizadas sobre o universo ou sobre partes do universo. Cada registro indica o octante onde se realiza a transformação.

| Código Obj. | Código obj.Or. | Operação | Origem | Transf. em X | Transf. em Y | Transf. em Z |
|---|---|---|---|---|---|---|
| 999 | 999 | 9 | 9 | 999.99 | 999.99 | 999.99 |
| : | : | : | : | : | : | : |
| : | : | : | : | : | : | : |
| 999 | 999 | 9 | 9 | 999.99 | 999.99 | 999.99 |

FIG. 5.4 Organização dos dados no arquivo Transformados

### 5.2.5 Primitivos

O arquivo Primitivos mantém informações sobre cada objeto primitivo utilizado na construção das entidades. Cada registro (figura 5.5) especifica o código do objeto, a sua categoria geométrica (cubo, cone, cilindro, esfera, prisma e pirâmide), os parâmetros utilizados na sua geração (que pode ser automática ou pré-definida), informações sobre a octree que o representa e o código do primitivo que o identifica como instância de um primitivo já utilizado.

| Cód. Obj. | Categ. Geom. | Parâmetros | | Tipo Ger. | OCTREE | | | Cód. Inst. |
|---|---|---|---|---|---|---|---|---|
| | | Par.1 | Par.2 | | Nív. | Nro Reg. | Ender. | |
| 999 | 99 | 999 | 999 | 9 | 99 | 99999 | 999999 | 999 |
| : | : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : | : |
| 999 | 99 | 999 | 999 | 9 | 99 | 99999 | 999999 | 999 |

FIG. 5.5 Organização dos dados do arquivo Primitivos

5.2.6 Geometria de Primitivos e de Objetos

Estes dois arquivos mantêm as informações geométricas dos objetos na forma de octrees.

O arquivo Geometria de Primitivos contém todos os objetos primitivos pré-definidos no sistema como: cubo, cone, cilindro e esfera. É mantido permanentemente pelo sistema e atualizações, pelo usuário, não são permitidas.

O arquivo Geometria de Objetos contém as informações geométricas dos objetos primitivos gerados automaticamente pelo sistema e dos objetos transformados e combinados.

Os campos "tipo de geração" dos arquivos Objetos e Primitivos indicam qual dos arquivos de informações geométricas (Geometria de Primitivos e Geometria de Objetos) deve ser usado para buscar a octree correspondente a um determindado objeto.

Os dois arquivos são listas sequenciais que mantêm os endereços dos nodos cheios da octree conforme apresentado na seção 4.3.2.2.

## 5.3 Funções de Modelagem de Sólidos

A atividade de modelagem envolve a construção de um modelo que permite obter adequadamente as propriedades de um objeto, e quando integrada aos recursos da computação gráfica interativa, permite que o modelo seja definido, modificado e exibido graficamente.

Esta seção apresenta os algoritmos de manipulação de sólidos implementados no N-MOS para suportar as operações oferecidas ao usuário.

As operações de modelagem consistem das operações de combinação, transformação geométrica e geração de primitivos. Adicionalmente, foi implementada uma função de seccionamento do universo.

Os algoritmos apresentados a seguir são aplicados a octrees. Os endereços de nodos são os valores resultantes da subdivisão dos octantes. São representados por uma série de números inteiros indicando a ordem de caminhamento na árvore, desde o nodo pai até o nodo terminal (cf. seção 4.3.2.2).

O nível de resolução máxima do universo (RU) é 10. Entretanto, por motivos de economia no espaço de armazenamento e tempo de processamento, os algoritmos foram implementados considerando RU = 6.

O sistema de coordenadas do universo é limitado em $[-2^N, 2^N]$ para X, Y e Z, com $N \leq RU$. A origem (0, 0, 0) é o centro do universo conforme visto na figura 5.6.

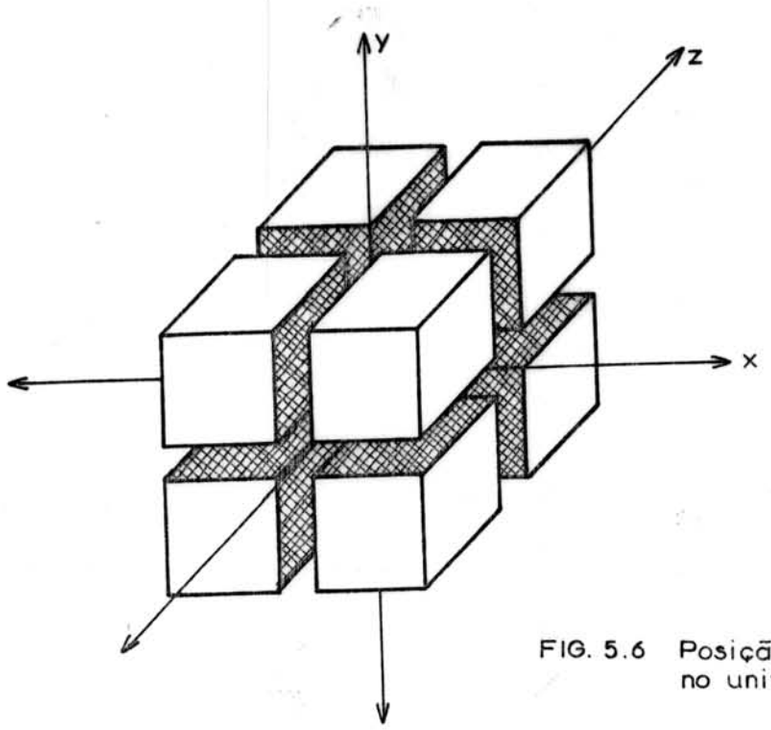



FIG. 5.6 Posição das coordenadas no universo de modelagem

Os algoritmos que tratam das operações de combinação devem inicialmente verificar se as duas árvores possuem o mesmo nível de resolução. Se possuirem diferentes níveis deve-se converter a árvore de menor nível em outra com igual nível ao da árvore de nível maior. Por exemplo, sejam S1 e S2 dois objetos descritos por octrees com nS1=2, nS2=3 e

S1 = {0X, 30, 31, 32, 33, 34, 35, 36} ;

S2 = {01X, 03X, 100, 102, 103, 104, 105, 106, 107, 21X, 30X,
320, 322, 324, 326}.

Sendo nS1 < nS2, então

S1 = {0XX, 30X, 31X, 32X, 33X, 34X, 35X, 36X}.

onde X indica o fator de compressão quando oito suboctantes cheios pertencem ao mesmo octante. Por exemplo, 21X é o mesmo que (210, 211, 212, 213, 214, 215, 216, 217). Nos endereços dos nodos, o fator de compressão X será substituído pelo dígito 8.

Tanto as operações de combinação quanto as de transformação geométrica e de seccionamento devem manter as octrees dos objetos a serem operados em memória (duas octrees para as operações de combinação e uma octree para operações de transformação geométrica e de seccionamento). Cada objeto é mantido em memória numa lista encadeada principalmente porque o número de registros necessários para representar um objeto varia. O algoritmo permite que sejam alocadas novas posições de memória quando for necessário.

### 5.3.1 União

Considerando-se os dois sólidos S1 e S2 descritos anteriormente, a união é uma fusão entre S1 e S2 onde os nodos comuns ou nodos que pertencem ao conjunto maior serão considerados uma única vez (figura 5.7):

S1 U S2 = {0XX, 100, 102, 103, 104, 105, 106, 107, 21X, 30X, 31X, 32X, 33X, 34X, 35X, 36X}

Depois da união é aplicado o fator de compressão (se for necessário). Supondo que S1 = {122, 123, 301, 303, 305, 307} e S2 = {12X, 300, 302, 304, 306}:

S1 U S2 = {12X, 300, 301, 302, 303, 304, 305, 306, 307}

pode ser transformado, pelo fator de compressão, para

S1 U S2 = {12X, 30X}

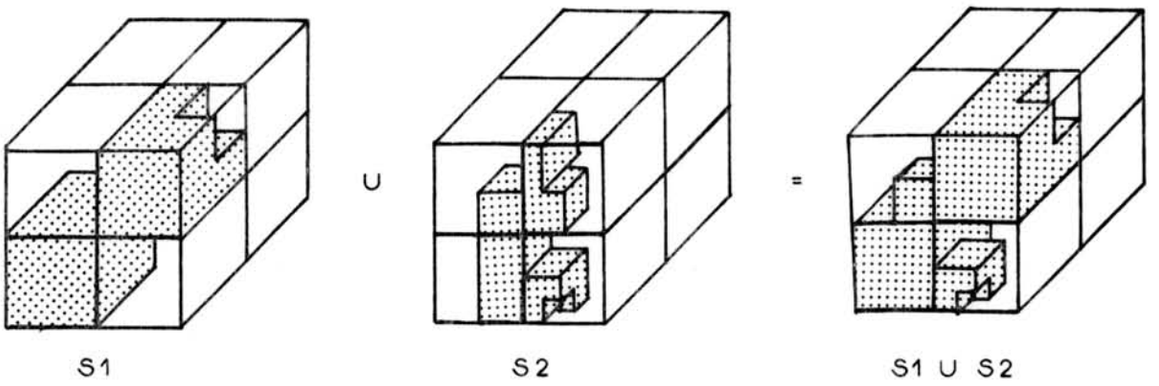



FIG. 5.7 União de dois sólidos

Segue uma descrição geral do algoritmo da união:

procedure união (S1,S2,SR);

(* S1, S2 e SR são listas encadeadas. Nodo é o campo que contém o endereço de nodo. Prox é o campo apontador para o próximo componente da lista e Apont conterá o valor do campo apontado. Inicialmente, Apont aponta para o primeiro componente da lista. λ indica valor nulo. Nro-de-Níveis possui o valor referente ao número de níveis das duas octrees, ou seja, o número de dígitos que compõe

```
    o endereço de nodo. *)
begin
    while S1.Apont ≠ λ and S2.Apont ≠ λ
    do
       if S1.Nodo > S2.Nodo then
       begin
          for N:=1 Nível0 to Nro-de-Níveis
          do
             if S1.Nodo [N] > S2.Nodo [N]and
                S1.Nodo [N]≠ 8 then
             begin
                N:= Nro-de-Níveis;
                SR:= S2; (*  Sólido  resultante recebe  conteúdo  de
                             S2 *)
             end;
          S1.Apont:=S2.Prox;
       else
          if S1.Nodo < S2.Nodo then
          begin
             for N:=Nível0 to Nro-de-Níveis
             do
               if (S1.Nodo[N] < S2.Nodo[N] and S2.Nodo ≠ 8) or
                   S1.Nodo[N] >S2.Nodo[N] and S2.Nodo ≠ 8) then
               begin
                   N:=Nro-de-Níveis;
                   SR:=S1;
               end;
             S1.Apont:=S1.Prox
```

```
        else
            begin
                SR:=S1;
                S1.Apont:=S1.Prox;
                S2.Apont:=S2.Prox;
            end;

    if S1.Apont = λ and S2.Apont ≠ λ then
        while S2.Apont ≠ λ do
        begin
            SR:=S2;
            S2.Apont:=S2.Prox;
        end
    else
        if S1.Apont ≠ λ and S2.Apont= λ then
            while S1.Apont ≠ λ do
            begin
                SR:=S1;
                S1.Apont:=S1.Prox;
            end;
    Fator-de-compressão;
end; (* Operação de união *)
```

### 5.3.2 Intersecção

Para realizar a intersecção de S1 e S2 (figura 5.8) verifica-se quais nodos são comuns às duas árvores. Os nodos comuns farão parte

da nova árvore, constituindo, então, o objeto resultante.

Seja S1 = {0X, 30, 32, 33, 34, 35, 36} e

S2 = {01, 03, 10, 21, 30}

então

$s1 \cap s2 = \{01, 03, 30\}$

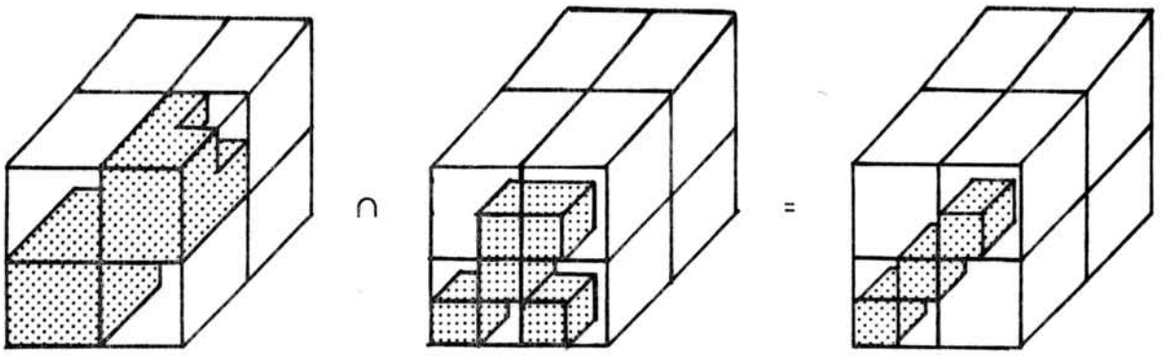



FIG. 5.8 Intersecção de dois sólidos

Segue uma descrição geral do algoritmo da intersecção:

```
procedure Intersecção (S1, S2, SR)
begin
    while S1.Apont ≠λ do
    begin
      (* Neste instante é verificado se um novo registro deve ser
         criado para SR*)

      for N:=Nível0 to Nro-de-Níveis
      do
         if S1.Nodo[N] = S2.Apont[N] then
            SR.Nodo[N]:= S1.Nodo[N]
         else
```

```
                if S1.Nodo[N] > S2.Nodo[N] then
                   if S1.Nodo[N] = 8 then
                      SR.Nodo[N] :=S2.Nodo[N]
                   else
                      N:= Nro-de-Níveis
                else
                   if S2.Nodo[N] = 8 then
                      SR.Nodo[N] :=S1.Nodo[N]
                   else
                      N:=Nro-de-Níveis;

          if S1.Nodo > S2.Nodo then
             S2.Apont:=S2.Prox
          else
             if S1.Nodo < S2.Nodo then
                S1.Apont:=S1.Prox
             else
                begin
                   S1.Apont:=S1.Prox;
                   S2.Apont:=S2.Prox;
                end;
       end;
    end; (* Operação de intersecção *)
```

5.3.3 Diferença

Para realizar a subtração S1 - S2 deve-se primeiro verificar os nodos comuns às duas árvores. Os nodos comuns serão desconsiderados assim como aqueles pertencentes somente a S2, permanecendo os nodos do primeiro conjunto não presentes no segundo. Portanto, a diferença permite que partes de um objeto sejam excluídas de outro resultando um novo objeto. Através desta operação podemos definir espaços vazios nos objetos.

A operação de diferença não possui a propriedade de ser comutativa, isto é, S1 - S2 = S2 - S1

Seja S1 = {0XX, 30X, 31X, 32X, 33X, 34X, 35X, 36X} e

S2 = {01X, 03X, 100, 102, 103, 104, 105, 106, 107, 21X, 30X, 320, 322, 324, 326}

então S1 - S2 = {00X, 02X, 04X, 05X, 06X, 07X, 31X, 321, 323, 325, 327, 33X, 34X, 35X, 36X} e

S2 - S1 = {100, 102, 103, 104, 105, 106, 107, 21X}

A figura 5.9 mostra a diferença entre dois sólidos S1 e S2 ilustrados na figura 5.7.

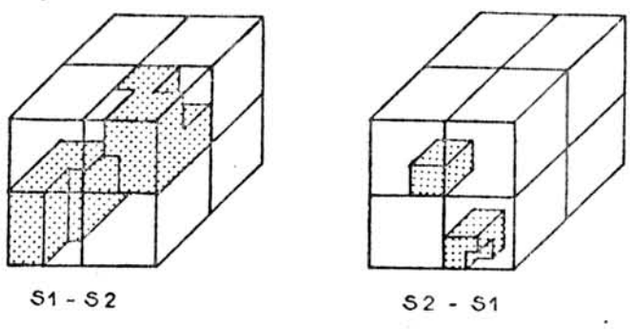



FIG. 5.9. Diferença de dois sólidos

Segue a descrição geral do algoritmo de diferença.

```
procedure Comparação-de-Desmembrados;
begin
   for I:0 to 7
   do
     if S1.Desmbr < S2.Nodo then
        for N:=Nível0 to Nro-de-Níveis
        do
          if S1.Desmbr[N] ≠ S2.Nodo[N] and S2.Nodo[N] ≠ 8 then
          begin
             SR:=S1.Desmbr;
             N:=Nro-de-Níveis;
          end
     else
         if S1.Desmbr > S2.Nodo then
            for N:=Nível0 to Nro-de-Níveis
            do
              if S1.Desmbr[N] ≠ S2.Nodo[N] then
                 if S1.Desmbr[N] ≠ 8 then
                 begin
                    SR:= S1.Desmbr;
                    N= Nro-de-Níveis
                 end
                 else
                 (* salva valores dos índices do "stack" de desmembra-
                    dos e desmembra níveis de S1.Desmbr *)
```

```
                Comparação-de-Desmembrados;

                (* Se "stack pointer" for igual a "base pointer" en-
                   tão S1.Apont recebe S1.Prox, senão "stack pointer"
                   recebe    "stack pointer" diminuído de   1      *)
             else
                S2.Apont:=S2.Prox;
                (*  verifica se S2.Apont é nulo e, realiza operações
                de salvamento se necessário *)

end; (* Comparação-de-Desmembrados *)

procedure Diferença (S1,S2, SR);
begin
   Intersecção (S1, S2, SR);
   (* transforma SR resultado da intersecção em S2 como o segundo só-
      lido a participar da operação *)
   while S1.Apont ≠ λ
   do
     (* se a leitura dos registros de S2 terminar antes dos registros
        de  S1  etão SR receberá os valores de registro restantes de
        S1 *)
     if  S1.Nodo = S2.Nodo then
     begin
        S1.Apont:= S1.Prox;
        S2.Apont:= S2.Prox;
     end
     else
```

```
if S1.Nodo < S2.Nodo then
begin
   SR:=S1;
   S1.Apont:=S1.Prox;
end
else
   (* verifica se em algum nível do endereço de nodo S1 possui o
      fator de compressão (8) *)
   if not existe-fator-compr then
      if S1.nodo < S2.Nodo then
         for N:=Nível0 to Nro-de-Níveis
         do
           if S1.Nodo[N] ≠ S2.Nodo[N] and
              S2.Nodo[N] ≠ 8 then
           begin
              SR:=S1;
              N:=Nro-de-Níveis;
              S1.Apont:=S1.Prox;
           end
      else
         S2.Apont:=S2.Prox
   else
      begin
         (* Desmembra níveis de S1.Nodo *)
         Comparação-de-Desmembrados;
         S1.Apont:=S1.Prox;
      end;
end; (* Operação de Diferença *)
```

### 5.3.4 Translação

O método utilizado no N-MOS para realizar a translação de objetos no espaço tridimensional foi proposto por Fujimura et alli em /FUJ 83/.

Dependendo do deslocamento, a translação pode ou não originar subárvores na octree original. Quando originar subárvores, a translação é conhecida como "translação com gap"; caso contrário, é dita "translação sem gap".

Para qualquer um dos dois tipos de translação é necessário conhecer o deslocamento (distância de movimento), isto é, a distância entre a posição inicial e final de um objeto a ser transladado. A distância de movimento $\underline{d}$ é expressa pela soma das potências de 2:

$$dx = 2^{x1} + 2^{x2} + \ldots + 2^{xi} \quad \text{onde } x1 > x2 > \ldots xi \geq 0 \text{ e } x1 \leq RU$$

$$dy = 2^{y1} + 2^{y2} + \ldots + 2^{yj} \quad \text{onde } y1 > y2 > \ldots yj \geq 0 \text{ e } y1 \leq RU$$

$$dz = 2^{z1} + 2^{z2} + \ldots + 2^{zk} \quad \text{onde } z1 > z2 > \ldots zk \geq 0 \text{ e } z1 \leq RU$$

Além da distância de movimento deve-se conhecer quais os níveis da árvore que serão afetados pela translação. Estes <u>níveis de</u>

translação são definidos subtraindo-se do nível máximo de resolução do universo, RU, as potências de dois que expressam a distância de movimento.

$$d_p = 2^{p1} + 2^{p2} + \ldots + 2^{pm} \quad \text{onde } p = x,y,z$$

$$nt_p = RU - p_k \quad \text{onde } k = m, m-1, \ldots, 1$$

Assim, para cada dimensão X, Y e Z obtém-se um conjunto de níveis de translação, de acordo com o valor de m. Por exemplo, se $d = 2^{x1} + 2^{x2} + 2^{x3}$ tem-se três níveis de translação RU-x3, RU-x2, RU-x1, ou seja, o procedimento será aplicado três vezes, afetando níveis diferenciados da árvore. Para identificação dos níveis de translação, os níveis da árvore são computados a partir de 0 (zero) até RU (figura 5.10).

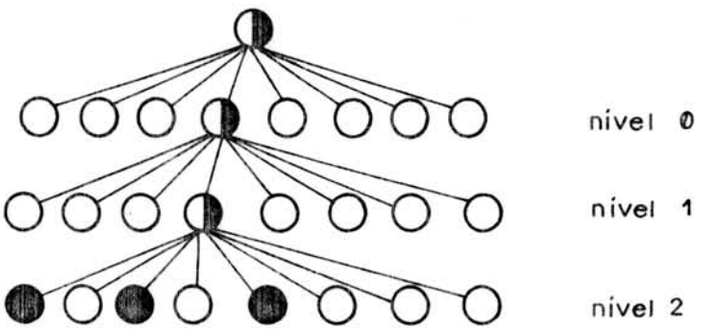



FIG. 5.10 Níveis de uma octree

Determinados os níveis de translação, o algoritmo toma cada nodo da octree e verifica se a sua transformação origina ou não subárvores. Isto é feito calculando-se o valor do "gap". Um gap é definido por:

gap = nt - n, onde n é o nível do nodo a ser transladado.

Se gap $\leq$ 0 então sem gap,

se gap > 0 então com gap.

É realizado, então, um conjunto de permutações dos códigos dos octantes ou a criação de novos octantes, segundo regras bem estabelecidas.

Para translação sem gap, na dimensão X, as regras de permutação são: 0$\leftrightarrow$1, 2$\leftrightarrow$3, 4$\leftrightarrow$5, 6$\leftrightarrow$7 (na dimensão Y: 0$\leftrightarrow$2, 1$\leftrightarrow$3, 4$\leftrightarrow$6, 5$\leftrightarrow$7 e na dimensão Z: 0$\leftrightarrow$4, 1$\leftrightarrow$5, 2$\leftrightarrow$6, 3$\leftrightarrow$7), onde o endereço do nodo é modificado do nível de translação até o nível 0 (zero) enquanto um elemento do conjunto dos octantes pares (0, 2, 4, 6) não for encontrado. Se houver um elemento do conjunto dos octantes pares, ele é modificado e cessa a permutação para aquele nível de translação. A figura 5.11(a) ilustra um exemplo de translação sobre a dimensão X, onde o endereço do nodo original é 0-0-1-1 e a distância de movimento é dada por $dx = 2^{10} + 2^{8} + 2^{7}$ . Para aplicar as regras de permutação deve-se calcular os níveis de translação como segue:

| | | |
|---|---|---|
| nt = RU - x | 0 - 0 - 1 - 1 | endereço do nodo ori- |
| nt = 10 - 7 = 3 | 0 - 1 - 0 - 0 | ginal |
| nt = 10 - 8 = 2 | 0 - 1 - 1 - 0 | |
| nt = 10 -10 = 0 | 1 - 1 - 1 - 0 | endereço do novo nodo |

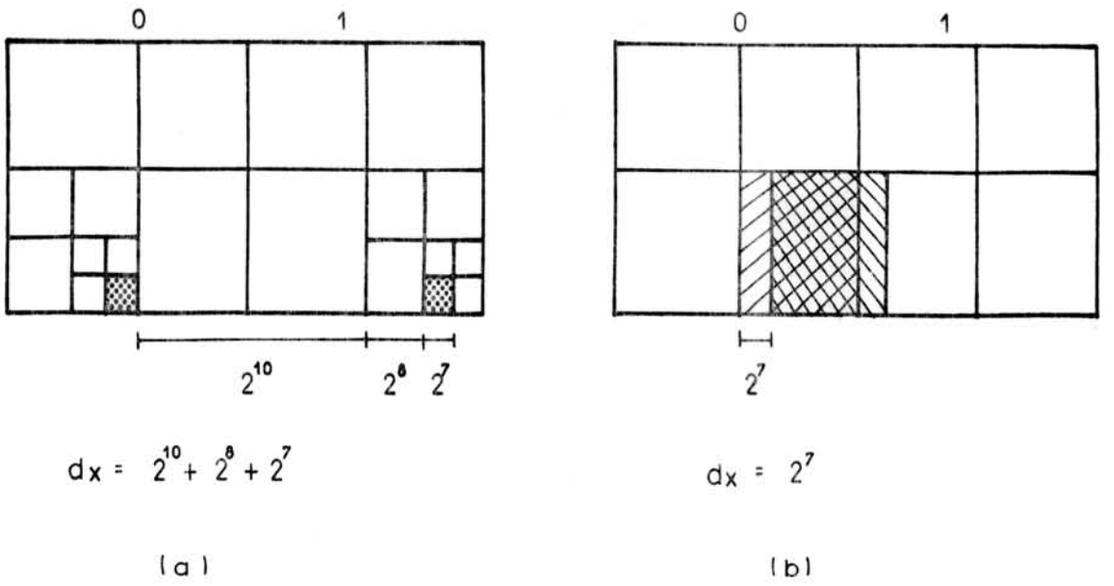



FIG. 5.11 Tipos de translação:
(a) sem gap
(b) com gap

Para a translação com gap, as regras de geração de subárvores, na dimensão X, estão descritas na figura 5.12. Nesta figura convencionou-se que x e y referem-se a um octante de tal forma que se x pertencer ao conjunto dos octantes pares P, então y = x + 1, e se x pertencer ao conjunto dos octantes ímpares I (1, 3, 5, 7), então y = x - 1; T e U são endereços de nodos representando o caminho até a raiz a partir do nodo x e y, respectivamente.

| combinações do nodo original / combinações de Gaps | T-x-0 | T-x-1 |
|---|---|---|
| gap-1 | T-x-0-I<br>T-x-1-P | T-x-1-I<br>U-y-0-P |
| gap-2 | T-x-0-P-I<br>T-x-0-I<br>T-x-1-P-P | T-x-1-P-I<br>T-x-1-I<br>U-y-0-P-P |
| gap-1-2 | T-x-0-I-I<br>T-x-1-P<br>T-x-1-I-P | T-x-1-I-I<br>U-y-0-P<br>U-y-0-I-P |

FIG. 5.12 Combinações de gap e endereços de nodos resultantes

A figura 5.11(b) ilustra um exemplo de translação com gap na dimensão X onde o endereço do nodo original é 0-1 e a distância de translação $dx = 2^7$ . A aplicação das regras de geração de subárvores para o exemplo são dadas a seguir:

nt = RU - dx  nt = 10 - 7 = 3

gap= nt - n  gap= 3 - 1 = 2

regra a ser aplicada: gap-2

1) T-x-1-P-f

0-1-0-1

0-1-0-3

0-1-0-5

0-1-0-7

0-1-2-1

...

0-1-6-7

2) T-x-1-1

0-1-1

...

0-1-7

3) U-y-0-P-P  (x ∈ P ⟶ y = x + 1)

1-0-0-0

...

1-0-6-6

A translação é aplicada em dois casos bem distintos:

1) quando um objeto qualquer, com exceção dos primitivos, deve ser transladado;

2) quando um objeto resultado de alguma operação deve ser combinado com outro objeto e este ou o anterior necessitam de translação, para alinhamento.

No primeiro caso, o objeto já está descrito em termos de coordenadas do universo. No segundo, um dos objetos está no universo e o outro está ainda definido no seu próprio sistema de coordenadas. Em ambos os casos a interface com o usuário utiliza o algoritmo descrito.

Segue uma descrição geral do algoritmo de translação:

```
procedure Translação (S, SR, Dimensão, Distância);
 (* S é a octree do objeto a ser transladado; SR é a octree resultante;  Dimensão indica em qual dos eixos se efetuará a translação;
    Distância de movimento em potências de 2 *)
begin
    (* Determinação dos níveis de translação através dos parâmetros
       informados *)
    while S1.Apont ≠ ^
    do
        (* para cada nível de translação determinado é realizado o
           cálculo do valor do "gap" e realizada a comparação a seguir
           *)
```

```
        if gap > 0 then
            Translação-com-gap (tipo-gap)
        else
            Translação-sem-gap;
    S.Apont:=S.Prox;
    (* SR recebe o endereço de nodo resultante da operação de permuta-
       ção no caso "sem gap" ou a lista de endereços de nodos origina-
       dos pela operação de translação "com gap" *)
end (* operação de translação *)


procedure Translação-com-gap (tipo-gap);
begin
   (* verificação de espaço disponível no universo para então ser efe-
      tuada a translação *)
   (* aplicação das regras de combinação de gap conforme o tipo de gap
      informado *)
   (* geração das subárvores *)
   (* armazenamento dos endereços de nodos originados na lista de
      subárvores *)
end (* translação com gap *)


procedure Translação-sem-gap;
begin
   (* verificação de espaço disponível no universo para então ser efe-
      tuada a translação *)
   N:=Nível-translação;
```

```
   while S[N] not in Octantes-Pares and N = -1 do
   begin
      (* realiza as permutações *);
      N:=N-1;
   end;
end; (* translação sem gap *)
```

5.3.5 Rotação

A rotação sobre os três eixos pode ser realizada simplesmente permutando-se os nodos da octree quando o ângulo de rotação for 90°, 180° ou 270° no sentido horário ou anti-horário.

As regras de permutação para estes ângulos considerando o sentido anti-horário, podem ser assim descritas:

1) sobre o eixo X

90° = (0←2), (1←3), (2←6), (3←7), (4←0), (5←1), (6←4), (7←5)

180° = (0←6), (1←7), (2←4), (3←5), (4←2), (5←3), (6←0), (7←1)

270° = (0←4), (1←5), (2←0), (3←1), (4←6), (5←7), (6←2), (7←3)

2) sobre o eixo Y

90° = (0←4), (1←0), (2←6), (3←2), (4←5), (5←1), (6←7), (7←3)

180° = (0←5), (1←4), (2←7), (3←6), (4←1), (5←0),
(6←3), (7←2)

270° = (0←1), (1←5), (2←3), (3←7), (4←0), (5←4),
(6←2), (7←6)

3) sobre o eixo Z

90° = (0←2), (1←0), (2←3), (3←1), (4←6), (5←4),
(6←7), (7←5)

180° = (0←3), (1←2), (2←1), (3←0), (4←7), (5←6),
(6←5), (7←4)

270° = (0←1), (1←3), (2←0), (3←2), (4←5), (5←7),
(6←4), (7←6)

Sendo x e y dois octantes; a relação (x←y) usada nas regras de permutação acima indicam que o octante x receberá o conteúdo do octante y.

A função do algoritmo de rotação é gerar uma nova árvore representando o objeto rotacionado no ângulo desejado.

Rotações sobre partes arbitrárias do universo também podem ser feitas, isto é, cada octante ou suboctante pode ser rotacionado separadamente sobre um sistema de coordenadas próprio. A figura 5.13 ilustra a rotação de apenas um octante.

Rotações sobre ângulos diferentes de 90°, 180° e 270° requerem outro algoritmo nada trivial com geração de subárvores. Atualmente, o N-MOS somente realiza a rotação para os ângulos indicados.

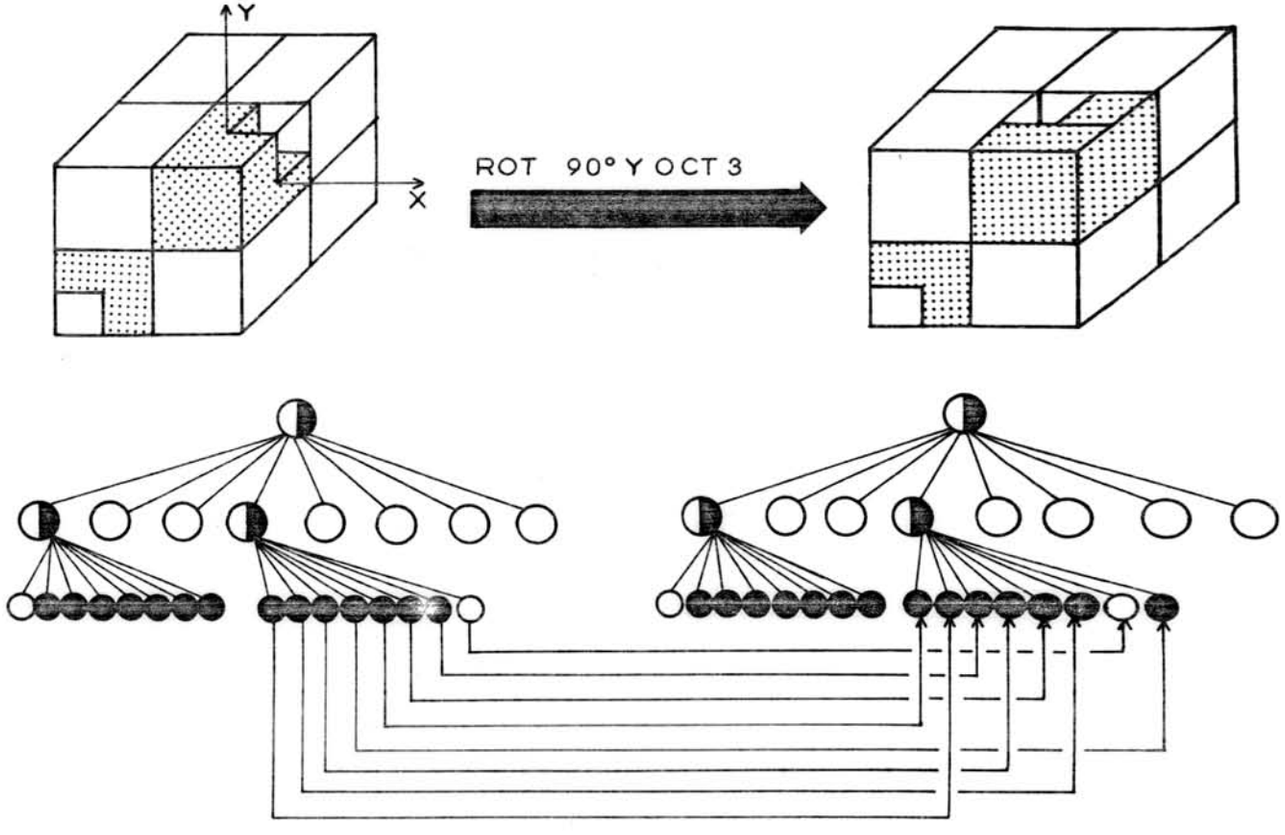



FIG. 5.13 Rotação de um octante

Segue a descrição geral do algoritmo de rotação.

```
procedure Rotação (ângulo, dimensão, origem);
  (* as permutações serão realizadas em cada nível do
     endereço de nodo *)
begin
   if origem = 8 then
   begin
      (* busca os nodos referentes ao octante que sofrerá a rotação *)
   end
   else
      begin
         (* a rotação se dará sobre todo o universo que engloba o  ob-
            jeto *)
      end
   while S.Apont =  do
   begin
      case dimensão of
           X: case ângulo of
                90: (* aplicação das regras de permutação para 90º *)
               180: (* aplicação das regras de permutação para 180º *)
               270: (* aplicação das regras de permutação para 270º *)
              end;
           Y: case ângulo of
                90: (* aplicação das regras de permutação para 90º *)
               180: (* aplicação das regras de permutação para 180º *)
               270: (* aplicação das regras de permutação para 270º *)
              end;
```

```
            Z: case ângulo of
                 90: (* aplicação das regras de permutação para 90º *)
                180: (* aplicação das regras de permutação para 180º *)
                270: (* aplicação das regras de permutação para 270º *)
               end;
        end;
        S.Apont:=S.Prox;
    end;
    (* grava nova árvore *)
end; (* operação de rotação *)
```

5.3.6 Mudanças de Escala

Reduzir ou ampliar um objeto é o mesmo que adicionar ou remover níveis da árvore que o representa, respectivamente.

Reduzindo-se o objeto por um fator de 2, ele será diminuído igualmente em todas as dimensões. Na árvore, é representada pela adição de mais um nível no topo, ou seja, uma nova raiz. A antiga raiz passa a ser um octante semi-preenchido e os outros 7 serão criados vazios (figura 5.14).

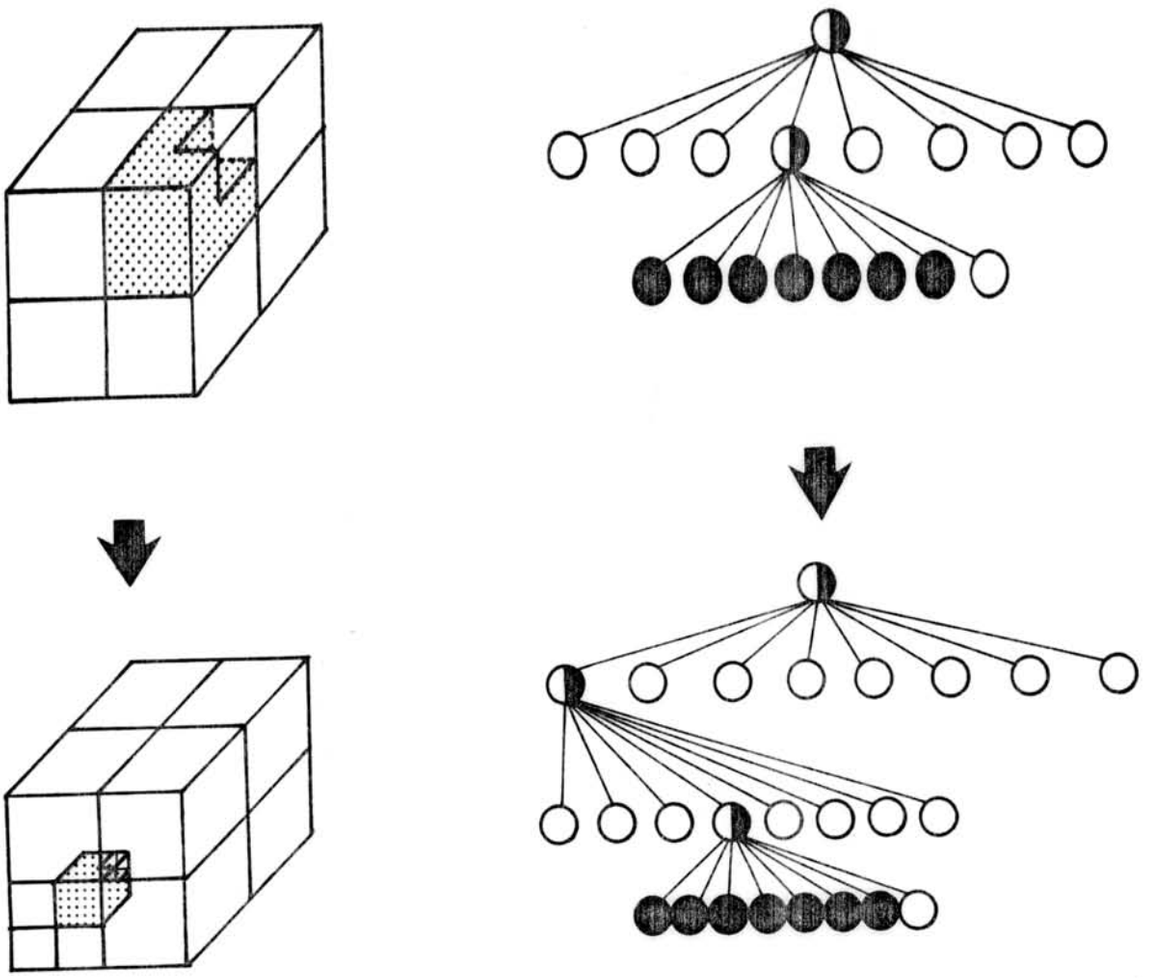


FIG. 5.14 Redução pelo fator 2

Ampliando-se um objeto pelo fator de 2 em todas as dimensões, os nodos filhos são desmembrados de seu nodo pai e uma nova árvore (ou várias) é gerada (ou são geradas). A figura 5.15 ilustra o processo de ampliação.

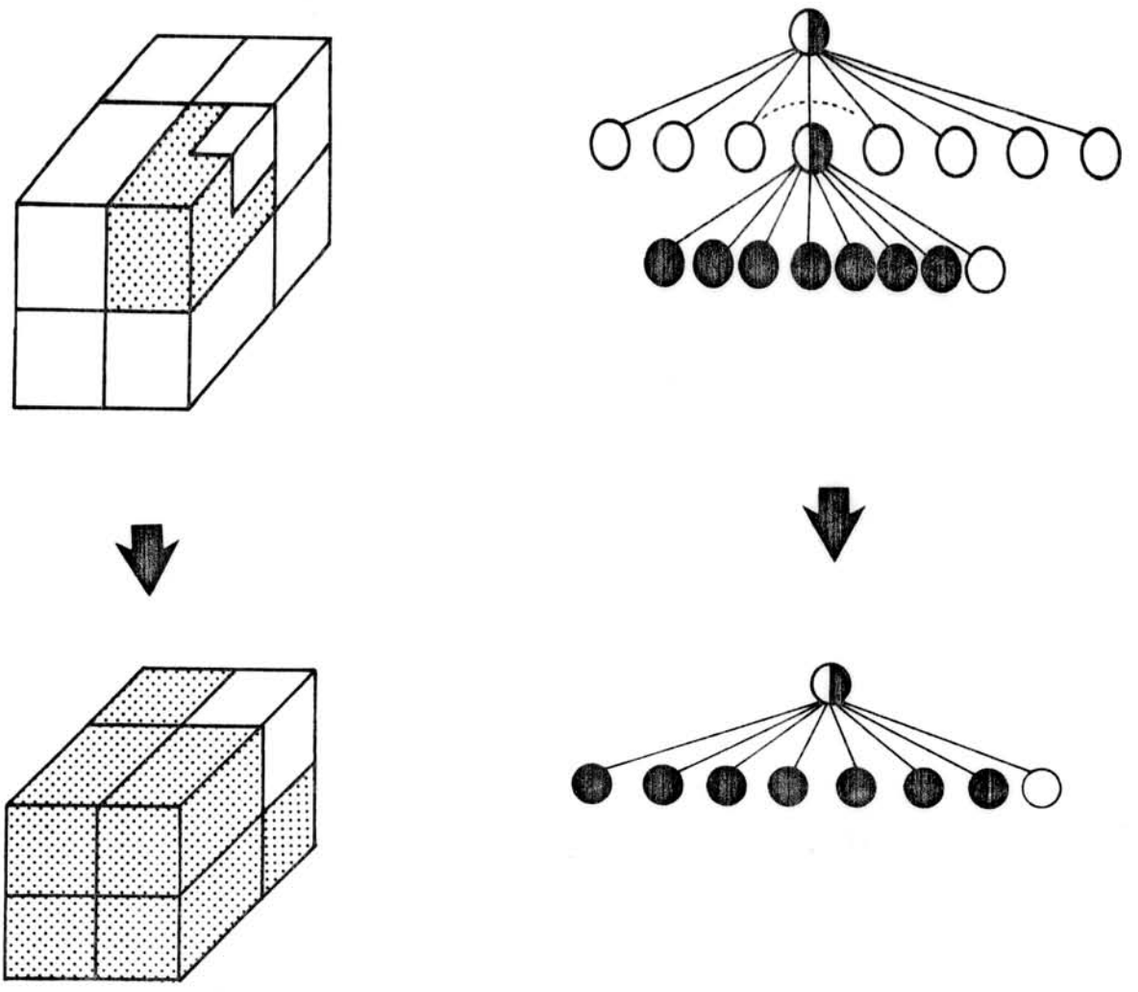


FIG. 5.15 Ampliação pelo fator 2

Segue a descrição geral do algoritmo para mudanças de escala.

```
procedure Escala (tipo);
 (* caso tipo for ampliação, serão geradas 8 novas árvores *)

begin
   case tipo of
      redução: begin
                  while S.Apont ≠ λ do
                  begin
                     (* adiciona o valor 0 no nível 0 de S.Nodo *)
```

```
                S.Apont:=S.Prox;
            end;
            (* grava nova árvore *);
        end;

ampliação: for f:=0 to 7
            do while S.Apont ≠λ and S.Nodo[0] = f
                do (* retira o valor de S.Nodo[0] *);
                (* grava nova árvore *)
end; (* operação mudança de escala *)
```

5.3.7 Seccionamento

A operação de seccionamento permite que sejam obtidos objetos seccionando-se o universo pela metade em qualquer uma das dimensões X, Y e Z. Serão gerados dois novos universos, (consequentemente, duas novas octrees) referentes as partes seccionadas do universo anterior (figura 5.16).

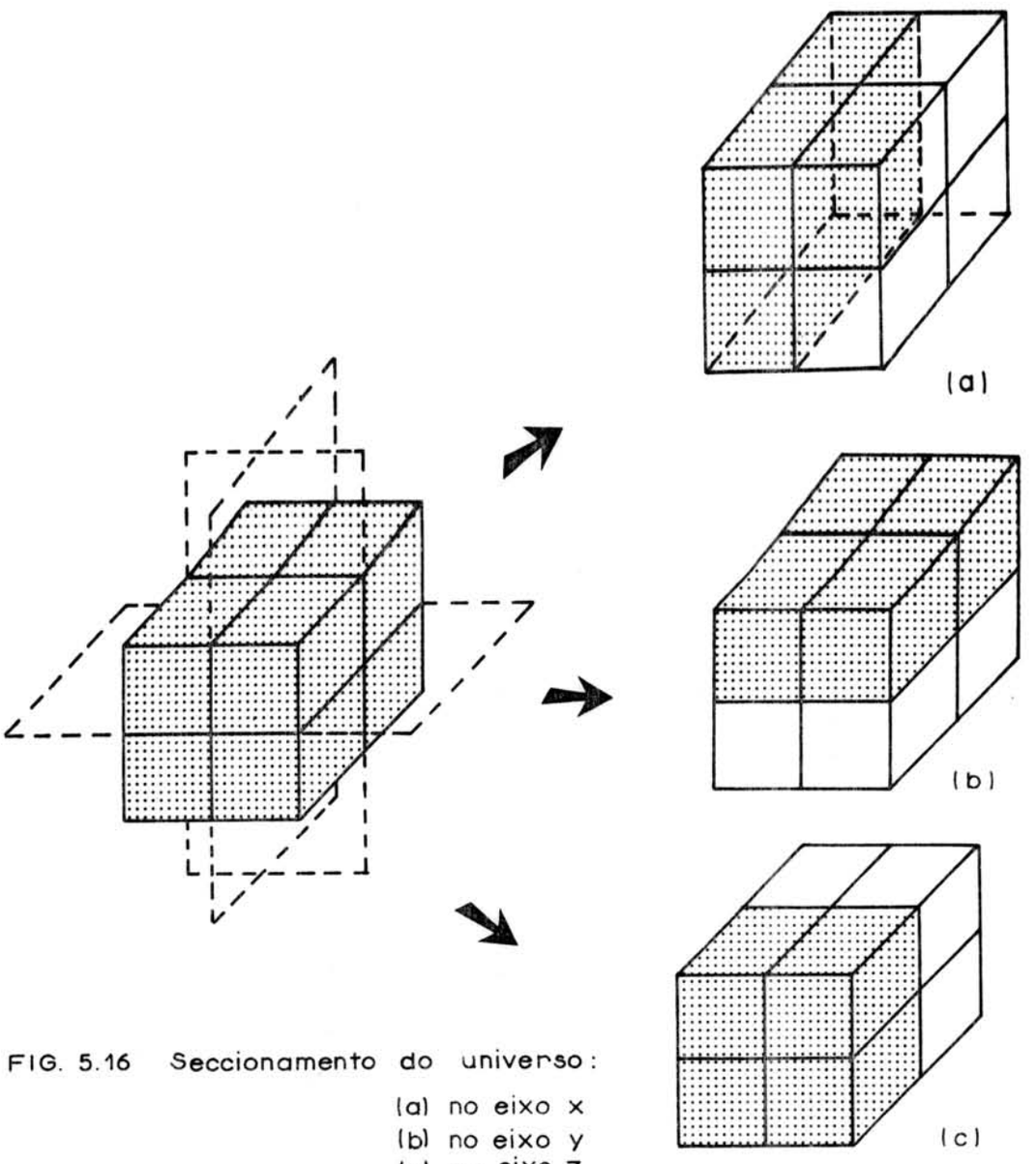



FIG. 5.16 Seccionamento do universo:
(a) no eixo x
(b) no eixo y
(c) no eixo z

Segue o algoritmo geral para a operação de seccionamento:

```
procedure seccionamento (dimensão);
begin
   case dimensão of
      x: begin
            octree-1:= (*conteúdo dos nodos referentes aos octantes 0,
                        2, 4, 6,*)
```

```
            octree-2:= (*conteúdo dos nodos referentes aos octantes 1,
                        3, 5, 7*)
         end;
      y: begin
            octree-1:= (*conteúdo dos nodos referentes aos octantes 0,
                        1, 4, 5*)
            octree-2:= (*conteúdo dos nodos referentes aos octantes 2,
                        3, 6, 7*)
         end;
      z: begin
            octree-1:= (*conteúdo dos nodos referentes aos octantes 0,
                        1, 2, 3*)
            octree-2:= (*conteúdo dos nodos referentes aos octantes 4,
                        5, 6, 7*)
         end;
end;
```

O algoritmo pode ser generalizado usando planos colocados em pontos arbitrários do universo, mas sempre ortogonais a um dos eixos e paralelos a outro, seguindo os procedimentos adotados para a geração de vistas de seções planas (cf. seção 6.4.2.2).

### 5.3.8 Geração de Primitivos

O N-MOS oferece ao usuário a possibilidade de gerar os objetos primitivos necessários à sua aplicação. Conforme já visto, há duas

classes de objetos primitivos: os pré-definidos e os gerados automaticamente.

O N-MOS suporta sólidos básicos pré-definidos como o cubo, cone, esfera e cilindro. Estes sólidos estão representados de forma normalizada e se não forem atribuídos valores iniciais de suas medidas, eles receberão os valores "default" do sistema.

Cada sólido primitivo é englobado sobremedida pelo seu próprio universo e quando for realizada uma operação de busca ele é levado para o universo de trabalho e posicionado no octante 0 (zero) (figura 5.17).

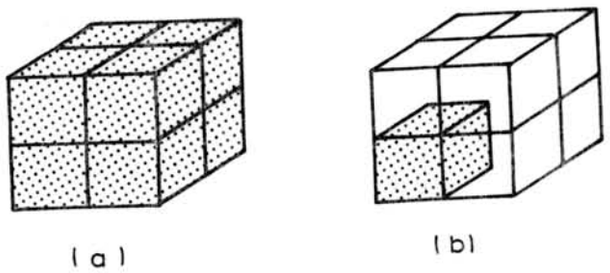



FIG. 5.17 Sólido primitivo visto:
(a) no seu próprio universo
(b) posicionado no universo de trabalho

Os valores de medidas, quando atribuídos, devem ser números inteiros de 0 a 5 ou 9 que indicam o nível de resolução de posicionamento do universo primitivo dentro do universo de trabalho (ver figura 5.18). O valor 9 indica que o universo do primitivo será do tamanho do universo de trabalho.

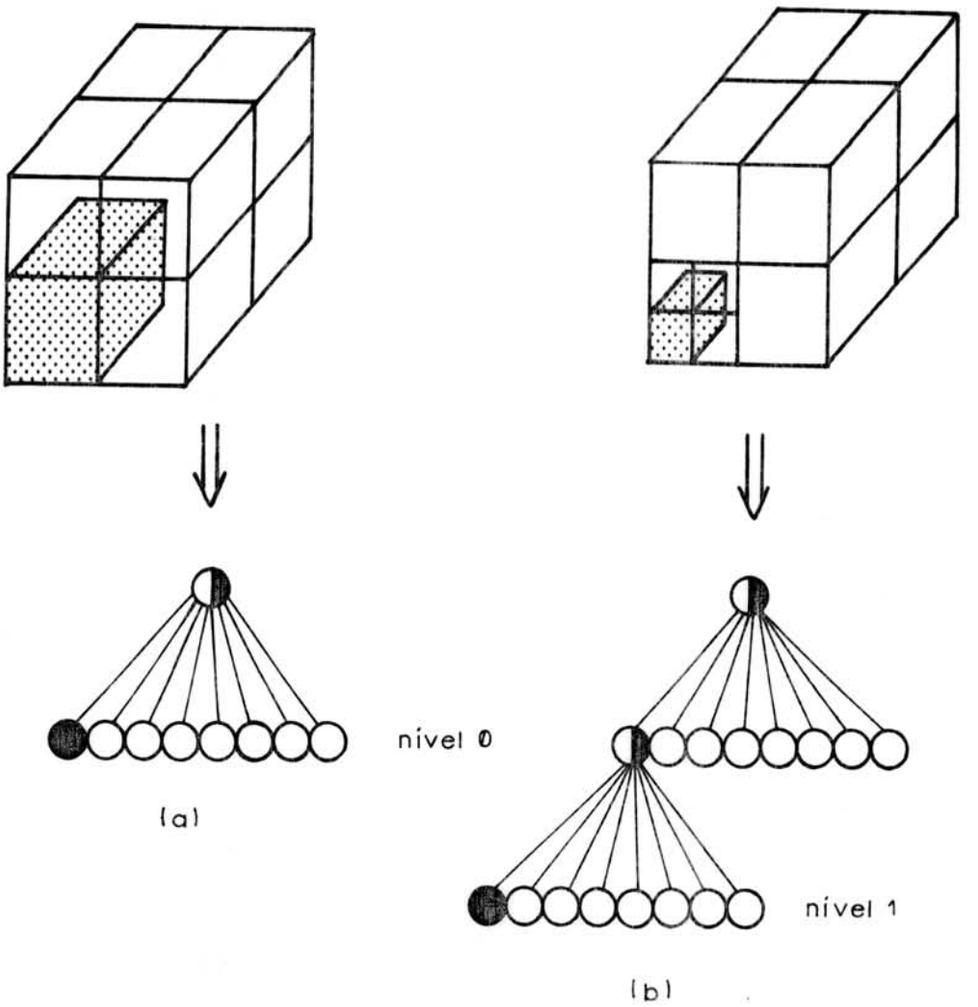



FIG. 5.18 Primitivo cubo englobado pelo universo de trabalho:
(a) no nível 0
(b) no nível 1

Os sólidos básicos gerados automaticamente, previstos no sistema, são os prismas e as pirâmides. Atualmente, está implementada a geração de prismas.

O algoritmo de geração de prismas permite que seja criado um prisma com até 15 lados. O tratamento inicial para o cálculo dos endereços de nodos é todo realizado sobre uma quadtree, onde é construído o polígono base. A construção do polígono base do prisma envolve o cálculo dos pontos iniciais e finais das arestas do polígono. Como ele é circunscrito por uma circunferência, é fácil calcular os pontos ini-

ciais e finais através das funções seno e cosseno.

O polígono gerado será uma figura regular, convexa e alinhada com a face frontal do universo. As arestas inicial e final terão os seus pontos inicial e final, respectivamente, posicionados no ponto de engatilhamento (x=1.0 e y=0.0).

Após a construção do polígono base do prisma são calculados os endereços de nodos através da dedução de quais quadrantes as arestas cortam e quais deles são totalmente preenchidos ou vazios. Isto é feito determinando-se se os pontos mínimo e máximo dos quadrantes estão dentro ou fora do polígono base.

Após serem calculados os endereços de nodos na quadtree é aplicado um algoritmo que estende para a dimensão Z todos os valores de endereços gerados, originando novos endereços de nodos, agora da octree.

A seguir é apresentado o algoritmo geral para a geração de prismas:

```
procedure Geração-de-Prismas (Nro-Lados)
begin
    Constrói-polígono-base;
    for I:=0 to 3 do
    begin
```

```
(* Inicializa base da pilha com os valores iniciais e finais de
  comparação. Para o quadrante:
  0: Pmin (-1,-1), Pmax (0,0)
  1: Pmin (0,0),   Pmax (1,-1)
  2: Pmin (-1,1),  Pmax (0,0)
  3: Pmin (0,0),   Pmax (1,1) *)
quadr:=0;
while SP < SB (* SP= "stack pointer"
                 SB= indicador de base da pilha *)
do
   case quadr of
      0: begin
            (* Calcula endereço para o quadrante 0 *);
            (* Verifica estado do endereço calculado: preenchi-
               do, vazio ou semi-preenchido *);
            quadr:=0;
         end
      1: begin
            (* Calcula o endereço para o quadrante 1 *);
            (* Verifica estado do endereço calculado: preenchi-
              do, vazio ou semi-preenchido *);
            quadr:=1;
         end
      2: begin
            (* Calcula endereço para o quadrante 2 *);
            (* Verifica estado do endereço calculado: preenchi-
              do, vazio ou semi-preenchido *);
            quadr:=2;
```

```
                end
            3: begin
                (* Calcula endereço para o quadrante 3 *);
                (* Verifica estado do endereço calculado: preenchi-
                  do, vazio ou semi-preenchido *);
                quadr:=3;
                end
        end
        (* calcula os endereços na dimensão Z *)
end (* geração de prismas *)
```

# 6 IMPLEMENTAÇÃO DE OUTRAS FUNÇÕES DO SISTEMA

## 6.1 Introdução

Para suportar a implementação do N-MOS de forma expansível, outras funções foram desenvolvidas e anexadas ao sistema de forma modular.

As estruturas de armazenamento foram implementadas e encapsuladas por uma interface de acesso, permitindo desta forma, que sejam substituídas futuramente, por outras. O N-MOS trata apenas das octrees.

Outro módulo implementado foi uma interface usuário-sistema simplificada para permitir a utilização experimental do núcleo implementado.

Finalmente, algumas funções de consulta envolvendo as informações geométricas dos objetos foram desenvolvidas para efeitos de estudo das octrees.

## 6.2 Módulo Consultor

O módulo Consultor implementado realiza o cálculo de algumas propriedades e verifica interferência espacial de dois objetos. As propriedades cujos cálculos foram implementados são: volume, peso e

área de superfície. A definição básica de cada procedimento é dada a seguir. Cabe salientar que os resultados dos cálculos são valores aproximados. Assim, quanto maior a resolução do universo, mais reais serão os valores resultantes.

### 6.2.1 Volume

O volume de qualquer objeto definido pela decomposição do espaço que ele ocupa em cubos, é obtido pela fórmula que calcula o volume de um cubo.

Sabendo-se que o cubo faz parte da família dos prismas e que o volume de um prisma (V) é o produto da área de base (B) pela medida de sua altura (H), temos no caso particular do cubo que

$V = B.H \quad B = L.L \quad V = L.L.L$ onde L é o tamanho do lado.

O algoritmo para cálculo de volume verifica o número de níveis da octree para determinar o tamanho L do cubo do nível de maior resolução e então aplica a fórmula de volume para cada nodo da árvore em estado de preenchido. A soma do volume de todos os nodos corresponde ao volume do objeto.

### 6.2.2 Área de Superfície

Atualmente é calculada apenas a área da base do objeto.

O algoritmo que trata do cálculo da área da base varre a árvore referente ao sólido indicado e verifica o número de nodos cujos endereços possuem os valores indicativos dos octantes 0, 1, 4 e 5. A área é dada por

Ab= N.L.L, onde N é o número de nodos e L é o tamanho do lado do cubo no nível de maior resolução.

O exemplo da figura 6.1 ilustra o cálculo da área da base.

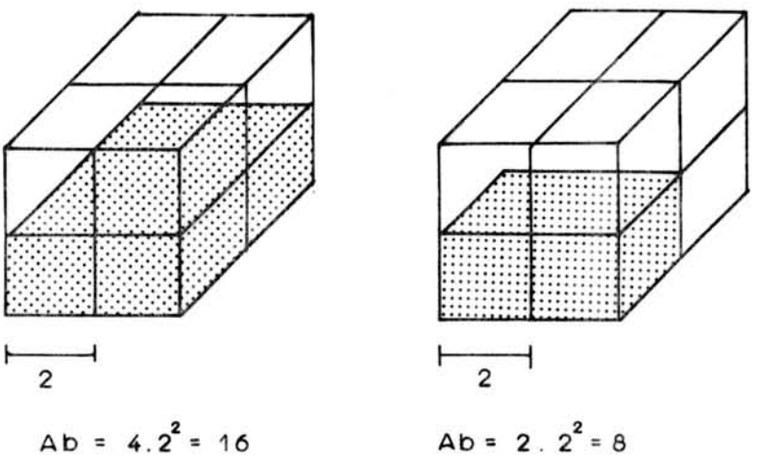



FIG. 6.1 Cálculo da área de base

6.2.3 Peso

Calcular o peso de um objeto representado por uma octree equivale a somar os pesos atribuídos a cada nodo considerado preenchido. O algoritmo é semelhante ao do cálculo de volume apresentado anteriormente.

### 6.2.4 Verificação de Interferência Espacial

Determinar interferências é uma tarefa importante no caso em que o usuário necessita ter certeza de que dois objetos não se sobrepõem. Pode-se utilizar um procedimento análogo para verificar se dois objetos não coincidem por faces ou arestas.

O algoritmo implementado percorre as árvores correspondentes aos objetos e verifica se os endereços de nodos de uma árvore são iguais aos da outra ou são vizinhos por face e aresta. Os dois objetos devem ser representados por árvores com igual número de níveis; portanto, uma conversão prévia é necessária nos casos em que isto não se verifica.

Para a verificação de vizinhança de um octante identificado por um endereço de nodo, o algoritmo deduz quais os octantes são seus vizinhos por face e quais são os seus vizinhos por aresta. A figura 6.2 ilustra três objetos A, B e C onde o processo de verificação de interferências determinará vizinhanças. Entre os objetos A e B, B e C há coincidência por uma de suas faces e, os objetos A e C, por uma aresta.

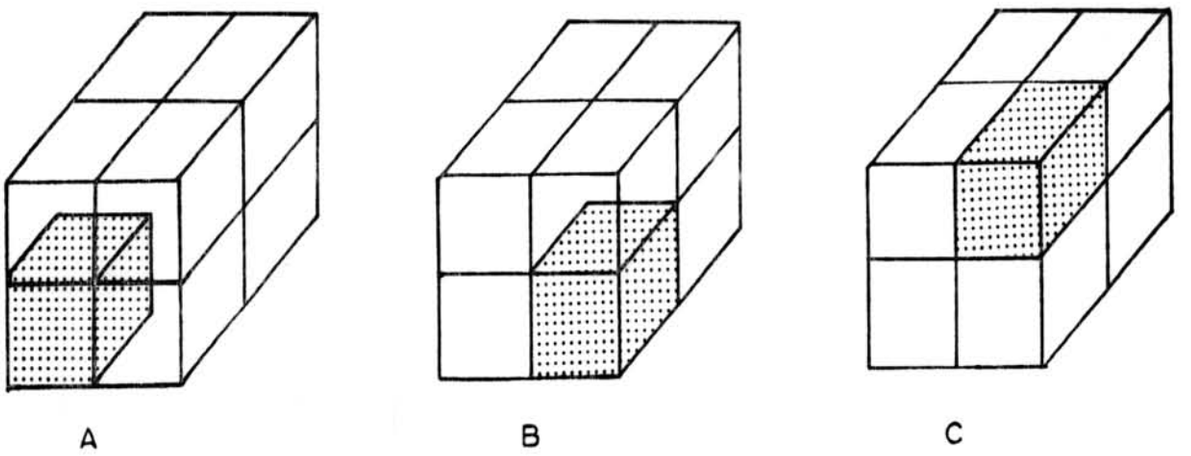



FIG. 6.2 Interferência por vizinhança

## 6.3 Interface de Acesso às Estruturas de Armazenamento

As operações de acesso utilizadas pelo N-MOS tem por função realizar as atualizações sobre as estruturas de dados. São quatro as operações de acesso: inserção de sólidos na estrutura de dados, remoção de sólidos da estrutura de dados, busca e alteração dos sólidos já armazenados na estrutura de dados. A operação de busca é também utilizada pelo Consultor para acesso às informações necessárias aos seus módulos internos.

### 6.3.1 Inserção

Usada para inserir uma nova entidade e um novo objeto e suas especificações na estrutura de dados. A inserção de uma entidade ou um objeto será feita quando o usuário efetivar uma operação de salvamento na estrutura de dados.

O sistema faz o controle automático a quais estruturas ele deve fazer acesso para a operação de inserção.

### 6.3.2 Remoção

Usada para uma possível eliminação do objeto da estrutura de dados. O N-MOS oferece dois tipos de operações de remoção: remoção lógica e física.

A remoção lógica remove temporariamente o objeto do conjunto em que ele aparece e, mesmo assim, continua armazenado na estrutura de dados. A remoção física elimina o objeto, de fato, da estrutura de dados. Se o objeto existe na área de trabalho, uma mensagem indica que ele não existe na estrutura de dados mas está presente na área de trabalho.

No caso do usuário executar uma remoção lógica, o sistema permite que o objeto seja restaurado através da opção "restauração" por ele indicada. Se ele desejar realmente que o objeto seja removido da estrutura de dados, basta executar a remoção física.

O sistema faz o controle automático a quais estruturas deve ser feito acesso para a operação de remoção.

### 6.3.3 Alteração

Usada para as possíveis alterações efetuadas sobre um objeto. Basicamente, uma modificação acontece quando forem executadas as operações de combinação, transformação e seccionamento.

A operação de alteração somente é vista a nível de usuário. Em termos de sistema, quando um objeto é modificado, ele é tratado como se fosse um novo objeto e assim inserido na estrutura de dados. O objeto anterior permanecerá no seu formato original na estrutura de dados ou então será removido após seu uso.

### 6.3.4 Busca

Usada para recuperar entidades e objetos das estruturas de dados. Através desta operação, o usuário busca o objeto armazenado na estrutura de dados em disco e traz para a área de trabalho em memória.

A busca será realizada pelo nome e mensagens indicativas de erro serão mostradas caso o nome do objeto não confira com os nomes armazenados nas estruturas de dados.

Esta operação não acarreta nenhuma mudança sobre as estruturas de dados.

### 6.3.5 Outras Operações

O N-MOS oferece, além das operações citadas acima, outras que são de auxílio ao usuário e que de certa forma exigem acesso às estruturas de dados. São operações de consulta e lista diretório.

As operações de consulta auxiliam o usuário a obter informações sobre os atributos de um objeto armazenado na estrutura Entidades.

A operação lista diretório devolve o nome de todos os objetos pertencentes a uma entidade.

## 6.4 Interface Usuário - Sistema

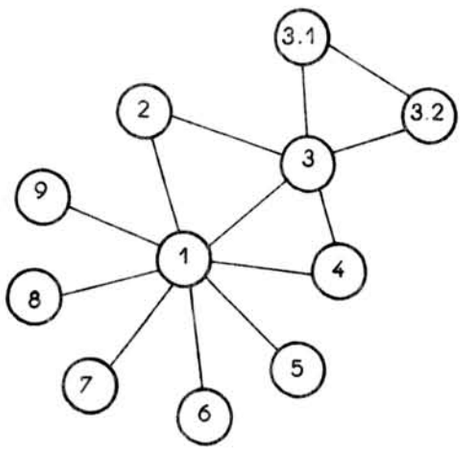
Para permitir acesso ao N-MOS e ao Consultor, foi implementada uma interface usuário-sistema simplificada. As entradas do sistema são feitas através de cardápios chamados via teclado alfanumérico e as saídas em vídeo e impressora.

6.4.1 Entradas do Sistema

A figura 6.3 mostra a arquitetura geral da interface.

O módulo de controle da interface que gerencia o primeiro nível de operação do sistema, permite que o usuário indique, através do teclado alfanumérico, a opção escolhida dentre aquelas apresentadas no cardápio. A nova tela, referente à opção escolhida, é exibida e o módulo é ativado.

A figura 6.4 ilustra como o sistema apresenta uma tela para o usuário. A tela é dividida em cinco áreas: área de apresentação do cabeçalho do sistema, área de apresentação do módulo de trabalho, área das opções do cardápio, área de exibição gráfica ou de trabalho e área de lembretes e mensagens para usuário.




1 - módulo de controle
2 - geração de primitivos
3 - criação e alteração de objetos
  3.1 - transformação
  3.2 - combinação
4 - busca de objetos
5 - remoção de objetos
6 - cálculo de propriedades
7 - consulta de propriedades
8 - exibição
9 - impressão

FIG.6.3 Módulo de gerenciamento de telas

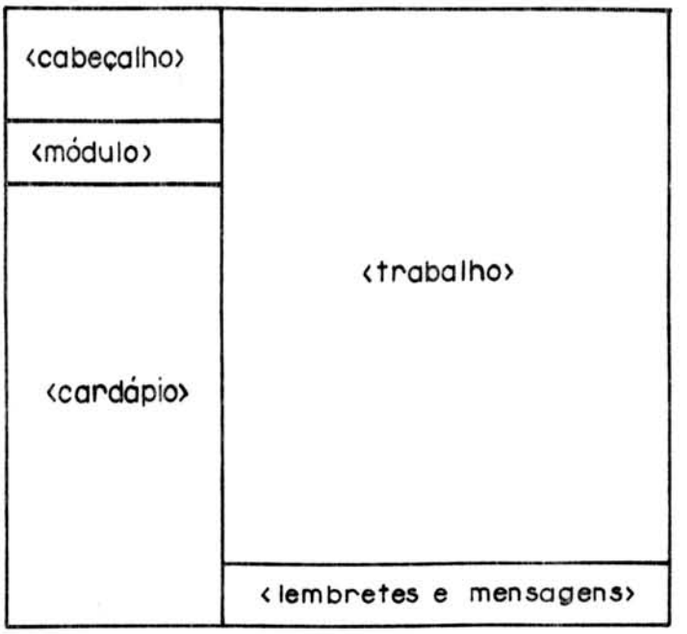



FIG.6.4 "Layout" da tela do N-MOS

O teclado é ainda utilizado para a atribuição de nomes aos objetos e durante a entrada de valores numéricos para as operações implementadas.

As operações disponíveis para o usuário nos diversos módulos mostrados na figura 6.3, são listadas abaixo.

| MÓDULO | FUNÇÕES | CHAMA MÓDULO |
|---|---|---|
| Controle | gera primitivos | Primitivos |
| | cria objetos | Criação/Aletração |
| | altera objetos | Criação/Aletração |
| | remove objetos | Remoção |
| | busca objetos | Busca |

| MÓDULO | FUNÇÕES | CHAMA MÓDULO |
|---|---|---|
| Controle | calcula propriedades | Cálculos |
| | consulta | Consulta |
| | exibe objeto | Exibição |
| | exibe primitiva | Exibição |
| | lista diretório | Diretório |
| | auxílio | Auxílio |
| | fim | - |
| Primitivos | gera prismas | Prismas |
| | gera pirâmides | Pirâmides |
| | auxílio | Auxílio |
| | retorno | Controle |
| Criação/Alteração | gera primitivos | Primitivos |
| | transforma objetos | Transformações |
| | combina objetos | Combinações |
| | busca objetos | Busca |
| | lista diretório | Dieretório |
| | auxílio | Auxílio |
| | retorno | Controle |
| Transformações | translada | Translação |
| | rotaciona | Rotação |
| | amplia/reduz | Escala |
| | cancela | Criação/Alteração ou Combinações |
| | auxílio | Auxílio |

| MÓDULO | FUNÇÕES | CHAMA MÓDULO |
|---|---|---|
| Transformações | retorno | Criação/Aleração ou Combinações |
| Combinações | transforma | Transformações |
| | une | União |
| | intersecciona | Intersecção |
| | subtrae | Diferença |
| | cancela | Criação/Alteração |
| | auxílio | Auxílio |
| | retorno | Criação/Alteração |
| Exibição | vistas ortográficas | Controle |
| | seções planas | Controle |
| | auxílio | Auxílio |
| | lista diretório | Diretório |
| Busca | entra parâmetros | Controle ou Criação/Alteração |
| Remoção | remoção lógica | Controle |
| | remoção física | Controle |
| Cálculos | volume | Controle |
| | peso | Controle |
| | área da base | Controle |
| | retorno | Controle |
| | auxílio | Auxílio |
| Consultas | material | Controle |
| | cor | Controle |

| MÓDULO | FUNÇÕES | CHAMA MÓDULO |
|---|---|---|
| Consultas | grau de acabamento | Controle |
| | auxílio | Auxílio |
| | retorno | Controle |

Os módulos que realizam as operações de combinação e de transformação geométrica solicita ao usuário os parâmetros necessários à execução das operações. O mesmo ocorrendo para os módulos de geração de prismas e pirâmides.

Uma operação de auxílio permite que seja oferecida ao usuário ajuda em qualquer etapa do trabalho. As instruções emitidas são referentes a etapa em que ele se encontra.

### 6.4.2 Saídas do Sistema

As saídas do sistema podem ser obtidas de duas formas: exibição em vídeo gráfico dos objetos tridimensionais e impressão em modo "hardcopy" dos objetos exibidos no vídeo gráfico.

A exibição gráfica dos objetos é essencial num sistema de modelagem. O objetivo é transformar os objetos tridimensionais em imagens bidimensionais correspondendo à vista do objeto a partir de um ponto qualquer do espaço e garantir que superfícies ocultas do objeto ou objetos que se encontram atrás deles não sejam exibidos.

Duas estruturas de árvores serão necessárias na exibição: a octree que representa o espaço do objeto e a quadtree que representa o espaço de visualização. A octree é utilizada como estrutura de entrada e a quadtreee como estrutura de saída. Dessa forma, todos os cubos visíveis representados pelos nodos da octree serão projetados na superfície de exibição.

As imagens podem ser obtidas através de diferentes tipos de projeções /CAR 78/, entre outras por vistas ortográficas e por seções planas. Projeções em perspectiva são as que mais aproximam a imagem do objeto com a da realidade vista pelo olho humano. Os algoritmos que tratam deste tipo de projeção para octrees são complexos e envolvem operações de multiplicação e divisão de inteiros /MEA 80/. Baseiam-se na aplicação de transformações geométricas que deforma o objeto de tal forma que uma projeção ortográfica produza o mesmo resultado visual. Como tais deformações não estão implementadas no N-MOS, atualmente, a exibição de objetos restringe-se à produção de vistas ortográficas e de seções planas.

6.4.2.1 Vistas Ortográficas

O algoritmo de exibição que proporciona a obtenção das seis vistas ortográficas comuns (frontal, posterior, lateral à esquerda, lateral à direita, superior e inferior) foi primeiramente proposto por Meagher /MEA 80/, /MEA 82b/.

O algoritmo percorre a octree numa ordem apropriada (dependendo do ponto de visualização) e todos os nodos preenchidos são exibidos no vídeo, isto é, os nodos são buscados (geralmente numa sequência de frente para trás) e mapeados para uma quadtree que representa a superfície de exibição. Assim, cada "face" do cubo que representa o universo de modelagem é convertida para uma quadtree. O algoritmo é escrito para a "face" frontal e a obtenção das outras vistas é feita mediante a aplicação prévia de uma rotação.

Entretanto, este algoritmo baseia-se fundamentalmente na exibição de faces preenchidas do objeto.

O algoritmo desenvolvido no N-MOS obtém a descrição das vistas ortográficas pela determinação das arestas limitantes das faces a serem exibidas. Esta descrição é representada numa quadtree.

A vista ortográfica será gerada percorrendo-se a estrutura em árvore que está armazenada em pré-ordem, isto é, as subárvores são armazenadas da esquerda para a direita onde o primeiro nodo na estrutura é o filho mais à esquerda. Assim, os octantes representados pelos nodos da árvore são analisados da face frontal para trás.

Para a determinação das arestas das faces a serem exibidas (as visíveis), o algoritmo deve ser capaz de verificar quais nodos são adjacentes àquele que se está analisando e quais nodos o escondem. Para determinar os nodos adjacentes e ocultantes é necessário aplicar

regras que garantam a generalidade do algoritmo. As regras descritas a seguir são aplicadas para os octantes 0, 1, 2, e 3. As mesmas regras serão aplicadas, respectivamente, para os octantes 4, 5, 6 e 7, substituindo-se os valores correspondentes, sem perda da generalidade.

Seja n, o nível do nodo e $D_1 D_2 \ldots D_n$ os dígitos componentes do seu endereço. Seja N, S, L, O, NE, SE, NO e SO as posições "geográficas" norte, sul, leste, oeste, nordeste, sudeste, noroeste e sudoeste, respectivamente; os nodos adjacentes determinados farão parte de uma das posições "geográficas" relativas ao nodo de interesse (figura 6.5).

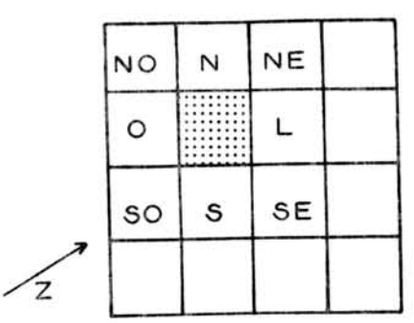



FIG. 6.5 **Posição** "geográfica" dos vizinhos do nodo cujo endereço é 21.

As regras para determinar os vizinhos de um nodo podem ser divididas em quatro, conforme o dígito terminal do endereço de nodo:

1) Para $D_1 D_2 \ldots D_n 0$ então

N : $D_1 D_2 \ldots D_n 2$

S : $D_1 D_2 \ldots (D_n - 2)2$ ou

$D_1 D_2 \ldots (D_{n-1} - 2)(D_n + 2)2$ ou

...

L : $D_1 D_2 \ldots D_n 1$

O : $D_1 D_2 \ldots (D_n - 1)1$ ou

$D_1 D_2 \ldots (D_{n-1} - 1)(D + 1)1$ ou

...

NE : $D_1 D_2 \ldots D_n 3$

SE : $D_1 D_2 \ldots (D_n - 2)3$ ou

$D_1 D_2 \ldots (D_{n-1} - 2)(D_n + 2)3$ ou

...

NO : $D_1 D_2 \ldots (D_n - 1)3$ ou

$D_1 D_2 \ldots (D_{n-1} - 1)(D_n + 1)3$ ou

...

SO : $D_1 D_2 \ldots (D_n - 3)3$ ou

$D_1 D_2 \ldots (D_{n-1} - 2)\{D_n + [(D_n + 3) \text{ MOD } 3]\}$ ou

...

2) Para $D_1 D_2 \ldots D_n 1$, então

N : $D_1 D_2 \ldots D_n 3$

S : $D_1 D_2 \ldots (D_n - 2)3$ ou

$D_1 D_2 \ldots (D_{n-1} - 2)(D_n + 2)3$ ou

...

L : $D_1 D_2 \ldots (D_n + 1)0$ ou

$D_1 D_2 \ldots (D_{n-1} + 1)(D_n - 1)0$ ou

...

O : $D_1 D_2 \ldots D_n 0$

NE : $D_1 D_2 \ldots (D_n+1)2$ ou

$D_1 D_2 \ldots (D_{n-1}+1)(D_n-1)2$ ou

...

SE : $D_1 D_2 \ldots (D_n-1)2$ ou

$D_1 D_2 \ldots (D_{n-1}-1)(D_n+1)2$ ou

...

NO : $D_1 D_2 \ldots D_n 2$

SO : $D_1 D_2 \ldots (D_n-2)2$ ou

$D_1 D_2 \ldots (D_{n-1}-2)(D_n+2)2$ ou

...

3) Para $D_1 D_2 \ldots D_n 2$, então

N : $D_1 D_2 \ldots (D_n+2)0$ ou

$D_1 D_2 \ldots (D_{n-1}+2)(D_n-2)0$ ou

...

S : $D_1 D_2 \ldots D_n 0$

L : $D_1 D_2 \ldots D_n 3$

O : $D_1 D_2 \ldots (D_n-1)3$ ou

$D_1 D_2 \ldots (D_{n-1}-1)(D_n+1)3$ ou

...

NE : $D_1 D_2 \ldots (D_n+2)1$ ou

$D_1 D_2 \ldots (D_{n-1}+2)(D_n-2)1$ ou

...

SE : $D_1 D_2 \ldots D_n 1$

NO : $D_1 D_2 \ldots (D_n+1)1$ ou

$D_1 D_2 \ldots (D_{n-1}+1)(D_n-1)1$ ou

...

SO : $D_1 D_2 \ldots (D_n-1)1$ ou

$D_1 D_2 \ldots (D_{n-1}-1)(D_n+1)1$ ou

...

4) Para $D_1 D_2 \ldots D_n 3$, então

N : $D_1 D_2 \ldots (D_n+2)1$ ou

$D_1 D_2 \ldots (D_{n-1}+2)(D_n-2)1$ ou

...

S : $D_1 D_2 \ldots D_n 1$

L : $D_1 D_2 \ldots (D_n+1)2$ ou

$D_1 D_2 \ldots (D_{n-1}+1)(D_n-1)2$ ou

...

O : $D_1 D_2 \ldots D_n 2$

NE : $D_1 D_2 \ldots (D_n+3)0$ ou

$D_1 D_2 \ldots (D_{n-1}+D_n)(3-D_n)0$ ou

...

SE : $D_1 D_2 \ldots (D_n+1)1$ ou

$D_1 D_2 \ldots (D_{n-1}+1)(D_n-1)1$ ou

...

NO : $D_1 D_2 \ldots (D_n+2)0$ ou

$D_1 D_2 \ldots (D_{n-1}+2)(D_n-2)0$ ou

...

SO : $D_1 D_2 \ldots D_n 0$

Para verificar os nodos ocultantes basta aplicarmos as mesmas regras acima considerando a relação Oeste e a dimensão Z (figura 6.6). Os fatores aplicados para os octantes 0, 2, 4 e 6 valem para os octantes 1, 3, 5 e 7. As regras para posição Oeste são:

1) Dado $D_1 D_2 \ldots D_n 0$ então $D_1 D_2 \ldots (D_n-4)4$ ou

$D_1 D_2 \ldots (D_{n-1}-4)(D_n+4)4$ ou

...

2) Dado $D_1 D_2 \ldots D_n 4$ então $D_1 D_2 \ldots D_n 0$

3) Dado $D_1 D_2 \ldots D_n 2$ então $D_1 D_2 \ldots (D_n-4)6$ ou

$D_1 D_2 \ldots (D_{n-1}-4)(D_n+4)6$ ou

...

4) Dado $D_1 D_2 \ldots D_n 6$ então $D_1 D_2 \ldots D_n 2$

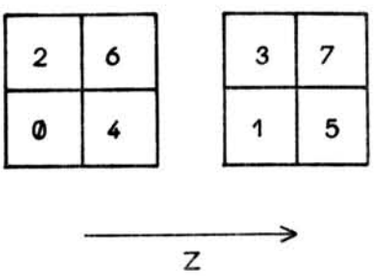



FIG. 6.6 Vista em profundidade dos octantes.

O algoritmo toma um nodo P armazenado na octree e verifica se ele é um nodo que faz fronteira com as bordas do universo ou se é um nodo fronteira do objeto porque seus vizinhos são nodos livres.

Caso o nodo P não seja fronteira, isto é, não possui vizinhos livres, ele é desconsiderado e um novo nodo é tomado. Contudo, se um de seus vizinhos é livre, o nodo é definido em termos de quadtree como um conjunto de quadrantes no grau máximo de resolução. Assim, são encontrados outros possíveis vizinhos livres e o nodo é descrito em termos de "quadtree imagem" (figura 6.7). O endereço do nodo P é mantido de lista de ocultantes.

A lista de ocultantes serve para determinar se um nodo é ocultado por outro. Os nodos ocultantes em relação aquele que está sendo analisado são obtidos aplicando-se as regras descritas anterior-

mente.

A cada nodo retirado da estrutura em árvores é verificado se não existe um que o oculta na lista de ocultantes. Se existir, o nodo é ignorado, caso contrário, o processo é retomado.

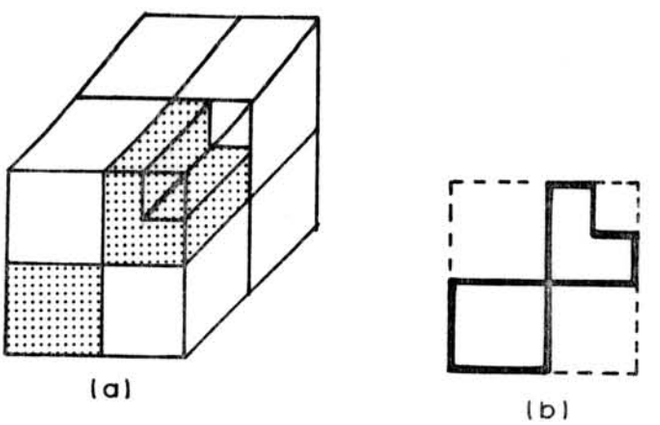



FIG. 6.7 Mapeamento da octree para "quadtree imagem":
(a) representação de um objeto no universo
(b) projeção do objeto numa "quadtree imagem"

6.4.2.2 Seções Planas

A obtenção de seções planas é feita considerando-se planos de cortes no universo (ver figura 6.8).

A única diferença para o algoritmo de obtenção de vistas ortográficas é a escolha do ponto onde inicia a produção da imagem

O algoritmo permite que sejam exibidos cortes do universo apartir de uma distância m onde será colocado o plano para a efetivação da seção Assim, $m = 2^{d1} + 2^{d2} + 2^{d3}$ onde a sequência d1, d2, ..., di é usada para deduzir o endereço do nodo onde a pesquisa deve começar.

O tamanho do universo no sistema varia de $[-2^N, 2^N]$ apesar de sofrer no máximo 6 subdivisões, assim, $5 \leq di \leq 10$ (figura 6.8).

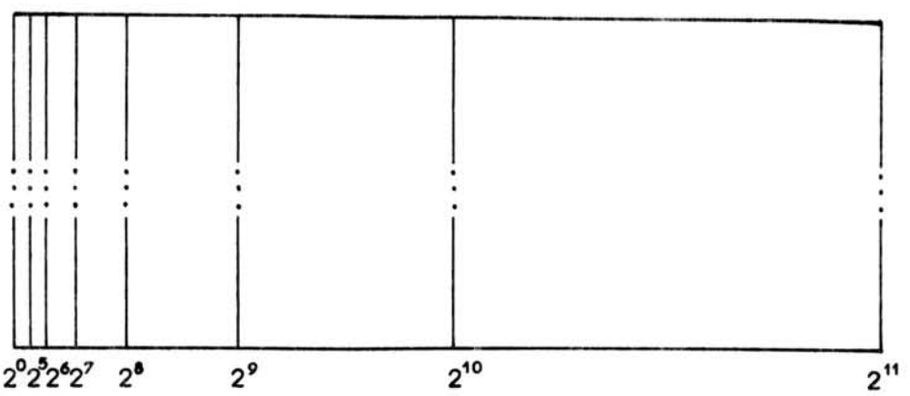



FIG. 6.8 Disposição dos planos de seccionamento

A dedução do endereço de nodo onde começa a pesquisa é feita calculando-se o número de dígitos 0 (zero) que devem preceder o dígito terminal 4. Seja Ed o endereço deduzido; o cálculo para determinar o endereço de nodo é dado por

$$Ed = 11 - (di + 1)\ 4$$

exceto para di = 11, que é o tamanho total do universo, e $0 \quad di \quad 4$. A figura 6.9 ilustra os endereços deduzidos apartir de di.

| di | | endereço |
|---|---|---|
| 10 | - | 4 |
| 9 | - | 04 |
| 8 | - | 004 |
| 7 | - | 0004 |
| 6 | - | 00004 |
| 5 | - | 000004 |

FIG. 6.9: Dedução do endereço de nodo onde a pesquisa deve começar

Depois de determinado o endereço de nodo onde a pesquisa deve começar, o algoritmo verifica se o número de níveis da árvore é menor que o número de níveis do nodo gerado. Se for, traduz-se a árvore para

o mesmo nível do endereço e inicia-se o algoritmo de vista ortográfica do nodo cujo endereço foi obtido.

É importante salientar que considerou-se o dígito terminal 4 porque o plano de seccionamento corta o universo pela profundidade. Para efeitos de vistas ortográficas, 4 é sinônimo de 0 (zero)

Outro detalhe a considerar é que rotações prévias podem ser aplicadas sobre o universo e o algoritmo continua tratando os octantes de "frente para trás", obtendo-se, assim, cortes alternativos.

# 7 ANÁLISE DA IMPLEMENTAÇÃO E CONCLUSÕES

## 7.1 Introdução

Este capítulo faz uma análise dos algoritmos implementados, como também, uma apresentação das limitações do sistema como um todo. São feitas considerações sobre possíveis extensões do sistema implementado e as conclusões finais sobre o trabalho.

## 7.2 Análise da Implementação

Dois motivos principais levaram a escolha da octree como forma de representação das informações geométricas dos objetos: (1) a simplicidade da implementação das funções de modelagem, particularmente, das operações de combinação, e (2) o fato de existirem poucos sistemas adotando esta abordagem. Além do mais, a octree representa explicitamente as porções dos objetos, côncavos ou convexos, com ou sem cavidades, no espaço de modelagem.

A árvore CSG foi utilizada como forma paralela de representação dos objetos. Desse modo, é possível manter uma descrição estruturada dos objetos durante o processo construtivo, podendo-se tratar cada componente de um objeto isoladamente e realizar retroações no processo construtivo quando necessário.

A opção de implementação em microcomputadores compatíveis com o IBM PCxt deveu-se simplesmente pela maior disponibilidade destes equipamentos, na UFRGS, à época de início do trabalho. Esta opção, obviamente, teve influência nos resultados obtidos em termos de performance e nas limitações em termos da resolução máxima do universo. Não houve preocupação quanto ao espaço de armazenamento, a não ser o suficiente para a realização dos testes, pois considerou-se a possibilidade da utilização do computador Burroughs A9P recentemente instalado no Centro de Processamento de Dados da UFRGS.

É comum, em sistemas de modelagem que utilizam outras formas de representação, a realização de operações de ponto flutuante, multiplicações e divisões de inteiros. Isto, de certa forma, diminui a performance desses sistemas, o que não ocorre no N-MOS, uma vez que não são realizadas operações desse tipo (com exceção da etapa de geração de primitivos).

Dos algoritmos que implementam as operações de combinação, o da diferença é o que apresenta maior complexidade de tempo. Os três algoritmos fazem comparações entre os nodos das octrees referentes aos dois sólidos a serem combinados e a octree resultante é uma intercalação dos nodos dependendo da operação aplicada. A diferença, além da comparação, necessita verificar se os nodos que apresentam fator de compressão devem ser desmembrados. O algoritmo de união é da ordem de n1+n2, onde n1 e n2 são os números de nodos das octrees que estão sendo unidas. O algoritmo de intersecção, no melhor caso, é da ordem de n1+1, se a segunda árvore possuir apenas um nodo. No pior caso, é da ordem de n1+n2. O algoritmo da diferença é proporcional a n1+n2,

acrescentando-se ainda o tempo gasto com o desmembramento de nodos com fator de compressão. Este tempo c é constante para cada nodo a ser desmembrado e esta análise é feita apenas para os nodos de uma das octrees. Logo, o algoritmo da diferença, no pior caso, é proporcional a (n1.c.8 + n2) quando todos os n1 nodos necessitam ser desmembrados e n1+1, no melhor caso, quando a segunda árvore possuir um nodo.

O espaço de memória usado para a execução dos algoritmos é dependente do número de nodos das duas octrees em comparação e da octree resultante. O número de nodos da octree resultante para a operação de união será no mínimo igual ou maior ao número de nodos da menor octree. Para a operação de intersecção, o número de nodos da octree resultante será, no máximo, igual ao da menor octree. E, no caso da diferença, o número de nodos da octree resultante será, no máximo, igual ao da primeira octree, ou maior, se existirem nodos com fator de compressão. Atualmente, para minimizar o espaço de memória, cada nodo da octree resultante é armazenado em memória secundária durante a sua geração. Dessa forma, há uma perda de tempo envolvendo o processo de gravação dos nodos.

Os algoritmos que realizam transformações geométricas são mais complexos se comparados com sistemas baseados em polígonos.

O algoritmo de translação não requer cálculos a não ser para determinar os níveis e o tipo de translação ("com gap" ou "sem gap"). O tempo gasto para a execução dos algoritmos é dependente do número de nodos da octree, isto é, do número de níveis de translação e do tempo necessário para a aplicação das regras dos dois tipos de translação.

O tempo de execução da translação "sem gap" é proporcional a n.t.p onde n é o número de nodos da octree, t é o número de níveis de translação e p é o tempo médio de realização das permutações nos endereços de nodos. Faz-se referência a um tempo p médio porque ele não é constante, e sim, dependente de cada nível de translação. O tempo de execução da translação "com gap" é proporcional a n.g, onde n é o número de nodos da octree original e g é o tempo médio de geração de subárvores conforme a regra a ser aplicada.

Para ambos os tipos de translação, a quantidade de memória necessária depende do número de nodos da octree original. No caso de translação "sem gap", o número de nodos da octree resultante não se altera em relação à original. O número de nodos da octree resultante apresenta um considerável acréscimo no caso "com gap" devido a decomposição da octree original em subárvores.

As operações de rotação e mudança de escala implementadas no N-MOS não requerem cálculos. As rotações são executadas simplesmente reordenando-se as subárvores e aplicando-se as permutações necessárias sobre os nodos. A mudança de escala é acompanhada pela adição ou remoção de níveis na raiz da octree. O tempo gasto no algoritmo de rotação é proporcional ao número de nodos da octree, uma vez que apenas são realizadas permutações, com tempo constante, sobre os endereços dos nodos. Na operação de mudança de escala, tanto na redução como na ampliação, o tempo de execução é proporcional ao número de nodos da octree.

Para as duas operações, o espaço em memória necessário é dependente do número de nodos da octree a ser transformada. O número de nodos da octree resultante para a rotação será igual à octree original; para a ampliação, menor, e para a redução, maior.

A geração de prismas envolve operações de ponto flutuante, para o cálculo dos pontos mínimo e máximo de cada quadrante da base do prisma no nível de resolução máximo, e operações de comparação, para a verificação dos pontos interiores, exteriores e de fronteira do prisma a ser gerado.

O tempo de execução do algoritmo é proporcional ao número de subdivisões (em quadrantes) do plano que contém a base do prisma. O número máximo de subdivisões é 4, porque o nível de resolução máximo do universo é 6. O número necessário de subdivisões é dependente do número de lados do polígono base.

O espaço de memória necessário é fixo, consistindo das pilhas que deduzem os quadrantes (da base) cheios, vazios e semi-cheios. O tamanho das pilhas depende do número de subdivisões realizadas no universo (no N-MOS o número máximo de subdivisões é 6). Cada nodo da quadtree gerada é gravado em memória secundária. Após a dedução dos nodos da quadtree, o algoritmo estende para o espaço tridimensional os nodos deduzidos através da permutação dos mesmos na terceira dimensão.

Um nodo gerado no espaço bidimensional dará origem a $2^n$ nodos no espaço tridimensional onde $n$ é o número de níveis do nodo deduzido. Estes $2^n$ nodos são gravados imediatamente em disco. Um prisma de

base triangular, por exemplo, necessita 4 Kbytes para sua representação. Para este prisma, serão gerados no espaço bidimensional 210 nodos no nível de resolução 6, 33 nodos no nível de resolução 5, 20 nodos no nível de resolução 4, 7 nodos no nível de resolução 3 e 1 nodo no nível de resolução 2.

Para a geração de vistas ortográficas e seções planas, os algoritmos necessitam percorrer a octree e, para cada nodo, determinar seus vizinhos e os nodos que o ocultam. Portanto, supondo pesquisa sequencial teríamos um tempo da ordem de $n^2 + nd$, onde n é o número de nodos da octree que representa o objeto e nd é o número médio de nodos na lista de ocultantes. Supondo pesquisa binária este tempo seria diminuído para $\log_2 n + nd$.

No módulo Consultor, os algoritmos de cálculo de peso e volume têm tempo de execução proporcional ao número de nodos da octree que representa o objeto. Já, o algoritmo de cálculo da área da base tem tempo de execução pproporcional ao número de nodos dos octantes 0, 1, 4 e 5 da octree. O espaço de memória necessário é apenas o da octree.

Finalmente, a interface com o usuário implementada é bastante simples visando apenas permitir a interação básica necessária para a realização dos testes dos módulos implementados.

## 7.3 Restrições e Limitações

Os objetos definidos no contexto do N-MOS são objetos tridimensionais não existindo nenhum tratamento para objetos bidimensionais. O menor objeto representável no universo terá o tamanho da unidade de volume (do menor cubo), ou seja, no maior nível de resolução do espaço de modelagem. Assim, um ponto no espaço de modelagem é representado pela unidade de volume.

O nível máximo de resolução no espaço de modelagem no N-MOS é 10 (dez) mas, na implementação atual, é limitado em 6 devido ao pouco espaço de armazenamento disponível.

A principal limitação é quanto ao volume de memória necessário para representar um objeto, principalmente objetos com superfícies curvas como cilindros, cones e esferas, que podem ocupar até 150 KB de memória no N-MOS, pois os nodos que representam os objetos são armazenados através de uma cadeia de caracteres ("string"). Esta limitação pode, no entanto, ser bastante sobrepujada se ao invés de usar uma cadeia de caracteres for usada uma cadeia de dígitos octais representados por 3 bits.

É importante salientar que qualquer objeto representado no N-MOS terá uma representação aproximada, tanto quanto os valores obtidos nos cálculos realizados sobre esse objeto.

## 7.4 Extensões Futuras

Com a aquisição, por parte da Universidade, de um computador de grande porte Burroughs da série A9P com capacidade para 24 MB no sistema monoprocessador e, da futura instalação de uma rede de comunicação entre microcomputadores da série PCxt utilizando o A9P como "mainframe", considera-se a possibilidade de implementar o N-MOS no A9P. Os microcomputadores serviriam como interface de entrada e saída. Dessa forma, seria resolvido o problema de memória (hoje, o maior problema) para representação dos objetos.

A organização funcional do N-MOS em módulos permite que modificações futuras possam ser feitas no módulo desejado sem provocar erros nos outros módulos. Para facilitar o uso do N-MOS seria interessante que a interface com o usuário fosse modificada de modo a introduzir outras técnicas interativas necessárias para melhorar o nível de interação com o usuário. A inclusão de novos módulos também é facilitado. A implementação dos algoritmos genéricos de rotação (ângulos arbitrários) e de mudança de escala (fatores e pontos de origem árbitrários), translação de partes do universo, alinhamento de uma face de um sólido com outra face de outro sólido e exibição em perspectiva será a continuação deste trabalho. Particularmente para a exibição em perspectiva, poder-se-ia adotar um algoritmo de "ray-tracing", possibilitando a produção de imagens com realismo /SAN 85/.

A inclusão do cálculo de novas propriedades, dependendo da aplicação, é desejável num modelador como o N-MOS. Atualmente, o modelo de dados suporta apenas algumas propriedades e o N_MOS realiza al-

guns cálculos como volume, peso e área da base de um objeto. É possível estender-se o modelo e o N-MOS para suportar outras propriedades e novos cálculos além dos já existentes.

Todos os algoritmos implementados no N-MOS, com exceção da geração de sólidos primitivos utilizam operações aritméticas simples, como adição e subtração de inteiros. O N-MOS não trabalha com operações de ponto flutuante. Isto facilitaria a implementação dos algoritmos em VLSI, possivelmente para realização em paralelo, e aumentar o desempenho do sistema.

Finalmente, um trabalho de maior porte seria a integração do N-MOS a um sistema de gerência de banco de dados que implementasse o modelo geométrico descrito através da abordagem Abstração de Dados. Tal SGBD seria orientado a objetos e deveria possuir os requisitos já enumerados na seção 3.3.3. Esta integração seria facilitada pela modularidade apresentada na implementação realizada.

## 7.5 Conclusão

O trabalho apresentou a descrição e a implementação do Núcleo de Modelagem de Objetos Sólidos (N-MOS) do CPD-PGCC da Universidade Federal do Rio Grande do Sul.

A maioria dos sistemas de modelagem de sólidos, com pouquíssimas exceções, é usada em aplicações específicas. Um dos principais objetivos do N-MOS é poder ser utilizado em diversas aplicações. Para

tal, a forma de representação de objetos tridimensionais adotada foi considerada por ser bastante genérica, de tal modo que ela pode ser aplicada, por exemplo, tanto em sistemas para engenharia mecânica quanto em sistemas voltados para a área médica.

Várias formas de representação foram propostas, ao longo dos últimos anos, com a finalidade de gerar e representar modelos tridimensionais em computadores. Quando aplicadas em sistemas de modelagem, elas trazem consigo certas limitações, como por exemplo, a característica de representar e processar um conjunto limitado de objetos. Entre as formas apresentadas, a CSG, B-rep e Sweep são as mais empregadas. Contudo, quando há necessidade de generalizar o uso de tais sistemas modificações substanciais devem ser realizadas.

A idéia de utilizar octrees como forma de representação no N-MOS permite que qualquer objeto possa ser representado até o limite máximo de resolução do universo, de tal forma que os objetos são vistos como sólidos reais. Os objetos deixam de ser representados por polígonos. Os nodos da octree correspondem às regiões do espaço preenchidas pelo objeto. Se recortes sobre o objeto forem executados, o interior será de "material sólido" em vez de uma simples cavidade. Isto permite que o N-MOS opere com representações verdadeiras dos sólidos, ou seja, os sólidos são representados não apenas por superfícies, mas também pelos seus interiores.

Por outro lado, a motivação no uso de SGBDs em aplicações gráficas que envolva modelagem de sólidos é grande.

Estudos sobre abordagens semânticas tem possibilitado grandes avanços no projeto de banco de dados para sistemas CAD. As estruturas de agregação e generalização apresentadas no capítulo 3 são ferramentas poderosas no tratamento de estruturas complexas como é encontrado na representação de dados gráficos e não-gráficos.

Neste trabalho, foram apresentadas as vantagens básicas de se utilizar banco de dados integrado a um sistema de modelagem de sólidos e os requisitos que um banco de dados deve suportar para atender as necessidades de tais sistemas.

O N-MOS na versão atual utiliza arquivos convencionais, podendo ser visto como um sistema experimental e, apesar das limitações do hardware onde foi implementado, seu desenvolvimento se reveste de importância dado o conhecimento adquirido durante o trabalho e a possibilidade de sua continuidade até torná-lo passível de uso por parte da comunidade da UFRGS.

# BIBLIOGRAFIA

/ABR 74/ ABRIAL, J. R. Data semantics. In: DATA BASE MANAGEMENT. Amsterdam, North Holland Publishing Company. 1974. Apud GRABOWSKI, H. e EIGNER, M. Semantic datamodel requeriments and realization with a relational datastructure. Computer-aided Design, Surrey, 11(3):158-68, May 1979.

/AIS 84/ AISH, Robert & NOAKES, Peter. Architeture without numbers - CAAD based on a 3D modelling systems. Computer-aided Design, Surrey, 16(6):321-28, Nov. 1984

/AYA 85/ AYALA, D. et alli. Objects representation by means of nonminimal division quadtrees and octrees. ACM Transactions on Graphics, New York, 4(1):41-59, Jan. 1985.

/BAE 79/ BAER, A.; EASTMAN, C.; HENRION, M. Geometric modelling: a survey. Computer-aided Design, Surrey, 11(5):253-72, Sept. 1979.

/BAR 84/ BARBIC, J.; DACAR, F.; SPEGEL, M. Oriented geometric objects in computer graphics and numerical control. Computer Graphics Forum, Amsterdam, 3(1):103-10, Mar. 1984.

/BOB 85/ BOBROW, James E. NC machine tool path generation from CSG part representations. Computer-aided Design, Surrey, 17(2):69-76, Mar. 1985.

/BOL 85/ BOLTZ, Richard J. & AVERY JR., J. Thomas. Mechanical design-productivity using CAD graphics - a user point of view. IEEE Computer Graphics and Applications, Los Alamos, 5(2):40-4, Febr. 1985.

/BOY 79/ BOYSE, John W. Interference detection among solids and surfaces. Communications of the ACM, New York, 22(1):3-9, Jan. 1979.

/BOY 82/ BOYSE, John W. & GILCHRIST, Jack E. GMSolid: interactive modeling for design and analysis of solids. IEEE Computer Graphics and Applications, Los Alamos, 2(2):27-40, Mar. 1982.

/BRA 75/ BRAID, I. C. The synthesis of solids bounded by many faces. Communications of the ACM, New York, 18(4):209-16, Apr. 1975.

/BRO 82/ BROWN, Christopher. PADL-2: a tecnical summary. IEEE Computer Graphics and Applications, Los Alamos, 2(2):69-84, Mar. 1982.

/BRU 85/ BRUNET, Pere & NAVAZO, Isabel. Geometric modelling using exact octree representation of polyhedral objects. In: EUROPEAN GRAPHICS CONFERENCE AND EXHIBITION, 6., Nice, Sept. 9-13, 1985. Proceedings. Amsterdam, North-Holland, 1985, p.159-69.

/BUC 84/ BUCHMANN, Alejandro P. Current trends in CAD databases. Computer-aided Design, Surrey, 16(3):123-26, May 1984.

/CAR 78/ CARLBOM, Ingrid & PACIOREK, Joseph. Planar geometric projections and viewing transformations. Computing Surveys, New York, 10(4):465-502, Dec. 1978.

/CAR 85/ CARLBOM, Ingrid & VANDERSCHEL, David. A hierarchical data structure for representing the spatial decomposition of 3-D objects. IEEE Computer Graphics and Applications, Los Alamos, 5(4):24-31, Apr. 1985.

/CHA 80/ CHANG, S. K. & CHENG, W. H. A methodology for structured database decomposition. IEEE Transactions Software Engineering, New York, SE-6(2):205-18, Mar. 1980.

/CHE 77/ CHEN, P. P. The Entity - relationship model toward a unified view of data. ACM Transactions on Database Systems, New York, 1(1):9-36, Mar. 1976.

/CHI 85/ CHIYOKURA, Hiroaki & KIMURA, Fumihito. A method of representing the solid design process. IEEE Computer Graphics and Applications, Los Alamos, 5(4):32-41, Apr. 1985.

/COD 70/ CODD, E. F. A Relational model of data for large shared data bank. Communicationns of the ACM, New York, 13(6):377-87, June 1970.

/DAT 82/ DATE, C. J. An Introduction to database systems. 3.ed. Reading, Addison-Wesley, 1982. v.1

/DAT 83/ DATE, C. J. An Introduction to database systems. Reading, Addison-Wesley, 1983. v.2

/DBT 71/ DBTG - CODASYL DATA BASE TASK GROUP REPORT. ACM. 1971. Apud FURTADO, A. L. & SANTOS, C. S. dos. Organização de banco de dados. Rio de Janeiro, Campus, 1979.

/DIT 85/ DITTRICH, Klaus R. & LORIE, Raymond A. Object-oriented database concepts for engineering applications. San Jose, IBM Reseach Laboratory, 1985. (Research Report, RJ 4691 (50029) 5/8/85)

/DOC 81/ DOCTOR, Louis J. & TORBORG, John G. Display techniques for octree-encoded objects. IEEE Computer Graphics and Applications, Los Alamos, 1(3):29-38, July. 1981.

/EAS 79/ EASTMAN, Charles M. & WEILER, Kevin. Gometric modeling using the Euler Operators. Pittsburg, Carnegie-Mellon University, Institute of Physical Planning, 1979. (Research Report nº 78)

/ELL 78/ ELLIOTT, W. S. Interactive graphical CAD in mechanical engineering design. Computer-aided Design, Surrey, 10(2):91-100, Mar. 1978.

/FEN 81/ FENVES, Steven J. Computer-aided desing in civil engineering. Proceeding of the IEEE, New York, 69(10):1240-8, Oct. 1981.

/FIS 79/ FISHER, W. E. PHIDAS - a database management system for CAD/CAM aplication software. Computer-aided Design, Surrey, 11(3):146-50, May 1979.

/FIT 81/ FITZGERALD, William; GRACER, Franklin; WOLFE, Robert. GRIN: interactive graphics for modeling solids. IBM J. Research Development, Armonk, 25(4):281-94, July 1981.

/FOL 82/ FOLEY, James D. & VAN DAM, Andries. Fundamentals of interactive computer graphics. Reading, Addison-Wesley, 1982.

/FUJ 83/ FUJIMURA, K. et alli. Octree algorithms for solid modelling. Tokyo, Departament of Information Science Faculty of Science, University of Tokyo, Jan. 1983. (TR 83-01)

/FUL 81/ FULLENWIDER, Donald R. & LEFEVER, James P. Computer graphics and the practice of architecture. IEEE Computer Graphics and Applications, Los Alamos, 1(4):18-26, Oct. 1981.

/GAR 82a/ GARGANTINI, Irene. An effective way to represent quadtrees. Communications of the ACM, New York, 25(12):905-10, Dec. 1982.

/GAR 82b/ GARGANTINI, Irene. Linear octrees for fast processing of three-dimensional objects. Computer Graphics and Image Processing, New York, 20:365-74. 1982.

/GRA 79/ GRABOSKY, H. & EIGNER, M. Semantic datamodel requeriments and realization with a relational datastructure. Computer-aided Design, Surrey, 11(3):158-68, May 1979.

/GUL 85/ GULIATO, Denise. Sistema gráfico interativo para o projeto de peças mecânicas. Porto Alegre, PGCC da UFRGS, 1985. (Dissertação de Mestrado).

/HAR 82/ HARDER, Theo & REUTER, A. Database systems for non standard applications. West Germany, University Kaiserlautern, 1982. (RT 54/82).

/HAS 82/ HASKIN, Roger L. & LORIE, Raymond A. On extending the functions of relational database systems. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DISTRIBUTED DATA PROCESSING (ACM), Paris, June 23-26, 1982. Proceedings. Amsterdam, North-Holland, June 1982. p. 207-12.

/HAT 84/ HATVANY, J. Computer-aided manufacture. Computer-aided Design, Surrey, 16(3):161-5, May 1984.

/HER 85/ HERBERT, Franz. Solid modelling for architectural design using ocpaths. Computer & Graphics, New York, 9(2):107-16. Apr. 1985.

/HIL 82/ HILLYARD, Robin. The build of solid modelers. IELE Computer Graphics and Applications, Los Alamos, 2(2):43-52, Mar. 1982.

/HUN 79/ HUNTER, G. M. & STEIGLITZ, K. Linear transformation of pictures reepresented by quad trees. Computer Graphics and Image Processing, New York, 10:289-96. 1979.

/JAC 80/ JACKINS, Chris L. & TANIMOTO, Steven L. Octrees and their use in representing three-dimensional objects. Computer Graphics and Image Processing, New York, 14:249-70. 1980.

/KAT 85/ KATZ, Randy H. Computer-aided design databases. IEEE Design and Test, California, 1(2):70-4, Feb. 1985.

/KLI 78/ KLINGER, Allen & ALEXANDRIDIS, Nikitas. Picture decomposition, tree data structures, and identifying directional symmetries as node combinations. Computer Graphics and Image Processing, New York, 8:43-77. 1978.

/KNO 84/ KNOWLES, N. C. Finite element analysis. Computer-aided Design, Surrey, 16(3):134-40. May 1984.

/LAN 84/ LANSDOWN, John & MAVER, Tom. CAD in architecture and building. Computer-aided Design, Surrey, 16(3):148-54, May 1984.

/LEE 83/ LEE, Y.C. & FU, K.S. Integration of solid modeling and database manegement for CAD/CAM. In: DESIGN AUTOMATION CONFERENCE, 20., Miami Beach, June 27-29, 1983. Proceedings. New York, ACM/IEEE 1983. p. 367-733.

/LON 86/ LONGHI, Magalí T. Representação de objetos sólidos por OCTREES. Porto Alegre, PGCC da UFRGS, 1986. (RP-052).

/MAN 82/ MANTYLA, Martti & SULONEN, Reijo. GWB: a solid modeler with-euler operators. IEEE Computer Graphics and Applications, Los Alamos, 2(7):17-31, Sept. 1982.

/MAN 83/ MANTYLA, Martti. Set operations of GWB. Computer Graphics Forum, Amsterdam, 2(2/3):122-34, Aug. 1983.

/MAR 80/ MARKOWSKY, George & WESLEY, Michael A. Fleshing out wire frames. IBM Journal Research Development, Armonk, 24(5):582-97, Sept. 1980.

/MEA 80/ MEAGHER, Donald J. R. Octree encoding: a new technique for the representation, manipulation and display of arbitrary 3-D objects by computer. Troy, Electrical and Systems Engineering Department, Rensselaer Polytechnic, Oct. 1980. (Technical Report IPL-TR-80-111)

/MEA 82a/ MEAGHER, Donald. Geometric modelling using octree encoding. Computer Graphics and Image Processing, New York, 19:129-47. 1982.

/MEA 82b/ MEAGHER, Donald. Efficient synthetic image generation of arbitrary 3-D objects. In: CONFERENCE ON PATTERN RECOGNITION AND IMAGE PROCESSING, June 1982. Proceedings. New York, IEEE, 1982. p. 473-78.

/MEA 84/ MEAGHER, Donald. The Solids Engine: a processor for interactive solid modeling. In: NICOGRAPH'84, Tokyo, 1984.

/MEA 85/ MEAGHER, Donald. The application of octree techniques to 3-D medical imaging. In: ANUAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 7., Chicago, Sept. 27-30, 1985. Proceedings. Chicago, IEEE, 1985.

/MYE 82/ MYERS, Ware. An industrial perspective on solid modeling. IEEE Computer Graphics and Applications, Los Alamos, 2(2):86-97, Mar. 1982.

/NEU 82/ NEUMANN, Thomas & HORNUNG, Christof. Consistency and transactions in CAD databases. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 8., Mexico, Sept. 8-10 1982. Proceedings. Saratoga, VLDB endwment, 1982, p. 181-88.

/OLI 83/ OLIVER, M. A. & WISEMAN, N. E. Operations on quadtrees encoded images. Computer Journal, New York, 1(26):83-91, Feb. 1983.

/PAT 82/ PATNAIK, L. M. & RAMESH, N. Implementation of a interactive relational graphics database. Comput. & Graphics, Great Britain, 6(3):93-6. 1982.

/REQ 77/ REQUICHA, A. A. G. & VOELCKER, H. B. Constructive solid geometry. Rochester, Production Automation Project, the University of Rochester, Nov. 1977. (TM-25)

/REQ 80/ REQUICHA, Aristides A. G. Representations for rigid solids: theory, methods and systems. Computing Surveys, New York, 12(4):437-64, Dec. 1980.

/REQ 82/ REQUICHA, A. A. G. & VOELCKER, H. G. Solid Modeling: a historical summary and contemporary assessment. IEEE Computer Graphics and Applications, Los Alamos, 2(2):9-24, Mar. 1982.

/REQ 83/ REQUICHA, A. A. G. & VOELCKER, H. B. Solid Modelling: current status and research directions. IEEE Computer Graphics and Applications, Surrey, 3(7):25-37, Oct. 1983.

/REQ 85/ REQUICHA, Aristides A. G. & VOELCKER, Herbert B. Boolean operations in solid modeling: boundary voluation and merging algorithms. Proceeding of the IEEE, New York, 76(1):30-44, Jan. 1985.

/ROB 81/ ROBINSON, Hugh. Database analysis and design. England, Adrian V Stokes, 1981. (Polytechnic Computer Science Series)

/ROW 85/ ROWLEY, T. The state of the art in modeling systems. In: ---. CAD and CAM - State of Art Report. England, Pergamon Infotech Limited. 1985. p. 61-9.

/SAM 80a/ SAMET, Hanan. Region representation: boundary codes from quadtrees. Communication of the ACM, New York, 23(3):171-9, Mar. 1980.

/SAM 80b/ SAMET, Hanan. Region Representation: quadtree from boundary codes. Communications of the ACM, New York, 23(3):163-70, Mar. 1980.

/SAM 80c/ SAMET, Hanan. Deletion in two-dimensional quadtrees. Communication of the ACM, New York, 23(12):703-10, Dec. 1980.

/SAM 84/ SAMET, Hanan. The quadtree and related hierarchical data structures. Computing Surveys, New York, 16(2):187-260 June 1984.

/SAN 85/ SANDOR, John. Octree data structure and perspective imagery. Computer & Graphics, New York, 9(4):397-405, Oct. 1985.

/SCH 75/ SCHMID, H. A. & SWENSON, J.R. On the semantics of relational data model. In: ACM SIGMOD INTERNATIONAL CONFRENCE ON MANAGEMENT OF DATA, San Jose, May 14-16, 1975. Proceedings. San Jose, IBM, 1975. p.211-23.

/SHE 82/ SHEPHARD, Mark S.; TONIAS, Constantine N. & WEIDNER, Theodore J. Attribute specification for finite elements models. Comput. & Graphics, Great Britain, 6(2):83-91. 1982.

/SHE 83/ SHENOY, R. S. & PATNAIK, L. M. Data definition and manipulation languages for CAD database. Computer-aided Design, Surrey, 15(3):131-4. May 1983.

/SID 80/ SIDLE, Thomas W. Weaknesses of commercial data base management systems in engineering applications. In: DESIGN AUTOMATION CONFERENCE, 17., Minneapolis, June 23-25, 1980. Proceedings. New York, ACM/IEEE, 1980. p. 57-61.

/SIS 86/ SISTEMA CAD/CAM EUCLID. São Paulo. COMPUGRAF - Centro Administrativo do Grupo Safra, 1986.

/SMI 77a/ SMITH, John Miles & SMITH, Diane C.P. Database abstractions: aggregation. Comunications of the ACM, New York, 20(6):405-13, June 1977.

/SMI 77b/ SMITH, John Miles & SMITH, Diane C.P. Databases abstraction: aggregation and generalization. ACM Transactions on Database Systems, New York, 2(2):105-33, June 1977.

/STA 86/ STALEY, Scott M. & ANDERSON, David C. Functional specification for CAD databases. Computer-aided Design, Surrey, 18(3):132-8, Apr. 1986.

/SUT 63/ SUTHERLAND, I. E. SKETCHPAD: a man-machine graphical communication system. Baltimore, Spartan Books, 1983. Apud FOLEY, James D. & VAN DAM, Andries. Fundamentals of interactive computer graphics. Reading. Addison-Wesley, 1982.

/THA 82/ THALMANN, Daniel; GUIMET, Daniel; s20, Christian. Drawing bridges by computer. Computer-aided Design, Surrey, 14(4):195-9, July 1982.

/TIL 80/ TILOVE, Robert Bruce. Set membership classification: a unified approach to geometric intersection problems. IEEE Transactions on Computers, New York, C-29(10):874-83, Oct. 1980.

/TOK 83/ TOKUNASU, Shinji et alli. Solid model in geometric modeling system: HICAD. In: DESIGN AUTOMATION CONFERENCE,20., Miami Beach, June 27-29, 1983. Proceedings. New York, ACM/IEEE, 1983. p. 360-66.

/TOR 86/ TORIYA, Hiroshi et alli. UNDO and REDO operations for solid modeling. IEEE Computer Graphics and Applications, Los Alamos, 6(4):35-42, Apr. 1986.

/ULF 81/ ULFSBY, Stig; MENN, Steiner; OIAN, Jorn. TORNADO: DBMS for CAD/CAM systems. Computer-aided Design, Surrey, 13(4):193-7, July 1981.

/ULF 82/ ULFSBY, Stig; MENN, Steiner; OIAN, Jorn. TORNADO: a database management system for graphics applications. IEEE Computer Graphics and Applications, Los Alamos, 3(2):71-9, May 1982.

/VOE 77/ VOELCKER, Herbert B. & REQUICHA, Aristides A.G. Geometric modeling of mechanical parts and processes. Computer, Los Angeles, 10(12):48-56, Dec. 1977.

/VOE 78/ VOELCKER, H. et alli. The PADL-1.0/2 system for defining and displaying solid objects. Computer Graphics, New York, 12(3):257-63, Aug. 1978.

/WEI 85/ WEILER, Kevin. Edge-based data structures for solid modeling in curved-surface environments. IEEE Computer Graphics and Applications, Los Alamos, 5(1):21-40, Jan. 1985.

/WES 80/ WESLEY, Michael A. Constructions and use of geometric models. In: COMPUTER-AIDED DESIGN MODELING, SYSTEMS ENGINEERING, CAD-SYSTEMS, Darmstadt. Sept. 8-19, 1980. Proceedings. Berlin, Springer-Verlag, 1980. p. 80-136.

/WIL 76/ WILLEY, D. S. Approaches to computer-aided architectural sketch design. Computer-aided Design, Surrey, 8(3):181-6, July 1976.

/WOO 84/ WOO, Tony C. Interfacing solid modeling to CAD and CAM: data structures and algorithms for decomposing a solid. Computer, Los Angeles, 17(2):44-9, Dec. 1984.

/WOR 84/ WORDENWEBER, B. Finite element mesh generation. Computer-aided Design, Surrey, 16(5):285-91, Sept. 1984.

/YOS 84/ YOSHIRA, Hiroshi; FUJIMURA, Kikuo; KUNII, Toslyasu. Top-down construction of 3-D mechanical object shapes from engineering drawings. Computer, Los Angeles, 17(2):32-40, Dec. 1984.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Pós-graduação em Ciência da Computação

# DEFINIÇÃO E IMPLEMENTAÇÃO DE UM MODELO DE DADOS PARA REPRESENTAÇÃO DE SOLIDOS

Dissertação apresentada aos Srs.

Prof. Dr. Leo Pini Magalhães

Prof. Dr. José Mauro Volkmer de Castilho

Profa. Carla Maria Dal Sasso-Freitas

Visto e permitida a impressão
Porto Alegre, 10/12/87.

Roberto Tom Price
Coordenador do Curso de Pós-Graduação em Ciência da Computação