# AMBIENTE INTELIGENTE PARA CONVERSÃO DE APLICAÇÕES EM LINGUAGEM NATURAL PARA UMA LINGUAGEM DE TRANSFORMAÇÕES

JOSÉ DEMISIO SIMÕES DA SILVA

# AMBIENTE INTELIGENTE PARA CONVERSÃO DE APLICAÇÕES EM LINGUAGEM NATURAL PARA UMA LINGUAGEM DE TRANSFORMAÇÕES

Dissertação apresentada ao Curso de MESTRADO em ENGENHARIA ELÉTRICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

ÁREA DE CONCENTRAÇÃO: PROCESSAMENTO DA INFORMAÇÃO

CARLOS ALBERTO DE OLIVEIRA
JOÃO MARQUES DE CARVALHO
Orientadores

CAMPINA GRANDE
AGOSTO - 1992


S586a Silva, Jose Demisio Simoes da
Ambiente inteligente para conversao de aplicacoes em linguagem natural para uma linguagem de transformacoes / Jose Demisio Simoes da Silva. - Campina Grande, 1992.
134f. : il.

Dissertacao (Mestrado em Engenharia Eletrica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Linguagem de Programacao 2. Linguagem Natural 3. Linguagem de Transformacoes 4. Dissertacao I. Carvalho, Joao Marques de, Ph.D. II. Oliveira, Carlos Alberto de, Dr. III. Universidade Federal da Paraiba - Campina Grande (PB). IV. Título

CDU 004.43(043)

AMBIENTE INTELIGENTE PARA CONVERSÃO DE APLICAÇÕES
EM LINGUAGEM NATURAL PARA UMA LINGUAGEM DE TRANSFORMAÇÕES

**JOSÉ DEMISIO SIMÕES DA SILVA**

DISSERTAÇÃO APROVADA EM 26.08.1992

JOÃO MARQUES DE CARVALHO, Ph.D., UFPB
Orientador

CARLOS ALBERTO DE OLIVEIRA, Dr., INPE
Co-Orientador

MAURO CAVALCANTE PEQUENO, M.Sc., UFC
Componente da Banca

MARIA DE FATIMA QUEIROZ VIEIRA TURNELL, Ph.D., UFPB
Componente da Banca

CAMPINA GRANDE - PB
AGOSTO - 1992

AGRADECIMENTOS

Aos Doutores Carlos Alberto de Oliveira e João Marques de Carvalho pelas sugestões, paciência, atenção, incentivo, colaboração e comprometimento com a orientação deste trabalho.

Ao Doutores Guilherme Bittencourt e Valter Rodrigues pela leitura e pelas opiniões, sugestões e críticas.

Ao colega Francisco de Assis Tavares Ferreira da Silva pela paciência em suportar e promover inúmeras discussões sobre o tema, e também pelas sugestões, opiniões, críticas e leitura do texto.

À Dra. Tânia Maria Sausen e aos demais colegas do Setor de Treinamento do INPE pelo incentivo permanente e a ajuda material para a realização desta dissertação.

À minha esposa Rozinete e ao meu filho Felipe, aos quais dedico este trabalho, pelo incentivo, compreensão, dedicação, paciência, tolerância e amizade que demonstraram durante todo o processo.

S U M Á R I O

LISTA DE ILUSTRAÇÕES

# RESUMO

Este trabalho descreve o desenvolvimento de um ambiente inteligente como um meio de otimizar a interação homem-máquina em sistemas dedicados e paramétricos. Este ambiente linguístico traduz solicitações de ação passadas em Linguagem Natural em sequências de ações primitivas (procedimentos), dentro de uma abordagem de programação automática. O trabalho aborda as técnicas e ferramentas da Inteligência Artificial (IA), bem como, a identificação do conhecimento necessário para a concepção do ambiente. O desenvolvimento de um protótipo do ambiente é descrito, para realizar ações no domínio da Computação Gráfica 2D. Este protótipo é desenvolvido dentro do conceito Quadro-Negro utilizando Frames e Dependência Conceitual como formalismos de representação do conhecimento.

# 1 - INTRODUÇÃO

Se, por um lado, o uso do computador nos diferentes segmentos da sociedade já é uma realidade indiscutível, uma grande quantidade dos softwares produzidos, apesar de eficientes nas aplicações, apresenta problemas de interface para a interação homem-máquina, tornando "difícil" seu aprendizado e manipulação. Assim, um dos problemas enfrentado pelos projetistas de softwares é torná-los "mais fáceis", ou seja, prover meios para que o usuário seja capaz de utilizá-los para resolver ou ajudar na solução de seus problemas diários, quer sejam profissionais ou domésticos, tendo-os como ferramentas eficientes.

Em geral os usuários consideram "difícil" um software quando este exige parâmetros e seqüências de procedimentos específicos, ou seja, quando se deve especificar com exatidão procedimentos e parâmetros que devem ser utilizados na realização da tarefa. Isto ocorre com frequência em softwares do tipo CAD (Computer Aided Design) ou especializados, como sistema de tratamento de imagens. O fato que os torna paramétricos resulta do uso de metodologias de desenvolvimento de aplicativos, utilitários e softwares em geral, com uma abordagem direcionada somente aos procedimentos ou aos dados. Isso significa que, para a utilização adequada dos sistemas, é necessário: a) conhecer com exatidão a passagem de parâmetros envolvidos; b) conhecer detalhadamente as seqüências de procedimentos

utilizados para resolver os problemas; e, c) conhecer bem o domínio de aplicação. Isto pode significar horas de treinamento, que podem representar investimentos extras na contratação de consultores especializados e/ou nas horas gastas com o próprio treinamento, tornando a relação custo/benefício alta e, como conseqüência, implicando em um lento retorno dos investimentos.

Uma forma de resolver os problemas envolvidos na utilização de sistemas paramétricos é o uso de métodos de desenvolvimento de interfaces que facilitem a interação homem-máquina (IHM). Estes métodos devem incluem a modelagem das características e necessidades do usuário do sistema. KOBSA e WAHLSTER (1988) comentam que para que o pragmatismo seja corretamente exibido em sistemas de Inteligência Artificial, estes devem incluir um modelo do usuário contendo considerações sobre o conhecimento do mesmo, assim como, metas e planos durante a consulta ao sistema. Muitas pesquisas têm sidos realizadas, investigando como tais considerações podem ser automaticamente criadas, representadas e exploradas pelo sistema durante a interação. Neste sentido, pode-se citar como exemplo o trabalho de KASS e FININ (1988) que trata da modelagem do usuário em sistemas de Linguagem Natural.

Com a introdução do modelo do usuário no sistema, este consegue especificar sua solicitação de ação de maneira mais clara e objetiva sem um envolvimento profundo com a especificação, por vezes complexa, de todos os parâmetros requeridos na ação a ser tomada pela máquina.

Por outro lado, o projeto de interfaces para atender às necessidades mencionadas anteriormente, requer o mapeamento completo de todas as possibilidades de ação dentro do domínio considerado. Portanto, é necessário que os projetistas de tais interfaces sejam profundos conhecedores do domínio de aplicação, e, dependendo do domínio, a forma de representação de todo esse conhecimento pode se tornar complexa e dificultosa para usuários neófitos.

Desse modo, a otimização da IHM tem sido objeto de estudo de vários pesquisadores. Estes têm trabalhado no desenvolvimento de métodos que viabilizem a representação do conhecimento necessário para a realização de tarefas em um domínio de aplicação. Assim sendo, o usuário pode operá-lo, sem a necessidade de conhecer detalhadamente a especificação de parâmetros e procedimentos para a realização de ações.

Uma das formas utilizadas para tanto é o uso de menus. Como exemplo, tome-se o trabalho de THOMPSON (1984) que projetou um sistema de menus para interfacear uma base de dados na determinação de valores. Neste sistema o usuário compõe sua solicitação através de uma árvore de decisão, à medida que avança pelos menus específicos. Um impasse exige que o usuário acrescente um novo conhecimento à árvore em questão. Isto faz com que o conhecimento necessário para a execução de uma dada tarefa, pelo sistema, seja paulatinamente agregado ao conhecimento já existente, tornando o aplicativo mais maleável e fácil de entender.

Os ícones também têm sido bastante utilizados para melhorar a IHM. Cite-se como exemplo o trabalho de AMBLER e BURNETT (1989). Nele mostra-se o uso de ícones no desenvolvimento de linguagens gráficas interativas, com os quais o usuário não necessita mais escrever programas para realizar suas tarefas. Estas são pré-programadas e podem ser ativadas a partir do acionamento de mouses ou de um outro dispositivo após a escolha.

Outros pesquisadores procuraram resolver o problema com interfaces em Linguagem Natural (LN).

BATES (1984) desenvolveu o sistema IRUS - Information Retrieval Using para interfacear o gerenciador de banco de dados "dbms system 1022". O objetivo desse sistema é interpretar as questões dos usuários antes de resolver como recuperar a informação solicitada, destacando assim a relevância da intencionalidade do usuário. O IRUS é um sistema baseado em conhecimento com componentes procedurais independentes de qualquer domínio particular e estrutura de base de dados.

Os componentes básicos do sistema IRUS são: a) o módulo de Linguagem Natural (LN) e b) o módulo de tradução de base de dados. O módulo de LN é baseado no analisador RUS, que aceita comandos e questões em inglês e interpreta-os sem o uso do conhecimento de qualquer sistema particular de base de dados. A interpretação resultante é mapeada em uma Linguagem Formal de Representação do Significado ( MRL - Meaning Representation Language ). O dicionário no módulo

de LN contém muitas abreviações e o sistema não exige a correção de pontuação. No módulo de tradução a interpretação em MRL é recebida e traduzida para uma representação adequada à recuperação de dados. Em seguida esta representação é transformada em uma seqüência de interações ou comandos, na linguagem de manipulação de dados do gerenciador da base de dados (dbms).

O processamento da Linguagem Natural é suportado por algumas fontes de conhecimento: a) dicionários de palavras do domínio; b) gramática do inglês e regras semânticas para que o interpretador sintático produza estruturas interpretáveis; c) módulo de aquisição do conhecimento para que o usuário acrescente novas entradas ao dicionário e regras semânticas para estender o domínio; e, d) linguagem de representação do significado (MRL), que tem uma semântica declarativa formal, podendo ser expressa tanto em cálculo de predicados quanto em semântica procedural formal.

HOEPPNER et ali (1983) conceberam o sistema HAM-ANS - HAMburg Application oriented Natural language System que permite diálogos amplos em LN, em alemão coloquial, para produzir respostas rápidas o suficiente, tornando-o prático para aplicações em tempo real. Este sistema é operacional para um sistema especialista, um sistema de visão (WAHLSTER et ali, 1983) e um sistema de banco de dados. Na sua concepção são usados vários formalismos de representação especializados nos vários modos de fonte de conhecimento do HAM-ANS. Os formalismos centrais são as linguagens de

apresentação SURF e DEEP, orientadas pela lógica. SURF e DEEP são linguagens declarativas, equivalentes a uma lógica de primeira ordem. A entrada para a unidade de geração é uma representação SURF de um enunciado completo. Devido a possibilidade de diversas aplicações, o HAM-ANS necessita que conceitos diversos sejam expressáveis na linguagem de representação adotada. Por exemplo, "verbos de locomoção" são representados por frames de caso em um domínio específico, como por exemplo "tráfego", ao invés de estruturas de termos e predicados que se prestam para formalizar a representação para descrição de estados no domínio "hotel".

OLIVEIRA (1990) desenvolveu o sistema IDEAL - Interface Dialógica em Linguagem Natural para a língua portuguesa, como uma ferramenta para a criação de interfaces para sistemas especialistas, que usam regras de decisão e/ou redes semânticas para a representação do conhecimento, e que descrevem tal representação através de uma linguagem de descrição dessa representação. Assim o IDEAL objetiva traduzir os conhecimentos de uma frase em LN, para uma dada representação de conhecimento, dentro de uma abordagem interlíngua como em RAMAN e ALWAR (1990).

CARBONELL (1983) concebeu o projeto XCALIBUR para prover acessos flexíveis através de diálogos em LN ao sistema especialista XSEL de McDermott, para produzir ordens de venda. O sistema compõe-se dos seguintes módulos: a) módulo DYPAR II que usa uma estratégia de múltipla análise; b) manuseador de informações para mediar as

comunicações entre o analisador, o sistema especialista subjacente e o gerador de LN; c) gerador de LN que processa em três fases: a solicitação do analisador ou do manuseador de informações é convertida em um grafo de dependência conceitual, o verbo é selecionado e os "slots" do grafo são mapeados em um frame de caso que é, então, traduzido para o inglês.

## 1.1 - A Problemática

As dificuldades na IHM atual estão no fato dos sistemas conjuncionais objetivarem atingir apenas as metas conjuncionais, sem tentar detectar a "intenção" do usuário, quando de uma solicitação deste. Assim, sempre se executa (ou não) a tarefa computacional desejada, sem chances de diálogo para o usuário, caso este não domine a linguagem exigida para a interação, caracterizando um problema de IHM.

Acrescente-se o fato de que, apesar das interfaces baseadas em ícones e menus, por exemplo, apresentarem vantagens ergonométricas para os usuários, as tarefas não são sempre executadas automaticamente, do seu ponto de vista. Ou seja, durante toda a execução da tarefa-ação o usuário é solicitado a interagir com a máquina, fornecendo parâmetros e escolhendo procedimentos que devem ser adotados. Além disso, algumas tarefas podem envolver objetos sofisticados, por vezes com dimensões pequenas que

dificultam a sua execução e/ou a composição de vários procedimentos paramétricos. Cite-se como exemplo, o uso de ícones acionados por mouses ou cursores, em um sistema CAD. Neste sistema, em algumas aplicações, os objetos de pequenas dimensões e sofisticados quanto à composição, devem ser ampliados para uma escala visível antes de qualquer transformação de suas características. De forma interativa o usuário especifica o fator de ampliação desejado. Entretanto, o valor fornecido pode não oferecer uma boa visualização de detalhes, necessitando de nova especificação, que envolve mais ciclos de máquina na transformação desejada.

Um exemplo disto, pode ser uma transformação em uma planta baixa de um prédio ("uso do CAD em arquitetura"), onde se deseja alterar as dimensões de uma caixa de passagem de condutores elétricos em uma sala específica. Para redesenhar a caixa nas novas dimensões faz-se necessário tomar um conjunto de procedimentos para visualizar e transformar o objeto. Entre estes procedimentos tem-se a ampliação da parte da planta com o centro no ambiente desejado, a navegação no modelo, a especificação do ângulo de visada para vista em perspectiva, o escalonamento do objeto, etc. Mesmo com um sistema com uma IHM facilitada por ícones, por exemplo, o usuário necessita conhecer todas as funções (ou ícones) e a seqüência de acionamento das mesmas para realizar sua tarefa.

Um outro exemplo onde ícones, menus, etc, podem ser usados eficientemente, mas que o usuário centraliza as operações no fornecimento de parâmetros e da seqüência de procedimentos a ser adotada, é no Processamento Digital de Imagens. Ao tomar-se uma imagem é necessário executar um conjunto de procedimentos para torná-la agradável à interpretação visual, por exemplo. Para isso, usam-se algoritmos específicos de tratamento e de filtragem de imagens. O fornecimento dos parâmetros pode não resultar no esperado pelo usuário, quando então este deve especificá-los novamente. O processo torna-se repetitivo e envolve bastante tempo de processamento, podendo não ser significativo para o usuário.

Assim, pode-se ver nos ambientes citados por AMBLER e BURNETT (op. cit.) que o processo exposto não se mostra totalmente automático do ponto de vista do usuário, pois este pode/deve continuar a interação com a interface durante todo o processamento. Além disso, algumas operações dentro de um mesmo domínio podem se tornar complicadas do ponto de vista do usuário, exigindo desse um conhecimento mais específico do domínio. Em situações desse tipo, a IHM não ocorre num nível de abstração alto, por que o usuário necessita considerar os detalhes na especificação precisa de parâmetros durante a execução das tarefas.

Se o objetivo da IHM é tornar a passagem de parâmetros e a utilização de ferramentas o mais transparente possível para os usuários, para que um número maior de pessoas possa usufruir das facilidades

introduzidas pelos computadores, uma vez que não há necessidade de cursos longos e cansativos para a utilização de softwares, a IHM deve ocorrer no mais alto nível de abstração possível.

Pode-se, então, pensar no uso da Linguagem Natural (LN) no desenvolvimento de interfaces que facilitem a IHM. Um ambiente em LN numa abordagem cognitiva, ou seja, baseado em conhecimento, permite que o usuário participe do processo de execução da tarefa-meta, explicitando interativamente sua "intenção" (OLIVEIRA, 1988). Usando sua linguagem de comunicação cotidiana, o usuário apresenta maior segurança no uso do computador, pois sente-se a vontade e menos receoso de errar, uma vez que não precisa dominar toda uma linguagem paramétrica antes de resolver algum problema.

Como exemplo do uso da LN na IHM, pode-se citar SCHMITT (1989), que desenvolveu um sistema para gerar seqüências de transformações morfológicas encadeadas, formando verdadeiros programas morfológicos. Assim qualquer usuário familiarizado com o domínio de aplicação (no caso, a Morfologia Matemática) pode executar suas tarefas. Sem este ambiente o usuário precisa conhecer uma linguagem paramétrica do domínio, para poder executar as transformações desejadas. No entanto, este sistema foi desenvolvido para o domínio (restrito) da Morfologia Matemática com interação em Língua Inglesa, ficando quase que inválido para usuários de outras línguas, visto que seria tomado como apenas mais uma linguagem a ser

aprendida. Assim, por força de uma "inércia cultural", a maioria das interfaces em LN são baseadas em modelos lingüísticos não pertinentes à LPB.

Outro dos problemas que surgem com o uso da LN está relacionado à representação do conhecimento. Os trabalhos em IA em um domínio, requerem um conhecimento substancial sobre o mesmo. A representação efetiva deste conhecimento é então a chave para o sucesso dos programas em IA (FIKES e KEHLER, 1985). O conhecimento necessário envolve várias formas, incluindo definições descritivas de termos específicos do domínio, descrições de objetos individuais do domínio e suas relações e os critérios para tomada de decisão. HOFFMAN (1987) (página 53) enfoca o problema de adquirir o conhecimento necessário para sistemas especialistas. No seu trabalho ele fala de métodos de extração do conhecimento e habilidades dos especialistas no domínio, ou seja, formas de caracterizar o conhecimento de um especialista. Um outro aspecto que deve ser considerado é a escolha das ferramentas e técnicas para representar o conhecimento de maneira efetiva. Muitas técnicas e ferramentas têm sido propostas, cada uma apresentando vantagens e desvantagens quanto ao uso e implementação. Entretanto, algumas apresentam-se mais adequadas para determinados domínios. Neste caso, enquadra-se o uso da representação por frames para o domínio em consideração.

## 1.2 - A Proposta Deste Trabalho

Este trabalho propõe o uso da LN numa abordagem cognitiva, ou seja, baseado em conhecimento para a concepção de um ambiente inteligente que permita uma IHM eficiente. O ambiente baseado em conhecimento, traduzirá uma solicitação de ação do usuário em LN para a ação ou conjunto de ações mais adequado para execução da tarefa desejada. Procurar-se-á detectar a "intenção" do usuário (o desejo do usuário), através de interações com o mesmo por um sistema de menus, ícones ou perguntas em LN, permitindo o mapeamento automático da solicitação na ação ou conjunto de ações a ser tomado. O ambiente é concebido para interações usando o sistema lingüístico da Língua Portuguesa do Brasil (LPB), onde as solicitações do usuário são feitas em linguagem natural livre dentro de um domínio de aplicação restrito.

O uso da LN na concepção de ambientes inteligentes mantém a IHM em um alto nível de abstração, permitindo que os usuários dediquem-se a tarefas mais nobres, no sentido de que eles não precisam compreender a necessidade de parâmetros, do ponto de vista do software, nem a seqüência de procedimentos que deve ser adotada para realizar a ação. Assim, o tempo antes utilizado para conhecer a funcionalidade do software, agora pode ser usado para conceber metodologias que prevêem o uso eficiente do software em aplicações reais.

O conhecimento resultante das interações com os usuários durante o processo de análise e interpretação, deverá ser "apreendido" pelo ambiente, estabelecendo um processo dinâmico de aquisição do conhecimento (OLIVEIRA, 1988) e aprendizado de máquina. Isto devido às dificuldades para um usuário especialista ou não, memorizar os limites do domínio de aplicação todo o tempo (WEXELBLAT, 1989). Além disso, quando estes limites mudam, as relações que dão significado a uma palavra são alteradas (OLIVEIRA, op. cit.).

Essa concepção tornará o ambiente capaz de mapear um conhecimento dentro do domínio para uma linguagem de transformações: Linguagem de Transformações Geométricas ou uma Linguagem de Processamento de Imagens - como em BATCHELOR (1986) e SCHOWENGERDT e WANG (1989), ou, ainda, uma Linguagem de Processamento de Sinais - para síntese e/ou projeto de filtros digitais (NIE et ali, 1991) ou (CARVALHO et ali, 1991), por exemplo, dentro de uma abordagem interlíngua (RAMAN e ALWAR, 1990).

O mapeamento significa a especificação de procedimentos que devem ser tomados para a execução da tarefa desejada. Os parâmetros ainda serão fornecidos pelo usuário, só que o sistema baseado no conhecimento armazenado, os requisitará de forma "amigável", por vezes explicitando a necessidade dos mesmos, ou seja, em um nível de abstração alto. Isso, caracteriza um processo de treinamento do usuário, através do próprio ambiente, onde qualquer falha de especificação pode ser detectada antes da

execução da tarefa. Essa característica torna os ambientes inteligentes para IHM atrativos para os usuários sem experiências em computação e no domínio de aplicação, podendo inclusive ser parte de um sistema de treinamento de especialistas em um determinado domínio, caracterizando-se como um ambiente propício para tarefas educacionais, onde o usuário sem experiência navega segundo as necessidades do domínio de acordo com as diretrizes propostas pelo ambiente. Assim, o ambiente poderá ser usado em diversas aplicações que requeiram o intercâmbio de informações envolvendo a especificação dos parâmetros e dos procedimentos. Um exemplo seria o Processamento de Imagens onde o usuário poderia interagir com o ambiente através de diálogos como a seguir:

**US > Suavizar a imagem.**
**AMBIENTE > Especifique a imagem a ser filtrada.**
**US > <Fornece o nome da imagem>**
**AMBIENTE : O filtro passa-baixa deve preservar a borda?**
**US > <O usuário (neófito) pode responder de acordo com a pergunta do sistema ou pode indagar o que significa preservar a borda, ao que o sistema pode atender explicando os efeitos da filtragem passa-baixa sobre a imagem e da preservação da borda (inclusive, ou preferencialmente, através de exemplos)>.**

O conhecimento do domínio considerado será representado utilizando-se a estrutura de frames discutido na seção 2.2, dentro de uma abordagem de Quadro-Negro discutido na seção 2.4. O domínio considerado será o da Computação Gráfica 2D. Assim, o conhecimento necessário estará restrito à representação de figuras em 2D e à formulação de regras que permitam executar tarefas neste domínio. Estas regras serão consideradas como ações

primitivas, as quais poderão compor seqüências de procedimentos para realizar várias tarefas do domínio. A linguagem de frames definida para a manipulação do conhecimento será a base para as primitivas comentadas.

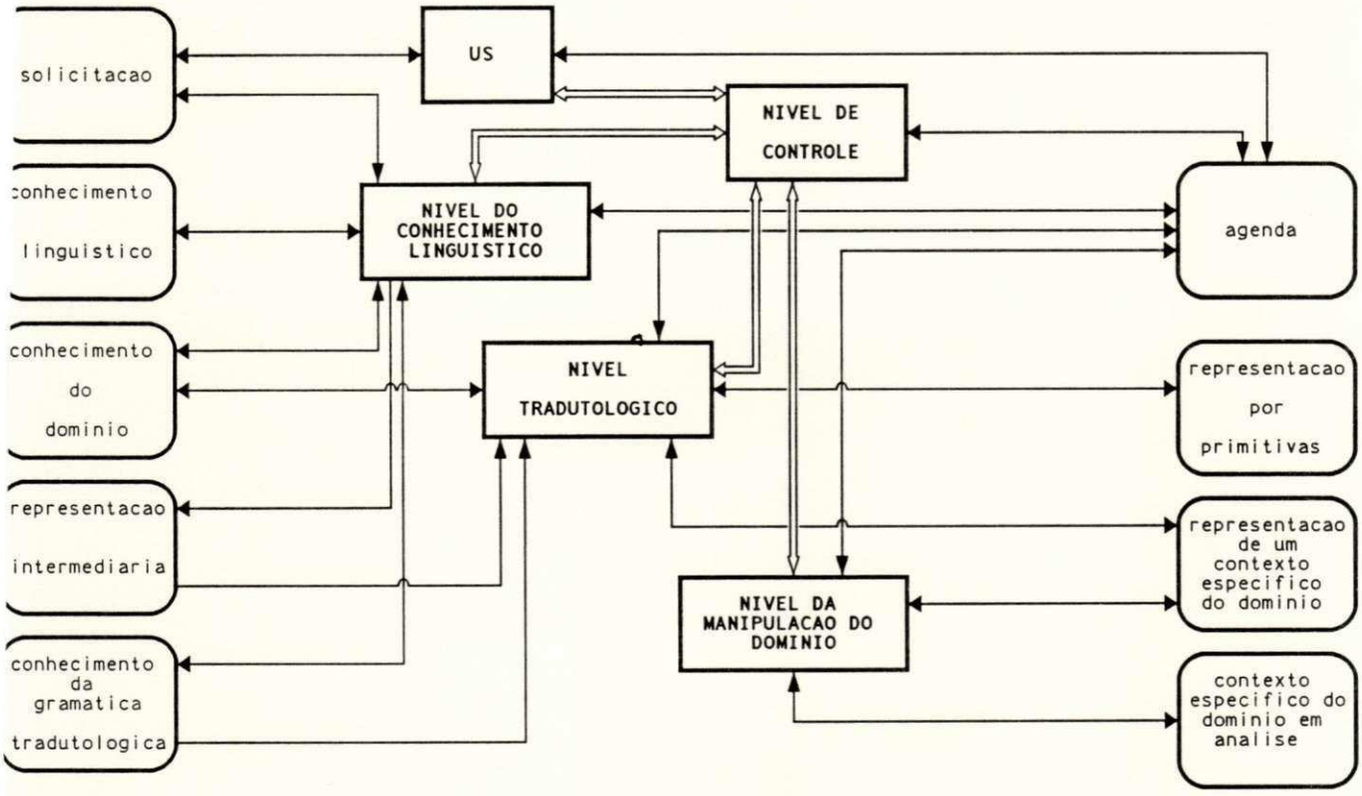



**Figura 1.1 - Visão Geral do Ambiente Proposto**

A figura 1.1 apresenta uma visão geral do ambiente proposto, mostrando que conhecimento pode estar envolvido no desenvolvimento do ambiente. Observe-se que, para melhor

explicitar os diversos estágios pelos quais a solicitação de ação do usuário passa, o conhecimento é classificado em níveis. Assim, tem-se o nível do tratamento lingüístico, o nível tradutológico, o nível de controle, o nível do usuário e o nível do software interfaceado. Cada nível está relacionado ao conjunto de regras necessário para desenvolver uma tarefa específica (fontes de conhecimento ativas). As bases de dados armazenam as premissas necessárias para as inferências em cada nível de conhecimento e podem ser compartilhadas por níveis distintos, dependendo da necessidade de dados de cada um. Como cada nível pode ser ativado independente um do outro, pode existir uma certa concorrência no uso de algumas bases de dados, caracterizando-se portanto numa arquitetura de Quadro-Negro. Assim, cada nível é ativado dependendo das estratégias de ações agendadas no mecanismo de agenda e de controle implementados, para que as tarefas sejam adequadamente realizadas, uma vez que algum nível pode não ser ativado até que um outro nível tenha processado os dados necessários. Isto pode levar a se pensar em processamentos seqüenciais, eliminando portanto o paralelismo existente no modelo Quadro-Negro. Entretanto, isto ocorre do ponto de vista da realização da tarefa-meta do usuário, mas não invalida o modelo Quadro-Negro porque é função de cada nível saber as condições adequadas à realização eficiente de sua tarefa. Sendo assim, os processamentos seqüenciais mencionados seriam resultantes da necessidade de cada nível de conhecimento de informações processadas.

A arquitetura da figura 1.1 mostra-se independente do domínio de aplicação. Isto significa que a mesma filosofia de desenvolvimento pode ser empregada para outros domínios. O processamento lingüístico é o núcleo do ambiente. Neste trabalho considera-se a potencialidade do sistema IDEAL, discutido anteriormente, que é próprio para interfacear sistemas especialistas.

## 1.3 - Considerações Finais

A LN incorpora características essenciais à interação homem-máquina, tornando-a útil para sistemas paramétricos. Entretanto, ambientes em LN ainda não são totalmente atrativos para os projetistas de software, por que muitos dos problemas inerentes à LN ainda não foram totalmente resolvidos, tornando seu uso difícil para o desenvolvimento de sistemas. Entre os problemas citados está a necessidade de representação de todo o conhecimento lingüístico necessário às interações. Em se tratando de uma linguagem de comunicação, a LN é dinâmica, no sentido de que a cada instante novas expressões e regras são concebidas, principalmente em grandes extensões territoriais, onde a LN se adapta aos costumes e as culturas locais, dando uma característica livre a LN.

A característica livre da LN implica na necessidade de armazenamento dinâmico de informações. Assim, o processamento pode tornar-se suficientemente grande e

complexo na medida que a LN permite que uma mesma informação seja comunicada de formas diferentes, comprometendo o rendimento das máquinas no tocante a performance das regras estabelecidas. Além disso, a representação do conhecimento de um domínio específico pode ter alto grau de complexidade, fazendo com que o projetista procure ser um profundo conhecedor do domínio ou interaja com engenheiros do conhecimento, especialistas no domínio, para decidir o conhecimento que deve ser representado, ou seja, estabelecer os limites do domínio de aplicação. Um outro problema importante é a escolha das ferramentas e técnicas de representação que devem ser utilizadas. Todas oferecem vantagens e desvantagens. Entretanto, algumas apresentam-se mais adequadas para certos domínios de aplicação.

O ceticismo existente em relação a LN residiu durante muito tempo, na necessidade de capacidade de memória para o armazenamento da representação do conhecimento, quando se utilizava máquinas limitadas. Entretanto, a tecnologia digital atual permite a produção de máquinas com capacidade de armazenamento alta, o que já permite considerar o uso da LN para facilitar a IHM.

A IHM com LN, pode ser melhorada utilizando-se diferentes recursos para facilitar o intercâmbio entre o usuário e a máquina. Dentre estas cita-se os ícones, mouses, etc.

Apesar das desvantagens resultantes do baixo desempenho computacional de ambientes com LN, estes ainda representam aspectos positivos, na medida que os usuários não precisam ser treinados exaustivamente para usar um determinado sistema. A parametrização é fornecida interativamente livrando o usuário da necessidade de memorizar a sintaxe e os domínios dos parâmetros formais necessários.

Objetiva-se, pois, construir um protótipo para o domínio específico da Computação Gráfica 2D, como demostração da factibilidade e de alternativa de solução para os problemas comentados anteriormente. Desta forma, a estrutura do trabalho compreende o descrito a seguir.

O capítulo 2 discorre sobre a teoria da dependência conceitual na seção 2.1, a teoria da representação por frames na seção 2.2 e sobre o modelo de Quadro-Negro na seção 2.3.

No capítulo 3, apresenta-se o conhecimento envolvido num ambiente como o proposto no trabalho, adaptado para o domínio da Computação Gráfica 2D, considerando o sistema IDEAL como núcleo para o tratamento lingüístico. A seção 3.1 comenta sobre os conhecimentos lingüístico e pragmático necessários para o tratamento das solicitações do usuário. Nela se discorre sobre o uso da dependência conceitual para representar as propostas de significado das solicitações. A seção 3.2 discute o conhecimento tradutológico necessário para executar a tarefa de tradução para seqüências de

primitivas, e apresenta as quatro primitivas utilizadas no domínio mencionado anteriormente. Concluindo o capítulo 3, a seção 3.3 apresenta o conhecimento utilizado do domínio e fala sobre sua representação através de estruturas de frames.

O capítulo 4 apresenta a descrição do desenvolvimento de um protótipo do ambiente. A seção 4.1 descreve a arquitetura concebida para o ambiente e as seções subseqüentes mostram cada módulo que compõe o protótipo, fazendo considerações sobre o que foi implementado para cada um.

O capítulo 5 fala das conclusões sobre a implementação conseguida, faz considerações sobre os problemas encontrados durante a implementação, e mostra as perspectivas de aperfeiçoamento e desenvolvimento do ambiente para máquinas maiores, utilizando outras linguagens de programação.

# 2 - FUNDAMENTOS TEÓRICOS

Esta seção trata dos fundamentos teóricos necessários para a concepção do ambiente em proposição. No apêndice A têm-se exemplos práticos da fundamentação teórica apresentada.

## 2.1 - A Representação de Conhecimento por Primitivas

A teoria da Dependência Conceitual (DC) (SCHANK 1975) é uma forma de representação do conhecimento do significado interlíngua. Esta representação é independente da linguagem e tem como princípio básico que: "duas sentenças com significado idêntico devem ter uma única representação". Para se utilizar a DC, é necessário quebrar as sentenças nos elementos constituintes e, em seguida estabelecer uma sintaxe das relações conceituais possíveis, e um conjunto de categorias conceituais que as relacionem. No seu modelo, SCHANK (op. cit.) estabeleceu as categorias conceituais, tais como vistas na figura 2.1.

**PP - Objetos do mundo real.**
**ACT - Ações do mundo real.**
**PA - Atributos dos objetos.**
**AA - Atributos de ações.**
**T - Tempo**
**LOC - Localizações.**

**Figura 2.1 - Categorias Conceituais de Schank.**

Alem das categorias citadas na figura 2.1, SCHANK (op. cit.) estabeleceu 16 regras de sintaxe conceitual, que as relacionassem de formas específicas, e 12 ações primitivas, nas quais qualquer verbo de uma construção conceitual pode ser dividido, com preservação da informação. A figura 2.2 ilustra algumas das 16 regras concebidas por SCHANK (op. cit).

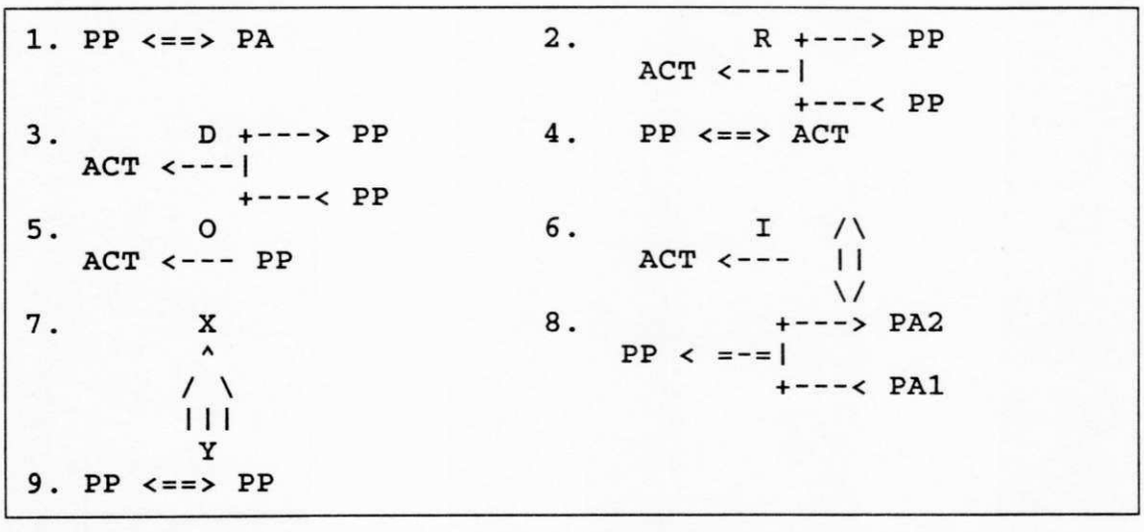



**Figura 2.2 - Algumas Regras da Sintaxe Conceitual de Schank**

A interpretação das regras na figura 2.2 é feita da seguinte forma (RICH, 1988):

| | |
|---|---|
| **Regra 1** | - Descreve as relações entre um PP e o PA que está sendo afirmado como descrevendo-o, ou seja, objeto (PP) possui atributo (PA). |
| **Regra 2** | - Descreve a relação entre uma ACT (ação) e a fonte recipiente da ACT, ou seja, o recipiente /dono do objeto dentro da ação. |
| **Regra 3** | - Descreve a relação entre uma ACT (ação) e sua origem e destino físico, ou seja, a direção do objeto dentro da ação. |
| **Regra 4** | - Descreve as relações entre um ator e o evento que ele causa, ou seja, o ator age. |
| **Regra 5** | - Descreve a relação entre uma ACT e o PP que é o objeto da ACT, ou seja, PP é objeto de uma ação. |
| **Regra 6** | - Descreve a relação entre uma ACT e o instrumento com o qual é realizada, ou seja, conceituação instrumental para uma ação. |
| **Regra 7** | - Descreve a relação entre uma conceituação e outra que a causou, ou seja, conceituação X causou a conceituação Y. |
| **Regra 8** | - Representa a relação entre um PP e um estado em que ele começou e outro em que ele terminou, ou seja, o estado de mudança de um objeto. |
| **Regra 9** | - Descreve a relação entre dois PP, um dos quais pertence a um conjunto definido pelo outro. |

A seguir, comentam-se as funções das primitivas estabelecidas por SCHANK (op. cit.):

| | |
|---|---|
| **ATRANS** | - Transferência de uma relação abstrata (Por exemplo, dar) |
| **PTRANS** | - Transferência de localização física de um objeto (Por exemplo, ir) |
| **PROPEL** | - Aplicação de força física a um objeto (Por exemplo, empurrar) |
| **MOVE** | - Movimento de uma parte do corpo de alguém (Por exemplo, chutar) |

| | |
|---|---|
| **GRASP** | - Segurar um objeto por um ator (Por exemplo, jogar) |
| **INGEST** | - Ingestão de um objeto (Por exemplo, comer) |
| **EXPEL** | - Expulsão de algo do corpo de um animal (Por exemplo, chorar) |
| **MTRANS** | - Transferência de informação mental (Por exemplo, dizer) |
| **MBUILD** | - Construir nova informação da antiga (Por exemplo, decidir) |
| **SPEAK** | - Produção de sons (Por exemplo, dizer) |
| **ATTEND** | - Enfocar o órgão de sentido em um estímulo (Por exemplo, ouvir) |

No trabalho de SCHANK e ABELSON (1977) a representação do significado de sentenças baseia-se no princípio de que duas sentenças com significado idêntico devem ter uma única representação, independentemente da linguagem. Assim, qualquer informação na sentença que está implícita, deve ser explicitada na representação do significado desta sentença. Com este princípio SCHANK e ABELSON (op. cit.) conceberam um **esquema** inicial para compactar a representação do significado. Neste esquema, as propostas de significado de uma expressão sob a linguagem são chamadas **CONCEITUAÇÕES**, conforme expresso a seguir:

| **CONCEITUAÇÃO** | **FORMA** |
|---|---|
| ATIVA<br>ESTÁTICA<br>valor ) | AGENTE(ATOR) AÇÃO OBJETO DIREÇÃO<br>OBJETO (está em) ESTADO ( com |

As conceituações ativas são analisadas como um AGENTE fazendo algo a um OBJETO numa DIREÇÃO, e as

conceituações estáticas como uma declaração do ESTADO do OBJETO (SCHANK, 1975).

A vantagem na utilização dessa representação está em que ela restringe as categorias de conceitos que podem ser considerados, dependendo do domínio em análise, e permite que as propostas de significado sejam mapeadas através de ações primitivas.

## 2.2 - A Representação de Conhecimento por Frames

Segundo FIKES e KEHLER (1985) o sucesso de um sistema especialista está na forma como o conhecimento detalhado do domínio é efetivamente representado.

Para WINSTON (1979) uma representação é definida como um conjunto de convenções para descrever objetos, situações, ações, etc, de um domínio, cujo projeto, quando bem elaborado, é a chave para tornar simples os problemas complicados.

Para representar então o conhecimento do domínio, é necessário primeiro decidir que conhecimento deve ser utilizado e escolher uma forma de representação adequada, que permita a recuperação (lembrança) do que foi apreendido de forma eficiente.

A teoria de "frames" concebida por Minsky (WINSTON, 1975) é uma forma de representação simbólica, que provém da hipótese de que o homem não analisa novas situações a partir do zero e forma novas estruturas de conhecimento para descrevê-las. Ao invés disso, ele dispõe de um conjunto extenso de estruturas, representando suas experiências anteriores com objetos, situações e pessoas. Assim, na análise de uma nova experiência ele tenta selecionar na memória uma dessas estruturas previamente armazenada ("um frame") (RICH, 1988). Esta estrutura é relembrada e adaptada à realidade, alterando-se os detalhes de acordo com a necessidade do evento. Portanto um frame é uma estrutura de dados usada para representar situações estereotipadas, como por exemplo ir a uma festa de aniversário infantil, ou objetos do mundo real, como por exemplo uma cadeira ou uma sala. Vários tipos de informações podem ser reunidas em uma estrutura de frames, sendo armazenadas em seus terminais (slots, atributos). As informações referem-se às experiências e situações anteriores, que criam expectativas do que possa acontecer a seguir e às orientações sobre o que fazer se as expectativas não forem confirmadas. A figura 2.3 abaixo mostra um exemplo de um frame para um estádio de futebol (PASSOS, 1989 e ARARIBÓIA, 1989).

```
Frame Estádio_de_Futebol
      Nome: <valor> = sequência_de_caracteres
      Proprietário: <valor> = particular_ou_estado
      Capacidade: <valor> = um_inteiro
      Dimensões: <default> = oficiais
```

**Figura 2.3 - Exemplo de Frame Estádio_de_Futebol genérico.**

Observe-se na figura 2.3 que alguns slots têm rótulos denominados <valor> e outros <default>. O rótulo <valor> significa uma atribuição confiável, enquanto o rótulo <default> significa uma atribuição baseada em generalizações e estereótipos. Um exemplo de um frame que descreve um estádio de futebol específico, está na figura 2.4.

```
Frame Estádio Maracanã
      Nome: <valor> = Mário Filho
      Proprietário: <valor> = Estado
      Capacidade: <valor> = 200.000 pessoas
      Dimensões: <valor> = oficiais
```

**Figura 2.4 - Exemplo de Frame Estádio_de_Futebol específico.**

Observe-se na figura 2.4, que no slot denominado Dimensões, o rótulo é <valor>. Isso acontece porque as dimensões do estádio Maracanã são oficiais. No caso de tratar-se de um outro estádio do qual não se tenha informações suficientes, o rótulo atribuído seria <default>. Os exemplos mostram o que usar baseado em situações anteriores e como proceder se não houver correspondência com a realidade.

Minsky (WINSTON, 1975) considera uma estrutura de frames como uma rede de nós e relações. Nesta rede os níveis mais altos (topo) são fixos e usados para representar tudo que é sempre verdadeiro sobre uma situação suposta. Os níveis mais baixos têm vários atributos (slots) que são preenchidos por instâncias específicas ou dados.

Cada slot pode especificar condições que seus valores devem satisfazer. Os próprios valores são geralmente sub-frames menores. As condições simples são especificadas por marcadores que devem requerer que uma atribuição terminal seja uma pessoa, um objeto de valor suficiente, ou um ponteiro para o sub-frame de um determinado tipo. Condições mais complexas podem especificar relações entre o que foi atribuído à vários atributos (slots).

Os conjuntos de frames relacionados são ligados uns aos outros formando sistemas de frames. Os efeitos das ações importantes são espelhadas nas transformações entre os frames de um sistema. Estes são usados para facilitar alguns tipos de cálculos, representar mudanças de ênfase e atenção, e para explicar as propriedades de imagens (figuras).

Na análise de cenas visuais, os frames diferentes de um sistema descrevem a cena de pontos de vista diferentes, e as transformações entre um frame e outro representam os efeitos de mover de um lugar para outro. Nos frames do tipo não visual, as diferenças entre frames de um sistema podem representar ações, relações causa-efeito, ou mudanças do ponto de vista metafórico.

Os diferentes frames de um sistema podem compartilhar os mesmos terminais (slots). Este é o ponto crítico que permite a coordenação da informação reunida a partir de pontos de vista diferentes.

A força fenomenológica da teoria de frames, depende da inclusão de expectativas e outras suposições. Os atributos (slots) de um frame são normalmente preenchidos com atribuições <default>. Assim, um frame pode conter muitos detalhes cuja suposição não é garantida pela situação, mas que são usados na representação de informações gerais, de casos parecidos, de técnicas para relevar a lógica e maneiras de fazer generalizações úteis.

As atribuições <default> são feitas livremente aos slots, de modo que possam ser facilmente substituídas por novos itens que melhor especifiquem a situação em questão. Elas indicam que a informação tem origem em generalizações e estereótipos, e só devem ser usadas no caso de não se ter algo mais concreto. Assim, elas podem servir como "variáveis" ou "casos especiais" para "raciocinar por exemplos", ou como "casos de livros texto", e tornam desnecessário o uso de quantificadores lógicos.

Os sistemas de frames são ligados por uma rede de recuperação de informação. Quando um frame proposto não se adapta à realidade, ou seja, quando não é possível encontrar atribuições terminais que correspondem adequadamente às condições do marcador terminal, esta rede provê um frame substituto. Estas estruturas interframes tornam possível outras formas de representação do conhecimento sobre fatos, analogias e outras informações úteis ao entendimento.

Uma vez proposto um frame para representar uma situação, um processo de correspondência (matching) tenta atribuir valores aos terminais de cada frame, consistentes com os marcadores em cada local. O processo de correspondência é parcialmente controlado pela informação associada com o frame, que inclui informação de como tratar as surpresas, e pelo conhecimento sobre as metas do sistema em questão.

Em um sistema baseado em frames, estes são organizados em uma hierarquia de generalizações onde cada um herda informações do que está acima dele na hierarquia. O superior hierárquico pode ser encontrado através de um elo do tipo "ako" (A KIND OF - uma espécie de) como na figura 2.5.

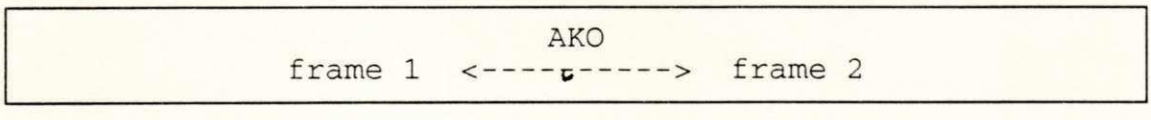



**Figura 2.5 - Relação AKO inter-frames.**

A teoria de frames foi proposta por Minsky na tentativa de resolver problemas simples de visão relacionados a psicologia Gestalt. No seu artigo ele considera a visão como um processo extenso, onde leva-se tempo para preencher detalhes, coletar evidências, fazer conjecturas, testar, deduzir e interpretar uma cena, que parece ser imediatamente interpretada ao se ter o primeiro contato, mas que depende do conhecimento, das expectativas e dos objetivos de quem está sendo submetido à experiência. Esta teoria vai ao encontro das idéias dos teóricos da

psicologia Gestalt, que tentam explicar vários fenômenos de visão em termos das propriedades globais de campos elétricos no cérebro.

Para exemplificar Minsky desenvolveu um sistema de frames simplificado para representar um cubo em perspectiva. A figura 2.6 mostra o frame para um determinado ângulo de visada.

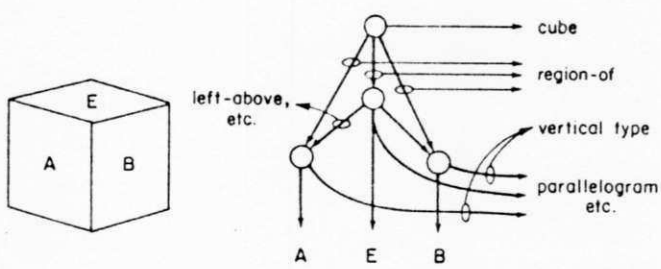



**Figura 2.6- Um frame representando a visão de um cubo. (Fonte: WINSTON, 1975, página 217)**

Na figura 2.6, o primeiro nível indica que o frame corresponde a visão de um cubo através do elo "Ê_UM". No nível seguinte as faces visíveis são representadas por cada nó. Os elos "TEM_DE_SER" descrevem uma restrição sobre os valores que podem preencher os slots de cada nó. Os elos dos nós de faces para os nós específicos A, E e B, indicam os valores para os slots de faces relativas a esta vista em particular.

A figura 2.7 mostra um sistema de frames representando a perspectiva de um cubo, considerando-o sob pontos de vista diferentes.

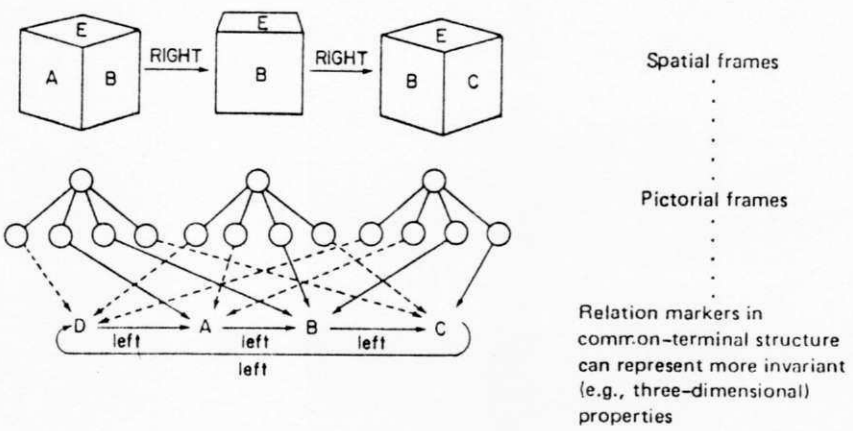



**Figura 2.7 - Esquema de um Sistema de Frames. (Fonte: WINSTON, 1975, página 219)**

Na figura 2.7, as linhas sólidas dos slots que representam faces para as faces particulares e que preenchem seus slots, indicam lados visíveis. As linhas pontilhadas indicam faces que são invisíveis daquele ponto de vista. Os elos entre os frames descrevem a relação entre os pontos de vista que os frames representam. Além dos nós explícitos, os próprios frames são ligados através de nós compartilhados que preenchem seus slots. Esses nós compartilhados, cada um sendo uma face do cubo, representam

descrições, vistas independentemente das faces. Neste exemplo, as faces são apresentadas como nós simples. Mas se contivessem padrões complexos, tais padrões seriam representados por estruturas mais complexas, possivelmente outro conjunto de frames. Como cada slot de um frame pode ser preenchido por outro frame e como um dado frame pode preencher mais de um slot, não precisa haver armazenamento redundante da informação que for comum a muitos pontos de vista.

Os atributos de um frame não se restringem apenas a valores. Também é possível atribuir-se regras para executar alguma tarefa específica do que está se representando. Estas podem ser declarativas ou procedurais, significando que um frame pode ativar uma regra para realizar alguma atividade.

O esquema do sistema de frames pode ajudar a explicar muitos fenômenos da inteligência humana. Ele é apropriado na interpretação de uma seqüência específica observada e é útil para prever a ocorrência de certos acontecimentos que não tenham sido mencionados anteriormente. Em particular, os sistemas de frames são úteis para domínios onde a forma e o conteúdo do dado desempenham um papel importante na solução do problema, tais como, interpretação visual de cenas ou entendimento da fala (PASSOS, 1989).

## 2.3.- A Linguagem de Frames

A estrutura de frames comentada em 2.2, é um formalismo de representação do conhecimento, ou seja, em última análise, uma estrutura de armazenamento de dados. Faz-se necessário, então, definir-se uma forma de efetivamente implementá-lo. Vários recursos podem ser utilizados para uma boa implementação. Neste trabalho, por exemplo, em alguns momentos implementa-se a estrutura de frames utilizando-se declarações PROLOG, tais como:

```
recordz(sinônimo,[mudar,mover],_____).
recordz(atributo,[cor : _______],_____).
```

Estas declarações estão organizadas em seqüência em um arquivo de dados denominado LING (dicionário lingüístico). Entretanto, a maneira mais eficaz para se implementar a estrutura de frames é ter-se uma linguagem de frames disponível. Muitos pesquisadores têm trabalhado em sistemas que objetivam facilitar o uso de modelos de representação do conhecimento através de linguagens, por exemplo, os trabalhos de BITTENCOURT e MARENGONI (1991) e de ARARIBÓIA (1989). Estas linguagens são implementadas para permitir a navegação em todo o conhecimento representado. BITTENCOURT e MARENGONI (op. cit.) desenvolveram uma linguagem para ambientes codificados em COMMON LISP, onde usa-se comandos para implementar o modelo de frames e um motor de inferência para a interpretação de regras. A linguagem em si é composta de primitivas para inicializar o sistema de frames, criar novos frames,

atribuir valores ou funções aos atributos dos frames, eliminar atributos de frames, recuperar informação dos frames, e visualizar mensagens para o usuário. A linguagem desenvolvida por ARARIBÓIA é implementada em PROLOG, portanto utilizando-se do motor de inferência da própria linguagem. Esta linguagem é compacta na medida que apenas se implementa regras para manipulação do conhecimento (armazenamento, recuperação, associação, etc, do conhecimento do domínio envolvido). Quatro comandos compõem a linguagem que implementa a estrutura de frames (ARARIBÓIA, 1989): fponha, fremova, ligue, fpegue. Estes quatro comandos são suficientes para navegar em toda a estrutura de frames e manipular o conhecimento representado. Suas funções específicas são como a seguir:

| |
|---|
| **fponha** - coloca uma Regra no Frame. |
| **fremova** - remove uma Regra do Frame. |
| **ligue** - conecta dois frames por um elo do tipo AKO. |
| **fpegue** - recupera o conteúdo de um slot de um frame. |

O exemplo a seguir explicita o uso dos comandos.

**Exemplo:**

| |
|---|
| **Armazenamento:**<br>**fponha(aposento(dormir,L,C),valor,mobília(cama)).**<br>**ligue(quarto(L,C),aposento(dormir,L,C)).** |
| **Recuperação:**<br>**fpegue(quarto(5,6),mobilia(M)).**<br>**M = cama ->.**<br>**yes** |

No exemplo, durante a fase de armazenamento criou-se um frame denominado **aposento** com características próprias, ou seja, função (dormir) e dimensões (C - comprimento, L - largura). **mobília(cama)** é atribuído como um valor ao frame, pois trata-se de uma premissa conhecida. Da maneira como foi declarada, mobília é uma regra (procedimento) para se inferir que tipo de mobília um **aposento** de **dormir** pode ter. Esta declaração é uma particularização de um frame chamado **aposento**. No comando seguinte (**ligue**), conecta-se uma instância de quarto, com dimensões conhecidas (L,C), com a particularização de aposento de dormir, através de um elo do tipo AKO. Na fase de recuperação (**fpegue**), procura-se saber que tipo de mobília pode existir em um quarto com dimensões L=5 e C=6. Observe-se que, a resposta é **cama**. Isto ocorre porque **quarto(5,6)** herdou todas as características de **quarto(L,C)**, que herdou de **aposento(dormir,L,C)**, devido ao elo existente entre eles do tipo AKO. Outras características do quarto(5,6) poderiam ser obtidas, se as regras correspondentes tivessem sido representadas. Por exemplo, seria possível computar o cálculo da área do quarto, através de uma regra declarada para esse fim.

## 2.4 - A arquitetura Quadro-Negro

A resolução de problemas em Inteligência Artificial (IA), exige procedimentos flexíveis. Assim, pode-se pensar em um modelo de resolução de problemas como um esquema de

organização do raciocínio e do conhecimento do domínio, para construção de uma solução para um problema.

O modelo **Quadro-Negro** (**QN**) é uma abordagem que possui algumas características necessárias à resolução de problemas. Essas características incluem flexibilidade e generalidade (NAKAMITI, 1991). Neste modelo o conhecimento é segmentado em diversos módulos, e a cada um associa-se uma máquina de inferência. Os módulos se comunicam através de um ambiente comum denominado **Quadro-Negro**. Portanto, o modelo QN é uma entidade conceitual e não uma especificação computacional, que provê funcionalidades suficientes para esboçar a solução de um determinado problema (NAKAMITI, op. cit.).

As fontes de conhecimento contribuem cooperativamente para a solução de um problema (OLIVEIRA, 1990). Todas as possibilidades de soluções parciais ou totais formam o espaço de soluções. Este espaço é organizado em uma ou mais hierarquias dependentes da aplicação (NII, 1986). As fontes de conhecimento são logicamente independentes e autoativadas, o que implica um forte paralelismo, e são as únicas responsáveis pelas alterações do Quadro-Negro (NII, op. cit.). As aplicações do modelo QN são implementadas com diferentes combinações de representações de conhecimento, diferentes esquemas de raciocínio e diferentes mecanismos de controle, e, além disso, os ambientes computacionais são seriais, em sua maioria. Logo, criou-se uma **arquitetura Quadro-Negro** para prover os recursos necessários para o projeto de sistemas

Quadro-Negro. Esta arquitetura contém descrições dos componentes básicos do sistema QN, que são partes de um sistema computacional real baseado no modelo (NII, op. cit.), e trata a solução do problema como um processo oportunístico de montagem de uma configuração satisfatória dos elementos da solução (HAYES-ROTH, 1985).

O modelo Quadro-Negro consiste basicamente de três componentes principais: **Fontes de Conhecimento**, **Estrutura de Dados Quadro-Negro**, e **Módulo de Controle**. A figura 2.8 ilustra um exemplo da arquitetura Quadro-Negro.

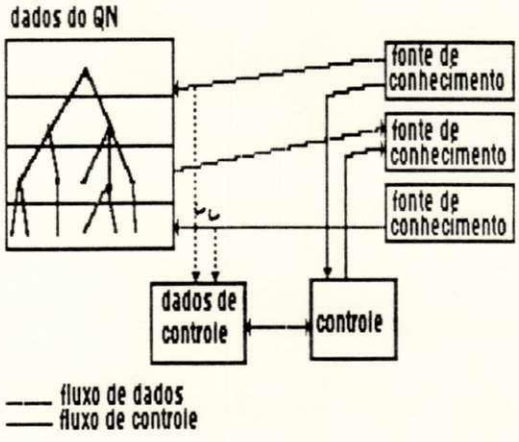



**Figura 2.8 - Arquitetura do Quadro-Negro. (Fonte: NII, 1986, página 44)**

A seguir, descreve-se cada componente da arquitetura.

### 2.4.1 - As Fontes de Conhecimento

O conhecimento do domínio necessário para a solução do problema é segmentado em **Fontes de Conhecimento (FC)** (NII, 1986). Estas podem ser procedimentos, conjuntos de regras ou asserções lógicas ou combinações entre eles. O objetivo de cada FC é contribuir com a informação que levará à solução do problema. Elas podem modificar apenas a estrutura QN e as estruturas de dados de controle. O QN é alterado apenas pelas fontes de conhecimento (NAKAMITI, 1991 e NII, 1986). As FCs devem ter pré-condições indicando as condições no Quadro-Negro sobre as quais devem atuar e contribuir para a solução do problema.

As FCs devem gerar hipóteses através do uso de regras que: adicionam ou modificam elementos da hipótese, ou adicionam ou modificam relações entre elementos. A forma como as regras são organizadas em uma FC e quais regras estarão presentes nela, dependem de sua funcionalidade no plano global da solução do problema (NAKAMITI, op. cit.). A organização em si pode representar modelos, onde as regras são agrupadas em função dos objetos, ou conceitos, de forma similar à representação por "frames", ou pode ser feita em função de eventos, onde funções similares podem ser agrupadas (NAKAMITI, op. cit.).

### 2.4.2 - A Estrutura de Dados Quadro-Negro

Todos os dados relativos ao estado da solução do problema, são armazenados em uma base de dados global denominada **Quadro-Negro**. As FCs produzem alterações no QN que levam incrementalmente a uma solução ou a um conjunto de possíveis soluções. As interações entre as FCs acontecem através das alterações no Quadro-Negro. O propósito da estrutura QN é manter dados computacionais e dados do estado da solução, produzidos e requeridos pelas FCs. Os objetos do espaço de soluções compõem a estrutura QN, podendo ser dados de entrada, soluções parciais, alternativas e soluções finais, cujo armazenamento é hierarquicamente organizado em níveis de análises. Assim, a informação associada a um nível serve como entrada para um conjunto de FCs, que por sua vez, põe nova informação no mesmo nível e em outros níveis. O vocabulário do espaço de soluções é definido pelos objetos e suas propriedades, que são representadas por pares de valores de atributos. Os relacionamentos entre os objetos são denominados **links**, que podem ser dos tipos: "**parte_de**" ou "**em_apoio_a**" para objetos em diferentes níveis, e "**próximo_de**" ou "**segue**" para objetos no mesmo nível. O Quadro-Negro pode ter múltiplos painéis do tipo Quadro-Negro, ou seja, o espaço de solução pode ser segmentado em hierarquias múltiplas.

### 2.4.3 - O Módulo de Controle

O **Módulo de Controle** (**MC**) deve existir devido às restrições do processamento serial. Sua função é monitorar o acesso ao QN e também decidir quais as ações que deverão ser tomadas a seguir. Vários tipos de informação estão disponíveis globalmente para o **MC**, podendo estar no QN ou mantidas separadamente. O uso da informação de controle determina o foco de atenção, que indica o que deve ser processado a seguir. O foco de atenção pode ser as fontes de conhecimento, os objetos do QN ou uma combinação de ambos (NII, 1986). O **MC** pode resumir-se a um escalonador síncrono ou ser composto por uma coleção de FCs. Além disso, deve haver critérios para que os módulos de controle reconheçam quando o processo chegou ao fim. Esta geralmente é uma tarefa de uma FC específica, que deve reconhecer se uma solução satisfatória foi encontrada ou se o sistema não pode continuar por falta de conhecimento ou dados.

A abordagem do Quadro-Negro foi experimentalmente utilizada dentro do contexto do projeto **HEARSAY-II** de compreensão da fala (NII, op. cit.). O HEARSAY-II entendia perguntas faladas sobre temas em ciência da computação, armazenados em uma base de dados, respondendo às perguntas formuladas. O modelo proposto no projeto HEARSAY superou os limites dos sistemas de reconhecimento de voz, incorporando a importância do contexto, sintaxe, semântica e regras fonológicas, reconhecidas como essenciais para o reconhecimento de voz. O projeto considerava duas estratégias de resolução de problemas: a primeira era uma

estratégia "bottom-up" onde as interpretações eram sintetizadas diretamente dos dados, trabalhando-se a hierarquia de abstração; a segunda era uma estratégia "top-down" que produzia sentenças alternativas a partir de conceitos sentenciais, seqüências alternativas de palavras de cada sentença, seqüências alternativas de fonemas de cada palavra e assim sucessivamente. O objetivo desse processo recursivo era produzir uma seqüência no nível paramétrico que fosse consistente com os dados de entrada.

## 2.5 - Considerações Lingüístico-Pragmáticas

A comunicação entre os seres humanos é feita através de sistemas de signos. O sistema de signos lingüísticos é um dos principais sistemas para comunicação humana. O signo lingüístico compõe-se de uma Expressão e um Conteúdo relacionados (OLIVEIRA, 1990). Esta relação é utilizada na codificação/decodificação do signo, ou seja, na sua tradução, que é um dos objetivos ambiciosos do cientistas da computação (RAMAN e ALWAR, 1990). Tradicionalmente a tradução está associada à relação significante/significado. Entretanto, OLIVEIRA (op. cit.), comenta que "conforme mudam-se os domínios, mudam-se também as relações que dão significação a uma dada 'palavra'". A figura 2.9 ilustra alguns exemplos de alteração das relações comentados por OLIVEIRA (op. cit.).

| |
|---|
| **DOMÍNIO 1 - SOL <---- antonímia ----> CHUVA** |
| **DOMÍNIO 2 - SOL <-- particularização_de --> CORPO CELESTE**<br>**LUA <-- particularização_de --> CORPO CELESTE** |
| **DOMÍNIO 3 - SOL <---- sinonímia ----> DIA**<br>**LUA <---- sinonímia ----> NOITE**<br>**SOL <---- antonímia ----> LUA** |

**Figura 2.9 - Relações significativas da "palavra" sol em domínios diferentes (Fonte: OLIVEIRA, 1990, página 7).**

OLIVEIRA (op. cit.) salienta que a tradução também pode ser intra-códigos, quando se estabelece um conjunto de relações que delimitam e permitem o trânsito entre domínios. Logo podemos tomar a tradução como a tarefa de mapear expressões de um domínio específico em um outro. Neste mapeamento o Conteúdo de uma expressão em um domínio é primeiro traduzido para uma representação semântica independente dos domínios, e em seguida mapeado para o outro domínio. A representação semântica, permite então, que a tradução seja realizada em dois estágios: um para análise e armazenamento semântico e outro para gerar expressões a partir da semântica obtida.

# 3 - O CONHECIMENTO DO SISTEMA

Neste capítulo descreve-se o conhecimento necessário para conceber um ambiente em LN que permita melhorar a interação homem-máquina em softwares especialistas.

O tratamento da Linguagem Natural, no domínio restrito, tem como base o sistema **IDEAL** de OLIVEIRA (1990). Este sistema, através das interações com o usuário, provê o universo lingüístico suficiente para compreender, interpretar e traduzir frases em LN (em Língua Portuguesa do Brasil) para uma representação de conhecimento. Assim, a aquisição do conhecimento léxico-semântico-pragmático das solicitações dos usuários, uma das tarefas mais complexas, é executada pelo sistema IDEAL, adaptado para o ambiente em questão.

A figura 3.1 ilustra a arquitetura do sistema IDEAL. O Sistema IDEAL compõe-se basicamente dos seguintes módulos: a) **Quadro-Negro (QN)**; b) **Gerenciador de Ações por Objetivos (GAO)**; c) **Módulo de Tradução intra-sistema**; e, d) **Módulo de Tradução inter-sistema**. Esta arquitetura usa o modelo Quadro-Negro que considera a solução de um problema por resolvedores distintos, onde cada um contribui, oferecendo soluções parciais ou totais para o problema. Estes resolvedores (fontes de conhecimento) compartilham o mesmo conhecimento representado do domínio.

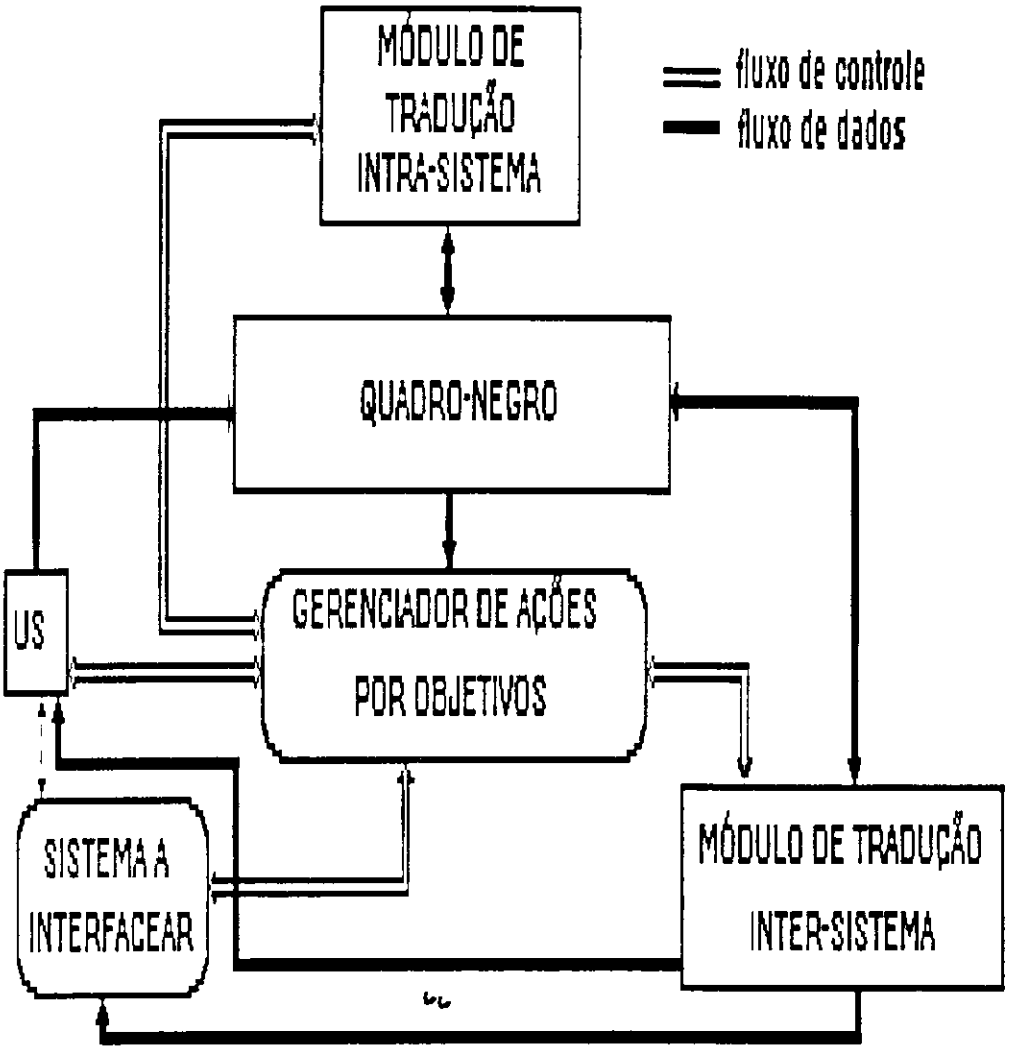



**Figura 3.1 - Arquitetura do Sistema IDEAL. (Fonte: OLIVEIRA, 1990, página 39)**

No módulo de **Quadro-Negro** do sistema IDEAL, se tem a inserção de dois níveis: de **controle** e de **usuário**. O nível de **controle** existe por que cada fonte de conhecimento sabe quem deve ser ativado após sua atuação; o nível de **usuário** existe por que o usuário é considerado como uma das fontes de conhecimento do sistema que produz novas informações a

partir de inferências sobre o conhecimento que possui e ao qual é exposto.

O módulo **Gerenciador de Ações por Objetivos - GAO**, controla e planeja as ações que devem ser desenvolvidas, gerenciando a ação dos módulos e a comunicação entre todos os elementos do Sistema. Para isso, este módulo gera e/ou atualiza as Estruturas de Ações dirigidas por Objetivos (EAO). A necessidade do GAO está associada ao fato de cada elemento do Sistema ser autônomo, no sentido de que suas funções e especialidades concorrem com o processo de codificação/decodificação, quando sua interferência é solicitada. Desde que a comunicação entre os módulos dá-se pelo QN, faz-se necessário um gerenciador que implemente esta comunicação de maneira escalonada. As fontes de conhecimento podem gerar dados de controle para ativar/desativar sub-processos em cadeia até que a solução do problema seja encontrada. Uma EAO ativada pode gerar outras EAOs, que serão gerenciadas pelo gerador e atualizador de EAOs do próprio GAO.

O **Módulo de Tradução intra-sistema** compõe-se das seguintes fontes de conhecimento: a FC para realizar a **busca estereotípica**, a FC para a **análise morfo-semântico-sintática** e a FC para a **interpretação**. A busca estereotípica inicia-se com a inserção da frase do usuário no QN. Neste instante o GAO cria uma EAO que desencadeia o processo. Esta busca é feita através de vários passos, enumerados na página 119 de OLIVEIRA (1990). A análise morfo-semântico-sintática é realizada também através de FCs

individuais para cada tipo de análise, morfológica, semântica e sintática. Estas fontes estão explicitadas nas páginas 122 a 130 de OLIVEIRA (op. cit.). O módulo de interpretação é responsável pela escolha de uma das hipóteses de interpretação, que satisfaça todas as restrições impostas por uma frase em particular (RICH, 1988, página 363). O universo de possibilidades para a interpretação, é restringido à medida que o módulo interpretativo estabelece/confirma as hipóteses causais, atuando sobre o conjunto lexia, lexema e listas de casos. As regras desta FC estão nas páginas 131 a 134 de Oliveira (op. cit.)

O **Módulo de Tradução inter-sistemas** permite que o conhecimento armazenado em uma representação, possa ser mapeado para uma outra. Neste sentido o sistema IDEAL mostra características de flexibilidade por aceitar e utilizar uma linguagem de descrição de representação de conhecimento adotada por um sistema especialista, e de transportabilidade por poder ser utilizado por qualquer outro sistema especialista. Neste módulo, os sistemas que efetivamente interagem são: o Usuário, o próprio IDEAL e o Sistema Especialista no domínio.

O conhecimento do domínio de aplicação é representado através de estruturas de FRAMES (através da linguagem de Frames de ARARIBÓIA (1989)), que permite descrever um objeto ou uma situação, de forma estereotipada. A hierarquia na representação por FRAMES estabelece um mecanismo de herança que evita que o

conhecimento seja duplicado. Por outro lado, algumas características dos OBJETOS são descritas por regras de inferência associadas à representação por FRAMES. Como exemplo de uma representação de um OBJETO, tome-se o domínio da Computação Gráfica, para a representação da descrição de um "QUADRADO" no espaço bidimensional (2D) como na figura 3.2:

```
Classe do objeto : figura_2D
Instância : quadrado1
Atributos :
     Ângulos de 90°
     Número de ângulos 4
     Centro
     Coordenadas do Centro - (Xc,Yc)
     Comprimento do lado
     Numero de lados de comprimento iguais
     Cor da aresta
```

**Figura 3.2 - Frame de representação de um quadrado.**

Na figura 3.2, a **instância** está associada à descrição de um objeto genérico da classe descrita, que pode ser usado, por exemplo, como referência para o "matching" durante a identificação de um OBJETO em análise. A esta referência pode-se inclusive acrescentar outros slots que podem ser preenchidos com atributos referenciais, como por exemplo o conceito de grande, pequeno, etc. Os atributos do objeto em descrição são associados aos slots por atribuição de valores, que podem ser do tipo <valor>, para valores conhecidos e confiáveis, ou seja, sobre os quais se tem certeza da origem e da veracidade, e, <default>, para valores sobre os quais não se tem garantia quanto à veracidade, porque são baseados em generalizações.

As informações sintáticas e semânticas, suficientes para o processamento lingüístico do domínio considerado, e a representação da gramática tradutológica são armazenadas em dicionários. Esta representação é feita utilizando-se frames construídos através de declarações de armazenamento de cláusulas no banco de dados do PROLOG. Na inicialização do ambiente estas cláusulas são carregadas em memória, podendo ser relembradas em qualquer momento por pesquisas ao banco.

As semânticas representadas contêm a definição de verbos primitivos e derivados que devem ser usados para classificar novas entradas (verbos) do usuário. Durante a classificação são estabelecidos os atributos do verbo e dos objetos, manipuláveis pelo verbo. Por exemplo, o verbo MUDAR pode ser classificado como um verbo que altera o atributo posição de um objeto. Quando solicitado é necessário especificar a posição de origem e a posição de destino, ou seja, de onde e para onde o objeto será deslocado. Na classificação também se estabelece em que ações primitivas ou derivadas o verbo pode ser dividido. No exemplo citado anteriormente, o verbo MUDAR pode ser dividido em outros verbos primitivos ( ações ou processos ) para identificar, localizar o objeto e alterar seu atributo posição, que é a tarefa explicitada na solicitação de ação. A hierarquia existente entre as primitivas está explicitada nas regras que as definem e é efetivamente executada através da agenda do mecanismo de controle (escalonador) do Quadro-Negro.

O ambiente é concebido na arquitetura de Quadro-Negro que incorpora um controle com a agenda implementada através do dicionário. Nela as regras executadas (por exemplo, as primitivas de ação) sinalizam a finalização ou não de sua execução e a necessidade por dados. O mecanismo de controle implementado assemelha-se a um escalonador de ações devido à hierarquia existente na execução de muitas tarefas, que precisam verificar a agenda em busca de ponteiros de execução das tarefas que as antecedem e dados processados.

## 3.1 - O Conhecimento Lingüístico-Pragmático

As propostas de significado no ambiente concebido são representadas, segundo a teoria de SCHANK (1975), através de CONCEITUAÇÕES. SCHANK (op. cit.) classifica as propostas de significados de uma sentença como CONCEITUAÇÕES ATIVAS ou ESTÁTICAS.

A conceituação ativa caracteriza-se por ser segmentada em AGENTE, AÇÃO, OBJETO e DIREÇÃO, ou seja, um AGENTE fazendo uma AÇÃO, sobre um OBJETO em uma DIREÇÃO. O termo ATIVA significa que o OBJETO sofre uma transformação devido à AÇÃO. A conceituação estática, significa que não há transformação sobre o objeto. Sua forma é OBJETO em um ESTADO.

Para o propósito do ambiente em descrição a máquina e o usuário exercem o papel de AGENTE das conceituações ativas alternadamente, portanto, o AGENTE não é considerado na representação do significado das solicitações dos usuários no ambiente, assim, as CONCEITUAÇÕES ATIVAS consideradas no ambiente são da forma ( **AÇÃO OBJETO DIREÇÃO** ).

A classificação das **CONCEITUAÇÕES** requer que as frases sejam segmentadas em seus elementos constituintes, conduzindo ao conceito de ações primitivas, porque cada ação explicitada por um verbo pode ser dividida em uma ou mais ações. Algumas delas são ações que não mais se dividem em outras ações, neste caso, nenhuma outra ação pode antecedê-las. Estas são classificadas como ações primitivas. Este fato, induz a pensar em classes de verbos como AÇÕES e PROCESSOS. A classificação dos verbos, pode ser conforme BORBA (1985a; 1985b; 1985c) que mostra os critérios de classificação dos verbos da Língua Portuguesa em verbos de AÇÃO, de PROCESSO, de AÇÃO-PROCESSO e de ESTADO. Os verbos de AÇÃO são aqueles cujo sujeito se caracteriza como AGENTE, isto é, o elemento instigador, capaz de, por si mesmo, desencadear a AÇÃO, a dinâmica da frase. Já os verbos de PROCESSO possuem sujeitos que são afetados por algo externo a eles, ou seja, os sujeitos dos verbos de PROCESSO sofrem modificação com o PROCESSO. A classificação dos verbos evita a redundância de armazenamento do conhecimento, sendo portanto, importante para a concepção do modelo representativo.

As AÇÕES e PROCESSOS comentados anteriormente são mapeados através de uma gramática tradutológica simples composta de: **VBO**, **PROCESSO**, **AÇÃO** e **PRIMITIVAS**, que apresentam relacionamentos variáveis semelhantes conforme a figura 3.3.

```
VBO:=PROCESSO/AÇÃO
    PROCESSO:=AÇÃO/PROCESSO/<VAZIO>
    AÇÃO:=PRIMITIVAS/AÇÃO/<VAZIO>
    PRIMITIVAS:={ conjunto de ações de nível mais baixo }
```

**Figura 3.3 - Esboço do mapeamento da gramática tradutológica**

O ambiente objetiva prover um meio de programação automática dentro de um domínio de aplicação restrito. A figura 3.3 mostra uma árvore genérica de AÇÕES/PROCESSOS resultante da divisão da solicitação, de onde se extraiu a "intenção" do usuário. Esta divisão permitirá que se monte a seqüência de primitivas necessárias para realizar a tarefa requisitada, sem a participação do usuário, caracterizando-se como uma programação automática do seu ponto de vista.

A gramática mencionada na figura 3.3 é fornecida "a priori" na sua forma mais simples e incrementada/alterada durante as interações com o usuário. A árvore na figura 3.4 ilustra uma forma de representação da gramática requerida.

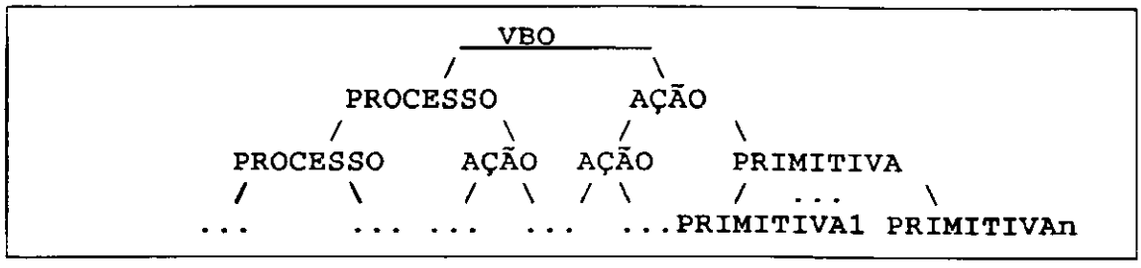



Figura 3.4 - Esquema gráfico resultante da classificação das ações.

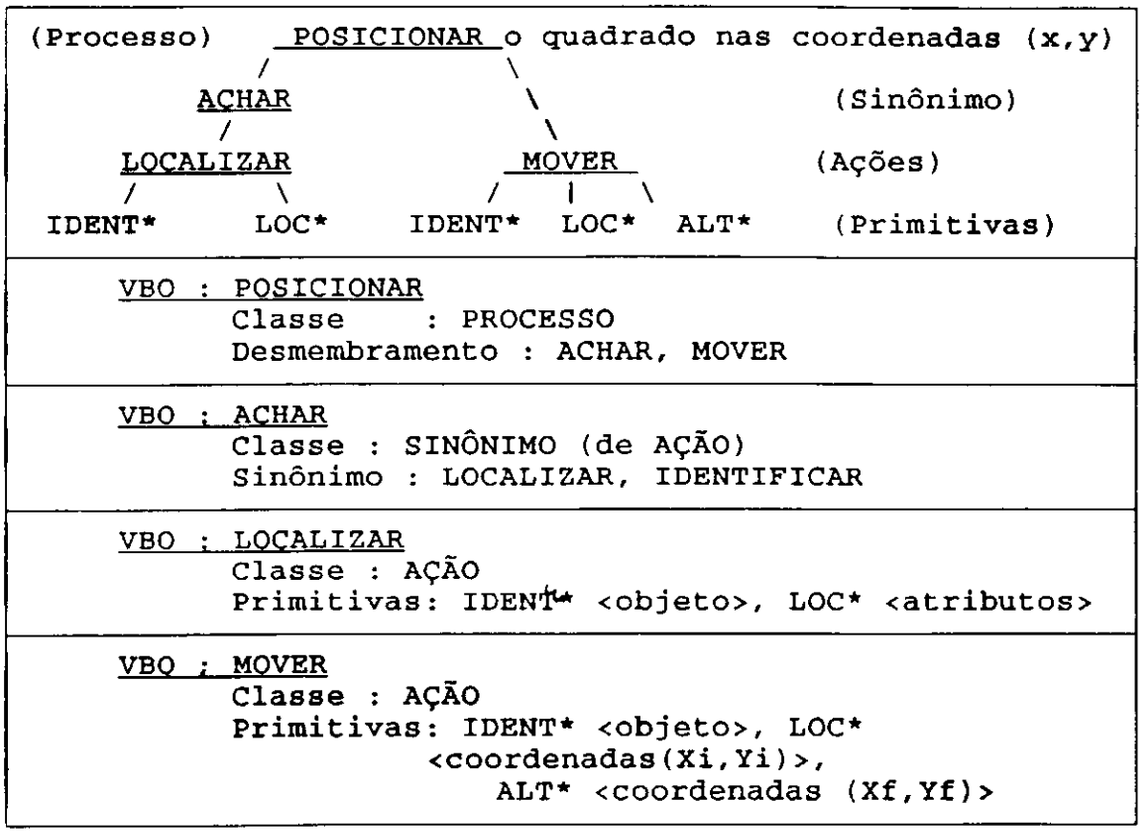



Figura 3.5 - Exemplo de Representação do Conhecimento Tradutológico.

A figura 3.4 exemplifica que dentro de um domínio de aplicação, o VBO pode representar um PROCESSO ou uma AÇÃO, os quais ainda podem representar outros PROCESSOS ou AÇÕES, e assim sucessivamente, até se atingir uma ação primitiva.

É neste sentido que os verbos apresentam relacionamentos variáveis e semelhantes.

Como exemplo, considere-se a seguinte solicitação no domínio da Computação Gráfica (2D) : "**POSICIONAR** o quadrado nas coordenadas (x,y)". Tomando-se as considerações de BORBA (op. cit), o verbo **POSICIONAR** pode ser classificado como um PROCESSO. Além disso, sua execução, pode ser dividida em outras AÇÕES e/ou outros PROCESSOS. Assim, o verbo "POSICIONAR" pode ser representado, de acordo com esquema da figura 3.4, como está ilustrado na figura 3.5:

Na representação da figura 3.5, o verbo POSICIONAR, classificado como um PROCESSO, é dividido nas AÇÕES ACHAR e MOVER. Por outro lado, a AÇÃO ACHAR tem como sinônimo LOCALIZAR. A AÇÃO MOVER é dividida nas PRIMITIVAS IDENT*, LOC* e ALT*. O verbo ACHAR, classificado como sinônimo de LOCALIZAR, herdou toda a classificação do verbo LOCALIZAR. No final, a representação de mais baixo nível seria, eliminando-se as redundâncias se houverem:

```
IDENT* <objeto>, LOC* <atributos>,
LOC* <coordenadas(xi,yi)>, ALT* <coordenadas(xf,yf)>
```

Esta representação, por si só, lançaria expectativas semântico-sintáticas que conduziriam a análise do restante da frase, ou seja, conforme figura 3.5, "o quadrado em (x,y)".

Assim, "quadrado" deverá ser o <objeto> sob análise e "x,y" coordenadas desse objeto. A primitiva IDENT* deve verificar, a existência do objeto "quadrado" e LOC* deve extrair todos os atributos de todas as instâncias desse objeto. A outra chamada da primitiva LOC* deve extrair do conjunto de todos os atributos anteriormente extraídos o subconjunto de todas as coordenadas (x,y), ficando assim implícito, mesmo na ausência da palavra "coordenada", ou similares desta, que se referencia às "coordenadas de um objeto que se parametrizam por algum (x,y)".

Como o "x,y" da frase não se identifica com o subconjunto de coordenadas, ele será mapeado para a primitiva ALT*. Dessa forma, o preenchimento da representação ficará numa forma frasal para melhor entendimento:

| "ALT* o objeto genérico QUADRADO com o conjunto coordenadas (x,y)n para a coordenada x,y". |
|---|

Neste contexto, o usuário será chamado a dar as coordenadas que identificam a instância do objeto genérico, que serão então conferidas com as do subconjunto extraído por LOC*, determinando assim o QUADRADO específico sobre o qual se fala. Só então, completada a representação e passível de execução, ela será passada para o processo computacional seguinte como:

| "ALT* as coordenadas do QUADRADOn para x,y". |
|---|

Restrições devem ser estabelecidas em vários níveis da representação do conhecimento, pois o ambiente precisa ser consistente para evitar interpretações inócuas ou trabalho desnecessário, provendo uma boa automatização e eficiência na execução das tarefas. Solicitações tais como "Girar (sinônimo de "Rotacionar") um círculo de 30°", no domínio da Computação Gráfica 2D, devem ser descartadas pelo sistema por serem inócuas. Restrições deste tipo, podem ser incluídas na representação do conhecimento do domínio através do preenchimento de um slot nos frames de descrição dos objetos. Este preenchimento significa acrescentar valores ou indicações, enfatizando a propriedade de que não faz sentido aplicar uma transformação de rotação sobre o objeto.

Toda a primeira fase do ambiente proposto está baseada no conhecimento lingüístico da LPB e no conhecimento que o usuário, de intuitiva e normalmente tem sobre sua língua materna. E as questões levantadas pelo ambiente são aquelas que se fazem num diálogo cotidiano, como no exemplo a seguir:

**US > Mostre a posição de todos os triângulos.**
**AMBIENTE > { conjunto de tuplas }.**
**US > Mude a cor do triângulo da posição (x,y).**
**AMBIENTE > Para que cor?**
**US > Verde.**
**AMBIENTE > { Lista de inteiros correspondentes as cores }**
**AMBIENTE > Indique o inteiro correspondente.**
**US > 3.**
**AMBIENTE > "posição" é sinônimo de "coordenada" ?**
**US > Sim**

O conhecimento lingüístico inclui desde regras até elementos gramaticais da LPB (artigos, preposições, pronomes, etc) e suas funções. Como exemplo, na figura 3.6 tem-se algumas regras para toda a gama de combinações entre preposições.

```
mover de <LOC* atual> para <ALT* destino>
posicionar em <ALT* destino>
mover para <ALT* destino> o objeto em <LOC* atual>
```

**Figura 3.6 - Combinações entre preposições**

Observe-se que a combinação das preposições de e para na figura 3.6 denota a intenção em alterar-se a posição do objeto, ou seja, independente da explicitação ou não do atributo posição, é fato que ao se solicitar "mudar o quadrado de 30,30 para 40,40", estar-se-á na realidade solicitando implicitamente que se mude o quadrado da posição 30,30 para a posição 40,40. Qualquer outro atributo que se deseja mudar será normalmente especificado pelo usuário, se não considere-se a solicitação "Mudar a cor do quadrado em 30,30 para 2". Esta combinação de preposições faz parte do conhecimento lingüístico intuitivo do usuário e deve constar no conhecimento lingüístico do ambiente. Isto, caracteriza uma modelagem do usuário, uma vez que se está representando expectativas de ações que podem ser solicitadas.

Além do conhecimento representado a priori o ambiente apreende novos conhecimentos nos diálogos com o usuário, quando pode-se estabelecer relações de sinonímia,

como no diálogo do exemplo anterior onde a palavra "verde" e o número 3 são sinônimos, ou definir a função de quantificadores, por exemplo "todos", pela "confirmação/orientação" das perguntas. Este raciocínio é válido para as demais interações.

## 3.2 - O Conhecimento Tradutológico

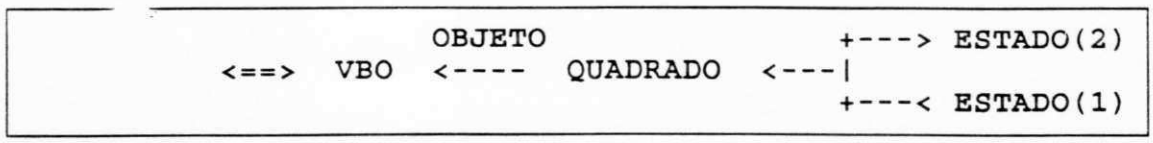
Esta seção descreve as primitivas concebidas para realizar as tarefas do domínio em questão. Cada primitiva é declarada através de cláusulas em PROLOG usando-se uma linguagem de frames em um nível mais baixo (ARARIBÓIA, 1989), que permite a navegação na estrutura de frames utilizada na representação do conhecimento do domínio.

As primitivas sendo ações de baixo nível, não podem ser subdivididas em outras AÇÕES e/ou PROCESSOS. Todas as ações, entretanto, devem ser realizadas através destas primitivas que são responsáveis pela execução das tarefas específicas no domínio de aplicação. Logo conforme o domínio mude, as primitivas também devem ser alteradas. As primitivas executam no nível do conhecimento através da navegação na estrutura de frames do conhecimento. Esta navegação é feita através da linguagem de frames que cria e manipula as estruturas de frames definidas na fase de representação. A linguagem de frames utilizada está conforme discutido na seção 2.3.

Considerando o domínio da Computação Gráfica (2D), pode se estabelecer as seguintes primitivas: **IDENT***, **LOC***, **ALT*** e **GEN***. Todas as ações deste domínio são mapeadas em seqüências destas primitivas a serem executadas. Observe-se que o domínio é restrito, no sentido de que não objetiva-se cobrir todas as possíveis tarefas, nem todo o seu conhecimento. Portanto, as primitivas definidas podem não ser suficientes para mapear todas as ações existentes do domínio, sendo necessário então, em caso de ampliação das fronteiras do domínio, definir novas primitivas que permitam o mapeamento do maior número de ações possível.

As primitivas têm funções individualizadas. A primitiva IDENT* toma um conjunto de atributos de um OBJETO e faz o "matching" com a descrição do OBJETO armazenada na base de conhecimento do domínio (estrutura de frames). A primitiva LOC* toma um conjunto de atributos (que inclui a descrição do OBJETO) e retorna uma conceituação estática do tipo "OBJETO na posição (x,y)", onde (x,y) pode representar, por exemplo, as coordenadas do atributo 'CENTRO' do objeto. ALT* é a primitiva usada para mudar (ou alterar) os atributos desejados. A primitiva GEN insere novos objetos no contexto do domínio em análise, executando no nível do conhecimento, ou seja, dispensando o uso interativo de um software específico do domínio (no caso, software gráfico).

A solicitação "POSICIONAR o quadrado em (x,y)" da figura 3.5, pode ser representada com as mesmas regras de sintaxe conceitual de SCHANK (1975).




O mapeamento através das primitivas IDENT, LOC e ALT, é feito segundo o esquema da figura 3.7.

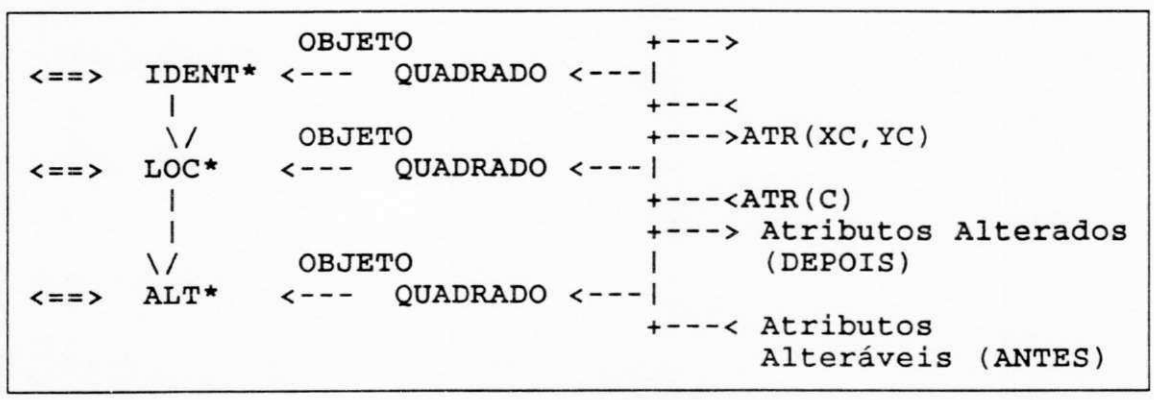



**Figura 3.7 - Mapeamento através de Primitivas segundo a DC.**

O verbo "POSICIONAR" pode ter sinônimos tais como "MOVER", "COLOCAR", e outros, que poderiam ser utilizados com a mesma semântica. Este conhecimento de sinonímia é acrescentado através do IDEAL que provê o conhecimento lingüístico necessário para apreendê-lo e armazená-lo nos dicionários descritos anteriormente, através das interações com o usuário.

### 3.2.1 - A Primitiva IDENT

Esta primitiva é responsável pela identificação dos objetos no domínio. Para a execução da solicitação do usuário, é necessário averiguar se o objeto em questão é

manipulável, isto é, se ele pertence ao domínio de aplicação em questão e se está presente no contexto atual considerado. No caso de tratar-se de um objeto do domínio, e se presente no contexto, esta primitiva sinaliza para o sistema (põe em agenda) que confirma sua existência no domínio e no contexto, em seguida retorna o controle para a regra que a chamou. Caso contrário, o usuário é notificado através de mensagem, que o objeto em questão não está definido para este domínio ou que não existe no contexto atual (na tela atual). Assim, o usuário tem a opção de escolher entre uma nova solicitação ou a definição do objeto da solicitação atual.

A primitiva IDENT recebe como parâmetro o nome do objeto sobre o qual se deseja processar alguma ação, conforme solicitação do usuário. A figura 3.8 ilustra o grafo e-ou da primitiva IDENT.

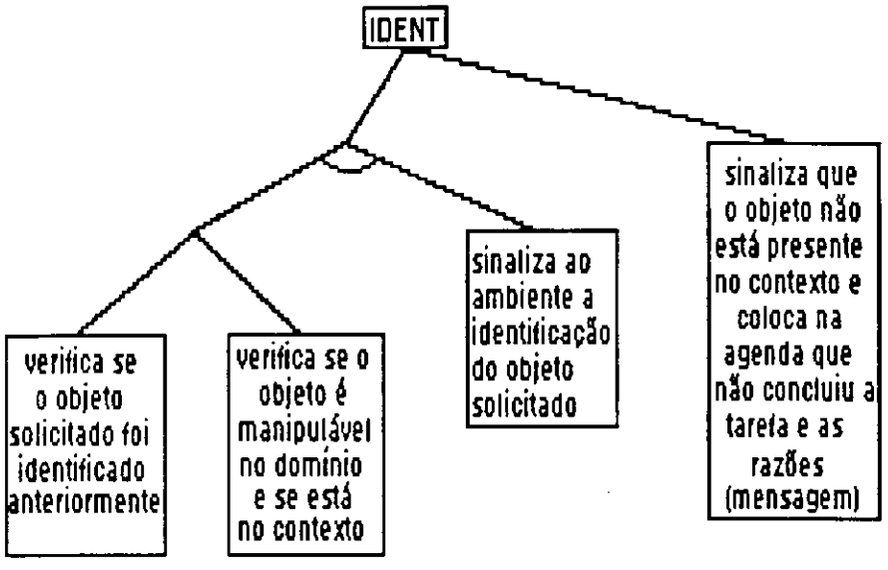



**Figura 3.8 - Grafo e-ou da Primitiva IDENT.**

### 3.2.2 - A Primitiva LOC

A função desta primitiva é exibir todos os atributos dos objetos requisitados. Sua execução sempre sucede a primitiva IDENT conforme a solicitação do usuário.

Quando chamada com algum parâmetro, como por exemplo, a posição do objeto, LOC retornará todos os atributos do objeto na posição. Quando não se especifica parâmetro algum para a busca, a primitiva localiza todas as instâncias do objeto da solicitação, exibindo seus respectivos atributos. Em seguida exibe o número de objetos do tipo referido na solicitação, presentes no contexto.

Terminada sua tarefa, esta primitiva sinaliza (põe em agenda) para o ambiente que a tarefa de localização foi concluída e retorna o controle para a regra que a chamou.

O grafo e-ou da primitiva LOC está ilustrado na figura 3.9.

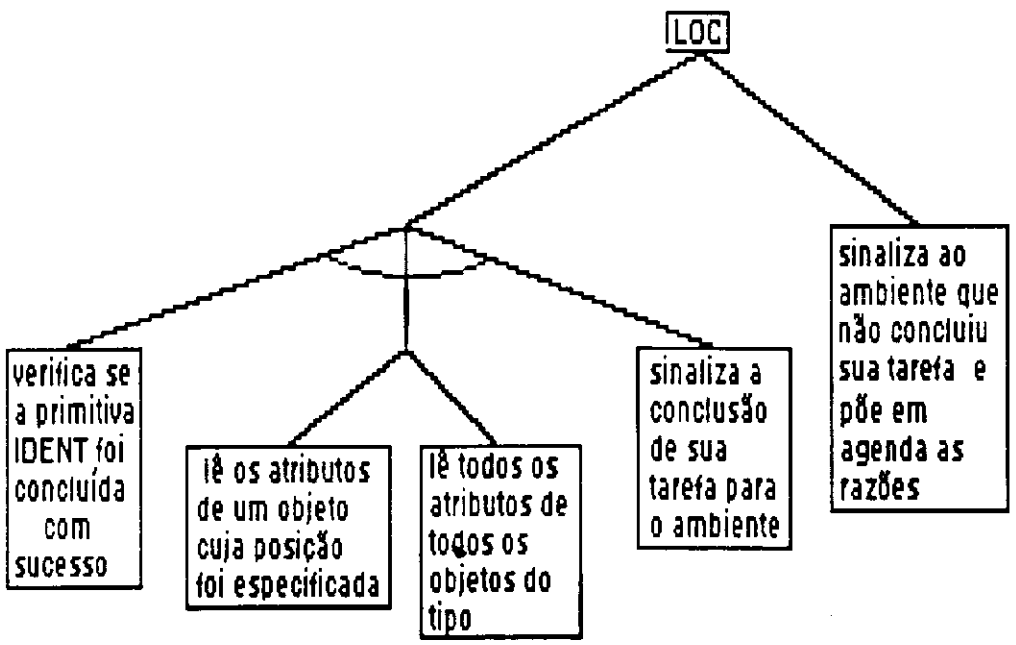



**Figura 3.9 - Grafo e-ou da Primitiva LOC.**

O exemplo a seguir ilustra o uso das primitivas IDENT e LOC, após a tradução de uma solicitação de ação hipotética do usuário ("exibir o quadrado em x,y") em primitivas na linguagem PROLOG.

```
ação:[ident,[quadrado]])
(ação:[loc,[coordenada,(X,Y)]])
```

### 3.2.3 - Primitiva ALT

Toda e qualquer alteração dos atributos de um objeto no contexto será sempre executada por esta primitiva. Sua ativação sempre sucederá a ativação da regra primitiva LOC; caso esta não seja concluída a primitiva ALT não será ativada. O sistema de tratamento lingüístico além de analisar morfológica, sintática, semântica e pragmaticamente a solicitação do usuário, identifica a parametrização necessária para a execução da tarefa. Assim, é tarefa do módulo de tradução relacionar e passar os parâmetros envolvidos na ação a ser desenvolvida, para as primitivas no "Módulo de Execução".

Apesar da primitiva LOC exibir todos os atributos do objeto em questão em uma ordem para alteração de um objeto, a primitiva ALT, só alterará o atributo solicitado e/ou interpretado na fase de descoberta da intencionalidade do usuário.

É possível que não apenas um, mas que também vários atributos sejam alterados em uma mesma interação. Nestes casos, o ambiente passa uma lista de atributos para a primitiva ALT, que passa então a ser ativada recursivamente

até que todos os atributos tenham sido alterados. Concluídas as alterações, a primitiva sinaliza para o sistema (põe em agenda) a sua conclusão.

Observe-se o grafo e-ou desta primitiva na figura 3.10.

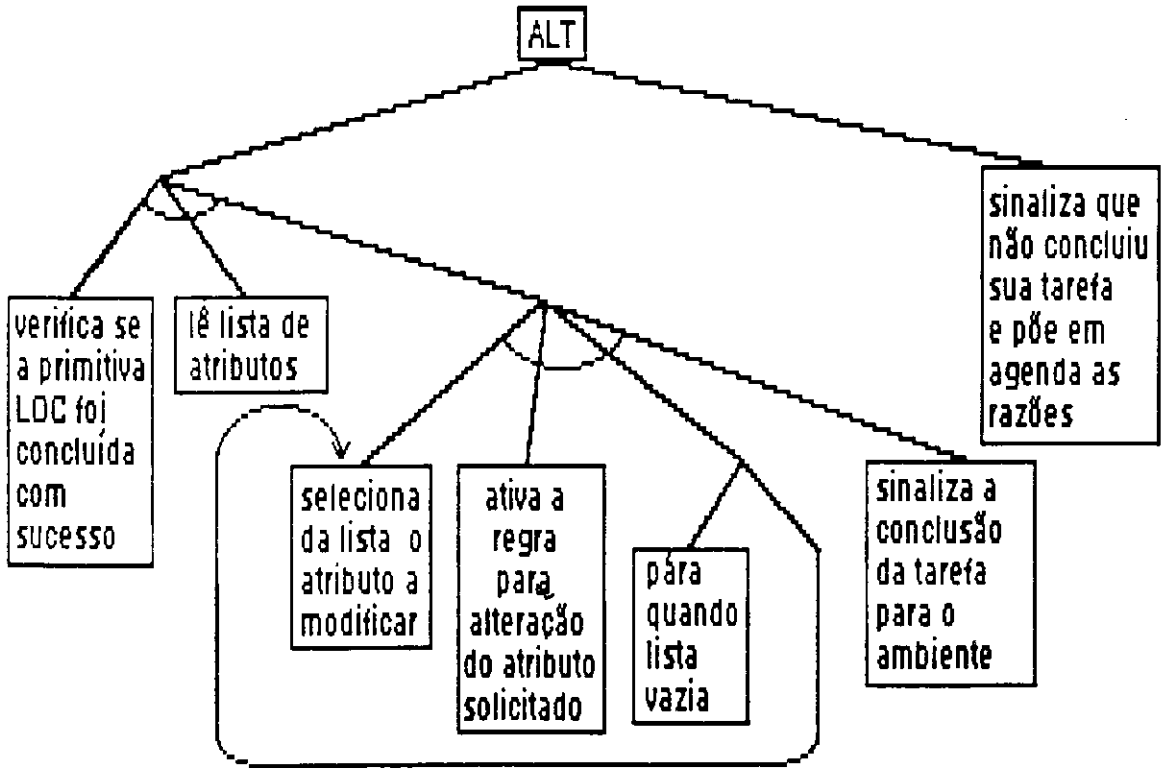



**Figura 3.10 - Grafo e-ou da Primitiva ALT.**

O exemplo que segue mostra o uso desta primitiva para a frase "Mudar o quadrado da posição (Xin,Yin) para (Xfin,Yfin)."

```
(ação:[ident,[quadrado]])
(ação:[loc,[coordenada,(Xin,Yin)]])
(ação:[alt,[coordenada,(Xfin,Yfin)]])
```

### 3.2.4 - Primitiva GEN

Novos objetos podem ser acrescentados ao contexto inicial de forma declarativa no nível do conhecimento, através da primitiva GEN. Esta primitiva pode criar ou gerar (desenhar) figuras que não estão presentes no contexto, tornando o ambiente capaz de realizar tarefas resultantes de entradas do seguinte tipo: "Criar um quadrado na posição X,Y com a cor W". O usuário explicitará todos os procedimentos e parâmetros necessários através de um diálogo com o ambiente. A execução desta primitiva é precedida da ativação da primitiva IDENT.

O grafo na figura 3.11 ilustra a regra que gera um objeto:

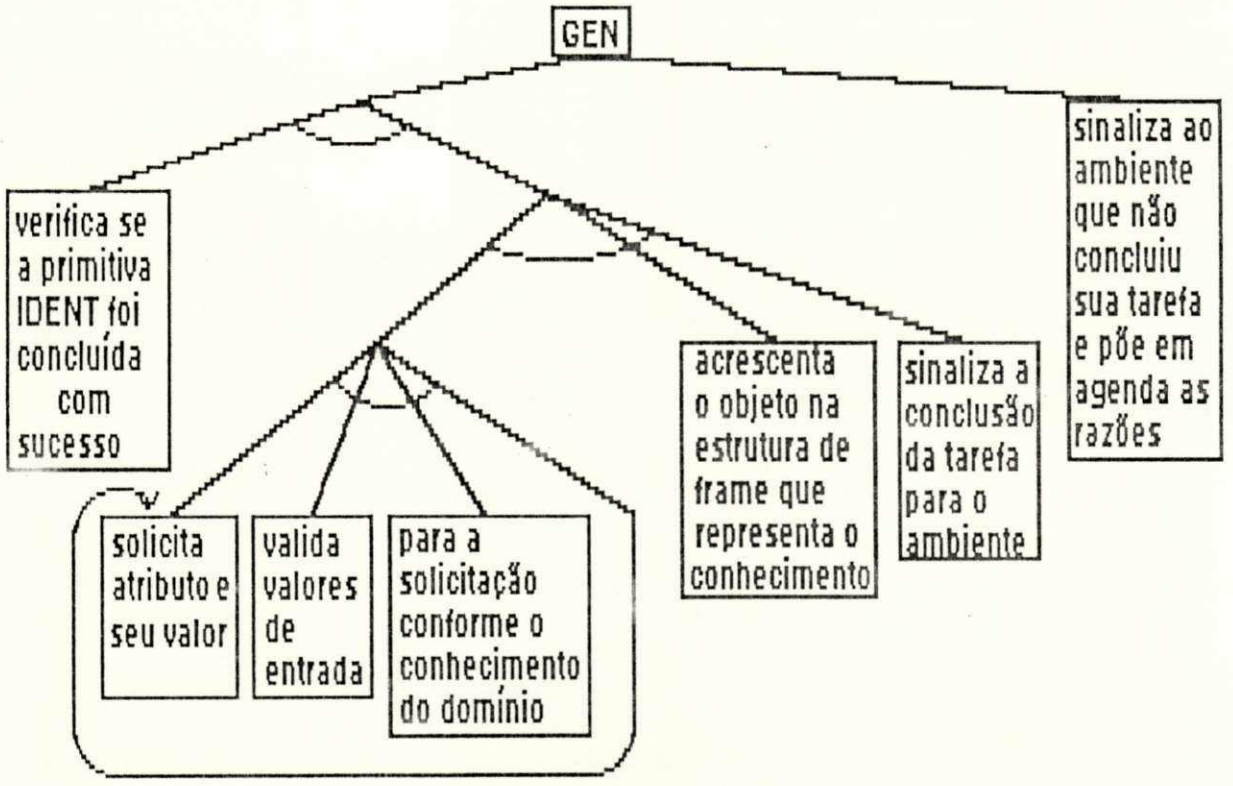



**Figura 3.11 - Grafo e-ou da Primitiva GEN.**

Observe-se nos exemplos anteriores que a primitiva LOC sempre sucede a ativação da primitiva IDENT e a primitiva ALT sempre sucede a ativação da primitiva LOC. Isso explicita então a existência de uma hierarquia entre as primitivas.

Supondo que foi detectado que a intenção do usuário é desenhar um objeto, por exemplo um quadrado, o ambiente, segundo a concepção atual, verifica que atributos descrevem o objeto referido e requisita os valores do usuário, através de diálogos como a seguir:

```
US > Desenhar um quadrado?
AMBIENTE > Qual a posição?
US > 10,10.
AMBIENTE > Qual a cor?
US > 2.
AMBIENTE > Qual o tamanho do lado?
US > 10.
AMBIENTE > Qual a orientação?
US > 0.
AMBIENTE > Confirme os dados?
AMBIENTE > {Mostra todos os dados inseridos}
US > Ok.
```

## 3.4 - O Conhecimento do Domínio

Para o desenvolvimento de um protótipo do ambiente proposto neste trabalho, considera-se o domínio (restrito) da Computação Gráfica em 2D, onde é possível realizar algumas transformações sobre objetos do mesmo.

A restrição do domínio deve-se ao fato de não se procurar realizar todas as transformações permitidas sobre todos os possíveis objetos do mesmo. Portanto, as primitivas discutidas anteriormente não são suficientes para mapear as diversas possibilidades de ação de todo o domínio da Computação Gráfica 2D. O conhecimento necessário para o domínio considerado engloba desde a descrição de objetos em 2D até as regras para cálculos de alguns atributos e para a realização das possíveis transformações sobre os mesmos, tais como, rotação, escalonamento, translação e alteração de cor.

Todo o conhecimento é representado utilizando o formalismo de representação por frames concebido por Minsky (WINSTON, 1975). A escolha deste formalismo deve-se ao fato do domínio considerado no trabalho ser similar ao domínio utilizado por Minksy (WINSTON, op. cit), quando da formulação teórica da representação por frames. As figuras 2.6 e 2.7 exemplificam a representação por frames de um cubo, segundo o concebido por Minsky (WINSTON, op. cit.).

No trabalho a estrutura de frames é implementada efetivamente de acordo com o descrito em ARARIBÓIA (1989), que concebeu uma linguagem de frames para implementar o modelo. Esta linguagem independe do domínio de aplicação em consideração. No caso do ambiente em proposição, ela é adaptada para um melhor entendimento das funções que seus comandos realizam.

Decidido o domínio de aplicação, faz-se necessário estabelecer que conhecimento deve ser utilizado. Esta é uma tarefa que envolve a quantificação do conhecimento necessário para a execução eficiente e efetiva das ações solicitadas, e deve ser feita por especialistas do domínio, conforme HOFFMAN (1987). Para o caso em consideração, este conhecimento está associado à descrição dos objetos manipuláveis, à declaração de regras que permitam realizar determinadas ações dentro do domínio e à descrição do contexto gráfico tomado como ponto inicial. O objetivo do trabalho é mostrar que, dada uma determinada situação dentro de um domínio, é possível realizar tarefas dentro do mesmo sem um envolvimento grande com a passagem de

parâmetros e a especificação dos procedimentos necessários, ou seja, no mais alto nível de abstração possível, do ponto de vista do usuário, caracterizando-se portanto em uma boa interação homem-máquina.

Considera-se a existência de uma fonte de conhecimento que processa a leitura do contexto gráfico e o representa em uma estrutura conhecida. Esta fonte pode ser composta de procedimentos escritos declarativamente ou proceduralmente utilizando-se qualquer linguagem de programação. Nenhuma restrição é feita a esta fonte, que, por exemplo, pode consistir em procedimentos utilizados por um sistema de processamento de imagens, procedimentos usados por um software do tipo CAD, um sistema gráfico que realize transformações em 2D, procedimentos de um sistema de processamento de sinais como em CARVALHO et ali (1991), etc, dependendo apenas do domínio em consideração. Uma outra fonte de conhecimento lê este contexto, já representado em uma estrutura conhecida, e o representa na estrutura de frames utilizando a linguagem de frames comentada anteriormente. Na figura 3.12 tem-se um esboço da distribuição espacial resultante da representação de uma dada situação de contexto (tela gráfica).

Neste esboço, o conhecimento utilizado engloba desde generalizações, até especificações detalhadas de objetos presentes no contexto gráfico referido anteriormente. Em um nível de abstração alto, para a representação, estão as generalizações dos objetos manipuláveis no domínio. Por exemplo, observe-se a descrição generalizada para objetos

de quatro lados e quatro ângulos. Esta descrição é conectada a um objeto específico do domínio, no caso o quadrado, que tem particularidades específicas que o distingue de outros objetos da mesma classe generalizadora (por exemplo, retângulo). A conexão referida é do tipo "A KIND OF" como em FIKES e KEHLER (1985).

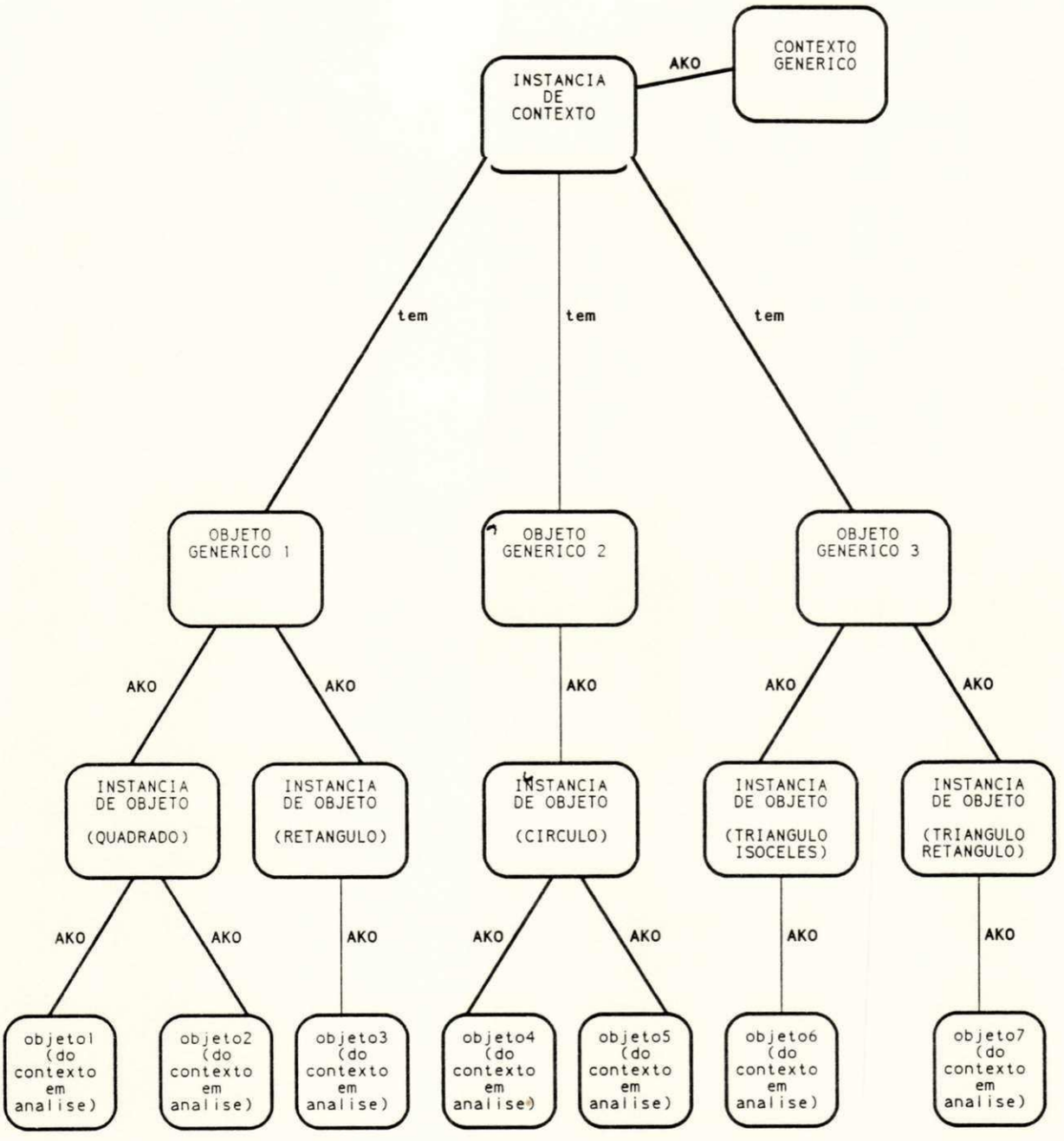



**Figura 3.12 - Esboço da Representação do Conhecimento dos objetos por Frames**

A representação do contexto significa mapear em frames todas as informações referentes ao contexto gráfico considerado. Assim, o quadrado referido anteriormente é usado como padrão para as instâncias presentes no contexto. Na figura 3.12 têm-se conexões entre os objetos do contexto, indexados pela ordem na qual são identificados no contexto, e as instâncias das generalizações, também através de ligações do tipo "A KIND OF". Observe-se que em alguns slots da descrição de alguns objetos, tem-se regras para realizar tarefas relativas especificamente ao objeto referido. Estas regras podem ser declarativas ou procedurais, ou seja, o modelo de frames permite o "procedural attachment" que significa o preenchimento de um "slot" com um procedimento para executar uma determinada tarefa.

O nível de detalhamento do conhecimento pode ser tão grande quanto o domínio requeira ou conforme a sua necessidade. A representação da figura 3.12 mostra um esquema de representação do conhecimento para um domínio específico e restrito. A estruturação está de acordo com a teoria de frames comentada no capítulo 2. Observe-se os tipos de ligação entre frames que permitem o mecanismo de herança entre os frames distintos. Na figura 3.13 tem-se um exemplo de preenchimento dos slots dos frames na figura 3.12. As informações contidas resultam da análise da necessidade do conhecimento para resolver problemas do domínio. Muitas informações dos objetos poderiam ser omitidas ou acrescentadas, dependendo dos limites estabelecidos para o domínio.

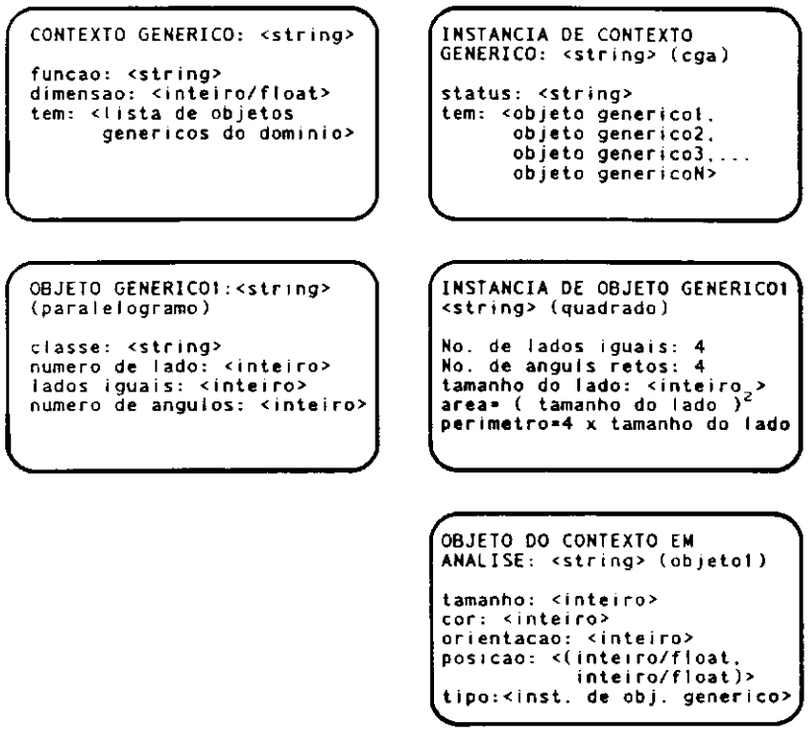



**Figura 3.13 - Exemplo de Preenchimento de Frames para um Domínio.**

No início da operação do ambiente, faz-se a ativação de todas as bases de dados contendo as premissas do conhecimento necessário do domínio. Em seguida o usuário é requisitado, caso esta seja a primeira utilização do ambiente, a definir a gramática tradutólogica e as primitivas de ação. Após esta inicialização ("setup") o conhecimento pode então ser acessado através de regras declaradas em PROLOG, que utilizam a linguagem de frames para manipular a informação representada. Estas regras são as primitivas definidas na seção 3.2. As requisições de

ação do usuário são antes traduzidas para uma representação, utilizando a gramática definida na seção 3.1, esta, por sua vez, é utilizada por um Módulo Executor de Tarefas que realiza a ação no nível do conhecimento.

Como exposto anteriormente, é evidente que ambientes como o proposto podem possuir dois tipos básicos de usuários: o que inicializa e o que usa o ambiente como usuário final. Entretanto, os limites entre eles não são simples de estabelecer, porque o usuário que inicializa também pode ser um usuário final. A diferença básica está no grau de especialização dos usuários no domínio em questão. Ou seja, a inicialização deve ser preferencialmente executada por um especialista que detém o conhecimento necessário para resolver problemas dentro do domínio. Como o ambiente é proposto para domínios restritos, dentro de uma abordagem lingüística livre, estas restrições dependerão da "experiência do especialista treinador.

A fase de treinamento (inicialização) engloba a definição de vários formas de conhecimentos. É nesta fase que o usuário treinador deve definir a gramática tradutológica, o conhecimento lingüístico suficiente para tradução e os objetos do domínio.

Todos os dados para representação podem estar armazenados em arquivos ou podem ser interativamente fornecidos pelo usuário ao ambiente. Para ilustrar tome-se

o exemplo abaixo, com a classificação dos elementos ainda não classificados de uma frase hipotética.

| |
|---|
| AMBIENTE > moveres é plural de mover? |
| US > não. |
| AMBIENTE > então confirme a AÇÃO desejada. |
| US > mover. |
| AMBIENTE > quadrados é plural de quadrado? |
| US > sim. |

No início da classificação o ambiente tenta descobrir a classe da palavra 'mover', perguntando se se trata de um substantivo. Com a resposta do usuário o ambiente passa a considerá-la como verbo e pede para que o usuário confirme a ação solicitada. Em seguida passa-se a análise da palavra 'quadrado', que é classificada como substantivo. O resultado desta interação é então armazenado na base de dados lexêmicos do conhecimento lingüístico. Há também uma fase de representação do conhecimento do domínio através de interações com o usuário, onde se estabelece os objetos e regras manipuláveis no domínio.

# 4 - A ARQUITETURA DO SISTEMA

## 4.1 - Descrição do Ambiente

O enfoque principal do trabalho está na possibilidade de uso da LN integrada a ferramentas e técnicas de inteligência artificial, para melhorar a IHM em sistemas paramétricos como por exemplo, sistemas gráficos, sistemas de processamento digital de imagens, sistemas de processamento digital de sinais, dentre outros.

Neste sentido de demonstrar a exeqüibilidade do ambiente proposto, desenvolveu-se um protótipo para executar ações dentro do domínio da computação gráfica (2D). Este protótipo foi desenvolvido utilizando-se a linguagem PROLOG.

O conhecimento necessário para realizar os objetivos do usuário é representado utilizando-se o modelo de representação por frames como no capítulo 3, sendo que a representação do conhecimento lingüístico está de acordo com a Dependência Conceitual de SCHANK (1975).

O conhecimento referido anteriormente, engloba tanto as bases de dados como as fontes de conhecimento, responsáveis pelas inferências no ambiente. Assim, optou-se por estruturar todo o conhecimento dentro de uma abordagem do modelo Quadro-Negro. Neste as fontes de conhecimento

compartilham a bases de dados do Quadro-Negro, sob o controle de uma fonte de conhecimento de controle, conforme mostra a figura 4.1.

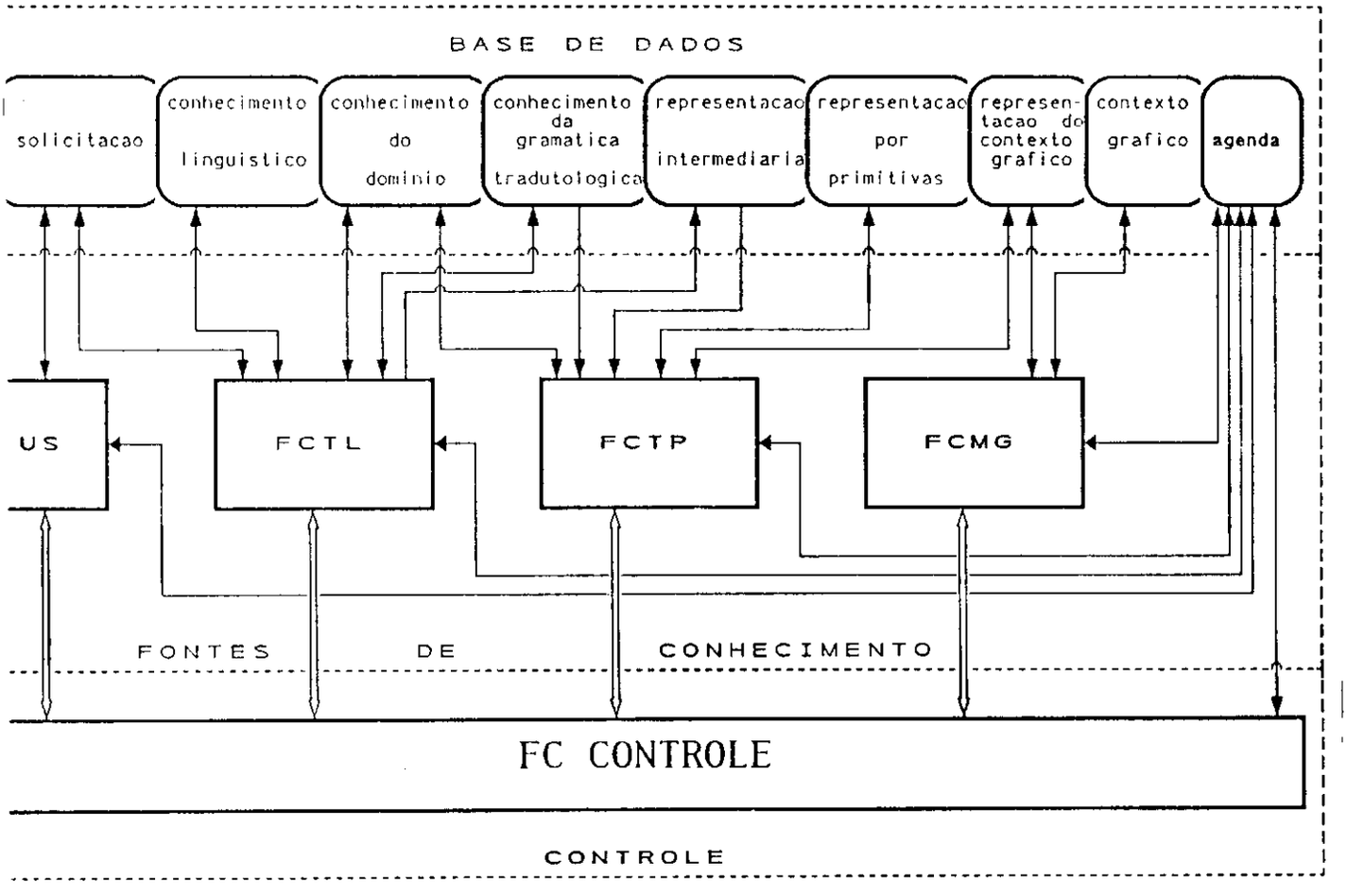



**Figura 4.1 - Arquitetura do Ambiente - Visão Quadro-Negro**

Considerando-se o objetivo principal do trabalho, não se tenta aqui conceber um ambiente para cobrir todas as possibilidades de ação dentro do domínio sobre todos os possíveis objetos do mesmo, restringindo-se a

processamentos básicos em figuras geométricas regulares em 2 dimensões, tais como, translação, rotação, escalonamento, exclusão e inclusão de novos objetos em um contexto gráfico.

No caso de se estender o domínio é necessário reavaliar se o conhecimento já representado (que inclui dados e regras), é suficiente para atender às novas possibilidades de ação.

A figura 4.1 ilustra a arquitetura do ambiente proposto, apresentando todas as fontes de conhecimento e bases de dados utilizadas para atender os propósitos do usuário dentro do domínio, explicitados pela sua intencionalidade na solicitação de ação.

Na arquitetura são apresentados três módulos básicos, referindo-se às fontes de conhecimento ativas do protótipo. O módulo FCTL (Fonte de Conhecimento para Tratamento Lingüístico) é responsável pelo tratamento lingüístico da solicitação do usuário. O módulo FCTP (Fonte de Conhecimento para Tradução em Primitivas) é composto por dois sub-módulos que executam as seguintes ações: a) a tradução da representação obtida no módulo FCTL para a representação por primitivas; e b) a interpretação (no sentido de execução) da representação por primitivas que resulta na realização da tarefa-meta do usuário. O módulo FCMG (Fonte de Conhecimento para Manipulação Gráfica) é responsável pela manipulação gráfica que executa a montagem das cenas (contextos gráficos), permitindo a visualização

do novo contexto gráfico, após a execução da(s) tarefa(s) no domínio do conhecimento (este módulo é o próprio sistema gráfico utilizado).

Na arquitetura da figura 4.1, cada módulo pode utilizar várias bases de dados que representam os vários tipos de conhecimentos necessários para realizar sua tarefa específica. O tratamento lingüístico requer o conhecimento das regras do sistema lingüístico utilizado, do domínio de aplicação em questão, da gramática tradutológica envolvida no processo de transformação da representação e da pragmática do usuário. A tradução em primitivas e sua respectiva interpretação (execução) utilizam-se do conhecimento do domínio, do contexto gráfico considerado, da gramática tradutológica, do conhecimento resultante do tratamento lingüístico (representação intermediária) e da seqüência de primitivas resultante da tradução.

Observe-se que dois outros módulos são considerados dentro da arquitetura na figura 4.1. Os módulos Usuário e de Controle. O Usuário é considerado como uma fonte de conhecimento porque pode inferir novos dados e informações a partir do conhecimento que possui. Para tanto interage com o Quadro-Negro através da base de dados referente a sua solicitação. O módulo de Controle é responsável pela ativação dos diversos módulos que compõem o ambiente. Sua existência está associada ao fato de que cada fonte de conhecimento sabe quem deve ser ativado após sua atuação e tem conhecimento para gerar informações que possam efetivar tal ativação. Praticamente este módulo consulta os dados em

agenda e avalia se a solução do problema foi alcançada em algum instante, quando então deve sinalizar que uma solução satisfatória foi encontrada, ou passar o controle para um módulo que pode prover uma solução ou melhorar a que estiver em consideração.

## 4.2 - Fonte de Conhecimento para o Tratamento Lingüístico - FCTL

Todas as solicitações de ação no contexto considerado do domínio de aplicação, são passadas ao ambiente através de solicitações (comandos ou ordens) em linguagem natural, que são interceptadas e interpretadas por esta fonte de conhecimento, sendo portanto, o nível mais próximo do usuário.

Assim, a FCTL tem como objetivo a compreensão do que o usuário redigiu no seu comando, ou seja, a interpretação da solicitação em linguagem natural para que seja representada em uma outra forma escolhida para corresponder a um conjunto de ações disponíveis que podem ser realizadas. Para isso faz-se os tratamentos morfossintático e semântico-pragmático do que o usuário redigiu, inferindo-se a partir do conhecimento morfológico, sintático e semântico previamente estruturado e da pragmática do usuário.

A realização da tarefa de compreensão da intenção do usuário, é feita por vários módulos de sistema, cada um com uma função específica, segundo o sistema IDEAL. A figura 4.2 ilustra a arquitetura da FCTL.

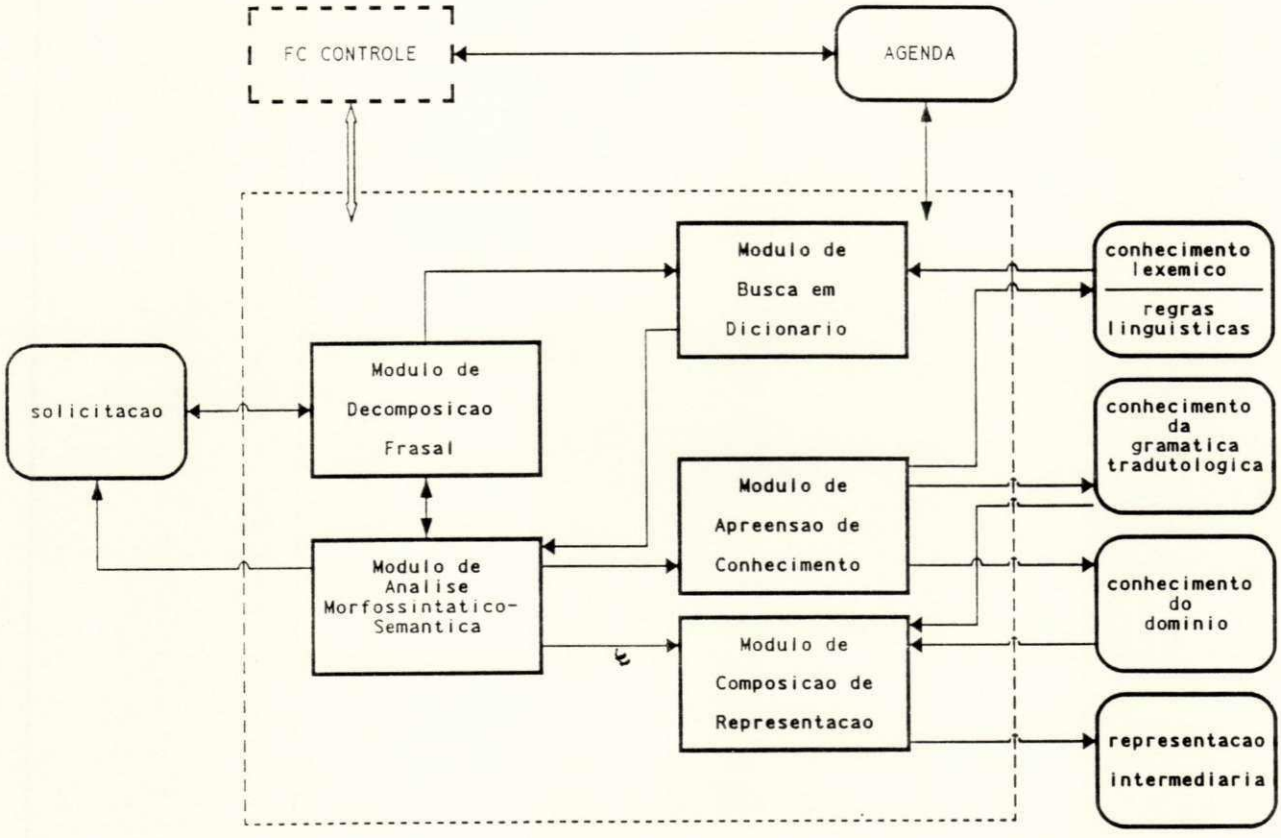



**Figura 4.2 - Arquitetura da FCTL**

A análise da solicitação do usuário começa após sua interceptação, quando o sistema inverte toda a frase e começa a segmentá-la em palavras decompondo-a caractere por caractere, observando a existência de espaços em branco que delimitam uma palavra. Ao detectar uma palavra essa passa

por uma busca em dicionário. Caso esteja presente, passa a ser analisada morfossintaticamente, ou seja, verifica-se a classe da palavra (verbo, nome, gramema, etc). Esta verificação é feita pela a análise das terminações da palavra invertida. Primeiro tenta-se inferir se a palavra é um verbo. Em seguida um nome e por último um gramema.

Para que uma palavra seja classificada é necessário ativar as regras que definem as várias classes previstas. Estas regras fazem parte do conhecimento Lingüístico e formam um subconjunto das regras construídas por OLIVEIRA, (1990). Ou seja, a FCTL é uma adaptação do sistema desenvolvido por OLIVEIRA (op. cit.). Poder-se-ía utilizar todo o conhecimento desenvolvido para o IDEAL, entretanto, esta adaptação se deve ao fato de não haver necessidade de tratamentos lingüísticos complexos, uma vez que o domínio de aplicação é restrito.

As conceituações ativas no ambiente são as próprias requisições dos usuários. O AGENTE sempre será a máquina ou o usuário, dependendo do momento em que um esteja requisitando algo do outro. Então, como mencionado anteriormente, os AGENTES não são representados e as conceituações são formadas por AÇÃO, OBJETO e DIREÇÃO. As ações devem ser especificadas sempre no infinitivo/imperativo, isto é, o usuário "deve" escrever seus propósitos de ação através de frases como no exemplo: "Mudar o quadrado de X1,Y1 para X2,Y2". As solicitações segundo frases como: "Gostaria que mudasse o quadrado de posição", não são tratadas atualmente pelo protótipo do

ambiente, pois o verbo que indica transformação (mudar) está precedido de um verbo que não expressa transformação (gostar), o que implica em redundância e processamento desnecessário uma vez que a intenção do usuário na primeira frase é a mesma da segunda, ou seja, "mudar o quadrado de posição" (é assim que o ambiente interpretaria a frase em questão). Entretanto, para efeito de uma maior aproximação do homem à máquina, através de melhoramentos significativos na IHM, é perfeitamente possível se inserir muitas regras gramaticais que permitam processamentos lingüísticos mais complexos, lembrando-se entretanto em manter um compromisso ótimo entre o desempenho do software e sua praticidade de uso.

Terminada a análise morfossintático o ambiente passa ao processo de análise semântico-pragmática da requisição, quando procura dar sentido à frase, através de buscas nas fontes de conhecimento previamente treinadas. Se nenhuma semântica for instanciada para a frase em questão, o sistema passa para uma fase pragmática quando então o usuário é chamado a explicitar sua verdadeira intenção. Todas as definições feitas pelo usuário serão armazenadas nas fontes de conhecimento, de maneira que entradas que possuam a mesma intencionalidade sejam automaticamente interpretadas em sessões posteriores. Isto caracteriza uma apreensão de conhecimento, que é fornecido em um alto grau de abstração para o ambiente, através de interações em LN. Como exemplo, suponha-se que a entrada do usuário seja "Enfatizar os quadrados", no sentido de exibir todos os atributos de todos os quadrados no contexto e que o verbo

enfatizar não tenha sido classificado. O ambiente após consultas à base de dados lingüísticos inicia a fase pragmática através de diálogos com o usuário como no exemplo:

```
QUESTÃO > A ação "Enfatizar" ainda não foi classificada.
          Você deseja classificar? (s/n)
USUÁRIO > s
```

```
QUESTÃO > Você deseja:
          1 - classificar entre as ações derivadas?
          2 - classificar entre as ações primitivas?
          3 - abortar a frase?
USUÁRIO > 1
```

```
QUESTÃO > Qual, então, a ação derivada de "enfatizar"?

mover      alterar (posição_origem, posição_destino) do
          (objeto)
criar      gerar (TUDO) do (objeto)
apagar     alterar (cor=0) do (objeto)
exibir     localizar (TUDO) do (objeto)
fazer      gerar (TUDO) do (objeto)

USUÁRIO > exibir.
```

Neste diálogo o usuário classifica o verbo enfatizar como exibir, classificado em sessões anteriores ou na fase de aprendizagem do ambiente. Esta classificação é feita através de uma relação de sinonímia. Logo, enfatizar herda toda a classificação de exibir e o mapeamento em primitivas será sempre igual ao mapeamento do verbo exibir. Observe-se que o conjunto de ações derivadas não restringe-se ao apresentado no exemplo de diálogo, podendo ser tão extenso quanto for a classificação no treinamento.

O processo de interpretação e compreensão da frase do usuário no ambiente, envolve o conhecimento da

linguagem, o conhecimento do domínio de aplicação e o conhecimento das convenções de uso da linguagem em questão (gramática tradutológica). O resultado do processamento da FCTL é armazenado na base de dados do Quadro-Negro, em uma representação intermediária, que será utilizada pela FCTP.

## 4.3 - Fonte de Conhecimento para Tradução em Primitivas - FCTP

Esta fonte de conhecimento executa a tradução final da solicitação para uma representação por primitivas, dentro da filosofia da implementação de SCHMITT (1989), ou seja, a representação obtida é na realidade uma seqüência de procedimentos que executam as tarefas solicitadas dentro do domínio, comparando-se a um sistema de programação automática.

Conforme mostra a figura 4.3, a FCTP é composta por quatro módulos principais. Cada um desempenhando uma função específica que deve ser completada para que outras fontes dentro ou fora deste módulo, realizem suas respectivas tarefas. A seguir comenta-se sobre a funcionalidade dos módulos individuais.

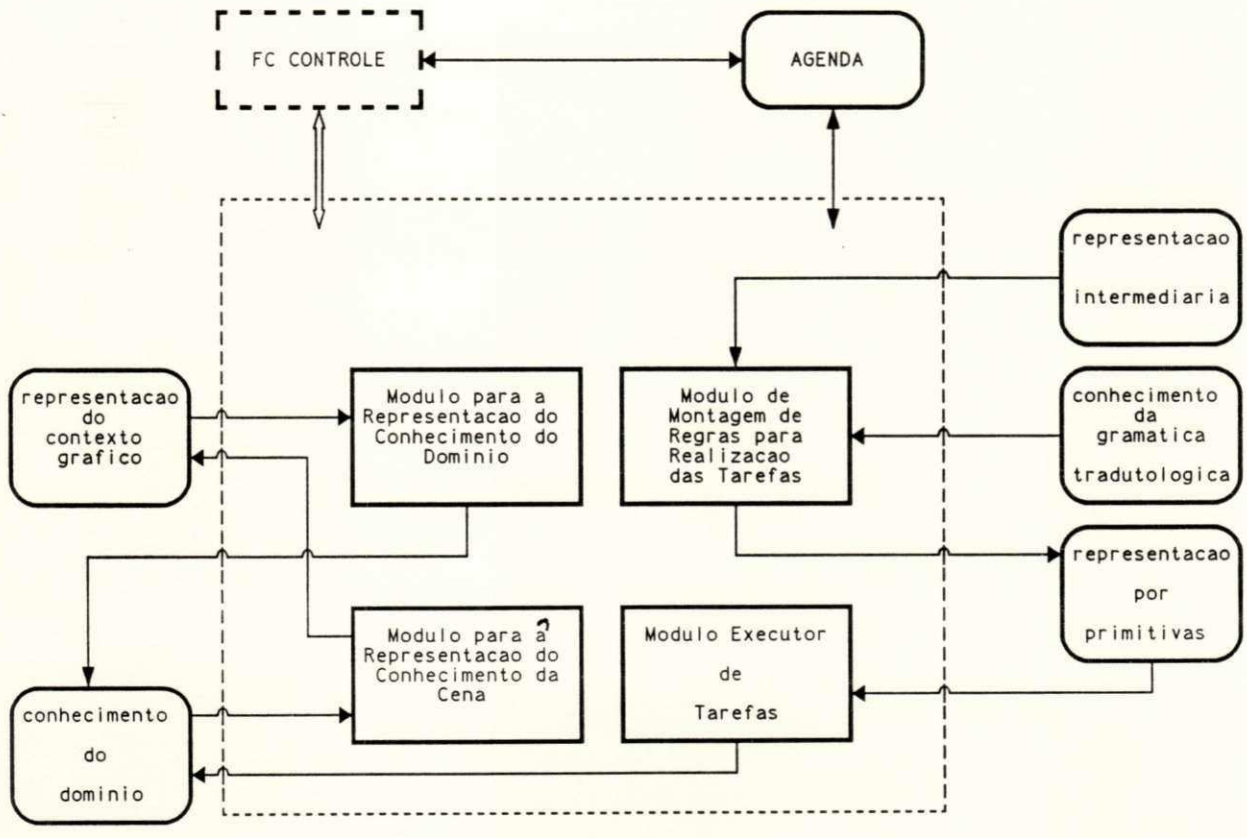



**Figura 4.3 - Arquitetura da FCTP**

O <u>módulo para representação do conhecimento do domínio</u> consulta a base de dados do Quadro-Negro, onde lê os dados armazenados relativos ao contexto sobre o qual o usuário deseja agir. Em seguida monta este conhecimento na estrutura de frames, utilizando a linguagem de frames comentada no capítulo 2 e as descrições dos objetos manipuláveis e das regras que permitam realizar inferências. As descrições dos objetos e das regras são fornecidas pelos usuários especialistas do domínio, através da especificação de arquivos que contêm as informações das

generalizações e instâncias dos objetos, e das regras necessárias para a realização das ações dentro do domínio. A escolha dos arquivos referidos depende do domínio em consideração, podendo ser especificados sempre que este mudar. Os dados do contexto estão armazenados em uma estrutura conhecida previamente definida para permitir compatibilidade entre os módulos de sistema do ambiente que necessitam destas informações. Assim, os dados do contexto são representados em uma estrutura de arquivo, onde cada linha do arquivo é um vetor com 80 caracteres contendo todos os atributos que identificam os objetos em cena. O número de linhas do arquivo traduz o número de objetos na cena observada. A figura 4.4 ilustra a estrutura utilizada, onde cada atributo é delimitado por espaços em branco.

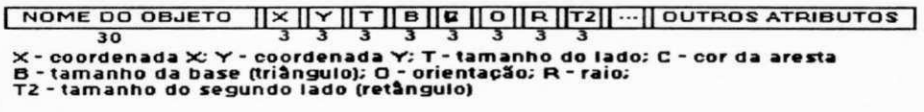



**Figura 4.4 - Estrutura básica do arquivo de contexto**

A estrutura da figura 4.4 é usada na representação de qualquer objeto do domínio presente no contexto, independente dele possuir ou não um determinado atributo. Os atributos que não pertencem ao objeto em consideração são preenchidos com cadeias "***", que serão interpretadas em outros níveis como não existentes. Todos os atributos

possuem o mesmo número de caracteres (3 por atributo) por objeto com exceção do nome que é representado com 30 caracteres independente do tamanho da palavra que identifica os objetos, por exemplo, "quadrado" e "triângulo_isóceles", ambos ocupam o mesmo número de caracteres (30). Para que qualquer outra fonte de conhecimento atue sobre o domínio representado, é necessário que este módulo tenha executado sua tarefa de representação e sinalizado adequadamente para o módulo de controle.

O módulo de montagem de regras para a realização da tarefa, se comporta como um sistema de programação automática e necessita do conhecimento da gramática tradutológica e da representação intermediária gerada na FCTL, significando portanto que sua ativação depende da conclusão da interpretação da solicitação do usuário, o que deve ser informado e verificado pelo mecanismo de controle e agenda implementado. Quando ativo, este módulo interpreta a representação gerada na FCTL e a traduz para uma linguagem de transformação (representada por primitivas de ação -IDENT, LOC, ALT e GEN), resultando em uma representação de conhecimento por primitivas, dentro de uma seqüência lógica de ativação das mesmas.

A seqüência obtida segundo o comentado anteriormente pode então ser usada pelo módulo executor de tarefas da FCTP que é responsável pela execução do que foi requisitado pelo usuário. Este módulo é composto por regras escritas em PROLOG que fazem chamadas às regras primitivas IDENT, LOC,

ALT e GEN, portanto interpretando a representação resultante da ativação do módulo de tradução da FCTP, e realizando o que foi especificado em um nível de abstração alto no âmbito do conhecimento do domínio e do contexto gráfico representado. Entretanto esta execução poderia ocorrer também no âmbito do próprio sistema manipulador do domínio, por exemplo, através do sistema gráfico para o domínio considerado.

Evidentemente que para ser possível executar todo e qualquer tipo de tarefa do domínio, é necessário representar melhor o conhecimento do domínio e definir novas regras e/ou melhorar as existentes. Uma outra consideração é que as regras utilizadas na FCTP dependem do domínio considerado. Assim, o chaveamento de domínios pode exigir a redefinição de todos os conjuntos de regras necessários no novo domínio.

Com a finalização da execução da ação requisitada o ambiente pode utilizar um módulo para representação do conhecimento da cena que atualiza a base de dados do Quadro-Negro relativa ao novo contexto gráfico resultante, que poderá ser usado na montagem final da cena gráfica para apreciação do usuário através da FCMG. Assim, o módulo referido acessa o conhecimento do domínio que está representado, utilizando a linguagem de frames disponível para buscar todas as informações presentes para montar a representação do novo contexto gráfico. Após a leitura dos dados este módulo monta-os em cadeias de caracteres de tamanho fixo, dentro de uma ordem pré-estabelecida conforme

comentado nas considerações do módulo para a representação do conhecimento do domínio. A seguir as cadeias são então representadas na estrutura escolhida para se representar o contexto gráfico.

A arquitetura da FCTP na figura 4.4, adota o modelo de Quadro-Negro onde os módulos comentados sinalizam para o mecanismo de controle, a conclusão ou a necessidade de dados ainda não disponíveis.

## 4.4.- Fonte de Conhecimento para Manipulação Gráfica - FCMG

A arquitetura na figura 4.5 representa a FCMG que permite a manipulação dos objetos do domínio presentes no contexto em análise. Suas funções estão divididas em dois módulos : a) o módulo reconhecedor de contextos e b) o módulo de montagem da contexto.

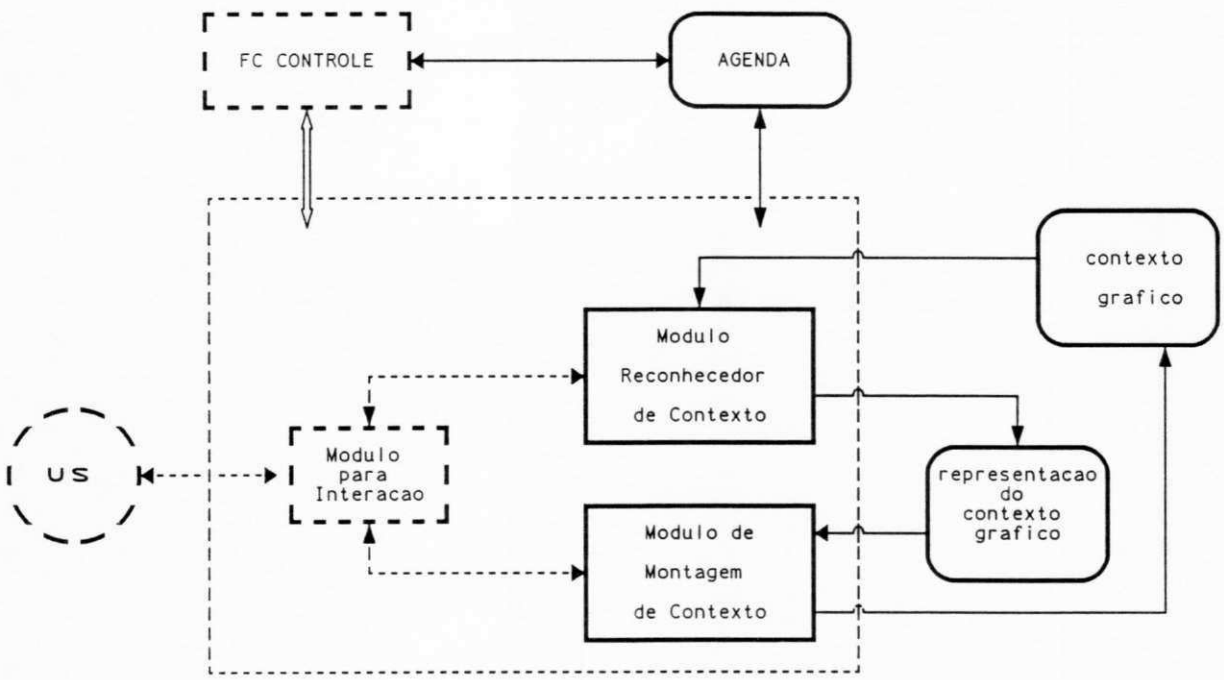



**Figura 4.5 - Arquitetura da FCMG**

O módulo reconhecedor de contextos compõe-se de funções baseadas em conhecimento para o reconhecimento de objetos em tela, usando como heurística a forma e a descrição geométrica dos mesmos. Sua função é identificar os objetos na cena (contexto tela) e seus respectivos atributos. Os parâmetros identificados são então representados em uma estrutura conhecida (representação do contexto gráfico), que permite a compatibilidade com outros sistemas e/ou linguagens.

Além da identificação dos objetos e seus atributos, este módulo poderia ser capaz de identificar também o

contexto gráfico em uso com todos seus atributos, ou seja, um ambiente CGA, VGA, EGAVGA, etc, (inclusive ambiente bem particulares, como o Sistema de Visualização de Imagens e Sistema Geográfico de Informações do Instituto Nacional de Pesquisas Espaciais) com as suas respectivas dimensões. O módulo compõe-se de várias funções cada uma responsabilizando-se pela identificação de um tipo específico de objeto (figura) dentro da cena.

Os dados obtidos neste módulo são concatenados num único vetor de tamanho fixo (80 caracteres), conforme comentado na seção 4.3, e em seguida são representados dentro de uma estrutura de arquivo, onde cada linha designa um objeto do contexto.

O módulo de montagem de contextos responsabiliza-se pela formação da contexto gráfico na tela. Para compor a tela este módulo lê a representação resultante do módulo reconhecer de contextos, onde busca toda a parametrização necessária para atualizar a tela. Para cada objeto (figura) a ser desenhado na tela, tem-se uma função executora. Os dados são lidos na forma de vetores de 80 caracteres. Em seguida estes vetores são segmentados em sub-vetores cada um identificando um atributo do objeto. Obtidos os atributos faz-se um tratamento destes dados para compatibilizá-los com os tipos de dados usados nas funções de montagem dos objetos na tela.

Este trabalho não objetiva o desenvolvimento de um ambiente com as funções comentadas anteriormente.

Entretanto, para efeito de demonstração de factibilidade, implementou-se algumas rotinas com base num sistema desenvolvido por SCHILDT (1989), que implementou funções para montar figuras geométricas 2D no domínio da computação gráfica, junto com funções de reconhecimento destes mesmos objetos no contexto cena. O reconhecimento é feito pela varredura vertical e horizontal da tela, testando se cada ponto está ativo (cor diferente do fundo da tela) ou não. Caso o ponto esteja ativo, guarda-se suas coordenadas e passa-se a análise dos seus vizinhos. Se um vizinho está ativo verifica-se pelo ângulo que figura se está identificando. As funções desenvolvidas podem reconhecer objetos dos tipos quadrado e triângulo. Neste trabalho restringiu-se ao desenvolvimento do módulo de geração de contextos gráficos. Entretanto, pressupõe-se a existência do módulo capaz de realizar a tarefa de reconhecimento de objetos do contexto observado.

O módulo de montagem de contextos gráficos compõe-se de várias funções, cada uma contendo regras de montagens de objetos específicos do domínio no contexto. Assim, objetos como "quadrado", "triângulo_isóceles", "linha", "círculo", etc., são construídos por funções distintas.

A FCMG também incorpora um módulo que permite interações diretas com o usuário, sem necessidade de se entrar no âmbito da representação do conhecimento. Isto significa que o usuário pode a qualquer momento desconsiderar o uso do ambiente proposto, sempre que a tarefa a ser executada não envolver um conhecimento

complexo. Além disso, este módulo explicita que o protótipo do ambiente proposto neste trabalho, realmente está servindo para interfacear o diálogo entre homem-máquina, uma vez que todos os processamentos possíveis podem ser também executados no âmbito da FC que executa ações no domínio.

## 4.5 - O Usuário e o Módulo de Controle

O usuário é considerado como uma fonte de conhecimento pois pode inferir novas informações a partir dos dados que possui ou que analisa, podendo então fazer várias inferências a partir das respostas dadas pelo ambiente durante as interações. Toda e qualquer fonte de conhecimento do ambiente pode interagir com ele através da arquitetura utilizada. Porém, suas respostas e perguntas são todas interceptadas e interpretadas pela FCTL.

O módulo de Controle responsabiliza-se pela ativação dos diversos módulos que compõem o ambiente. Sua existência está associada ao fato de que cada fonte de conhecimento sabe quem deve ser ativado após sua atuação e tem conhecimento para gerar informações que possam efetivar tal ativação. Praticamente este módulo consulta os dados em agenda e avalia se a solução do problema foi alcançada em algum instante, quando então deve sinalizar que uma solução satisfatória foi encontrada, ou passar o controle para um

módulo que pode prover uma solução ou melhorar a que estiver em consideração.

O controle é tido como uma FC porque pode inferir quais as FC's que precisam ser ativadas e em que ordem, a partir da análise dos dados em agenda. Assim, suponha-se que em determinado instante a agenda esteja como no estado 1 da seguinte configuração:

**AGENDA:**

| **Estado 1:** | Estado 2: |
|---|---|
| **(solicitação)** | **US** |
| **FCTL (falta parâmetro) - US** | **FCTL** |
| **FCTP** | **FCTP** |
| **FCMG** | **FCMG** |

O exemplo mostra no estado 1 da agenda a ordem de ativação da FC's e os dados após a entrada do usuário. Se, por exemplo, a FCTL verificar a falta de parâmetros na solicitação, ela anotará em agenda o motivo pelo qual não realizou a tarefa e quem pode contribuir no fornecimento do parâmetro faltoso. A FC de Controle avalia as informações e configura a agenda para o estado 2, onde as FC's que deveriam ser ativadas após a FCTL no estado 1, serão ativadas depois de solucionar-se o problema da falta de parâmetro. Assim, o controle é passado para o US.

# 5 - CONCLUSÃO

Apesar do avanço da tecnologia tornar as máquinas cada vez mais atrativas do ponto de vista de design, os softwares utilizados ainda apresentam dificuldades para os usuários neófitos. Estes necessitam de treinamento antes de conseguir produzir efetivamente, tornando o investimento inicial pouco rentável, a não ser pelo prazer da descoberta.

Os problemas associados aos softwares estão relacionados às metodologias de desenvolvimento que objetivam apenas a execução da tarefa-meta do usuário, sem detectar sua intenção e sem lhe permitir chances de diálogos nos quais suas intenções poderiam ser explicitadas. Como citado no capítulo 1 muitos pesquisadores têm trabalho em métodos que viabilizem a IHM, destacando o uso de menus e ícones, por vezes tentando incorporar os modelos dos possíveis usuários no desenvolvimento das interfaces para melhorar a IHM (CARBERRY, 1988). A IHM, portanto, é freqüentemente feita utilizando-se menus e ícones (AMBLER e BURNETT, 1989) que representam a ação a ser tomada. Entretanto, pouco é dito sobre a seqüência dos ícones e a entrada dos parâmetros que devem ser tomadas para se resolver um determinado problema dentro de um domínio. Assim, os softwares desenvolvidos estão orientados à execução da tarefa-meta do usuário sem permitir que este interaja com o sistema em uso, caso não domine a linguagem. Uma outra consideração é que a modelagem das características para a IHM pode direcionar o

software a um grupo de usuários restrito, não permitindo a popularização do software.

Outro aspecto importante considerado é o compromisso entre o tempo de processamento e a eficiência na resolução do problema. Os sistemas com menus e ícones apresentam vantagens ergonométricas para o usuário, ou seja, menor desprendimento de energia para resolver um problema, entretanto, podem introduzir muitos ciclos de máquina em uma execução. Como exemplo cita-se o uso de ícones em ambientes CAD na Arquitetura no Capítulo 1.

A IHM objetiva manter o diálogo com o usuário em um alto nível de abstração onde este não necessite considerar os detalhes de baixíssimo nível, como a especificação da seqüência de procedimentos a ser tomada ou o formato de entrada de determinados parâmetros para a resolução de um problema. Para isso, a IHM pode ser implementada através do uso da Linguagem Natural que é a forma mais cabal de comunicação com os usuários.

Muitos trabalhos foram desenvolvidos utilizando a LN na IHM, conforme é mencionado no Capítulo 1. Entretanto, na sua grande maioria os sistemas foram desenvolvidos para atender às necessidades específicas de softwares particulares.

A partir dessa problemática, optou-se por construir-se um ambiente em LN numa abordagem cognitiva que

detectasse a intenção do usuário e traduzisse sua solicitação de ação em um conjunto de procedimentos que a executasse de forma eficiente. Neste sentido a IHM ocorreria no mais alto nível de abstração possível, uma vez que a comunicação seria feita através da própria língua do usuário. A figura 5.1 mostra uma visão resumida do ambiente enfatizando o módulo responsável pelo tratamento lingüístico, que é baseado no sistema IDEAL (Oliveira, 1990) desenvolvido para interfacear sistemas especialistas.

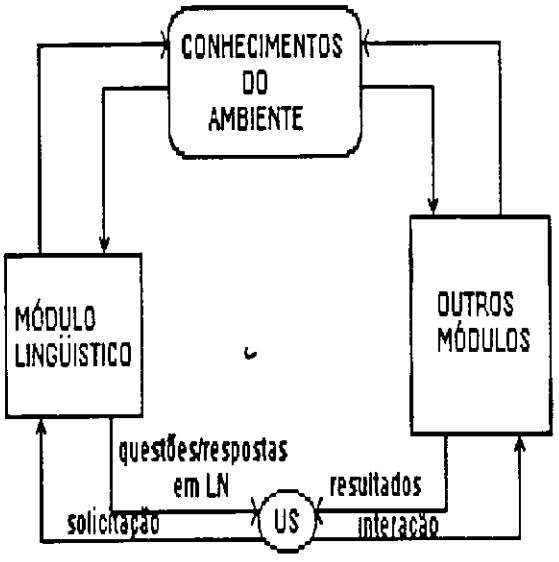



**Figura 5.1 - Visão resumida do ambiente**

Os outros módulos de sistema do ambiente são o sistema computacional que se deseja interfacear e o módulo responsável pelo mapeamento das solicitações de ações na linguagem de transformação composta de primitivas do sistema computacional do domínio. Os módulos referenciados

são na realidade fontes de conhecimento que atuam sobre os dados previamente estruturados. Como as fontes são independentes e atuam sobre bases de dados, por vezes iguais, o ambiente é concebido dentro do modelo de Quadro-Negro e os dados são representados segundo o formalismo da representação por frames.

A arquitetura do ambiente está de acordo com a figura 4.1, compreendendo fontes de conhecimento distintas, como a Fonte de Conhecimento para o Tratamento Lingüístico - FCTL, a Fonte de Conhecimento para a Tradução em Primitivas - FCTP, a Fonte de Conhecimento do Domínio (no caso, Manipulação Gráfica - FCMG), o Usuário e a Fonte de Conhecimento de Controle. A FCTL intercepta e interpreta as entradas do usuário. A FCTP traduz a representação obtida da FCTL nas primitivas de ação de acordo com o conhecimento tradutológico e executa as ações no âmbito do conhecimento. A FCMG é responsável pela manipulação do contexto do domínio em análise.

Além da adaptação do sistema IDEAL, a FCTL incorpora um módulo de apreensão de conhecimento que permite a definição de regras e conhecimento necessários para a correta interpretação da intenção do usuário. Ou seja, através deste módulo o usuário pode especificar, por exemplo, qual o conjunto de primitivas e quais as regras heurísticas que deverão ser utilizadas sempre que o domínio de aplicação muda, bem como toda a gramática tradutológica necessária ao mapeamento para a Linguagem de Transformação. Além disso, este módulo permite que o conhecimento do

domínio seja estendido, pela incorporação de novas regras e dados, de maneira que um maior número de tarefas possa ser realizado.

O ambiente como proposto aqui assemelha-se com o sistema desenvolvido por SCHMITT (1989). Entretanto, diferentemente do proposto, o sistema de SCHMITT (op. cit.) foi concebido para interações em Língua Inglesa, ficando quase que inválido para usuários da Língua Portuguesa, uma vez que se tornaria mais uma linguagem de interação a ser aprendida.

Utilizando sua própria língua de comunicação o usuário pode manter um diálogo com o ambiente, permitindo a detecção de qualquer falha de especificação antes das ações solicitadas serem executadas. Assim, o ambiente só passa a execução após ter corretamente detectado a verdadeira intenção do usuário. Este processo interativo pode ser executado através de ambientes como descritos por AMBLER e BURNETT (op. cit.) ou através de perguntas e respostas em LN.

O ambiente como exposto neste trabalho pode constituir-se em um modelo para o desenvolvimento de interfaces inteligentes, uma vez que a filosofia de desenvolvimento empregada pode ser utilizada para outros domínios, senão o considerado no desenvolvimento do protótipo comentado no capítulo 4.

Pode-se indagar aqui da validade ou necessidade de um ambiente baseado em conhecimento como interface para um sistema que executa tarefas semelhantes àquelas executadas pelo sistema proposto por SCHMITT (op. cit.), ou seja, a detecção de linhas fraturadas numa imagem. Provavelmente, todas aquelas tarefas poderiam ser completamente viabilizadas por uma interface gráfica interativa com ajuda de um "mouse" (e/ou ícones). Entretanto, do ponto de vista ergonômico o usuário desprenderia muito trabalho numa tarefa com alto grau de repetitividade, tornado a tarefa praticamente manual e dependente da acuidade visual, em se tratando de imagens de contraste e brilho baixos, por exemplo. O ambiente em LN aumenta a eficiência na realização da tarefa uma vez que a responsabilidade de execução é da seqüência de primitivas de ação resultante da ativação da FCTP, que não incorpora o elemento fadiga física de um operador, por exemplo.

A vantagem do uso da linguagem natural pode ser ainda explicitada ao comparar-se duas versões de um mesmo sistema, sendo uma com interação por menus e outra com interação com LN.

Para exemplificar tome-se o sistema de Tratamento de Imagens - SITIM, desenvolvido pelo Instituto Nacional de Pesquisas Espaciais - INPE, na forma de interação atual, onde o usuário neófito ou especialista no domínio do Processamento de Imagens, deve especificar cada ação que deve ser executada e fornecer os parâmetros segundo formatos pré-definidos. Assim, supondo uma tarefa onde o

usuário pretende exibir uma imagem na tela de um monitor, ele precisa:

1. Selecionar no menu principal o grupo de funções que manipula imagens. ( apresentam-se submenus ).

2. Selecionar no submenu a função responsável pela apresentação da imagem na tela do monitor.

3. Especificar o nome da imagem desejada.

4. Selecionar bandas ( faixas espectrais ) que devem ser exibidas.

5. Especificar a escala na qual ele deseja exibir a imagem.

Supondo agora o SITIM com interação em LN, utilizando os recursos do ambiente concebido neste trabalho, o usuário passaria a fazer suas solicitações através de sua linguagem natural de comunicação, sendo que os parâmetros seriam fornecidos de forma amigável, por exemplo por meio de diálogos como:

USUÁRIO > Apresente a imagem na escala de 50000.
AMBIENTE > Qual o nome da imagem?
USUÁRIO > <nome da imagem>
AMBIENTE > Quais as bandas?
USUÁRIO > 5,4 e 3.

Vê-se então que o tempo gasto pelo o usuário para especificar sua tarefa é reduzido, uma vez que ele não precisa navegar na estrutura de menus do sistema.

Além desta forma de entrada dialogada, o usuário poderia especificar sua solicitação com uma única frase como:

USUÁRIO > Apresente a imagem X, nas bandas 5,4 e 3, na escala de 50000.

Nesta solicitação o usuário especifica a ação e fornece os parâmetros, sem necessidade de conhecer formatos de parâmetros e seqüências de procedimentos a serem tomadas.

O ambiente como concebido neste trabalho apresenta-se propício para usuários sem experiências no sistema utilizado, mas pode introduzir trabalho desnecessário em se tratando de usuários especialistas que conhecem sistematicamente a maneira como especificar procedimentos e parâmetros para o sistema em uso. Estes usuários especialistas veêm a uso da LN como uma desvantagem. Para então diminuir esta desvantagem dos sistemas que usam LN, ambientes como o concebido, podem também incorporar módulos que permitam escolher entre entradas por LN, por menus ou por ícones.

Uma outra desvantagem no uso de LN está relacionada a sua característica de ser livre, pois, dependendo de sua experiência, o usuário pode demorar-se na classificação de novas possibilidades lingüísticas por considerar-se, às vezes, com pouco conhecimento da língua.

Além dessas considerações, a LN pode não ser adequada para situações onde o uso de outros recursos é claramente melhor. Por exemplo, considere-se um processamento simples para "mover" um objeto 2D de uma posição para outra sem exigir-se precisão na transformação. Nesta situação a LN não se mostra atrativa uma vez que o usuário desprenderia menos tempo se apenas selecionasse a ação iconizada a ser tomada e escolhesse o objeto através de um mouse, deslocando-o em seguida para posição desejada.

Entretanto, há situações em que outros recursos podem ser utilizados, mas que o uso da LN, como utilizada no ambiente concebido neste trabalho, permite menor tempo e complexidade na escolha de procedimentos a serem tomados. Tome-se então, o exemplo de uma transformação que envolve a combinação de várias outras transformações. Por exemplo, suponha-se uma solicitação para "deformar um objeto". Dentro do domínio considerado no trabalho, a utilização de ícones para a transformação anterior é totalmente viável. Contudo a seleção dos procedimentos e a passagem de parâmetros são feitas pelo usuário. Neste caso, faz-se necessário que o usuário conheça a seqüência de transformações e os formatos dos parâmetros a serem fornecidos, o que implica que usuários semi-leigos tomam um tempo maior para especificar as ações do que se usassem o ambiente em LN, que transformaria sua solicitação no conjunto de ações a serem tomadas. Esta observação também é válida para os usuários especialistas, que, dependendo da transformação, precisaria, utilizando ícones ou menus,

repetir a especificação de comandos de ações que devem ser tomadas por um certo número de vezes.

## 5.1 - Considerações sobre o Protótipo

Foi desenvolvido um protótipo do ambiente na linguagem PROLOG tomando-se o domínio restrito da Computação Gráfica em 2D onde fosse possível realizar algumas transformações tais como translação, rotação, apagamento e escalonamento, sobre objetos como quadrados, triângulos, círculos, linhas e retângulos, sem a preocupação de cobrir todo o espectro de ações e objetos do domínio. Assim, as primitivas IDENT, LOC, ALT e GEN podem não ser suficientes para mapear todas as possibilidades de ação do domínio.

O conhecimento do domínio considerado, engloba desde a descrição dos objetos até regras para cálculos de atributos e para a realização das transformações possíveis sobre os mesmos, sendo representado através de estruturas de frames como no capítulo 2. A navegação é feita utilizando a linguagem de frames segundo ARARIBÓIA (1989), adaptada à situação para prover um melhor entendimento das funções. Assim, o usuário pode inclusive utilizar-se da própria, para acessar o conhecimento sem utilizar as heurísticas das primitivas concebidas. A estrutura de frames mostrou-se bastante adequada aos propósitos do protótipo do ambiente uma vez que se tratava da

representação de objetos concretos do mundo real, dentro de um domínio similar ao utilizado por Minsky (WINSTON, 1975) para conceber o modelo. Provavelmente, dependendo do domínio, esta pode não ser a melhor forma de representação do conhecimento tendo, portanto, os projetistas destes ambientes, a tarefa de verificar a adaptabilidade dos formalismos dos vários métodos de representação do conhecimento existentes, às condições do ambiente e necessidades do domínio de aplicação considerado.

Um dos problemas enfrentado durante o desenvolvimento do protótipo esteve relacionado a quantificar o conhecimento necessário para a execução eficiente e efetiva das ações pretendidas, o que deve ser feito por especialistas do domínio, conforme HOFFMAN (1987). Como restringiu-se o domínio, o conhecimento quantificado estava associado à descrição dos objetos manipuláveis, à declaração de regras para realizar as ações no domínio e à descrição do contexto gráfico tomado como referência inicial. Este dimensionamento é difícil porque podem existir diversas maneiras de se abordar um problema para se conseguir soluções satisfatórias, e porque pessoas distintas também enxergam as soluções sob pontos de vistas diferentes, o que implica na impossibilidade de se representar todas as possíveis visões do espaço de soluções de um problema, dependendo da complexidade do domínio. O objetivo do desenvolvimento do protótipo foi mostrar a possibilidade de, dada uma determinada situação dentro de um domínio restrito, realizar tarefas no mesmo com transparência para o usuário na passagem de parâmetros e na

especificação de procedimentos, ou seja, no mais alto nível de abstração possível do ponto de vista do usuário, caracterizando-se em um ambiente de programação automática, onde as regras que executam as tarefas não são pré-programadas, mas construídas durante as interações com usuário e armazenadas pelo ambiente para futuras interações.

A linguagem de programação utilizada para o desenvolvimento poderia ser qualquer além do PROLOG. A escolha do PROLOG deveu-se ao fato do sistema IDEAL, o núcleo do ambiente, existir apenas na referida linguagem. Entretanto, outros módulos foram escritos em linguagem C, como por exemplo a FCMG, onde considerou-se a existência apriori do módulo reconhecedor de contextos gráficos, como discutido no capítulo 4.

No geral, a linguagem natural por se tratar de uma linguagem de comunicação, é muito dinâmica na medida que constantemente novas regras lingüísticas são criadas, dependendo de vários fatores que a norteiam. Caso se deseje aumentar a quantidade de processamentos lingüísticos, para que se trate casos lingüísticos mais complexos envolvendo um conhecimento do domínio mais estendido, haverá a necessidade de maiores espaços de memória para o armazenamento das novas regras. Este fato pode levar a sobrecarga de muitas das máquinas pessoais de hoje. Logo, apesar dos benefícios para usuários sem experiência computacional e no domínio de aplicação, um ambiente em LN pode tornar-se pouco cobiçado para atividades

computacionais, devido à complexidade que pode ser envolvida na formalização do domínio de aplicação e à quantidade de conhecimento necessário para tornar a máquina suficientemente capaz.

O protótipo do ambiente usa o recurso computacional de "overlay" o que facilita o gerenciamento de memória de trabalho. Com ele os módulos do sistema são carregados em memória de acordo com a necessidade do processamento.

Durante a fase de inicialização o módulo gerente (principal) configura as bases de dados lingüísticos e do domínio que serão utilizados na interpretação da solicitação do usuário. Além disso, também se coloca em memória o conhecimento descritivo do domínio onde as tarefas serão executadas.

O protótipo desenvolvido utiliza-se de um interpretador PROLOG que deve ser carregado em memória para que seja possível executar os programas do ambiente. Como o interpretador engloba recursos gráficos e de texto, a memória de trabalho fica reduzida para carregar estes programas. Também durante o processamento a FCTL pode criar, solicitar, armazenar ou modificar muito conhecimento lingüístico durante as sessões. Os resultados destas transformações são armazenados na memória de trabalho antes de serem armazenados definitivamente em arquivos.

Pelo exposto vê-se que o protótipo para sua eficiência máxima necessita de máquinas rápidas com boa capacidade de armazenamento. No desenvolvimento, entretanto, utilizou-se um PC com processador INTEL 80486 com 640 Kbytes de memória e com 3 Mbytes de expansão de memória. Utilizando-se do recurso da expansão de memória o interpretador manipulou eficientemente os diversos módulos do sistema.

O desempenho do protótipo do ambiente, em termos de tempo de resposta, compreendido entre a solicitação e a execução da tarefa, depende de fatores inerentes ao sistema IDEAL ( núcleo da FCTL ), da busca em dicionários e da busca na estrutura que representa o conhecimento do domínio. Os fatores inerentes ao IDEAL são:

1. Exibição de menus, leitura e compreensão do usuário e a decisão de optar.

2. O número de menus necessários para uma dada interação, por exemplo, para a classificação de novas possibilidades lingüísticas.

A busca em dicionários pode ser demorada dependendo da quantidade de casos lingüísticos já tratados. A busca na estrutura que representa o conhecimento do domínio pode ser lenta dependendo da expressividade da representação. Quanto mais expressivo, maior o tempo de processamento para a recuperação da informação, ou seja, menor a eficiência.

Em geral o tempo entre a solicitação do usuário e o mapeamento em primitivas leva de 2 a 18 segundos dependendo do tipo de entrada. A tabela a seguir explicita qualitativamente a relação entre o tipo de entrada e o tempo estimado para seu processamento.

| ENTRADA | TEMPO |
|---|---|
| 1. Ações e objetos já classificados | 02 - 08 segundos |
| 2. Ações não classificadas | 05 - 12 segundos |
| 3. Objetos não classificados | 05 - 12 segundos |
| 4. Ações e objeto não classificados | 10 - 18 segundos |

No caso 1 da tabela o tempo exposto corresponde ao tempo de busca em dicionário. Dependendo do tipo de classificação ( se envolve muitas ações derivadas ), e da forma na qual o dicionário está organizado, o tempo pode extrapolar os 8 segundos considerado como valor máximo para este caso.

Nos caso 2 e 3 o tempo exposto engloba as considerações feitas nos parágrafos anteriores para a FCTL. Nestes casos, o usuário deve definir as ações e os objetos, o que significa sua exposição à menus que devem ser lidos e compreendidos antes de optar.

No caso 4 considera-se a tentativa de classificar ações e objetos, quando o usuário é exposto aos menus, mas classifica de forma incoerente com os conhecimentos lingüísticos e do domínio já definidos. O ambiente faz uma

série de tentativas antes de passar para a fase pragmática, quando explicitamente requisita que o usuário entre com a AÇÃO, o OBJETO e os ATRIBUTOS da tarefa solicitada.

Os tempos considerados foram estimados utilizando-se máquinas do tipo PC com o processador INTEL 80286, tendo-se como usuários uma turma de 20 alunos do curso de Processamento de Dados que, na ocasião, estavam cursando a disciplina CAD (Computer Aided Design). Além de realizar estas estimativas de tempo procurou-se também realizar testes de compreensão dos menus que implementam os diálogos com o usuário. Estes testes mostraram a necessidade de efetuar-se mudanças nas sentenças usadas no diálogo com o usuário.

Os testes de tempo foram também realizados utilizando-se uma máquina do tipo PC com processador 80486 e com usuários neófitos no domínio de aplicação ( no total 4 usuários ). Observou-se que o desempenho do ambiente no tipo de máquina referido apresentou um excelente ganho (em torno de 3 a 5 vezes mais rápido, em relação ao PC com o processador INTEL 80286 ). As reações às interfaces de diálogo foram diversificadas. Um dos usuários preocupou-se com a estrutura gramatical da sua solicitação e esperava respostas e requisições do ambiente de forma semelhante. Os demais limitaram-se ao diálogo entrando com suas solicitações segundo o esperado pelo ambiente, ou seja, especificações de ações sempre no infinitivo ou no imperativo.

Na tabela apresentada anteriormente não se considera o tempo de processamento do sistema interfaceado. Dois sistemas foram testados executando o mesmo tipo de processamento. Um deles utilizou adaptadores gráficos ( do tipo CGA, EGA, VGA, etc. ) e o outro utilizou os recursos gráficos do Sistema de Tratamento de Imagens - SITIM, dispositivo para processamento de imagens e computação gráfica desenvolvido pelo INPE. O sistema com adaptadores gráficos ( do tipo CGA, EGA, VGA, etc. ) mostrou-se mais rápido no processamento. O SITIM apresentou um tempo de processamento longo por engloba características de "software" e de "hardware" que o tornam lento para o objetivo deste trabalho.

## 5.2 - Perspectivas

Ao longo do trabalho enfatizou-se o aqui denominado "Domínio da Computação Gráfica 2D", onde apenas se trabalhou com alguns objetos no plano. O conhecimento do domínio restringiu-se à representação conforme as figuras 3.12 e 3.13. Este, entretanto, pode não ser suficiente para resolver todos as possibilidades de ação necessitando ser melhor estruturado nas aplicações reais.

Conforme comentou-se anteriormente, a metodologia empregada no desenvolvimento do protótipo do ambiente pode ser utilizada em diferentes domínios de aplicação. Contudo, alguns domínios podem apresentar problemas quanto ao que

deve ser representando, por envolverem um conhecimento complexo e, portanto, a representação do conhecimento por frames pode não ser a mais indicada e não haver a necessidade de se empregar o modelo Quadro-Negro. Muitos pesquisadores têm analisado os métodos de representação do conhecimento existentes em diversos domínios. Logo, para que o ambiente proposto neste trabalho seja empregado em diversas aplicações, pode-se tomar os trabalhos destes pesquisadores no tocante a melhor forma de representação. Assim, por exemplo, para se estender o domínio considerado no desenvolvimento do protótipo do ambiente deste trabalho, poder-se-ía tomar as considerações de WALKER et ali (1988) que falam do uso de frames para representar e raciocinar sobre objetos em 3D para a visão. Para o domínio do Processamento de Imagens, poder-se-ía considerar, por exemplo, os trabalhos de FIKES e KEHLER (op. cit), que usam frames para representação de cenas, SILVA e BITTENCOURT (1991) que usam sistemas híbridos de representação dentro de uma abordagem do modelo Quadro-Negro, para a representação de imagens meteorológicas de radar e WOODS et ali (1989) que usam frames no desenvolvimento de um sistema para modelagem e execução de tarefas visuais.

Como comentado a FCTL responsável pelo tratamento lingüístico das solicitações do usuário, tem como base uma adaptação do sistema IDEAL de OLIVEIRA (op. cit). Esta adaptação deveu-se ao fato de não ser necessário fazer processamentos lingüísticos complexos, uma vez que tratava-se de um protótipo de ambiente para demonstração em domínio restrito. Assim, o ambiente foi concebido para o sistema

lingüístico da Língua Portuguesa do Brasil (LPB), onde as solicitações de ação do usuário são feitas em linguagem natural livre. Contudo, apenas algumas regras gramaticais foram representadas para atender algumas das ações do universo de possibilidades citado neste trabalho. Estabeleceu-se que as ações deveriam ser solicitadas sempre no imperativo ou infinitivo, sempre procurando utilizar verbos de LPB que denotem ação. Pois a utilização de outros verbos poderia acarretar em processamento desnecessário, uma vez que a intenção do usuário é a mesma para qualquer uma das frases: "Mudar o quadrado de 30,30 para 40,40" e "Gostaria de mudar o quadrado de 30,30 para 40,40".

Pretende-se, entretanto, estender o processamento lingüístico para o mesmo domínio, para que o ambiente possa interpretar solicitações de ação que contêm elementos referenciais, tais como 'perto de', 'menor que', 'maior que', 'a direita de', 'a cima de', etc.

O protótipo do ambiente foi desenvolvido para computadores do tipo PC com placas gráficas como CGA, VGA, EGA, HÉRCULES, etc., utilizando-se as linguagens PROLOG e C, como já comentado. Entretanto, já se começou o desenvolvimento do mesmo ambiente para Workstations com o sistema operacional UNIX, utilizando-se as linguagens LISP e C.

A metodologia de desenvolvimento empregada poderá ser utilizada com modificações, para conceber as interfaces de um sistema para "Extração de informações em imagens de

sensoriamento remoto: do tratamento ao processamento simbólico", dentro do domínio do Processamento de Imagens ( projeto conduzido pelo Instituto Nacional de Pesquisas Espaciais - INPE ), e do sistema SIPREX ( CAVALHO et ali, 1991 ) para projeto de filtros digitais, no domínio do Processamento de Sinais.

Além destas aplicações pretende-se também desenvolver uma interface baseada no ambiente concebido para um sistema CAD. Nesta interface o usuário especificará seus comandos de construção de objetos através de especificações em LN, livrando-se da necessiade de conhecer a priori toda a linguagem utilizada pelo CAD.

Como perspectiva de trabalho futuro, considerar-se-á o uso de sistemas de reconhecimento de voz como entradas para ambientes como o concebido. Com isso, o usuário especificará suas solicitações de ação através de frases faladas.

Entretanto, a adaptabilidade do ambiente concebido neste trabalho para outras aplicações, deverá ser precedida pela utilização de técnicas de engenharia de software com o objetivo de otimizar o desenvolvimento destes ambientes, quanto a velocidade de processamento, o desempenho das regras, o projeto de interfaces necessárias a interação, etc. Neste sentido a metodologia descrita neste trabalho, deverá ser modificada quando da transformação ou adaptação do ambiente para um produto ou para outras aplicações.

O ambiente como descrito neste trabalho deverá ser modificado nos aspectos seguintes, no caso de uso em domínios diferentes ao considerado no trabalho.

A FCTL sendo parte do sistema IDEAL (OLIVEIRA, 1990), não necessita ser modificada a menos da inclusão de outras regras lingüísticas, que permitam o tratamento de um maior número de casos lingüísticos, o que já é previsto como expansão do protótipo desenvolvido. Além disso, o dicionário lingüístico deve conter as descrições de casos, objetos, ações, primitivas, etc., pertinentes ao novo domínio. O módulo de apreensão do conhecimento, através do qual o usuário treinador declara os conhecimentos lingüísticos, do domínio e da gramática tradutológica a serem utilizados, deve ter sua interface de diálogo alterada para as necessidades do novo domínio. O apêndice B apresenta a interface utilizada neste trabalho, através da qual o usuário treinador classifica as ações e objetos do domínio.

As maiores modificações dever ser realizadas na FCTP, que deve ter todos os módulos redefinidos, pois as primitivas de ação, o formalismo de representação do conhecimento e as regras de inferências, podem mudar de acordo com o domínio.

A FCMG que referencia a fonte de conhecimento numérica do domínio, deve ser o próprio sistema computacional a ser interfaceado. Entretanto, este sistema deve conter um conjunto de primitivas que permitam realizar

as tarefas solicitadas pelos usuários, e que serão utilizadas como parte da gramática tradutológica no mapeamento da solicitação no conjunto de ações primitivas que realizarão as tarefas requeridas.

No estágio atual de desenvolvimento o protótipo não permite o aprendizado de máquina, restringindo-se apenas à "apreensão" do conhecimento. Esta "apreensão" significa o armazenamento em dicionários de casos, das informações fornecidas pelo usuário resultantes da classificação de novas ações e objetos presentes nas solicitações. Estas informações servem como casos para futuras solicitações.

Entretanto, com uma outra perspectiva de trabalho futuro, prever-se a incorporação de um módulo que realize o aprendizado de máquina, permitindo inferências sobre os fatos, regras e situações já armazenadas. Para isso deve-se decidir que método aprendizado utilizar na implementação eficiente deste módulo. Várias métodos estão disponíveis: Aprendizado Aleatório por Redes Neurais, Aprendizado por Hábito, Aprendizado por Ajustamento de Parâmetros, Aprendizado por Analogia, entre outras ( RICH, 1988 ). A escolha de uma destas técnicas deve levar em consideração o formalismo de representação do conhecimento utilizado no ambiente em uma aplicação específica. Contudo, deve-se prever o uso de mais de um método de aprendizado, devido à portabilidade do ambiente para domínios distintos, que podem utilizar formalismos de representação do conhecimento diferentes.

## REFERÊNCIAS BIBLIOGRÁFICAS

**AMBLER, A. L. ; BURNETT**, M. M.. Influence of Visual Technology on the Evolution of Language Environment. *Computer*, Los Alamitos, CA, 22(10): 9 - 22, October 1989.

**ARARIBOIA, G.**. *Inteligência Artificial: Um Curso Prático*, Rio de Janeiro, Livros Técnicos e Científicos Editora Ltda, 1989, 282 p.

**BATCHELOR, B. G.**. Merging the AUTOVIEW Image Processing Language with PROLOG. *Image and Vision Computing*, London, 4(4): <página-página>, November 1986.

**BATES, M.**. Accessing a Database with a Transportable Natural Language Interface. In: 1st CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS. Denver, CO, 1984. Proceedings. Los Angeles, CA, IEEE Computer Society, 1984, pp. 2 - 8.

**BITTENCOURT, G.; MARENGONI, M.**. Uma Ferramenta Funcional para Geração de Motores de Inferência, In: VIII Simpósio Brasileiro de Inteligência Artificial, Brasília, DF, 1991. ANAIS. Brasília, DF, 1991, Universidade de Brasília, pp. 105 - 109.

**BORBA, F. S. ; IGNÁCIO, S. E.**. Critérios para Identificação dos Verbos de Ação e de Processo. *Grupo de Estudos Linguísticos*, Bauru, SP, 10(1): 1 - 3, 1985a.

**BORBA, F. S. ; DEZOTTI, J. D..** Critérios para Identificação dos Verbos de Ação - Processo. *Grupo de Estudos Linguísticos*, Bauru, SP, 10(1): 4 - 5, 1985b.

**BORBA, F. S. ; NEVES, M. H. M. ; DALL'AGLIO, M. M..** Critérios para Identificação dos Verbos de Estado. *Grupo de Estudos Linguísticos*, Bauru, SP, 10(1): 6 - 10, 1985c.

**CARBERRY, S..** Modeling The User's Plans and Goals. *Computational Linguistics*, Rochester, NY, 14(3): 23 - 27, September 1988.

**CARBONELL, J.G.; BOGGS, W.M.; MAULDIN, M.L..** The XCALIBUR Project: A Natural Language Interface to Expert Systems. In: The 8 th INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, Karlsruhe, Germany, 1983. Proceedings. Los Altos, CA, 1983, v. 2, pp. 653 - 656.

**CARVALHO, J.M.; PEQUENO, M.C.; SILVA FILHO, A.M.; HATTORI, M.T..** SIPREX: An Expert System for Digital Filter Design. In: TREIZIEME COLLOQUE GRETSI, Juan-Les-Pins, 1991. ANAIS. Juan-Les-Pins, Septembre, 1991, pp. 945 - 948.

**DODHIAWALA, R.; JAGANNATHAN, V.; BAUM, L.; SKILLMAN, T..** The First Workshop on Blackboard Systems. *AI Magazine*, Menlo Park, CA, Summer: 77 - 80, 1988.

**ERMAN, L.; HAYES-ROTH, F.; LESSER, V.; REDDY, R..** The Hearsay II Speech Understanding System. *Computing Surveys*, New York, NY, 12(2): 213 - 253, June 1980.

**FIKES, R.; KEHLER, T..** The Role of Frame-Based Representation in Reasoning, *Communications of the ACM*, New York, NY, 28(28): 904 - 920, September 1985.

**HAYES-ROTH, B..** A Blackboard Architecture for Control, *Artificial Intelligence*, Amsterdam, Netherlands, 26(3): 260 - 261, July 1985.

**HOEPNNER, N.; CHRISTALLER, T.; MARBURGER, H.; MORIK, K.; NEBEL, B.; O'LEARY, M.; WAHLSTER, W..** Beyond domain-independence: experience with the development of a German language access system to highly diverse background system. In: THE 8th INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, Karlsruhe, West Germany, 1983. Proceedings. Los Altos, CA, 1983, v. 1, pp. 588 - 594.

**HOFFMAN, R.R..** The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology, *AI Magazine*, Menlo Park, CA, Summer: 53 - 67, 1987.

**KASS, R.; FININ, T..** Modeling the User in Natural Language Systems, *Computational Linguistics*, Rochester, NY, 14(3): 5 - 22, September 1988.

**KOBSA, A.; WAHLSTER, W..** Prefácio da Computational Linguistics, *Computational Linguistics*, Rochester, NY, 14(3): 1 - 4, September 1988.

**NAKAMITI, G.S..** BLACKBOARD: Uma visão do Modelo e da Arquitetura, Universidade Estadual de Campinas - UNICAMP. *Manuscrito*. Campinas, São Paulo, 1991.

**NIE, G. L. ; MORIZET-MAHOUDEAUX, P. ; GAILLARD, P..** Application of the AI techniques to signal processing: A knowledge-based-system for digital filters synthesis. *Signal Processing*, Amsterdam, Netherlands, 23(2): 121 - 136, May 1991.

**NII, H. P..** Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, *AI Magazine*, Menlo Park, CA, 7(2): 38 - 53, Summer 1986.

**OLIVEIRA, C. A..** Sobre Interfaces em Linguagem Natural. *Publicação INPE*. INPE-4681 PRE/1376, Agosto 1988.

**OLIVEIRA, C. A..** IDEAL, uma Interface Dialógica em Linguagem Natural para Sistemas Especialistas. *Tese de Doutorado em Computação Aplicada*, INPE-5151-TDL/424, Outubro 1990.

**PASSOS, E. L.**. *Inteligência Artificial e Sistemas Especialistas ao Alcance de Todos*, Rio de Janeiro, Livros Técnicos e Científicos Editora Ltda, 1989, 196 p.

**RAMAN, S.; ALWAR, N.**. An AI-Based Approach to Machine Translation In Indian Languages, *Communications of the ACM*, New York, NY, 33(5): 521 - 527, May 1990.

**RICH, E.**, *Inteligência Artificial*, São Paulo, McGraw-Hill 1988, 563 p.

**SCHANK, R. C.**. *Conceptual Information Processing*, North-Holland, Holland Publishing Company, Amsterdam 1975, 372 p.

**SCHANK, R. ; ABELSON, R.**. *Scripts Plans Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey 1977, 248 p.

**SCHILDT, H.**. *Inteligência Artificial Utilizando Linguagem C*. São Paulo, McGraw-Hill 1989, 349 p.

**SCHMITT, M.**. Mathematical Morphology and Artificial Intelligence: An Automatic Programming System. *Signal Processing*, Amsterdam, Netherlands, 16(4): 389 - 401, April 1989.

**SCHOWENGERDT, R.A.; WANG, H.L..** A General Purpose Expert System for Image Processing, *Photogrammetric Engineering and Remote Sensing*, Bethesda, MD, 55(9): 1277 - 1284, September 1989.

**SILVA, F.A.T.F.; BITTENCOURT, G..** Uma Representação de Conhecimento para a Interpretação de Imagens de Fenômenos Meteorológicos, In: VIII SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, Brasília, DF, 1991. ANAIS. Brasília, DF, 1991, Universidade de Brasília, pp. 83 - 88.

**THOMPSON, C. W..** Recognizing Values in queries or commands in a natural language interface to database. In: 1st CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS, Denver, CO, 1984. Proceedings. Los Angeles CA, IEEE Computer Society, 1984, v. 1, pp. 25 - 30.

**WAHLSTER, W.; MARBURGER, H.; JAMESON, A.; BUSEMANN, S..** Over-Answering Yes-No Questions: Extended Responses in a NL Interface to a Vision System. In: THE 8 TH INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, Karlsruhe, Germany, 1983. Proceedings. Los Altos, CA, August 8 - 12, 1983, v. 2, pp. 643 - 649.

**WEXELBLAT, R. L..** On Interface Requirements for Expert Systems. *AI Magazine*, Menlo Park, CA, Fall: 66 - 78, Fall 1989.

**WINSTON, P. H.**, *The Psychology of Computer Vision*, New York, McGraw-Hill 1975.

**WINSTON, P. H.**, *Artificial Intelligence*, Massachusetts, Addison-Wesley 1979.

**WOODS, P.W.; PYCOCK, D.; TAYLOR, C.J.**. A frame-based system for modelling and executing visual tasks. *Image and Vision Computing*, London, 7(2): 102 - 108, May 1989.

# APÊNDICE A

Este apêndice contém considerações gerais e exemplos dos conceitos da fundamentação teórica exposta no capítulo 2.

## Representação do Conhecimento por Primitivas

Esta representação é feita de acordo com a teoria da Dependência Conceitual (DC) concebida por SCHANK (1975) e como está exposto na seção 2.1.

Tomando as considerações de SCHANK ( op. cit. ), observe-se o exemplo que segue.

Exemplo:

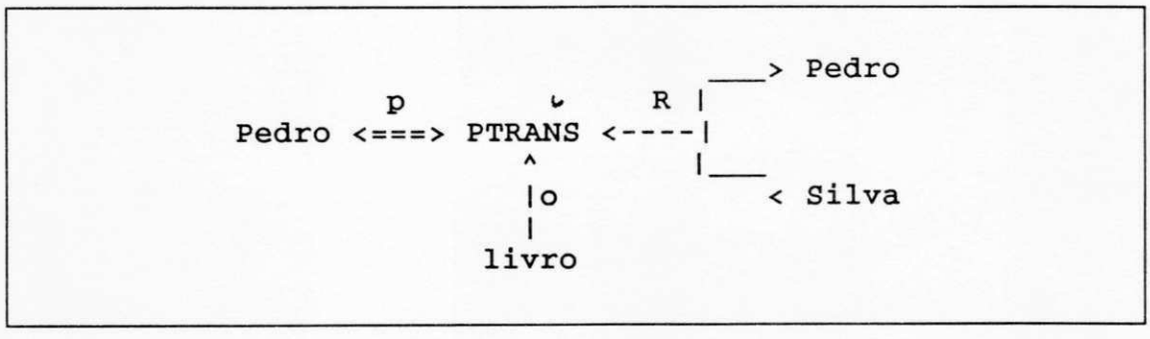
**Pedro comprou o livro de Silva.**

No exemplo a seguir pode-se identificar todos os elementos de uma conceituação ativa:

**Agente -> Pedro**
**Ação -> comprou**
**Objeto -> livro**
**Direção -> de Silva.**

Analisando cada componente da conceituação, verifica-se uma relação entre o **agente** e a **ação** ( o agente age ), conforme a regra 4 na figura 2.2. Observa-se também que **livro** é o objeto da ação, verificando-se, portanto, uma relação conforme a regra 5 na figura 2.2. A ação **comprar** traduz a transferência de posse de um objeto, identificada pela primitiva **PTRANS**, cujos recipientes são **Silva** e **Pedro**, como na regra 2 na figura 2.2.

Utilizando-se das regras na figura 2.2, pode-se representar a conceituação do exemplo anterior como:




O "p" na relação entre Pedro e PTRANS significa que a ação ocorreu no passado.

Representação do conhecimento por frames

A teoria de representação por frames concebida por Minsky ( WINSTON, 1975 ), baseia-se no princípio de que o homem analisa novas situações partindo de generalizações, objetos e situações já vividas. Em particular frames são úteis para representar situações e objetos estereotipadas, como por exemplo uma sala.

Quando se utiliza frames na representação do conhecimento, estes são construídos atribuindo-se aos seus atributos ( slots ), as características pertinentes ao que se está observando.

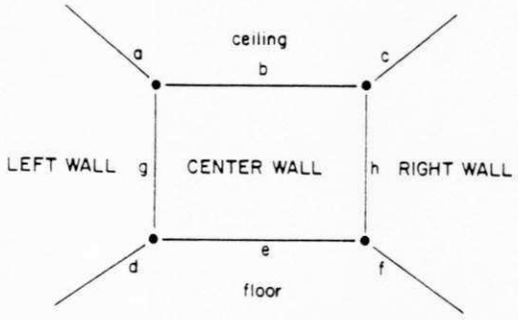
Assim, um frame representando o conceito "sala" deve ser preenchido com os atributos de uma sala, ou seja, número de paredes, número de portas, número de janelas, cor das paredes, tipo de assoalho, cor do teto, etc.. O frame representando o conceito "sala", dentro de um domínio seria preenchido como abaixo:

```
frame: sala

        slot: número de paredes : <inteiro>
        slot: número de portas  : <inteiro>
        slot: número de janelas : <inteiro>
        slot: cor das paredes   : <inteiro>
        slot: tipo de assoalho  : <string>
        slot: cor do teto       : <inteiro>
```

Esta seria uma representação em frame do conceito "sala" dentro de um certo domínio. Em domínio distinto, por

exemplo da visão, a representação do mesmo conceito poderia ser diferente, procurando-se enfatizar aspectos relevantes ao domínio, como por exemplo os aspectos geométricos de uma sala. Logo, o frame representando o "sala" poderia ser como:




```
frame: sala

        slot: parede esquerda: a g d
        slot: parede direita : c h f
        slot: parede central : b h e g
        slot: teto           : a b c
```

O preenchimento do frame anterior parte do fato de que para se preencher, por exemplo, os slots "parede esquerda" e "parede direita", deve-se observar os conjuntos de arestas "a g d" e "c h f" respectivamente.

Observe-se entretanto que o frame anterior representa a visão de um certo ângulo de visada do observador. Mudando-se o ângulo o preenchimento do frame anterior pode ser totalmente alterado.

Outros exemplos da representação do conhecimento por frames estão nas figuras 2.3 e 2.5 na seção 2.2.

# APÊNDICE B

Este apêndice mostra as interfaces com diálogos usados na fase de treinamento inicial do ambiente.

Como o treinamento deve ser feito por especialistas do domínio, a interação neste caso é feita utilizando-se menus. Após carregar os diversos módulos do ambiente em memória, o módulo gerente busca no dicionário a cláusula seguinte que indica se o ambiente já foi treinado anteriormente.

**classificação(sim,_).**

Se a cláusula anterior não for instanciada no banco de dados do interpretador, o controle passa para o módulo de definição apresentando a tela abaixo:

```
QUESTÃO > Você quer:
      (1) - definir ou
      (2) - ver ou
      (3) - apagar ou
      (4) - sair ?

RESPOSTA >
```

Este menu permite que o usuário defina novas ações e objetos, veja o que foi classificado até então, exclua ações e objetos definidos anteriormente e volte ao módulo principal.

Escolhendo-se a opção 2 o usuário verá todas as classificações feitas até então. As apresentações dos dados são feitas através de telas como:

```
Tela 1.

================== ATRIBUTOS E VALORES ==================

[lado : _08EC]
[orientação: _08EC]
[cor : _08EC]
[raio : _08EC]
[posição_atual : _08EC]
[tipo : _08EC]
[base : _08EC
[altura : _08EC]


TECLE ENTER >
```

```
Tela 2.

=================== OBJETO E ATRIBUTOS ==================

quadrado / Atributos -->
     [posição_atual,orientação,cor,lado]
triângulo / Atributos -->
     [posição_atual,orientação,cor,base,altura,tipo]
círculo / Atributos ->
     [posição_atual,cor,raio]


TECLE ENTER >
```

```
Tela 3.

================== PRIMITIVOS DE AÇÃO ==================

[ident, -->, identifica os objeto do domínio]
[loc, -->, recupera os valores dos atributos dos objetos]
[alt, -->, altera os valores dos atributos dos objetos]
[gen, -->, cria novos objetos do domínio no contexto]


TECLE ENTER >
```

```
Tela 4.

=================== AÇÕES PRIMITIVAS ====================

alterar          <atributo> do <objeto>
localizar        <atributo> do <objeto>
gerar            TUDO do <objeto>



========== AÇÕES DERIVADAS DAS AÇÕES ACIMA =============

mover            alterar [posição] do <objeto>
mudar            alterar [posição] do <objeto>
apagar           alterar [cor : 0] do <objeto>
exibir           localizar TUDO do <objeto>
```

```
TECLE ENTER >
```

Ao escolher-se a opção 1 para definir, as mesmas telas são apresentadas ao usuário, que deve confirmar se deseja ou não fazer a classificação. Assim, a Tela 1 é apresentada seguida da mensagem:

**QUESTÃO > Você que definir ATRIBUTOS ? <s/n>**

No caso do usuário responder "s", o ambiente passa a pedir que ele digite o nome do atributo. Em seguida o usuário define que tipo dado deve preencher o atributo referido. É possível classificar os tipos com <inteiro>, <string>, <float> e ainda <funções> que são previstas no formalismo de representação do conhecimento por frames (ligação por procedimentos). Caso o usuário responda que não deseja classificar atributos, o ambiente exibe a lista de objetos e repete a pergunta. Isto prossegue até que o usuário tenha concluído o trabalho de classificação de todos os itens especificados nas Telas de 1 a 4.

Escolhendo-se a opção 3 do menu principal, o ambiente exibe a tela a seguir, e pede para o usuário especifique o tipo de entrada que ele deseja remover da base de dados.

```
Tela 5.

     QUESTÃO > Você deseja eliminar:
           (1) - um primitivo de ação
           (2) - uma ação primitiva
           (3) - uma ação derivada
           (4) - um objeto
           (5) - um atributo
           (6) - NDA ?

     RESPOSTA >
```

Para qualquer uma das alternativas apresentadas no menu Tela 5, o ambiente requisita que o usuário especifique o dado que deseja eliminar.

Observe-se que as interfaces anteriores não são otimizadas. Entretanto, é possível adaptar-se novas formas de diálogos ao ambiente, de maneira que o usuário possa fazer suas definições no mais alto nível de abstração possível, baseado no conhecimento do domínio já representado.

No geral o ambiente concebido sempre terá dois tipos de usuários. O usuário especialista que "sabe que o ambiente não sabe o que ele sabe", e que portanto é o usuário-treinador, e o usuário "leigo" que "sabe que o ambiente sabe que ele não sabe". Portanto, deve-se prever a modelagem do usuário quanto às interfaces apresentadas para o diálogo, que devem conter formas de diálogos as mais didáticas possíveis, para que os usuários "leigos" possam usufruir da potencialidade do sistema, interfaceado sem necessidade de um aprendizado exaustivo deste.

As interfaces apresentadas nas Telas de 1 a 5 devem evidentemente ser alteradas quando da mudança do domínio de aplicação.