UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

CONTRIBUIÇÕES AO ESTUDO MATEMÁTICO DE SISTEMAS INTELIGENTES

Autor: Ricardo Ribeiro Gudwin

Orientador: Prof. Dr. Fernando Antonio Campos Gomide

Tese apresentada ao Departamento de Engenharia de Computação e Automação Industrial da Faculdade de Engenharia Elétrica e de Computação da UNICAMP, como parte integrante dos requisitos para obtenção do grau de Doutor em Engenharia Elétrica

Campinas, 1996

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G934c

Gudwin, Ricardo Ribeiro
Contribuições ao estudo matemático de sistemas inteligentes / Ricardo Ribeiro Gudwin. --Campinas, SP: [s.n.], 1996.

Orientador: Fernando Antonio Campos Gomide
Tese (doutorado) - Universidade Estadual de Campinas Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas inteligentes de controle. 2. Inteligência artificial. 3. Semiótica. 4. * Sistemas orientados a objetos. 5. * Orientação a objetos. I. Gomide, Fernando Antonio Campos. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Agradecimentos

Agradeço a todos aqueles que me incentivaram e apoiaram na elaboração deste trabalho, especialmente minha família: minha mãe, minha avó, meus irmãos, minha esposa Helena, pela paciência e compreensão, meu filho Lucas, por me trazer alegria e realização nos momentos estressantes, meus tios, primos, sogros, cunhadas e cunhado. Agradeço aos colegas de laboratório, que presenciaram todo o desenvolvimento aqui realizado, especialmente ao Maurício Figueiredo, pelas importantes discussões filosóficas que ajudaram a semear este trabalho, e ao João Fabro, que contribuiu gentilmente com algumas simulações que foram utilizadas para efeito de comparação. Agradeço ao Prof. Dr. Márcio Andrade Netto, pelo seu apoio e suporte, que permitiram a minha participação em congressos internacionais importantes para meu desenvolvimento, ao Prof. Dr. Mario Jino e Prof. Dr. Manuel de Jesus Mendes, pelos comentários construtivos e sugestões, principalmente na elaboração do capítulo referente aos objetos matemáticos, e ao meu orientador, Prof. Dr. Fernando Gomide, que sempre esteve presente nos momentos cruciais, dando-me empenho para prosseguir nas etapas difíceis, permitindo a conclusão deste trabalho. Agradeço também ao CNPq por seu suporte financeiro, através de uma bolsa de estudos, que permitiu que eu pudesse me dedicar integralmente ao desenvolvimento desta tese.

Dedico esta obra a todos aqueles que insistem no sonho, e fazem com que este se torne realidade por meio da perseverança e do esforço sincero.

Resumo

Neste trabalho, apresenta-se uma fundamentação para o estudo matemático de sistemas inteligentes. A relação entre semiótica e inteligência é investigada, e utilizada para a geração de estruturas formais de representação do conhecimento. Estas estruturas são utilizadas então para compor os módulos que integram um sistema inteligente.

Inicialmente, apresenta-se uma análise da relação entre semiótica e sistemas inteligentes. Em seguida, inicia-se a fundamentação formal, com a definição matemática de objeto e de redes de objetos. Apesar destas definições serem apresentadas visando seu uso na elaboração das estruturas de representação do conhecimento, elas contém características que permitem denotá-las como fundamentos de uma teoria geral dos objetos que engloba, além disso, sistemas orientados a objetos e programação orientada a objetos.

As definições formais de objetos e redes de objetos são utilizadas então para a definição de estruturas de representação e processamento do conhecimento, para os diferentes tipos de conhecimento identificados a partir da análise entre semiótica e sistemas inteligentes.

Por fim, é analisada uma aplicação-exemplo de um sistema inteligente, construído a partir das idéias anteriormente expostas. Essa aplicação considera o problema do controle de um veículo autônomo.

Palavras Chave: Sistemas Inteligentes, Inteligência Artificial, Semiótica, Engenharia do Conhecimento, Objetos, Sistemas Orientados a Objetos.

Abstract

In this work, we present the foundations for a mathematical study of intelligent systems. The relation between semiotics and intelligence is addressed and it is used to generate formal structures for representing and processing knowledge. Those structures are then used to compose the modules that integrate an intelligent system.

Initially, we present an analysis for the relation between semiotics and intelligent systems. After that, we begin the formal development, with the mathematical definitions of objects and objects networks. Besides those definitions being presented concerning its use in the elaboration of knowledge representation structures, they have characteristics that allow its use as a foundation for a general theory for objects, which addresses as well, object-oriented systems and object-oriented programming.

The formal definitions of objects and objects networks are then used to define the knowledge representation and processing structures, starting from the analysis between semiotics and intelligent systems.

At the end, we analyze an application example of an intelligent system, built within the spirit of the former ideas. This application considers the navigation problem, i.e., the control of an autonomous vehicle.

Keywords: Intelligent Systems, Artificial Intelligence, Semiotics, Knowledge Engineering, Objects, Object-Oriented Systems.

ÍNDICE

1. Introdução

2. Semiótica e Sistemas Inteligentes

3. Fundamentos para uma Teoria dos Objetos

1. Introdução

1.1 Prólogo

A inteligência humana foi sempre um fenômeno que despertou interesse e curiosidade no mundo científico. As primeiras incursões no sentido de compreender e modelar este fenômeno datam dos tempos de Platão (Platão, 1991), com suas especulações, sendo um tema constante na filosofia desde então. Com o advento do computador digital, foi possível passar da especulação à prática. Agora, havia máquinas que poderiam incorporar um comportamento dito inteligente, desde que o modelo desta "inteligência" fosse correto e bem especificado. Turing, foi um dos pioneiros no estudo da relação entre computadores e inteligência (Turing, 1950). Ficou famoso o teste (que leva seu nome), para determinar se um computador (ou programa de computador) poderia ser considerado inteligente ou não.

Na década de 60, fundamentada na lógica matemática nascia a "Inteligência Artificial" (Nilsson, 1980). Nos anos 70 e 80, diversos modelos para o conhecimento humano foram propostos. Newell introduziu seu sistema de símbolos físicos (*physical symbol system*) (Newell, 1982), como uma classe de sistemas que corporifica a natureza essencial dos símbolos. Segundo ele, este seria uma condição necessária e suficiente para a construção de agentes inteligentes generalizados. O conhecimento, segundo Newell, seria de uma natureza superior à dos programas ordinários de computador, devendo ser tratados em um nível mais alto, chamado por Newell de nível do conhecimento (*knowledge level*).

Um dos primeiros modelos de representação do conhecimento que teve repercussão foi o KL-ONE (Brachman & Schmolze, 1985). Em seguida, vieram o SOAR (Laird et.al. 1987; Norman, 1991), o ACT* e o PUPS de Anderson (1989), que pretensamente tratavam-se de arquiteturas para uma inteligência "geral". Em 1989, Collins & Michalski (1989) publicaram o que seria o núcleo de uma teoria para a inferência plausível. Por inferência plausível, eles denominavam todo tipo de inferência que um ser humano poderia efetuar.

Em 1991, Albus (1991) publicou um protótipo do que seria uma teoria da inteligência e Brooks (1991) organizou um manifesto argumentando que para comportamentos ditos inteligentes, não seria necessário nem representação nem inferência.

Mais recentemente, aspectos parciais da inteligência, tais como o raciocínio usando conhecimentos vagos ou incompletos, o aprendizado e a predição vêm sendo estudados na área nascente da "Inteligência Computacional" (Zurada, 1994). Nesta, técnicas como os sistemas e a lógica fuzzy, as redes neurais e os sistemas evolutivos vêm sendo estudados, dando contribuições significativas no entendimento da natureza da inteligência humana.

Paralelamente, nas ciências humanas, também se buscava um modelo para a inteligência e o comportamento inteligente. Dentre outros, vale a pena destacar o

estudo do desenvolvimento da inteligência nos seres humanos, por Piaget (Boden, 1983), e o desenvolvimento da semiótica, por Peirce e Morris (Peirce, 1975; Peirce 1990; Morris, 1947; Morris, 1964; Morris, 1971).

1.2 Motivação

Apesar do grande esforço empregado nas tarefas de desvendar os mistérios que cercam a inteligência humana e as dificuldades intrínsicas na criação de máquinas ou programas de computador que emulem um comportamento inteligente, existem poucas tentativas de se analisar o fenômeno da inteligência de uma forma integrada e organizada. A maioria dos trabalhos encontrados lidam com aspectos bem particulares do fenômeno da inteligência. Um dos poucos trabalhos que se propõem a analisar a inteligência sob um aspecto global, de uma maneira realista, é o já citado trabalho de Albus (1991). Neste trabalho, Albus tenta sistematizar o estudo da inteligência, efetuando uma descrição das diferentes partes que a compõem, de modo que a integração de todas essas partes possam levar a um comportamento inteligente. Entretanto, como o próprio Albus enfatiza, a maioria de suas definições são verbais, e seus teoremas, além de verbais, não apresentam provas. Isso ocorre pela falta de um sistema formal para a descrição dos fenômenos da inteligência. O uso da linguagem verbal para as definições e teoremas significa que não existem, ainda hoje, estruturas matemáticas adequadas para representar o fenômeno da inteligência como um todo. As que existem, estão por demais vinculadas a aspectos particulares do fenômeno da inteligência, sendo inadequadas para uma sistematização globalizada deste.

É neste contexto que se apresenta o trabalho desenvolvido nesta tese. O objetivo primeiro deste trabalho é o de criar fundamentos matemáticos que possam ser utilizados, no futuro, para a elaboração de uma teoria formal da inteligência. Não se pretende aqui, desenvolver uma teoria geral da inteligência. Esta seria uma meta muito ambiciosa, principalmente observando-se que uma grande parcela do que se entende por inteligência ainda não pode ser explicada pela ciência. O que se pretende aqui é propor subsídios a modelos matemáticos, para tratar classes de sistemas inteligentes.

Dentre os modelos formais e não-formais analisados, observou-se uma perspectiva nos trabalhos de Peirce e Morris versando sobre a semiótica. Apesar de carecer de um fundamento matemático adequado (Morris tenta esboçar algo como uma formalização em (Morris, 1971)), a semiótica tem uma concepção que, em princípio, se mostra adequada no fornecimento de subsídios para uma teoria geral da inteligência.

Um dos pontos levantados nessa tese, é a identificação de quais seriam esses subsídios. Nessa perspectiva, desenvolve-se o que poderia ser chamada de uma "semiótica matemática", onde algumas das principais idéias da semiótica são formalizadas pela criação de modelos matemáticos que as representem. O que se almeja, é que essa semiótica matemática possa prover os fundamentos matemáticos necessários para se modelar o fenômeno da inteligência.

Durante o desenvolvimento teórico nesta direção, tornou-se premente a necessidade de um modelo matemático para o que é conhecido hoje como sistemas orientados a objeto. Como não existia um modelo formal adequado aos nossos propósitos, foi necessário desenvolver um ferramental matemático para tratar dessa

questão. Assim, como um sub-produto desta tese, têm-se também os fundamentos básicos para uma teoria matemática dos objetos. Como a ênfase deste trabalho é na modelagem de sistemas inteligentes, seguiu-se, na modelagem dos objetos, somente até o ponto em que foi necessário para seu uso no modelo computacional de inteligência. Para abranger um escopo mais amplo, que incorpore os sistemas orientados a objeto como um todo, serão necessárias extensões que, apesar de indicadas (vide capítulo 3), não são desenvolvidas neste trabalho.

1.3 Estrutura da Tese

A estrutura da tese é a seguinte. No capítulo 2, é apresentada uma análise da relação entre semiótica e sistemas inteligentes. São avaliadas as potenciais contribuições da semiótica no estudo dos sistemas inteligentes. Apesar de baseadas na semiótica, essas contribuições são colocadas dentro da terminologia da inteligência artificial. É feita uma comparação entre a tríade básica da semiótica, envolvendo signo, objeto e interpretante, com uma tríade no contexto da inteligência artificial envolvendo o modelo de representação, o fenômeno e o conhecimento. São identificados três tipos básicos de conhecimentos, chamados de conhecimentos remáticos, dicentes e argumentativos, baseados em uma das diversas tríades descritas por Peirce (1990), que identifica três tipos de signos: remas, dicentes e argumentos. Esses tipos de conhecimento são analisados e classificados, organizando uma taxonomia dos tipos de conhecimento. Por fim, coloca-se uma outra classificação dos tipos de conhecimento, de acordo com sua utilização prática. Nesse caso, os conhecimentos podem ser chamados de designativo, quando apenas descrevem um determinado fenômeno, apraisivo, quando fazem uma apreciação, uma avaliação de um determinado conhecimento para o fim de se atingir um determinado objetivo, e prescritivo, quando efetivamente atuam sobre o ambiente. Essa classificação é inspirada nos interpretantes designativo, apraisivo e prescritivo descritos por Morris (1971).

No capítulo 3, são desenvolvidos os elementos básicos para uma teoria dos objetos. Nesse capítulo, a partir da teoria de conjuntos, desenvolve-se um ferramental teórico com a finalidade de modelar o que seriam os objetos (no sentido de sistemas orientados a objetos). É feita uma apreciação das características dos sistemas orientados a objetos, e como essas características são mapeadas no modelo matemático desenvolvido. Descrevem-se os sistemas de objetos e as redes de objetos, que são colocadas como um tipo especial de sistemas de objetos. As redes de objetos são então analisadas, por meio de exemplos, sendo sugeridos alguns mecanismos de análise do comportamento de tais redes. Por fim, é desenvolvida uma discussão sobre o modelo proposto e outras tentativas de se modelar tais tipos de sistemas. São apontadas as extensões ao modelo de modo que este possa abranger um escopo maior das aplicações ditas orientadas a objeto, e é feita uma comparação entre as redes de objetos e as redes de Petri, mostrando-se que as redes de Petri podem ser vistas como um tipo especial de redes de objetos, sendo que as redes de objetos apresentam um comportamento adaptativo que não pode ser modelado em termos de uma rede de Petri. Sendo assim, estas têm um poder de representação maior do que o das redes de Petri.

No capítulo 4, apresenta-se um modelo formal para as estruturas de representação do conhecimento introduzidas no capítulo 2. Para cada um dos tipos de conhecimento: conhecimento remático sensorial específico, conhecimento

remático sensorial genérico, conhecimento remático de objeto específico, conhecimento remático de objeto genérico, conhecimento remático de ocorrência específico, conhecimento remático de ocorrência genérico, conhecimento dicente e conhecimento argumentativo, é desenvolvido um modelo formal para sua representação.

No capítulo 5, desenvolve-se um exemplo de aplicação, a navegação de um veículo autônomo. Nesse capítulo, é feito um histórico do problema, analisando-se as diversas abordagens de solução encontradas na literatura. Em seguida, faz-se uma descrição mais elaborada do problema, e desenvolve-se uma proposta de solução, baseada no modelo desenvolvido no capítulo anterior. Apresenta-se então, os resultados dos experimentos de simulação realizados.

No capítulo 6, por fim, tem-se a conclusão, onde analisam-se as principais contribuições e a indicação de trabalhos futuros.

O capítulo 7 contém as referências bibliográficas utilizadas.

1.4 Resumo

Neste capítulo, apresentou-se uma caracterização da tese. Fez-se um breve histórico sobre o estudo do fenômeno da inteligência e, em seguida, colocaram-se as principais motivações que originaram o desenvolvimento deste trabalho. Por fim, a estrutura da tese foi descrita, evidenciando resumidamente o conteúdo de cada capítulo.

O próximo capítulo procede com uma análise entre a semiótica e os sistemas inteligentes. Diversos tipos de conhecimento são identificados e organizados em uma taxonomia do conhecimento, de uma maneira ainda informal, tendo-se como inspiração a semiótica de Peirce e Morris.

2. Semiótica e Sistemas Inteligentes

2.1 Introdução

Neste capítulo, apresentaremos alguns conceitos fundamentais no estudo dos sistemas inteligentes. Sistemas inteligentes são, em princípio, sistemas que apresentam um comportamento análogo ao dos seres humanos, no que se refere a sua capacidade de interagir com um ambiente que nem sempre é de todo conhecido e que pode ter suas características modificadas em função do tempo. A designação *sistema inteligente*, entretanto, é por demais genérica, podendo incluir desde sistemas que emulem aspectos parciais da inteligência humana, tais como o uso de conhecimento heurístico, bem como sistemas mais complexos, dotados de mecanismos de adaptação, aprendizado e predição. O fato é que o termo **inteligência** é vago e amplo, permitindo inúmeras interpretações. Sendo assim, qualquer sistema que apresente alguma característica que possa ser rotulada de inteligente, em princípio é um candidato a ser chamado de sistema inteligente.

O fenômeno da inteligência vem sendo estudado a muito tempo, tanto no âmbito das ciências humanas, quanto nas ciências exatas. Especificamente no campo da ciência de computação, diferentes aspectos da inteligência vêm sendo modelados e utilizados na implementação de sistemas computacionais com características inteligentes, notadamente na área de **inteligência artificial** (Barr & Feigenbaum, 1981a, Barr & Feigenbaum, 1981b, Cohen & Feigenbaum, 1981, Barr, Cohen & Feigenbaum, 1981).

Nas ciências humanas, o fenômeno da inteligência e do conhecimento vêm sendo estudado e sistematizado em uma disciplina chamada **semiótica** (Peirce, 1990; Peirce 1975; Coelho Netto, 1980; Eco, 1980). Fundamentalmente, a semiótica estuda os aspectos básicos dos fenômenos da **cognição** e da **comunicação**. A cognição trata da apreensão e compreensão dos fenômenos que ocorrem no ambiente. A comunicação trata de estudar como os fenômenos apreendidos e compreendidos podem ser transmitidos entre os seres inteligentes. A estrutura básica que é utilizada para esta tarefa é denominada **signo**, ou **representâmen**, sendo definido como qualquer coisa que, sob certo aspecto ou modo, representa algo para alguém (Peirce, 1990). A semiótica, portanto, estuda como os signos são formados, como representam os diferentes aspectos dos fenômenos e como podem ser utilizados para o armazenamento e transmissão de informação.

Apesar de uma motivação comum, a inteligência artificial e a semiótica, como disciplinas científicas, seguiram desenvolvimentos distintos. A inteligência artificial seguiu pelo caminho de criar estruturas matemáticas que emulassem características particulares da inteligência. A idéia fundamental era originar sistemas computacionais que exibissem um comportamento que pudesse ser chamado de inteligente. A semiótica, por outro lado, seguiu pelo caminho de identificar, classificar e sistematizar as diferentes características que, em conjunto, podem ser chamadas de inteligência. Para tal, cada disciplina adotou um diferente tipo de modelo. O modelo usado na inteligência artificial é o modelo formal, ou modelo

estrutural, onde se identificam estruturas matemáticas que incorporem o comportamento inteligente. O modelo utilizado na semiótica é o modelo descritivo, onde o comportamento inteligente é descrito de um modo mais subjetivo, utilizando conceitos e nomenclaturas que não encontram um suporte imediato na matemática. Dada essa diferenciação, obteve-se que os modelos originados na inteligência artificial são formalmente mais exatos e precisamente definidos, ao passo que os modelos provenientes da semiótica são mais intuitivos e vagos. A contrapartida é que os modelos oriundos da semiótica conseguiram avançar mais profundamente na explicação dos fenômenos que compõem a inteligência, ou seja, são capazes de explicar mais facetas da inteligência do que os modelos da inteligência artificial. É possível fazer um mapeamento entre os modelos de ambas disciplinas, sendo que para alguns modelos da semiótica, não se encontra um análogo nos modelos da inteligência artificial. Apesar disso, mesmo na semiótica não se encontra uma sistematização adequada dos diferentes modelos de modo a gerar um modelo unificado e integrado de inteligência. Neste capítulo, apresentaremos o que se pretende ser uma tradução dos principais modelos para os fenômenos da inteligência oriundos da semiótica. Inicia-se com com uma análise do fenômeno da cognição, seguida da introdução de alguns conceitos fundamentais da semiótica, tais como a trilogia signo-objeto-interpretante e os diferentes tipos de conhecimento associados aos diferentes tipos de signo. Em seguida, é feita uma proposta de sistematização destes conceitos em uma taxonomia do conhecimento.

2.2 Cognição e Semiótica

Nesta seção, efetuaremos uma análise dos fenômenos que se denominam genericamente por **cognição**. Para tal, utilizaremos alguns termos e conceitos empregados na semiótica, tendo em vista seu uso dentro da teoria de sistemas, de modo a modelar os elementos básicos que determinam o fenômeno cognitivo. As definições aqui colocadas, inspiram-se na cognição humana, mas referem-se, em princípio, a um tipo genérico de cognição, visando sistemas computacionais.

Iniciaremos com uma descrição filosófica do que se entende por um sistema cognitivo, ou seja, qual a sua natureza. Descreve-se portanto o seguinte cenário. Existe um mundo, ou ambiente, que é povoado por objetos. Estes objetos, que podem ser criados ou destruídos, são caracterizados por seus atributos, os quais podem ser modificados ao longo do tempo. Um sistema cognitivo é um sistema que, a partir de uma interface de entrada, consegue identificar objetos do mundo, e modelá-los por meio de uma representação interna ao sistema. Do mesmo modo, ele deve detectar uma modificação nos atributos destes objetos, a criação de novos objetos e a destruição de objetos já existentes, representando estas modificações no modelo.Além disso, a partir de seus modelos internos, um sistema cognitivo pode planejar uma alteração na configuração dos objetos do mundo (o que pode corresponder a criar novos objetos, destruir objetos existentes ou modificar atributos de objetos existentes) e efetivamente atuar sobre o mundo (ambiente), por meio de sua interface de saída, implementando as modificações planejadas.

O único meio pelo qual um sistema cognitivo pode tomar ciência de um determinado objeto do mundo, é por intermédio de sua interface de entrada. Observe, entretanto, que a interface de entrada não corresponde a um mapeamento completo do mundo, e sim a um mapeamento parcial deste. Este mapeamento é limitado por condições tais como o número e tipos de sensores, os limites dos valores mensuráveis pelos sensores e o próprio modelo utilizado para representar os

objetos, que pode ser mais ou menos detalhado. Além disso, o sistema cognitivo também é um objeto do mundo, devendo extrair o conhecimento de si próprio também por meio de seus sensores. Este conhecimento de si próprio como objeto, estará sujeito às mesmas limitações encontradas na representação dos outros objetos do mundo. Devido a todos estes fatores, o processo pelo qual os objetos do mundo são modelados em representações internas ao sistema cognitivo não é único, nem exato. Um sistema cognitivo tenta identificar objetos a partir dos dados sensoriais que adentram sua interface de entrada, sem entretanto ter uma garantia que estes objetos existem de fato no mundo real. É como se ele estivesse naturalmente preparado para identificar objetos do mundo, mas sem uma garantia explícita que os objetos representados internamente correspondam de fato a objetos do mundo. O fenômeno pelo qual um sistema cognitivo julga que reconheceu um objeto do mundo e, portanto, evocou uma representação interna de objeto é chamado de **interpretação**. Uma interpretação ocorre, quando se identifica uma representação interna de objeto, tanto a partir de dados sensoriais como a partir de outra representação interna de objeto. A fonte de informação que deu origem à interpretação é chamada de **signo**, sendo que a representação interna de objeto que é evocada recebe o nome de **interpretante**. Observe que os dados sensoriais que adentram a interface de entrada podem constituir um signo, mas um interpretante também pode ser um signo, na medida que ele gere outro interpretante, ou seja, outra representação interna de objeto distinta de si própria. Com isso, um único signo oriundo da interface de entrada pode gerar uma cadeia de interpretantes, identificando múltiplas facetas do mundo real.

A tripla (signo, objeto, interpretante) modela portanto o fenômeno básico da cognição. Esta tripla é chamada de um processo sígnico, ou *semiosis* por Morris (Morris, 1971). Observe algumas particularidades do modelo. Nem sempre o objeto de um processo sígnico corresponde de fato a um objeto do mundo real, mas sim a um objeto presumível no mundo real. Uma vez que o sistema cognitivo não tem um conhecimento prévio de quais são os objetos do mundo real, estes objetos são somente presumíveis, sendo representados por seus interpretantes. Um fenômeno cognitivo muito comum que exemplifica esta situação é quando alguém olhando para as nuvens do céu identifica objetos que efetivamente não existem. O sistema cognitivo humano presume que exista um objeto formado pelo contorno das nuvens e cria um interpretante para ele. Efetivamente, o objeto não existe. Isso não impede, entretanto, o sistema cognitivo humano de interpretá-lo. Ou seja, a existência real do objeto não é necessária para que o fenômeno da interpretação ocorra O sistema cognitivo está preparado para detectar objetos, mesmo onde eles não existem.

Outra questão importante do fenômeno cognitivo é sua característica dinâmica. É possível analisar o processo cognitivo como um processo estático. Neste caso, considera-se que um modelo do mundo real já existe previamente no sistema e as informações da interface de entrada são confrontadas com esse modelo estático de modo a gerar as decisões de controle. Entretanto, esta visão é por demais simplista. Um sistema cognitivo é, essencialmente, um fenômeno dinâmico. Se a partir da informação da interface de entrada não é possível identificar nenhum objeto representado pelo modelo, o sistema cognitivo cria uma nova representação interna (um novo interpretante), de modo a justificar a informação oriunda dos sensores. Esse processo de criar novas representações internas para novos objetos que vão sendo identificados é essencialmente dinâmico e se caracteriza por sua característica de adaptação e aprendizado. Com isso, um sistema cognitivo pode iniciar com muito pouca informação a respeito do mundo real, e gradativamente ir

construindo um modelo deste mundo. Este modelo é continuamente incrementado com a geração de representações internas para cada um dos objetos que o sistema presume existir, a medida que o sistema cognitivo interage com o ambiente. Basta que o sistema cognitivo tenha uma estrutura que permita uma modelagem consistente do objeto, e que os métodos de aprendizagem sejam utilizados.

A trilogia básica da semiótica, ou seja, a tripla (signo, objeto, interpretante), pode ser mapeada nos modelos da IA, também como uma tripla, conforme indicado na figura 2.1:

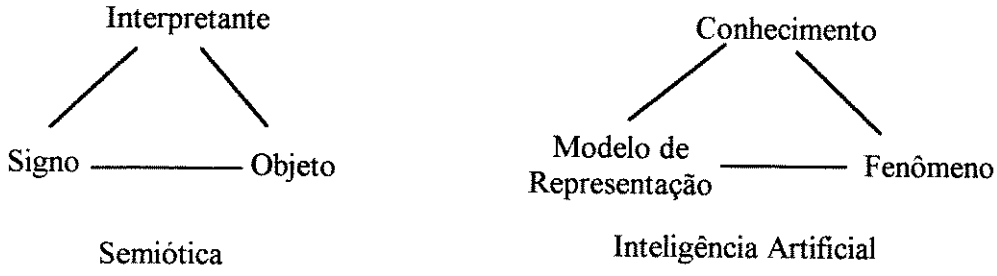


Figura 2.1 - Mapeamento entre a Semiótica e a Inteligência Artificial.

Observe a correspondência entre as triplas. Na tripla da semiótica, o signo é utilizado para representar o objeto, cuja compreensão por uma mente inteligente corresponde ao interpretante. Ou seja, o interpretante é a intelectualização do objeto. Na tripla da IA, um fenômeno do ambiente (que corresponde ao objeto da tripla da semiótica), interpretado como um conhecimento a respeito do ambiente, é representado por um modelo de representação do conhecimento, que corresponde ao signo. Na semiótica, o conceito de objeto não se limita a uma descrição de objetos físicos, ou que possam ter uma abstração como um objeto físico. Esse conceito é muito mais amplo, englobando objetos físicos, abstrações de objetos físicos, ações, modificações, verdades, raciocínios, etc. A melhor tradução para a idéia de objeto, em sua versão semiótica, é exatamente a idéia de fenômeno. Ele pode corresponder a um objeto físico (e suas abstrações), mas também pode se referir a modificações ou propriedades em tais objetos, ou ações realizadas por estes objetos, ou ainda a verdades relacionadas com tais objetos. Sendo assim, para evitar confusões no uso do termo objeto, a partir de agora, quando se referir ao termo "objeto", este designará uma abstração de um objeto físico, empregando-se explicitamente o termo "objeto físico" quando não se tratar de uma abstração mas a um objeto real do ambiente. Para se referir ao que na semiótica é entendido como objeto, utilizar-se-á explicitamente o termo "fenômeno".

Apesar da existência dessa analogia entre a tripla (signo-objeto-interpretante) e a tripla (modelo de representação-fenômeno-conhecimento) ser ilustrativa, nem todo o significado da primeira é traduzido na segunda. Por exemplo, a idéia que existe na semiótica de que um determinado interpretante pode ser um signo para outro interpretante não está representada na tripla (modelo de representação-fenômeno-conhecimento). Para representar esta idéia no âmbito da IA, precisaremos utilizar alguns artifícios que não são necessários quando se usa um modelo descritivo, mas que o são em um modelo estrutural (formal). Quando se diz que um determinado interpretante é um signo para outro interpretante, ele está expressando duas naturezas diferentes. Na primeira, como interpretante, ele corresponde a uma semântica, uma intelectualização de um fenômeno. Na segunda,

como um signo, ele ganha um caráter estrutural, que é utilizado para a geração do segundo interpretante. Essa dupla natureza do interpretante pode ser uma fonte de confusão quando se tenta especificar a trilogia (signo-objeto-interpretante) por meio de modelos formais. O modelo semiótico não é suficientemente claro quanto a uma especificação pragmática de como um elemento denominado signo dá origem a outro elemento chamado interpretante, e como esse interpretante passa a agir como um signo para um outro interpretante, principalmente quando estes elementos se encontram todos na mente do ser cognitivo. Em um modelo descritivo essa transição é simplesmente encarada como "natural". Em um modelo formal, é necessário apontar como se dá essa transição de signo para interpretante, ou pelo menos, de interpretante para um novo signo. Para contornar esses pontos, a idéia original da tripla signo-objeto-interpretante necessita de algumas alterações em sua transição para um modelo formal. O que se propõe é que uma determinada estrutura, que implicitamente representa um determinado conhecimento, seja transformada em uma outra estrutura, representando um outro conhecimento. Observe que com essa caracterização, a ligação entre signo e interpretante (ou estrutura-conhecimento) continua sendo uma coisa "natural", assim como na semiótica. E o problema onde um interpretante passa a ser um signo para um outro interpretante é resolvido por meio dessa transformação, que a partir do primeiro conhecimento (representado por uma estrutura), dá origem a uma nova estrutura (novo conhecimento). Com isso, consegue-se representar a natureza dual do interpretante, sem comprometer a analogia entre os dois modelos. Essa função de transformação é implementada por um tipo especial de conhecimento, chamado de conhecimento argumentativo, descrito na seção 2.3.3.

2.3 Tipos Elementares de Conhecimento

Apesar da denominação genérica "fenômenos do ambiente", na verdade esses fenômenos não são todos de uma mesma natureza. As diferentes naturezas dos fenômenos do ambiente vão dar origem a diferentes tipos de conhecimento. Os diferentes tipos de informação sobre o mundo real, representados por diferentes estruturas matemáticas, são chamados de tipos de conhecimento. Sistemas menos sofisticados irão abranger apenas um escopo limitado de tipos de conhecimento, ao passo que sistemas mais sofisticados terão uma abrangência maior. Nesta seção, faremos uma breve taxonomia dos tipos elementares de conhecimento, ou seja classificando-se os diferentes tipos de conhecimento sem se considerar sua aplicação ou finalidade em um sistema cognitivo. Não se pretende que essa sumarização seja completa, mas suficiente para que conhecimentos de diversos tipos sejam modelados. Na próxima seção, esses tipos elementares de conhecimento serão utilizados para modelar tipos de conhecimento mais sofisticados, onde presume-se uma finalidade no sistema cognitivo para cada conhecimento, gerando outra classificação.

Na acepção original, essa taxonomia do conhecimento é encontrada na forma de uma taxonomia de diferentes tipos de signos (Peirce, 1990). Pelo modelo da semiótica, esses signos estão diretamente acoplados a seus interpretantes. Sendo assim, a taxonomia dos signos induz uma outra taxonomia equivalente no espaço dos interpretantes. Na transição para um modelo na IA, é mais conveniente especificar essa taxonomia em termos de seus interpretantes ou seja, especificando-se diferentes tipos de conhecimento, que por sua vez darão origem a diferentes modelos de representação, ao invés de se seguir a convenção colocada na semiótica,

e organizar uma taxonomia dos diferentes modelos de representação. Essa preferência por classificar os tipos de conhecimento e não os modelos de representação decorre do fato de que, para um determinado tipo de conhecimento, podem existir diferentes modelos formais que o representem. Isso não ocorre na semiótica, pois a definição dos signos não é formal.

A taxonomia dos tipos elementares de conhecimento está ilustrada na figura 2.2 a seguir:

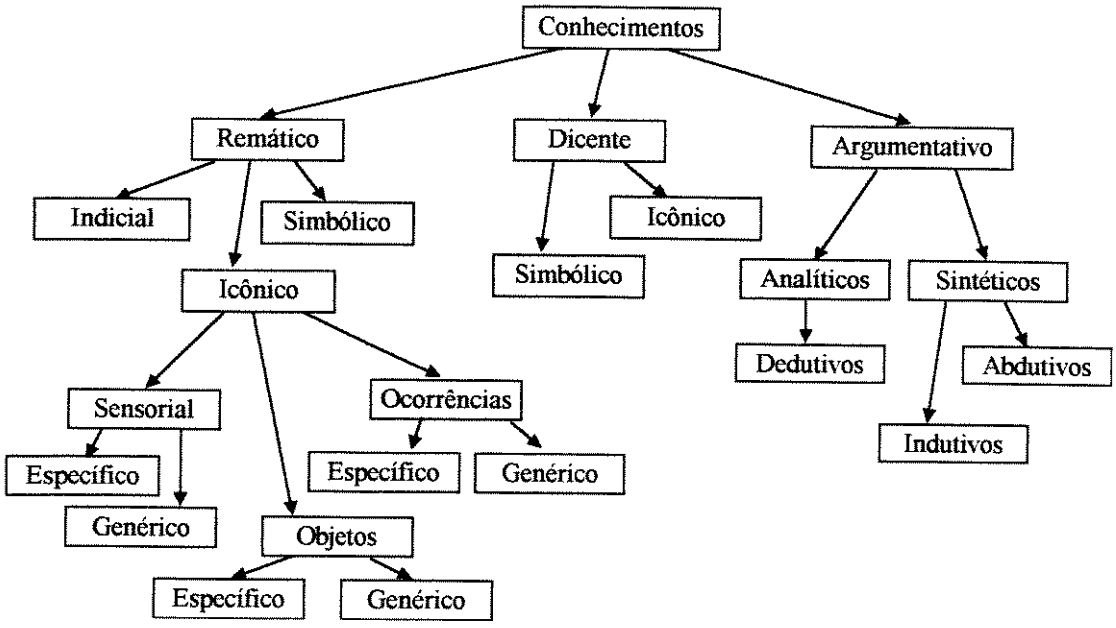


Figura 2.2 - Taxonomia dos Tipos Elementares de Conhecimento

2.3.1 Conhecimento Remático

O conhecimento remático é o tipo de conhecimento gerado pela interpretação de remas, ou termos. Esses termos são utilizados para referenciar fenômenos do ambiente, tais como experiências sensoriais, objetos, e ocorrências.

Existem três tipos de conhecimentos remáticos. O conhecimento remático simbólico corresponde a um nome. Por meio desse nome, pode-se referenciar indiretamente um fenômeno do ambiente.

O conhecimento remático indicial também é utilizado para se referenciar indiretamente um fenômeno do ambiente, embora não de maneira absoluta como no caso do conhecimento simbólico. Nesse caso, o fenômeno do ambiente é referenciado de maneira relativa, a partir de outro fenômeno previamente identificado. Exemplos desse tipo de conhecimento são a semântica dos pronomes relativos (esse, este, esta, etc), de índices que identificam termos em uma lista ordenada, ou um sistema de coordenadas em função de um referencial móvel.

O conhecimento remático icônico corresponde a um modelo direto dos fenômenos que representa. Pode ser dividido em três categorias básicas: o conhecimento sensorial, o conhecimento dos objetos e o conhecimento de ocorrências. Cada um destes, por sua vez, pode ser um conhecimento específico ou um conhecimento genérico.

Um conhecimento sensorial ocorre quando um signo que adentra a interface de entrada é identificado como um padrão conhecido. Alguns tipos de sistemas de processamento de conhecimento se limitam a modelar e a utilizar conhecimento deste tipo. Exemplos característicos são as redes neurais artificiais (Kosko, 1992) ou os sistemas de controle fuzzy (Kosko, 1992; Pedrycz, 1989), onde o conhecimento sensorial é classificado por meio de conjuntos fuzzy, e transforma-se um conhecimento sensorial de entrada em um conhecimento sensorial de saída que é enviado aos atuadores como um sinal de controle. Um conhecimento sensorial específico é a identificação de um padrão sensorial referente a uma instância particular e temporal. Ou seja, é por exemplo o conhecimento de que em um determinado instante de tempo bem definido, a entrada de um determinado sensor poderia ser classificada como "alta". Um conhecimento sensorial genérico ocorre quando se deseja referenciar todo um grupo de conhecimentos sensoriais, com características semelhantes. Nesse caso, não se identifica uma instância particular temporal, mas refere-se ao conhecimento de uma instância genérica que deve ser possível ocorrer. Tomando-se o exemplo anterior do sensor, esse tipo de conhecimento refere-se a todas as instâncias temporais onde a classificação da entrada do sensor é dada como "alta". De certa forma esse é um tipo de conhecimento prototípico, pois ele pode se referir a qualquer instância particular, embora nenhuma instância explícita seja assumida.

Observe que um conhecimento sensorial pode não se restringir a sensores individuais, mas sim a padrões multi-sensoriais e multi-temporais, o que por si só já leva a uma análise bem complexa.

O conhecimento de objeto acontece quando, por meio de conhecimentos sensoriais, supõe-se que esses conhecimentos são causados pela existência de um objeto do mundo exterior. Neste caso, o sistema passa a modelar um objeto cujas características estejam de acordo com os conhecimentos sensoriais obtidos até então. Do mesmo modo que no conhecimento sensorial, este tipo de conhecimento pode ser específico ou genérico. O conhecimento de objeto específico se refere a um objeto em particular. Isso implica na existência de um modelo para o objeto, com uma estrutura adequada para representar seus atributos em diferentes instantes de tempo. Este objeto, em particular, tem uma trajetória temporal bem definida, sendo gerado em determinado instante, existindo durante certo tempo e eventualmente tendo sido destruído em outro instante. O conhecimento de objeto genérico corresponde a um conhecimento prototípico do objeto, ou seja, diz respeito a toda uma classe de objetos que partilham características comuns. Um exemplo onde se distingue bem o objeto específico do objeto genérico pode ser encontrado no estudo dos sistemas de comunicação.Considere um canal de comunicação por onde são enviados caracteres referentes a um determinado código. Suponha que um dos caracteres constante do código seja o caracter "$". Uma determinada instância particular onde o caracter "$" é enviado corresponde a um conhecimento de objeto específico: o caracter "$"enviado no instante t. Quando se referencia o caracter "$" sem se identificar uma instância particular onde este ocorre (ou seja, qualquer caracter "$"), está se modelando um conhecimento de objeto genérico. Observe que no caso de um conhecimento de objeto genérico, este poderia em princípio ser qualquer uma de suas ocorrências particulares. Um outro exemplo de objeto genérico neste contexto, seria a referência a "um caracter". Neste caso, tem-se um outro objeto genérico, representando a classe de todos os caracteres, e nao somente os caracteres "$". Observe que a especificidade ou generalidade não estão relacionadas com o conhecimento da trajetória temporal completa do objeto, mas sim com a identificação da estrutura que guarda

informações do objeto. Em um objeto específico, pode-se não ter informações completas sobre os atributos do objeto em todas as instâncias temporais. Isso não significa que ele deixa de ser específico, mas sim que se houvesse o conhecimento dos valores dos atributos nestes instantes, eles estariam na estrutura que modela o objeto e não em outro lugar. No caso do objeto genérico, este representa todo um grupo de objetos específicos. Não se identifica uma estrutura em particular, mas um modelo geral que permita saber se uma determinada estrutura está ou não abrangida pelo seu escopo. Em outras palavras, o objeto genérico é uma abstração de um objeto específico, mantendo contato com o mundo real somente por intermédio de um objeto específico que esteja em seu escopo.

O conhecimento de ocorrências, ou ações, corresponde ao conhecimento dos valores dos atributos dos objetos do mundo, da mudança desses valores em função do tempo, bem como a geração ou destruição desses objetos. Pode ainda dizer respeito não aos objetos do mundo, mas a algum conhecimento sensorial. Nesse caso, este é corporificado para ser tratado como um objeto. Um conhecimento de ocorrências específico corresponde a uma ocorrência relacionada a um objeto ou conjunto de objetos, em uma instância temporal particular na existência desse(s) objeto(s). O conhecimento de ocorrências genérico é semelhante ao conhecimento de ocorrências específico, com a ressalva que a instância temporal é genérica, ou seja não necessita ser definida. Observe que uma ocorrência deve estar relacionada sempre com pelo menos um objeto, mas este tanto pode ser um objeto específico como um objeto genérico. O que caracteriza a ocorrência como genérica ou específica não são os objetos aos quais está relacionada, mas a definição de uma instância temporal particular (ou não) onde a ocorrência se concretizou. Podemos exemplificar esses conceitos utilizando-se o modelo de um veículo autônomo em um ambiente industrial. Imagina-se que durante sua trajetória diária, o veículo sai de um local inicial, percorre diversos outros locais e termina em um local final. Suponha que exista uma frota de 5 veículos no sistema. Um conhecimento de ocorrência específico seria saber que o veículo 1 saiu do local A as 8:00, chegou a B as 8:03, permanecendo até 8:05, dali saindo para o local C, onde chegou as 8:07. Este é um conhecimento de ocorrência específico com objeto específico. Do mesmo modo um conhecimento de ocorrência específico com objeto genérico seria saber-se que um veículo (não especificado) saiu do local A as 8:00, chegou a B as 8:03, permanecendo até 8:05, dali saindo para o local C, onde chegou as 8:07. Um conhecimento de ocorrência genérica com objeto específico seria saber-se que o veículo 1 saiu de A, passou por B e chegou a C. Observe que neste caso, não se especifica o horário em que o veículo 1 efetuou a trajetória. Se ele fez essa trajetória mais de uma vez durante o dia, este tipo de conhecimento em princípio não distingue qual das trajetórias se está referindo. Em princípio, ele se refere a todas elas. O conhecimento de ocorrência genérica com objeto genérico pode ser representado pelo seguinte conhecimento: um veículo saiu de A, passou por B e chegou a C. Nesse caso, não se especificou qual dos veículos e nem a que instância temporal particular deste se está referindo. Em princípio, pode-se pensar em um veículo genérico e uma ocorrência genérica. Outra ocorrência genérica neste contexto seria o conhecimento de que um veículo seguiu alguma trajetória, onde aqui, nem a trajetória é exatamente especificada, sendo uma trajetória prototípica. Observe que isso corresponde a um conhecimento, pois os veículos poderiam não ter executado nenhuma trajetória.

O conhecimento de ocorrências é sem dúvida nenhuma o mais complexo dos conhecimentos remáticos. O conhecimento sensorial é de um tipo mais primitivo, pois se encontra em contato direto com o mundo. O conhecimento de objetos já faz

uma extrapolação, quando presume que o conhecimento sensorial que obtém é oriundo de objetos do mundo. O conhecimento de ocorrências, além de tratar com estes objetos presumíveis, ainda extrapola quanto a mudanças nos atributos destes. Observe que o conhecimento sensorial é sempre um conhecimento de tipo simples e independe de outros conhecimentos. O conhecimento de objetos normalmente é simples, podendo ser composto, por exemplo, quando um objeto é parte de outro objeto. O conhecimento de ocorrências é sempre composto, pois necessariamente uma ocorrência tem de estar associada a um ou mais objetos. De um modo geral, podem haver ocorrências onde mais de um objeto está associado. Nesses tipos de ocorrências, há a necessidade que a modificação seja simultânea para os objetos associados, de modo que a ocorrência se caracterize. Por exemplo, o conhecimento de que um determinado veículo transporta uma determinada peça. Esse conhecimento é mais do que simplesmente dizer que a localização do veículo muda e a localização das peças mudam. Existe a necessidade de que sempre, a mudança na localização do veículo implique na mudança na localização da peça, e que não se tenha controle sobre a localização da peça, como se tem do veículo (pois se isso fosse possível, talvez a peça que estivesse transportando o veículo !). Neste caso, nitidamente existem dois objetos, a peça e o veículo, associados ao conhecimento de ocorrência "transporte", onde o veículo é o agente e a peça o elemento passivo.

2.3.2 Conhecimento Dicente

O conhecimento remático diz respeito a termos que caracterizam os fenômenos do ambiente, ou seja um determinado termo corresponde a um objeto, outro termo corresponde a uma ocorrência, e o sistema consegue interpretar o significado de cada um destes. No caso do conhecimento dicente, um termo ou uma sequência de termos é utilizada para representar uma expressão, que codifica uma proposição. O que caracteriza um termo ou sequência de termos como sendo uma proposição é o fato de existir um valor-verdade associado a ele. Esse valor verdade é uma medida da crença que o sistema cognitivo tem de que uma proposição é verdadeira. Usualmente, o valor verdade é representado por um valor entre 0 e 1. Um valor-verdade igual a 0 significa que o sistema acredita que a proposição é falsa. Um valor-verdade igual a 1 representa que o sistema acredita que a proposição é verdadeira. Na lógica clássica (Chang & Lee, 1973), assume-se que o valor-verdade de uma proposição somente pode assumir os valores 0 e 1, correspondendo aos valores "falso"e "verdadeiro". Em lógicas multivalores, tais como a lógica *fuzzy*, o valor-verdade pode assumir valores intermediários, entre 0 e 1. Esses valores correspondem a crenças maiores ou menores na veracidade/falsidade da proposição. Um valor 0.5 representa que o sistema não tem a menor idéia se uma proposição é falsa ou verdadeira. Outra maneira de representar essa crença seria utilizar valores de -1 a 1, onde -1 corresponderia a falsidade, 0 a dúvida total e 1 a veracidade.

Os termos que formam uma expressão podem ser proposições primitivas ou conectivos lógicos. Existem, fundamentalmente, dois tipos de proposições primitivas, as proposições icônicas e as proposições simbólicas.

Uma proposição icônica é definida como uma composição de termos formando uma sentença. Cada termo da sentença referencia direta ou indiretamente um conhecimento remático icônico, estando associado ao valor verdade da proposição. Uma sentença deve possuir uma ocorrência, e um ou mais objetos (ou conhecimentos sensoriais). O valor verdade associado à sentença corresponde à crença que o sistema cognitivo tem de que o conhecimento contido na proposição

icônica efetivamente representa o que ocorreu no mundo real, ou seja, que a ocorrência indicada efetivamente ocorre nos objetos indicados. Observe que nos exemplos de conhecimento de ocorrências utilizou-se implicitamente a noção de proposição icônica.

Em uma proposição icônica, a ocorrência é chamada de verbo (ou predicado), e cada um dos objetos relacionados é chamado de um relato (ou sujeito). O número de relatos necessários para uma ocorrência expressar seu significado é chamado de aridade da ocorrência, ou aridade do verbo.

As proposições simbólicas são nomes associados a outras proposições. Sendo assim, seu valor verdade deve corresponder ao valor-verdade da proposição que nomeia. Entretanto, apesar de não ter um conhecimento remático diretamente associado, seu valor verdade pode ser determinado por meio da interação com outras proposições.

Uma proposição é chamada de uma proposição simples, se ela não pode ser decomposta em mais de uma proposição, ou seja, é formada unicamente por uma proposição primitiva. Uma proposição composta é uma proposição formada pela composição de diversas proposições primitivas, ligadas por meio de conectivos lógicos.

Um tipo particularmente interessante de proposição é a chamada proposição condicional. Uma proposição condicional é uma proposição que pode ser representada na forma:

SE (proposição antecedente) ENTÃO (proposição consequente).

Uma proposição condicional é claramente uma proposição composta. O grande interesse por trás das proposições condicionais é que sendo esta verdadeira, a partir do valor verdade da proposição antecedente, pode-se determinar o valor verdade da proposição consequente. Uma proposição deste tipo é também chamada de uma regra. Uma proposição que tenha o valor verdade "verdadeiro" (ou uma crença com valor 1) é chamada de um fato.

O conhecimento dicente utilizando proposições simbólicas é muito utilizado na lógica clássica pois ele dispensa os detalhes semânticos do conhecimento remático contido nas proposições icônicas. Assume-se que o valor verdade de algumas proposições é conhecido, e a partir de regras envolvendo proposições simbólicas, determina-se o valor-verdade destas.

Utilizando-se tanto proposições simbólicas como icônicas, pode-se ampliar a aplicabilidade do conhecimento dicente, comparado a seu uso na lógica clássica. Na lógica clássica, assume-se que o valor-verdade de algumas proposições é previamente conhecido, e a partir das regras, determina-se o valor-verdade de outras proposições. Utilizando-se proposições icônicas, não é necessário partir-se do princípio de que o valor-verdade de algumas proposições é conhecido. O valor-verdade de proposições icônicas pode ser calculado, em função do conhecimento remático que a integra. Sabendo-se o valor-verdade destas proposições, pode-se determinar o valor-verdade de outras proposições (que podem ser simbólicas ou rematicas). Quando o valor-verdade de uma proposição icônica é determinado desta maneira, ele é utilizado para complementar o conhecimento remático que a integra, que pode se encontrar desconhecido. Com isso, obtém-se uma perfeita integração entre os conhecimentos rematicos e dicentes.

2.3.3 Conhecimento Argumentativo

No conhecimento dicente, tem-se a idéia de proposições, que são a composição de termos aos quais é associado um valor verdade. Para o caso do conhecimento argumentativo, tem-se a idéia de argumento, que corresponde a um agente de transformação de conhecimento. Um argumento tipicamente transforma um conjunto de conhecimentos, chamados de premissa (ou as premissas) do argumento, em um novo conhecimento, chamado de sua conclusão. Esta transformação é realizada por meio de uma função de transformação, chamada de função argumentativa, que caracteriza o tipo de argumento.

Os argumentos podem ser divididos em argumentos analíticos e argumentos sintéticos. Os argumentos analíticos são argumentos onde a conclusão, ou seja, o conhecimento gerado pela função argumentativa, já se encontra implicitamente nos conhecimentos utilizados como premissa. O mérito da função argumentativa é o de explicitar esse conhecimento por meio da análise dos conhecimentos nas premissas. Por exemplo, se o conhecimento utilizado nas premissas for um conhecimento dicente, pode-se gerar uma proposição (a conclusão) baseado na análise do valor verdade de outras proposições (as premissas). Os argumentos analíticos são também conhecidos como argumentos dedutivos e a função argumentativa analítica mais conhecida corresponde ao "modus ponens" (Enderton, 1972). Os argumentos sintéticos, por outro lado, criam um conhecimento novo, sintetizam um conhecimento novo, a partir dos conhecimentos em suas premissas. As conclusões de argumentos analíticos nunca entram em contradição com os conhecimentos já existentes no sistema cognitivo, uma vez que nada mais são do que a explicitação de um conhecimento já existente no sistema. As conclusões de argumentos sintéticos nem sempre têm esse comportamento, pois inserem um conhecimento novo no sistema, que potencialmente pode entrar em contradição com o conhecimento já existente. Entretanto, o uso de argumentos sintéticos permite ampliar e refinar o conhecimento de um sistema cognitivo, sendo extremamente úteis para o aprendizado de novos conhecimentos. Lembrando que um sistema cognitivo não pode ser iniciado contendo todo o conhecimento a respeito do mundo real, e que esse conhecimento adicional deve ser obtido a partir da interação com o mundo real, os argumentos sintéticos são a chave para esse aprendizado, gerando novos conhecimentos. Os argumentos sintéticos podem ser classificados como argumentos indutivos ou abdutivos. Um argumento indutivo é um argumento que modifica ligeiramente os conhecimentos utilizados nas premissas, na esperança de que diante de uma modificação pequena, o conhecimento resultante continue sendo válido. Por exemplo, um argumento indutivo pode fazer uma generalização. Nesse caso, a partir de um número de casos de ocorrências individuais, um argumento indutivo reforça a crença de que esses casos são instâncias de uma regra geral. Por exemplo, citando um exemplo de Peirce (1990): retira-se um feijão de um saco e o feijão é marrom. Retira-se outro, e este também é marrom. Retira-se um determinado número de feijões e estes são marrons. Conclui-se então que todos os feijões do saco são marrons. Esta conclusão não é uma dedução, pois para que o fosse, seria necessário inspecionar todos os feijões do saco. O argumento indutivo, nesse caso, é um mecanismo pelo qual se acredita na veracidade da proposição de que todos os feijões são marrons, mesmo na ausência de fatos que garantam isso. A base da argumentação indutiva, nesse caso, é que se um determinado conhecimento é verdadeiro em um determinado número de situações, e não é falso em nenhuma delas, isso significa que existe uma regra geral que regula esse conhecimento. Esse tipo de raciocínio pode ser perigoso algumas vezes, pois eventualmente ele pode

inserir contradições no sistema. Por exemplo, no caso dos sacos de feijão, se houvesse um único feijão branco no saco, a proposição de que todos são verdadeiros seria falsa (na lógica booleana). Entretanto, a argumentação indutiva é importante, pois existem situações onde não é possível verificar todas as situações possíveis, e sem um argumento indutivo, não seria possível usufruir desse conhecimento, que tem grandes chances de ser verdadeiro.

O outro tipo de argumento sintético é o argumento abdutivo. Nesse tipo de argumento, considera-se a validade de um conhecimento na medida em que este não entra em conflito com o conhecimento já existente no sistema. Assim, acredita-se que um determinado conhecimento é válido, simplesmente porque ele está de acordo com as outras crenças que o sistema possui. Em outras palavras, a conclusão do argumento abdutivo é validada porque ela explica outros conhecimentos do sistema. Um exemplo (também de Peirce) do raciocínio abdutivo seria a descoberta da equação que rege o movimento dos planetas por Kepler. A partir das medidas das posições dos planetas que Kepler havia efetuado, ele abduziu a fórmula que descrevia o movimento dos planetas. Ou seja, assumiu-se a fórmula como verdadeira, pois ela estava de acordo com as medições efetuadas. Observe que apesar de parecer consistente, esse raciocínio também pode levar a contradições. Por exemplo, sabe-se que a estimativa de uma função a partir de uma amostra de pontos não é única. No caso de Kepler, poderia haver uma outra função (que fosse a legítima) para a qual as amostras de Kepler também estariam corretas, mas para a qual outros pontos levariam a valores diferentes dos obtidos pela fórmula abduzida. A lógica do raciocínio abdutivo é a de que, se eventualmente uma conclusão assim obtida estiver errada, cedo ou tarde será constatado o erro e o conhecimento poderá ser retificado.

A classificação dos argumentos encontra-se sumarizada na figura 2.3 a seguir:

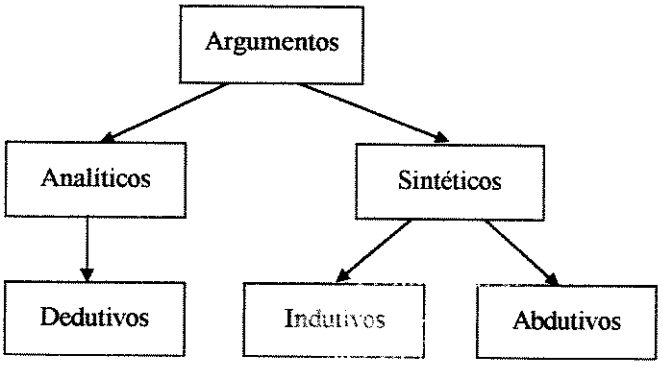


Figura 2.3 - Classificação dos Argumentos

Os argumentos indutivos e abdutivos podem trabalhar em estreita cooperação, de modo a proporcionar um aprendizado. Por exemplo, um argumento abdutivo cria uma nova proposição no sistema, inicialmente com valor verdade 0.5 (indicando nem veracidade nem falsidade), e um argumento indutivo modifica ligeiramente o valor-verdade desta proposição, a partir de fatos que evidenciem um aumento ou diminuição em seu grau de veracidade.

Observe que os conhecimentos utilizados nas premissas e gerados na conclusão do argumento podem ser em princípio de qualquer tipo. Podem ser conhecimentos remáticos, conhecimentos dicentes ou mesmo conhecimentos argumentativos. Com isso, um conhecimento argumentativo pode em princípio

gerar um outro conhecimento argumentativo, modificar um conhecimento argumentativo anterior ou eventualmente destruir um conhecimento argumentativo do sistema, que passe a ser considerado inválido ou superado em virtude de novos conhecimentos adquiridos pelo sistema. Por exemplo, considere-se um caso onde se generaliza uma situação que se repete diversas vezes. Ao invés de guardar cada uma das instâncias em que a mesma ocorre, após a geração de uma regra geral que explique a situação, o sistema pode descartar os conhecimentos individuais, referentes a cada caso, armazenando somente a regra geral.

Os argumentos dedutivos, indutivos e abdutivos são utilizados implicitamente em todos os sistemas que envolvem conhecimento, sendo que os argumentos indutivos e abdutivos são utilizados marcadamente em sistemas que envolvem aprendizado. Por exemplo, os sistemas de produção fuzzy (Pedrycz, 1989) nada mais são do que uma instanciação de um argumento dedutivo. As redes neurais (Kosko, 1992), utilizam notadamente um argumento dedutivo, quando utilizadas sem aprendizado, e um argumento indutivo, quando ocorre o aprendizado. Os algoritmos genéticos (Goldberg, 1989) utilizam argumentos abdutivos quando geram novos cromossomas por meio de crossover e argumentos indutivos quando geram novos cromossomas por meio de mutação e quando ocorre a seleção. O sistema de processamento de conhecimento mais interessante dentre todos estes, é o sistema classificador (Booker et.ali., 1989), que utiliza argumentos dedutivos, em sua parte de classificador, e argumentos indutivos e abdutivos em sua parte relacionada a algoritmo genético (algoritmo *bucket-brigade*).

Alguns autores, principalmente nos textos da IA (e.g. Rendell, 1987), referem-se aos argumentos sintéticos de um modo geral como "indução", chamando os argumentos abdutivos de "indução construtiva" (*constructive induction*) e os argumentos indutivos de "indução propriamente dita" (*induction proper*).

2.4 Conhecimento Aplicado

Nesta seção, uma outra classificação de tipos de conhecimento é utilizada, de acordo com a finalidade do conhecimento em um sistema cognitivo (vide figura 2.4).

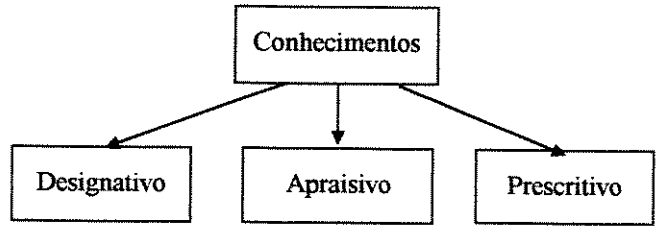


Figura 2.4 - Classificação dos Conhecimentos segundo sua Finalidade

Dependendo de sua finalidade, um conhecimento pode ser classificado como designativo, apraisivo[1] ou prescritivo. Essa classificação é ortogonal à classificação dos tipos elementares de conhecimento. Em princípio, qualquer conhecimento,

[1] O termo "apraisivo" é utilizado como uma tradução do inglês *apraisive*, apesar de não ser uma palavra regular da língua portuguesa. Outras traduções possíveis, tais como avaliativo ou apreciativo são consideradas inadequadas por não incluirem toda a semântica do termo original, no contexto em que este é utilizado.

independente de sua classificação elementar, pode ser designativo, apraisivo ou prescritivo. Em alguns casos, como iremos ressaltar, alguns tipos serão mais adequados que outros para exercer uma determinada finalidade. Essa nomenclatura foi adotada inicialmente por Morris (Morris, 1971; Morris, 1964; Morris, 1947).

2.4.1 Conhecimento Designativo

O conhecimento designativo é o conhecimento utilizado para modelar o mundo real. Para tanto, ele se serve de conhecimentos remáticos, dicentes e argumentativos, genéricos ou específicos. Pode também ser chamado de conhecimento descritivo, pois descreve, por meio de suas estruturas, uma representação do mundo. Normalmente, um sistema cognitivo inicia-se com muito pouco (ou nenhum) conhecimento designativo, o qual vai sendo adquirido a medida que o sistema interage com o mundo. Todos os conhecimentos apresentados como exemplo na seção anterior são do tipo designativo.

2.4.2 Conhecimento Apraisivo

O conhecimento apraisivo é um tipo de conhecimento que é utilizado como uma avaliação, um juízo, um julgamento, diante de um propósito. Em sistemas naturais o conhecimento apraisivo está relacionado com propósitos gerais de um ser vivo, tais como a reprodução, a sobrevivência do indivíduo, a sobrevivência da espécie e o aumento do conhecimento sobre o mundo, por exemplo. Diante desses propósitos gerais, ele adquire formas bem delineadas, tais como desejo, repulsa, medo, cobiça, ódio, amor, prazer, dor, conforto, desconforto, etc. Fundamentalmente, esse tipo de conhecimento avalia se uma determinada sensação, objeto ou ocorrência é boa ou ruim para o propósito relacionado. Essa avaliação será utilizada posteriormente, para a elaboração de um conhecimento prescritivo, que executará uma intervenção no mundo de modo a garantir que o propósito seja satisfeito. Apesar de, em seres vivos, esses propósitos serem gerais, nada impede que em sistemas artificiais, o conhecimento apraisivo seja utilizado para propósitos bem específicos.

O conhecimento apraisivo normalmente é uma característica inata dos sistemas cognitivos, pois expressam a finalidade para a qual se deseja que o sistema cognitivo atue no mundo. No caso dos seres vivos, essa finalidade pode ser apenas a de crescer e se multiplicar, mas em sistemas artificiais, pode-se especificar os conhecimentos apraisivos inatos do sistema, de modo a obter deste um comportamento desejado. Apesar de grande parte do conhecimento apraisivo de um sistema cognitivo ser inata, isso não significa que não exista aprendizado com conhecimentos apraisivos. Ele existe sim, pois como a maioria do conhecimento designativo não é inata, os conhecimentos designativos que vão sendo incorporados ao sistema pela experiência podem ser associados a conhecimentos apraisivos. No entanto, esse aprendizado será balizado pelo conhecimento apraisivo inato, expandindo-o a novas sensações e situações. O conhecimento apraisivo também é responsável, de certa forma, pelo próprio aprendizado. Um dos propósitos aos quais o conhecimento apraisivo faz referência, tanto em seres vivos como em sistemas artificiais, é o de aumentar o conhecimento a respeito do mundo real. Para tanto, o conhecimento apraisivo é utilizado para o próprio aprendizado de conhecimentos designativos e prescritivos.

O conhecimento apraisivo pode se constituir de sensações, e.g. quando se trata de uma avaliação não vinculada a nenhum objeto. Por exemplo, em seres humanos, quando se tem a sensação de algo bom ou algo ruim. Pode estar

relacionado com um conhecimento de objeto, quando o objeto é a fonte do conhecimento apraisivo. E pode também estar relacionado a ocorrências, quando é uma determinada ação que evoca o conhecimento apraisivo. Normalmente os conhecimentos dicentes e argumentativo não são utilizados em situações apraisivas.

Os conhecimentos apraisivos podem ser utilizados para diferentes propósitos. Com isso, determinadas situações ou objetos identificados podem ter associado um conhecimento apraisivo ambíguo. Por exemplo, diante de um propósito, o conhecimento apraisivo referente a um conhecimento designativo pode ser negativo e diante de outro propósito o mesmo conhecimento designativo pode evocar um conhecimento apraisivo positivo. Diante dessa situação, é necessária a definição de um conhecimento apraisivo global, que considere ponderadamente os diferentes conhecimentos apraisivos (obtidos dos diferentes propósitos) e determine um conhecimento apraisivo resultante, que balizará a atitude do sistema cognitivo quanto ao conhecimento prescritivo que será enviado para execução.

O conjunto de conhecimentos apraisivos de um sistema cognitivo corresponde ao sistema de valores do sistema cognitivo e é fundamental para que o sistema cognitivo atinja seus propósitos. Estes propósitos naturalmente devem ser ponderados por prioridades, de modo que em situações conflitantes, os propósitos mais importantes sejam priorizados.

2.4.3 Conhecimento Prescritivo

O conhecimento prescritivo está relacionado ao conhecimento utilizado para atuar no mundo real. Basicamente o conhecimento prescritivo é utilizado para traçar planos de atuação, e atuar efetivamente no mundo real por meio dos atuadores do sistema cognitivo. Deste modo, um conhecimento apraisivo de desejo, por exemplo, é transformado em um comportamento no sentido de se aproximar de um determinado objeto, enquanto que um conhecimento de repulsa leva ao afastamento do objeto em questão. Com isso, o tipo de conhecimento elementar mais utilizado como conhecimento prescritivo é o conhecimento dicente, pois um conhecimento prescritivo no fundo é uma proposição que deve ser verdadeira no futuro. Entretanto, podem haver alguns casos onde conhecimentos remáticos são utilizados como conhecimento prescritivo. Por exemplo, um sistema de ações automáticas (comportamento reativo) pode ser determinado por um conhecimento remático sensorial.

Apesar do conhecimento prescritivo estar relacionado com a atuação, nem todo conhecimento prescritivo implicará efetivamente em uma ação. Grande parte do conhecimento prescritivo é utilizado na elaboração de predições, estando armazenado em proposições condicionais do tipo: se eu fizer assim (conhecimento prescritivo), obtererei isso (conhecimento designativo e/ou apraisivo). De todos os conhecimentos prescritivos, somente aqueles selecionados para uma atuação é que realmente irão atuar sobre o mundo real.

O que caracteriza um conhecimento como prescritivo é ele poder ser mapeado (por meio dos argumentos adequados) em um conhecimento remático que corresponda diretamente aos valores dos atuadores do sistema cognitivo. Esse mapeamento pode ser direto, ou pode estar estruturado hierarquicamente.

Basicamente, um conhecimento prescritivo corresponde a um comando. Esse comando pode, eventualmente, ser decomposto em subcomandos e esses por sua vez em sub-sub-comandos e assim por diante, até se chegar efetivamente ao nível dos atuadores, quando os comandos são então executados. Observe que um comando de alto nível pode corresponder a inúmeros comandos subsequentes ao

nível de atuadores, o que faz com que a execução de um comando de alto nível tenha um tempo de latência até que seja completado. Isso pode eventualmente causar um problema de sincronismo. Imagine que um comando de alto nível seja enviado para execução. Em seguida, enquanto o sistema ainda está executando os subcomandos individuais dos atuadores, um outro comando de alto nível é enviado para execução. Que comandos de atuador devem ser efetivamente utilizados ? O do primeiro comando ou o do segundo ? Diversas estratégias podem ser utilizadas no sentido de resolver este problema. Uma delas consiste em simplesmente abortar o primeiro comando de alto nível, uma vez que um novo comando seja enviado para execução. Outra estratégia poderia ser colocar os comandos em uma fila, e só executar o segundo comando quando terminado o primeiro. Uma terceira estratégia poderia ser uma hibridização destas duas estratégias, colocando prioridades nos comandos, de modo que comandos prioritários abortem comandos anteriores ainda não executados, e comandos de menor prioridade aguardem a conclusão de comandos anteriores mais prioritários. Uma outra estratégia, poderia verificar se os comandos podem ser executados em paralelo (ou seja, afetam diferentes atuadores do sistema) e quando não, se deve ser executado um, o outro ou uma combinação dos dois. Outras estratégias também podem ser imaginadas.

Quanto aos atuadores do sistema, uma vez que nenhum comando tenha sido enviado para execução, estes podem apresentar algum comportamento padrão. Esse comportamento padrão pode ser diferente para cada atuador. Por exemplo, um comportamento padrão pode ser o de manter o valor anterior, quando nenhum comando é enviado. Outro comportamento padrão seria o de se definir um valor padrão para o atuador tal que, na ausência de um comando, o valor do atuador assume o valor padrão. Outros comportamentos mais sofisticados podem ser elaborados, como por exemplo, um atuador pode assumir valores randômicos dentro de uma determinada faixa, ou pode ir se modificando gradativamente de seu valor mais baixo para o seu valor mais alto e daí de novo para seu valor mais baixo. Diversos comportamentos padrão podem ser especificados. Estes comportamentos podem, inclusive, ser programáveis, ou seja, por força de um comando, adota-se um comportamento padrão, que pode ser modificado quando do aparecimento de um outro comando pertinente. A diferença básica entre um comportamento padrão e um comando é que, no comando, existe um mecanismo de princípio, meio e fim, que é coordenado de modo a concluir o comando. Um comportamento padrão em princípio não tem fim. Pode continuar até que um comando determine sua mudança.

O aprendizado do conhecimento prescritivo utiliza bastante o conhecimento apraisivo. Por exemplo, o sistema cognitivo pode gerar abdutivamente uma série de conhecimentos prescritivos, e submeter esses conhecimentos prescritivos a um julgamento mediante conhecimentos apraisivos. Esse julgamento levará em conta a previsão dos conhecimentos designativos que ocorrerão caso o conhecimento prescritivo seja efetivamente empregado e por conseguinte dos conhecimentos apraisivos que serão gerados. Diante do resultado dessa previsão de conhecimento apraisivo, o sistema decide executar o conhecimento prescritivo que gerar uma previsão de conhecimento apraisivo mais adequada. Uma vez que a previsão ocorra conforme esperado, um aprendizado indutivo pode reforçar as regras de tal forma que no futuro o sistema precise efetuar menos previsões, e atue em função simplesmente do conhecimento designativo presente.

2.5 Resumo

Nesse capítulo foram apresentados alguns conceitos básicos relacionados ao que é entendido como comportamento inteligente, utilizando como base conceitos disponíveis na semiótica, traduzidos para a linguagem utilizada na inteligência artificial. Introduziram-se alguns conceitos da semiótica, tais como a idéia de signo, e a trilogia signo-objeto-interpretante. Foi estabelecida uma correspondência entre estes conceitos e seus equivalentes utilizados na IA. Em seguida, apresentou-se uma taxonomia básica dos tipos de conhecimento, iniciando-se com os tipos elementares de conhecimento, seguidos dos tipos de conhecimento aplicado.

No capítulo 3 a seguir, faremos uma pausa na temática inteligência e conhecimento, para introduzir um modelo matemático formal - as redes de objetos - que será utilizado posteriormente no capítulo 4 para uma modelagem formal dos conceitos introduzidos, informalmente, no presente capítulo.

3. Fundamentos para uma Teoria dos Objetos

3.1 Introdução

Neste capítulo, introduz-se um modelo formal para o conceito de objeto. A proposição deste modelo é considerada tendo-se em vista seu uso como um veículo de formalização para os diferentes tipos de conhecimento que foram apresentados, informalmente, no capítulo anterior. Apesar de se ter essa perspectiva, a definição de um modelo formal para objetos, como colocada neste capítulo, pode (e deve) ser considerada em um âmbito mais abrangente. As definições aqui colocadas podem ser utilizadas como fundamentos para uma futura "teoria geral dos objetos", que funcione como um arcabouço formal para as aplicações ditas "orientadas a objeto", tais como linguagens orientadas a objeto ou programação orientada a objeto.

A idéia de objeto, ou entidade, está intimamente relacionada ao pensamento humano. Aparentemente, a mente humana está preparada para, a partir do vasto manancial sensorial do corpo humano, identificar, representar e utilizar abstratamente objetos, de modo a interagir com o mundo real. Essa característica da mente humana vem sendo utilizada em programação, por meio das chamadas linguagens orientadas a objeto. Nestas linguagens, tenta-se modelar a idéia de objetos em estruturas de programação, de modo que os conceitos que se desejam inserir nos programas estejam mais próximos do modo como a mente humana lida com tais conceitos, facilitando a transformação destes em programas. Entretanto, apesar de tais conceitos (linguagens orientadas a objeto, programas orientados a objeto, etc) estarem amplamente divulgados e sendo utilizados em aplicações práticas, existe ainda hoje uma lacuna relativa a um modelo formal ou uma teoria dos objetos. Algumas propostas vêm sendo elaboradas (Wang, 1989), embora carecendo de um embasamento matemático simples, claro e consistente. Outras, apesar do rigor formal, envolvem apenas a especificação de uma semântica uniforme para linguagens orientadas a objeto, na forma de uma meta-linguagem, o que restringe seu uso a linguagens de programação (Wolczko, 1988). Neste capítulo, apresenta-se um modelo matemático formal para o conceito de objetos, baseado na teoria dos conjuntos. Esse modelo é estendido posteriormente para o de redes de objetos. As redes de objetos, serão utilizadas para modelar sistemas dinâmicos adaptativos, com capacidade de aprendizado. Estes serão utilizados como modelo de representação para os diferentes tipos de conhecimento apresentados no capítulo 2.

Inicialmente, serão colocadas algumas definições. Em seguida, parte-se para a formalização da idéia de objeto e, por fim, são apresentadas as redes de objetos, com exemplos práticos ilustrativos.

3.2 Definições Preliminares

Nesta seção, colocaremos algumas definições preliminares necessárias. Estas definições serão feitas para um domínio discreto N, normalmente associado a instantes de tempo ou passos de algoritmos. As extensões para um domínio contínuo são possíveis, mas não serão aqui consideradas. Usualmente N corresponderá ao conjunto dos números naturais, podendo entretanto ser qualquer conjunto enumerável, eventualmente finito.

Relativamente à notação, é necessário ressaltar que não será feita uma distinção maior entre os conceitos de função e grafo de uma função. Sendo assim, o autor utilizará as notações $f : A \rightarrow B$ e $f \subset A \times B$ em diferentes situações, para representar a mesma função f, conforme se deseje ressaltar a característica funcional, ou o fato de uma função também ser um conjunto.

DEFINIÇÃO 3.1 - Ênuplas

Sejam $q_1 , q_2 , \ldots , q_n$ elementos genéricos pertencentes aos conjuntos $Q_1 , Q_2 , \ldots , Q_n$ respectivamente.

Define-se uma **ênupla** como o agrupamento dos elementos $q_1 , q_2 , \ldots , q_n$ formando um único elemento composto denominado q. Para representar o agrupamento de n elementos, utiliza-se uma notação especial, onde os elementos são separados por vírgulas, e a ênupla é delimitada por parênteses, conforme a seguir:

$$q = (q_1 , q_2 , \ldots , q_n)$$

O nome ênupla se refere a um agrupamento genérico de n elementos. Para um número específico de elementos agrupados, utilizam-se denominações particulares. Assim, uma ênupla com dois elementos é um par, três elementos uma tripla, quatro elementos uma quádrupla, etc.

Os elementos que integram uma ênupla, chamados de suas componentes, podem ser referenciados por seu índice na ênupla, de acordo com a ordem em que aparecem na mesma.

Observe que:

- O conjunto gerado pelo produto cartesiano de n conjuntos corresponde ao conjunto de todas as ênuplas que têm como sua n-ésima componente um elemento do n-ésimo conjunto. Assim, os elementos do produto cartesiano de n conjuntos são ênuplas.
- **Uma ênupla não é um conjunto**. Se uma ênupla fosse um conjunto, uma ênupla com apenas um elemento, seria um conjunto unitário, e não um elemento. Uma ênupla com apenas um elemento é o próprio elemento.
- As componentes de uma ênupla, podem também ser ênuplas. Ênuplas deste tipo são chamadas de **ênuplas complexas**. Por exemplo:

$$q = (q_1 , (q_{21} , q_{22} , q_{23}), q_3 , q_4 , q_5)$$

A qualquer momento, pode-se referenciar a ênupla $(q_{21} , q_{22} , q_{23})$ por seu nome composto q_2 . A ênupla ficaria então:

$$q = (q_1 , q_2 , q_3 , q_4 , q_5)$$

DEFINIÇÃO 3.2 - Aridade de uma Ênupla

Seja uma ênupla $q = (q_1, q_2, \ldots, q_n)$.

Define-se a **aridade** da ênupla q representada por Ar(q), como o número de elementos que constituem a ênupla q. Por exemplo, para a ênupla q, Ar(q) = n.

Observe que para o caso de ênuplas complexas, a aridade diz respeito à ênupla principal.

Exemplos: q = (a,b,c), Ar(q) = 3

q = (a,(b,c),d), Ar(q) = 3

q = ((a,b,c),(d,(e,f),g)), Ar(q) = 2

DEFINIÇÃO 3.3 - Índice de Referência

Para a localização de uma componente em uma ênupla, associa-se um **índice de referência** do elemento dentro da ênupla. Para o caso de uma ênupla simples s, o índice de referência consiste de um número i, $1 \leq i \leq Ar(s)$. Para o caso de ênuplas complexas, o índice de referência será uma ênupla i, onde cada elemento i_k desta ênupla corresponde a um sub-índice dentro da ênupla de nível k. Cada sub-índice pode assumir valores entre 1 e a aridade da ênupla no nível k. O índice de referência pode ser utilizado também para a identificação do domínio do elemento.

Exemplos:

$s = (a,b,c)$, $S = S_A \times S_B \times S_C$

$i=1 \rightarrow s_i = a$, $S_i = S_A$

$i=2 \rightarrow s_i = b$, $S_i = S_B$

$i=3 \rightarrow s_i = c$, $S_i = S_C$

$c = (a,(b,d))$, $C = C_A \times (C_B \times C_C)$

$i=1 \rightarrow c_i = a$, $C_i = C_A$

$i=2 \rightarrow c_i = (b,d)$, $C_i = C_B \times C_C$

$i=(2,1) \rightarrow c_i = b$, $C_i = C_B$

$i=(2,2) \rightarrow c_i = d$, $C_i = C_C$

$c = (a,(b,(s,d,(e,f),g),h))$, $C = C_A \times (C_B \times (C_C \times C_D \times (C_E \times C_F) \times C_G) \times C_H)$

$i=(2,1) \rightarrow c_i = b$, $C_i = C_B$

$i=(2,2,3) \rightarrow c_i = (e,f)$, $C_i = C_E \times C_F$

$i=(2,2,3,2) \rightarrow c_i = f$, $C_i = C_F$

$i=(2,3) \rightarrow c_i = h$, $C_i = C_H$

$i=2 \rightarrow c_i = (b,(s,d,(e,f),g),h)$, $C_i = C_B \times (C_C \times C_D \times (C_E \times C_F) \times C_G) \times C_H$

DEFINIÇÃO 3.4 - Fórmula de Indução

Sejam:

- Uma ênupla $q = (q_1, q_2, \ldots, q_n)$.
- Uma expressão k formada por meio da gramática a seguir, onde i corresponde a um índice de referência da ênupla q :

$k \leftarrow [\, i \,]$

$i \leftarrow i, i$

$i \leftarrow [\, i, i \,]$

A expressão k é chamada de uma **fórmula de indução**.

Exemplos: $k = [\, i_1 , [\, i_2 , i_3 , i_4 \,] , i_5 \,]$

$k = [\, [i_1 , i_2 \,], [i_3 , [i_4 , i_5 \,] \,] \,]$

$k = [i_1 , i_2 , i_3 \,]$

onde i_j são índices de referência de uma ênupla q.

DEFINIÇÃO 3.5 - Indução de uma ênupla

Sejam:

- Uma ênupla $q = (q_1 , q_2 , \ldots , q_n)$, definida em $Q = Q_1 \times \ldots \times Q_n$.
- Uma fórmula de indução k.

Define-se a indução de q segundo k, como uma nova ênupla $q_{(k)}$, onde os elementos da ênupla original são agrupados seguindo-se a fórmula de indução k, substituindo os colchetes por parênteses e os índices de referência i_j pelos elementos originais q_{i_j} da ênupla q . O domínio $Q_{(k)}$ de $q_{(k)}$ pode também ser obtido seguindo-se a fórmula de indução k, omitindo-se os colchetes externos da fórmula, substituindo-se os colchetes internos por parênteses, as vírgulas por $\times$ e os índices i_j pelos sub-domínios originais Q_{i_j} de Q.

Exemplos: $q = (a,b,c,d)$, $Q = Q_1 \times Q_2 \times Q_3 \times Q_4$, $k = [1,3,4,2\,]$,
$q_{(k)} = (a,c,d,b)$, $Q_{(k)} = Q_1 \times Q_3 \times Q_4 \times Q_2$

$q = (a,b,c,d)$, $Q = Q_1 \times Q_2 \times Q_3 \times Q_4$, $k = [4,1]$,
$q_{(k)} = (d,a)$, $Q_{(k)} = Q_4 \times Q_1$

$q = (a,b,c,d)$, $k = [\, 1, [2, 3] , 4]$,
$q_{(k)} = (a, (b,c), d)$, $Q_{(k)} = Q_1 \times (Q_2 \times Q_3) \times Q_4$

$q = (a,(b,c),d)$, $Q = Q_1 \times (Q_2 \times Q_3) \times Q_4$, $k = [1,(2,1),(2,2),3]$,
$q_{(k)} = (a,b,c,d)$, $Q_{(k)} = Q_1 \times Q_2 \times Q_3 \times Q_4$

$q = (a, (b,c), d)$, $Q = Q_1 \times (Q_2 \times Q_3) \times Q_4$, $k = [3,2]$,
$q_{(k)} = (d,(b,c))$, $Q_{(k)} = Q_4 \times (Q_2 \times Q_3)$

$q = (a, (b,c), d)$, $Q = Q_1 \times (Q_2 \times Q_3) \times Q_4$, $k = [3,2,(2,1)]$,
$q_{(k)} = (d,(b,c),b)$, $Q_{(k)} = Q_4 \times (Q_2 \times Q_3) \times Q_2$

DEFINIÇÃO 3.6 - Sub-ênupla

Seja q uma ênupla e k uma fórmula de indução, conforme anteriormente.

Uma ênupla $q_{(k)}$ formada pela indução de q segundo k é chamada de uma **sub-ênupla** de q, se cada índice que aparece na fórmula de indução k é um índice unário, aparecendo uma única vez na fórmula e a fórmula só possui um par de colchetes.

DEFINIÇÃO 3.7 - Relação

Sejam n conjuntos $R_1 , \ldots , R_n$ e $R = \{ (r_{i1} , \ldots , r_{in}) \}$, $i = 1, \ldots , M$, um conjunto de M ênuplas de aridade n, onde $n > 1$ e $\forall i \in \{1, \ldots ,M\}$, $\forall k \in \{1, \ldots , n\}$, $r_{ik} \in R_k$.

O conjunto R, $R \subseteq R_1 \times \ldots \times R_n$ é dito ser uma **relação** em $R_1 \times \ldots \times R_n$..

O produto cartesiano $R_1 \times \ldots \times R_n$ é chamado de **universo** da relação.

DEFINIÇÃO 3.8 - Projeção de uma Relação

Seja $R = \{r_i\}$, $r_i = (r_{i1}, \ldots, r_{in})$ uma relação n-ária definida em $R_1 \times \ldots \times R_n$.

Seja k uma fórmula de indução formada apenas por índices unários, $k = [k_1, k_2, \ldots, k_m]$, $k_i \in \{1, \ldots, n\}$, $k_i \neq k_j$, se $i \neq j$, $i = 1, \ldots, m$, $j = 1, \ldots, m$, $m \leq n$.

Define-se a **projeção** de R em $R_{k_1} \times \ldots \times R_{k_m}$, $R{\downarrow}R_{k_1} \times \ldots \times R_{k_m}$, ou alternativamente, $R_{(k)}$, como a relação obtida pela união de todas as sub-ênuplas $r_{i(k)} = (r_{ik_1}, \ldots, r_{ik_m})$ de R, obtidas pela indução das ênuplas de R segundo k:

$$R_{(k)} = \cup\, r_{i(k)}$$

Exemplos: $A = \{1, 2\}$ $B = \{a,b,c\}$ $C = \{\alpha, \beta, \gamma\}$. $R=\{(1,a,\beta), (2,c,\alpha), (2,b,\beta), (2,c,\beta)\}$

$R \downarrow A \times C = \{ (1,\beta), (2,\alpha), (2, \beta) \}$

$R \downarrow C \times B = \{ (\beta,a), (\alpha,c), (\beta,b), (\beta,c)\}$

NOTA: Na projeção de uma relação, os elementos da ênupla que aparecem na sub-ênupla não necessariamente precisam aparecer na mesma ordem que na ênupla original. Sub-ênuplas de diferentes ênuplas de R que sejam iguais, aparecem somente uma vez na projeção. Com isso, o número de elementos da projeção de R será sempre menor ou igual ao número de elementos de R.

DEFINIÇÃO 3.9 - Projeção Livre de uma Relação

Sejam $R = \{ r_i \}$, uma relação definida em U e k uma fórmula de indução.

Define-se a **projeção livre** de R em $U_{(k)}$, $R \downarrow U_{(k)}$, ou alternativamente, $R_{(k)}$ como a relação obtida pela união de todas as sub-ênuplas $r_{i(k)}$ de R, obtidas pela indução das ênuplas de R segundo k:

$$R_{(k)} = \cup\, r_{i(k)}$$

NOTA: A projeção livre é uma generalização do conceito de projeção. Na projeção, somente as ênuplas induzidas por fórmulas formadas por índices unários são consideradas, o que implica em ênuplas definidas sobre as dimensões principais do domínio da relação. Para a projeção livre qualquer ênupla induzida pode ser utilizada. Isso implica que se a fórmula de indução for formada por índices unários que aparecem apenas uma vez, uma projeção livre torna-se uma projeção.

DEFINIÇÃO 3.10 - Extensão Cilíndrica de uma Relação

Seja $R = \{ (r_{i1}, r_{i2}, \ldots, r_{in}) \}$ uma relação n-ária, definida em $R_1 \times \ldots \times R_n$. Define-se a **extensão cilíndrica** P de R em $P_1 \times \ldots \times P_m$, $P = R{\uparrow} P_1 \times \ldots \times P_m$, onde $\forall k \in \{1, \ldots, n\}\ \exists P_j = R_k$, $1 \leq j \leq m$, como a maior (com maior número de elementos) relação $P \subseteq P_1 \times \ldots \times P_m$ tal que $P \downarrow R_1 \times \ldots \times R_n = R$.

Exemplo: $A = \{1, 2\}$ $B = \{a,b,c\}$ $C = \{\alpha, \beta, \gamma\}$. $R = \{ (1,a), (2,c) \}$

$R \uparrow A \times B \times C = \{ (1,a,\alpha), (2,c,\alpha), (1,a,\beta), (2,c,\beta), (1,a,\gamma), (2,c,\gamma) \}$

$R \uparrow C \times A \times B = \{ (\alpha,1,a), (\alpha,2,c), (\beta,1,a), (\beta,2,c,), (\gamma,1,a,), (\gamma,2,c,) \}$

NOTA: Do mesmo modo que na projeção, os elementos da extensão cilíndrica de R que aparecem nas ênuplas de R, não necessariamente precisam aparecer na mesma ordem que nas ênuplas de R.

DEFINIÇÃO 3.11 - Junção de Relações

Sejam:

• R e S duas relações definidas em $R_1 \times ... \times R_n$ e $S_1 \times ... \times S_m$ respectivamente.

• $P = P_1 \times ... \times P_o$ um universo onde $\forall i \in \{1, ..., n\}\ \exists P_k = R_i$ e $\forall j \in \{1, ..., m\}\ \exists P_h = S_j$, $o \leq n + m$.

Define-se a junção das relações R e S sob o universo P, $R * S |_P$ como sendo:

$$R * S |_P = R\uparrow P \cap S \uparrow P$$

NOTA: Se existir algum $R_i = S_j$, pode existir apenas um conjunto P_k com representantes nas ênuplas de R*S. Nesse caso, para uma ênupla ser incluída na junção das relações, o valor da componente relativa a P_k deve ser o mesmo na ênupla de R e na ênupla de S.

NOTA 2: Se $\forall i,j$, $R_i \neq S_j$, $R * S |_P \downarrow R_1 \times ... \times R_n = R$ e $R * S |_P \downarrow S_1 \times ... \times S_m = S$.

Exemplo: $A = \{1, 2\}$ $B = \{a,b,c\}$ $C = \{\alpha, \beta, \gamma\}$. $R = \{ (1,a), (2,c) \}$ $S = \{(a,\alpha), (b,\beta)\}$

$R * S |_{A \times B \times C} = \{ (1,a,\alpha) \}$

$R * S |_{A \times B \times B \times C} = \{(1,a,a,\alpha), (1,a,b,\beta), (2,c,a,\alpha), (2,c,b,\beta) \}$

DEFINIÇÃO 3.12 - Variável

Sejam N um conjunto enumerável, onde cada n denota um elemento de N e $X \subseteq U$ um subconjunto de um conjunto universo U.

Define-se uma **variável** x de tipo X como uma função $x : N \rightarrow X$.

Note que uma função é também uma relação, que por sua vez pode ser expressa por um conjunto. Sendo assim, $x \subset N \times X$.

Exemplos: $N = \{1, 2, 3\}$, $X = \{a, b, c\}$, $x(1) = a$, $x(2) = b$, $x(3) = c$

$x = \{ (1, a), (2, b), (3, c) \}$

$N = \{1, 2, 3, ...\}$, $X = \{a, b, c\}$, $x(1) = a$, $x(2) = b$, $x(3) = c$, ...

$x = \{ (1, a), (2, b), (3, c), ... \}$

DEFINIÇÃO 3.13 - Variável Composta

Seja x uma variável de tipo X. Se os elementos de X são ênuplas, a variável x é chamada uma **variável composta** ou **estrutura**.

O valor de uma variável composta, em um determinado instante de tempo, será sempre uma ênupla. Os valores individuais de cada sub-elemento dessa ênupla podem ser obtidos, referenciados por seu índice na ênupla, também conhecidos como campos da variável. De um modo especial, se o conjunto X corresponde ao produto cartesiano de n conjuntos X_i, ou seja, $X = X_1 \times ... \times X_n$, cada campo da

variável pode ser visto como uma projeção livre de x em $N \times X_i$, ou seja, é uma variável simples de tipo X_i .

Exemplo: $N=\{1, 2, 3\}$, $X_1 = \{a,b\}$, $X_2 = \{c,d\}$ $X = X_1 \times X_2 = \{ (a,c),(a,d),(b,c),(b,d)\}$

$x = \{ (1,(a,c)) , (2,(a,d)), (3, (a,d)) \}$

$x \downarrow N \times X_1 = \{ (1,a) , (2,a), (3, a) \}$

$x \downarrow N \times X_2 = \{ (1,c) , (2,d), (3, d) \}$

3.3 Objetos

A partir do conceito de variáveis, pode-se definir o conceito de objeto. A idéia de se definir um conceito matemático de objeto, está intimamente ligada ao conceito físico de objeto. Ontologicamente, um objeto corresponde a uma entidade do mundo real, sendo caracterizado por meio de suas propriedades. Tais propriedades são catalogadas por meio de atributos (Wand, 1989). A partir de um referencial de informações (*frame-of-reference*), é possível localizar atributos que distinguem os diferentes objetos. Esses atributos são utilizados para descrever os objetos. Entretanto, esta visão ontológica da idéia de objeto, não considera que além de "existir" em um mundo real, os objetos também "atuam" sobre esse mundo real. Assim, um conceito matemático de objeto deve, além de descrevê-lo quanto ao seu aspecto existencial, modelar o aspecto ativo destes.

Introduz-se a seguir, os conceitos intuitivos sobre objetos e redes de objetos que serão formalmente modelados mais a frente.

3.3.1 Características dos Objetos

A conceitualização da idéia de objeto não pode, em princípio, ser feita de maneira independente. Apesar de podermos imaginar a existência de um objeto por si só, devemos considerar também sua capacidade de interação com outros objetos. Assim, para introduzir os principais conceitos sobre objetos, precisaremos nos referir a sistemas de objetos.

Um sistema de objetos é um conjunto de entidades que existem e interagem entre si. Assume-se que a existência e a interação entre objetos seguem os seguintes princípios:

- Os objetos são únicos e identificados por seu nome.
- Cada objeto possui um conjunto de atributos e/ou partes.
- Um objeto pode possuir um conjunto de funções de transformação.
- Um objeto do sistema pode consumir outro objeto do sistema.
- Um objeto do sistema pode gerar outro objeto do sistema.
- Os objetos podem ser classificados hierarquicamente em função de seus atributos e funções de transformação.
- A interação entre objetos se limita ao consumo e geração de novos objetos por objetos do sistema.

Os objetos são únicos e identificados por seu nome

Por esse princípio se afirma a existência e unicidade de um objeto do sistema. Assim, dois objetos podem ser iguais em quase tudo, mas devem ter um nome diferente, de modo que sejam identificados como dois objetos distintos

Cada objeto possui um conjunto de atributos e/ou partes

Por esse princípio, seguindo-se a definição de seu nome, um objeto passa a ser caracterizado. Os atributos são características em diferentes domínios que são atribuídos ao objeto. Esses domínios denominam-se referencial de informações (*frame-of-reference*). Além de atributos, os objetos podem ser constituídos por partes. Partes são outros objetos, que por sua vez podem ser constituídos por atributos e partes. Observa-se que a definição é recursiva. Para finalizar a recursão, deve existir em algum ponto desta cadeia, objetos que não contenham partes, ou seja, sejam definidos somente por atributos.

Um objeto pode possuir um conjunto de funções de transformação

Por este princípio, caracteriza-se a interação de um objeto em um sistema de objetos. Um objeto **pode** possuir um conjunto de funções de transformação. Não necessariamente ele as possuirá. Caso um objeto possua funções de transformação, ele é denominado um **objeto ativo**. Caso não as possua, é chamado um **objeto passivo**. Uma função de transformação é uma função que tem por domínio o conjunto de atributos e partes de objetos externos ao objeto e o conjunto de atributos e partes internos ao objeto. O contradomínio de uma função de transformação será o conjunto de atributos e partes internos ao objeto, bem como o conjunto de atributos e partes de objetos externos. Uma função de transformação, sendo executada, poderá alterar os atributos internos do objeto, bem como os atributos internos dos objetos que são partes do objeto.

Um objeto do sistema pode consumir outro objeto
Um objeto do sistema pode gerar outro objeto

Esses princípios, em conjunto com o conceito de função de transformação, definem o caráter ativo do objeto no sistema. Um objeto ativo, pode consumir objetos do sistema, modifica-los por meio de suas funções de transformação, e gerar novos objetos no sistema. Do mesmo modo que no caso anterior, um objeto **pode** consumir e/ou gerar outros objetos. Não necessariamente o fará. Somente os objetos ativos é que consomem e/ou geram novos objetos. Os objetos passivos simplesmente existem, podendo ser consumidos e gerados por objetos ativos. Observe que um mecanismo adequado para a seleção de quais objetos devem ser consumidos deve ser implementado. Esse mecanismo deve levar em consideração as funções de transformação e algum critério de seleção, caso mais de um objeto possa ser consumido em um instante de tempo.

Objetos ativos que consomem objetos mas não geram nenhum outro objeto (ou seja, cujo contradomínio de suas funções de transformação correspondem somente a atributos internos) são chamados de **objetos vertedouros**. Analogamente, objetos ativos que somente geram objetos, não consumindo nenhum outro objeto (ou seja, cujo domínio de suas funções de transformação correspondem somente a atributos internos) são chamados de **objetos fonte**.

Os objetos podem ser classificados hierarquicamente em função de seus atributos e funções de transformação

Por esse princípio, cria-se uma taxonomia e classificação dos objetos. Essa classificação é corporificada pela idéia de classe (ou tipo). Assim, cria-se uma relação de equivalência entre objetos que possuam um mesmo referencial de informações, um mesmo conjunto de partes e um mesmo conjunto de funções de transformação. Uma classe ou tipo, passa a ser representada por uma lista de domínios de atributos, partes e funções de transformação, sendo que todo objeto que possuir domínios semelhantes é dito pertencer a tal classe. Dois tipos de hierarquia são identificados. O primeiro é a hierarquia das partes, ou seja, um objeto é parte de outro, ou tem outro como parte sua. Outra hierarquia é a hierarquia de tipo, ou seja, caso um objeto tenha os mesmos domínios de atributos e partes, e mesmas funções de transformação de uma classe, e além disso possua ainda **outros** domínios de atributos e partes ou funções de transformação, ele será um objeto de uma nova classe, que herda as características da classe anterior (chamada sua super-classe), além de incorporar novas características. Do mesmo modo, um objeto pode herdar características de mais de uma classe. Nesse caso, diz-se existir uma herança múltipla.

A interação entre objetos se limita ao consumo e geração de novos objetos

Por esse princípio, assume-se que a dinâmica de um sistema de objetos é definida pelo mecanismo de consumo e geração de objetos pelos objetos ativos do sistema. Em outras palavras, nenhum mecanismo adicional é necessário para que o sistema evolua dinamicamente no tempo.

Em (Snyder, 1993), são referenciadas diversas propriedades que, em princípio, caracterizam um objeto. Essas propriedades serão discutidas a seguir, mostrando-se que todas elas estão encapsuladas pelas propriedades propostas anteriormente. Os conceitos utilizados por Snyder podem ser mais propícios quando se utiliza uma abordagem de programação, mas restringem um pouco a idéia de objeto, que em nossa proposição, se torna mais genérica. Por exemplo, utilizando apenas os conceitos de Snyder, fica difícil descrever uma rede de objetos (que será introduzida mais a frente) como um modelo matemático para sistemas dinâmicos. Segundo Snyder, os conceitos fundamentais para identificar um objeto são:.

Os objetos são abstrações

Os objetos matemáticos que queremos modelar são abstrações de objetos reais, ou objetos imagináveis pela mente humana. Ou seja, os objetos "matemáticos" são modelos dos objetos físicos (Aqui, objeto físico significa ou um objeto real ou um objeto que poderia ser real, isto é, imaginado pela mente humana. Essa colocação é posicionada em contraposição a um objeto matemático, que deverá ser o modelo do objeto físico). Esse preceito é observado pelo nosso modelo, uma vez que estamos caracterizando um objeto por meio de suas sub-partes e atributos, que constituem o referencial de informação pelo qual entende-se um objeto físico. A identificação de um objeto físico por meio das habilidades cognitivas do ser humano se dá exatamente por meio deste referencial de informação. Observe-se que o

modelo será tão mais acurado quanto for o nível de detalhamento da descrição do objeto, em função de seus atributos, partes e funções de transformação.

Os objetos provêm serviços

Em nossa asserção, ao contrário de Snyder, os objetos podem **ou não** prover serviços. Como serviços, entendemos ou a geração de outros objetos no sistema, ou a modificação de atributos internos ao objeto, que podem ser de um caráter geral. Esta proposição é mais genérica que a de Snyder pois este conceitualiza somente objetos ativos, enquanto que em nosso enfoque admitimos, em adição, a existência de objetos passivos. Do mesmo modo, na asserção original de Snyder, ele ratifica a atuação de objetos no sistema quanto a sua utilidade: provêr serviços. Nossa asserção é mais cautelosa ao dizer que os objetos geram novos objetos no sistema. Os objetos gerados ou as modificações em atributos internos ao objeto, podem ser encaradas como um serviço, ou não. De qualquer forma, nossa asserção engloba a idéia de Snyder, tornado-a mais abrangente

Objetos clientes fazem requisições de serviços

Aqui, de novo, nossa proposição é um pouco mais genérica que a de Snyder. Objetos podem gerar outros objetos. Esses outros objetos podem ou não ser requisições de serviços (ou mensagens). Entretanto, a idéia principal desta proposição, que é o encapsulamento, é respeitada (veja a seguir).

Os objetos são encapsulados

Essa propriedade é garantida pela nossa última asserção, ou seja, a de que a interação entre os objetos se limita ao consumo e geração de novos objetos. Assim, se um determinado objeto deseja alguma informação sobre outro objeto, ele deve gerar um terceiro objeto (uma mensagem, ou requisição de informação), que será consumida pelo outro objeto. Este, gerará um outro objeto (uma resposta), que deverá ser consumida pelo primeiro, sendo este portador da informação desejada. Nenhum outro meio de acesso é permitido. Assim, o encapsulamento é garantido.

As requisições identificam os métodos a serem utilizados

Essa propriedade pode ser implementada por meio da definição adequada do objeto-requisição. Quando o objeto-requisição for consumido pelo objeto destino, esse poderá determinar qual função de transformação aplicar. Note-se que existe uma ligação direta entre o conceito de função de transformação e o de método.

As requisições podem referenciar seus objetos de origem

Essa propriedade exige que exista um meio de se referenciar os objetos, e que essa referencia seja utilizada pelas funções de transformação. Essa propriedade é garantida pela proposição de que cada objeto é único no sistema e tem um nome. Portanto, esse nome pode ser utilizado para se fazer referência aos objetos, tanto de origem quanto de destino. Assim, um objeto-mensagem pode conter tanto o nome do objeto que o originou, como o nome do objeto que deve consumí-lo.

Novos objetos podem ser criados

Essa propriedade é equivalente a: "um objeto do sistema pode gerar outro objeto".

Métodos podem ser genéricos

Essa propriedade diz respeito ao uso de métodos virtuais. Em nosso caso, não colocamos nenhuma restrição quanto ao nome das funções de transformação, e portanto pode-se em princípio utilizar o mesmo nome para diferentes funções em diferentes objetos de uma hierarquia. Perde-se aqui, talvez, a questão da obrigatoriedade dos objetos de uma classe filha implementarem um método, ou uma função de transformação. De qualquer forma, essa restrição é colocada a nível de classificação dos objetos e não da sua dinâmica.

Objetos podem ser classificados em termos de seus serviços

Em nossa proposição, os objetos são classificados hierarquicamente em função de seus atributos, partes e funções de transformação (métodos). A idéia de interface (discutida por Snyder) está também associada, além das funções de transformação, aos já citados mecanismos de seleção de consumo dos objetos.

Objetos podem ter uma implementação comum

Essa propriedade relaciona-se com a geração de objetos. Como os objetos podem ser classificados em hierarquias de classes, uma implementação comum corresponde a ter objetos que pertençam a uma mesma classe.

Objetos podem partilhar a implementação parcialmente

Essa propriedade também está relacionada com a geração de objetos. Mais especificamente com a questão da hierarquia de objetos. Assim, um objeto de uma classe e um outro de uma classe que herda a definição da anterior e acrescenta outros atributos, partes ou funções de transformação, partilham parcialmente sua implementação.

Como se vê, as propriedades que se esperam de um objeto, no sentido de modelar a idéia de objetos em estruturas de programação, também são preservadas pela definição conceitual de objeto proposta neste trabalho.

A seguir, detalharemos o mecanismo de consumo e geração de novos objetos.

3.3.2 Atividade dos Objetos

O mecanismo que envolve o consumo de objetos, a alteração de parâmetros internos ao objeto e a geração de novos objetos é relativamente sofisticado.

Cada objeto possui dois tipos de interface: interface de entrada e interface de saída. A interface de entrada é caracterizada por uma série de portas, chamadas de portas de entrada, por onde os objetos que vão ser consumidos adentram. Essas portas são seletivas, ou seja, elas são capazes de selecionar quais objetos podem ou não ser consumidos. A seleção não leva em consideração a disponibilidade de um determinado objeto ou tipo de objeto *de per si*, mas a **disponibilidade simultânea de todos os objetos necessários para disparar uma função de transformação**.

Assim, as portas ficam monitorando o sistema, até que todos os objetos necessários para ativar uma função de transformação estejam presentes. Nesse instante, se diz que a transformação está habilitada. Então, uma vez que o mecanismo de seleção assim o determine, a função de transformação é executada. A execução da função de transformação implicará na modificação dos atributos do objeto e na geração de novos objetos para o sistema. Os objetos criados são liberados para o sistema por meio de portas de saída, que constituem a interface de saída do objeto.

Uma analogia deste procedimento com sua implementação em uma linguagem orientada a objeto é a seguinte. O procedimento de consumo e geração de novos objetos por parte de um objeto correspondem, em uma linguagem orientada a objeto, à chamada de um método do objeto. Cada um dos parâmetros passados à chamada do método corresponde a um objeto que vai ser consumido. O programa que vai efetuar a chamada ao método deve então determinar o valor destes parâmetros. O processo de armazenar cada um dos parâmetros em uma variável corresponde, em nossa especificação, à criação de um objeto-mensagem (passivo) do sistema, para cada parâmetro, habilitando o objeto desejado a disparar. A chamada da função-método do objeto, corresponde à determinação da função de seleção para o objeto habilitado neste instante, gerando o disparo do mesmo.O mecanismo de disparo irá invocar uma função de transformação do objeto, tendo como parâmetros os objetos-mensagem. A função de transformação poderá utilizar-se também de variáveis internas do objeto para sua execução. Do mesmo modo, poderá armazenar valores em variáveis internas do objeto. Uma vez executada, a função de transformação pode ou não gerar uma saída, que em nosso modelo corresponde à geração de um ou mais objetos-mensagem, que são portadores de mensagens cujos valores são os retornados pelo método do objeto. Observe algumas particularidades de nosso modelo. O disparo de objetos está intimamente ligado às funções de seleção, que determinam quais objetos vão ser consumidos, quais vão ser gerados e que função vai ser executada para cada objeto ativo do sistema. Por meio da implementação adequada das funções de seleção, os objetos podem tornar-se entidades autônomas, ou seja, independentes de um mecanismo externo de sincronização. Nesse caso, existindo os objetos que satisfaçam o mecanismo interno de seleção do objeto, estes são consumidos. Do mesmo modo, utilizando as funções de seleção adequadas, pode-se implementar diversos mecanismos de sincronismo externos (que em alguns casos podem ser importantes, e.g. quando dois objetos diferentes querem consumir o mesmo objeto). Por exemplo, pode-se efetuar um ajuste conjunto dos mecanismos de seleção para cada objeto, evitando que dois objetos diferentes queiram consumir um mesmo objeto, ou que um objeto ativo que esteja consumindo um objeto em um instante n, também esteja sendo consumido no instante n, o que seria uma grave inconsistência. Outra possibilidade é o uso de mecanismos de seleção por endereço, quando um campo do objeto-mensagem claramente identifica qual objeto deve consumi-lo. Esta é a maneira utilizada para a chamada de métodos em um programa orientado a objeto. Esse mecanismo de seleção pode ser utilizado para criar objetos sincronizadores, gerados somente no momento apropriado para que os objetos consumam e disparem uma função de transformação adequada. Com isso, nenhum objeto está habilitado a disparar, até que exista um objeto sincronizador (possivelmente criado por um objeto controlador central). Entretanto, da maneira como o modelo encara o objeto, não há necessidade que exista o sincronismo. Com isso, o comportamento de objetos reais, com atividades assíncronas e em paralelo, pode ser modelado. Observe também, que em nosso modelo, os objetos consumidos e gerados não são necessariamente passivos. Isso possibilitará que um sistema de objetos altere sua

própria estrutura, podendo ser usado para modelar sistemas adaptativos e auto-organizáveis, especialmente sistemas com características de aprendizado.

3.3.3 *Definição Formal de Objeto*

DEFINIÇÃO 3.14 - Classe

Uma **classe** C é um conjunto cujos elementos c_i são ênuplas do tipo:

$$(v_1, v_2, \ldots, v_n, f_1, f_2, \ldots, f_m), \; n \geq 0, \; m \geq 0$$

onde $v_i \in V_i$, e f_j são funções

$$f_j : \bigtimes_{p \in P_j} V_p \rightarrow \bigtimes_{q \in Q_j} V_q$$

onde $\times$ corresponde ao produto cartesiano, $P_j \subseteq \{1, \ldots, n\}$ e $Q_j \subseteq \{1, \ldots, n\}$ são definidos para cada função f_j, e para cada ênupla $(v_1, v_2, \ldots, v_n) \in V_1 \times \ldots \times V_n$ deve existir uma ênupla correspondente em C.

DEFINIÇÃO 3.15 - Objeto

Seja C uma classe não vazia. Seja c o nome de uma variável de tipo C. A variável c é chamada então de um **objeto** da classe C.

Deve-se observar que a definição de um objeto pode encampar outros objetos. Uma vez que um objeto é uma variável cujos valores são ênuplas, que possui elementos v_i pertencentes a conjuntos V_i, se esses conjuntos V_i forem, por sua vez, classes, então uma variável de tipo V_i também será um objeto. Nesse caso, dizemos que tal objeto é uma **parte** do outro objeto. Assim, podemos ter objetos que são constituídos por partes, que por sua vez também são objetos.

Do mesmo modo, observe que pela definição, pode-se ter n=1 e m=0. Para este caso, a ênupla se reduz a um único elemento. Assim, uma variável é também um objeto. Uma variável nestas condições é chamada de um **objeto primitivo**. Observe que existe, também, uma classe vazia, quando n=0 e m=0, mas pela definição de objeto, não pode existir um objeto desta classe.

Do mesmo modo, uma estrutura, ou variável composta, também é um objeto. Veremos mais adiante, que uma estrutura também é chamada de um objeto passivo.

A questão da unicidade de um objeto está relacionada ao nome atribuído a este. O nome de um objeto caracteriza um identificador que é único e exclusivo para cada objeto. Assim, dois objetos podem ser exatamente iguais, porém, possuindo nomes distintos, serão objetos distintos. Em alguns casos, pode ser interessante incluir o nome na própria definição do objeto. Nesse caso, um dos conjuntos V_i pode ser um conjunto de nomes, sendo o correspondente elemento v_i, que não pode variar em função do tempo, um nome que é único e atribuído exclusivamente a cada objeto.

DEFINIÇÃO 3.16 - Instância de um Objeto

Seja c um objeto de uma classe C. Define-se como a **instância de um objeto** em um instante n o valor de c nesse instante: c(n). Lembrando-se que C é um conjunto de ênuplas, a instância de um objeto será um elemento de C, no caso, uma ênupla. Observe que a instância de um objeto c em um instante n é um elemento de C.

DEFINIÇÃO 3.17 - Superclasse e Subclasse

Seja C uma classe. Um conjunto D cujos elementos são sub-ênuplas dos elementos de C, todas pertencentes a um mesmo universo, de tal forma que para cada elemento em C corresponda um elemento em D, e D é uma classe, é chamado uma **superclasse** de C. Nesse caso, C é chamada de **subclasse** de D. Observe que uma classe pode ser definida a partir da definição de uma ou mais classes primitivas. Lembrando que uma classe é uma relação, uma nova classe pode ser gerada pela extensão cilíndrica de uma classe, pela junção de diversas classes ou mesmo pela extensão cilíndrica da junção de diversas classes. Em todos esses casos, as classes primitivas são superclasses da classe originada.

PROPOSIÇÃO 3.1 - Hierarquia de Classes

As definições de classe, projeção, extensão cilíndrica e junção induzem uma hierarquia entre as classes, onde a projeção de uma classe que também é uma classe corresponde a uma superclasse desta. De maneira reversa, a partir de uma classe e de sua extensão cilíndrica, obtém-se uma subclasse da classe original. Analogamente, a junção de duas classes é uma subclasse de ambas classes originais. Observe que qualquer classe é subclasse da classe vazia (vide exemplo na figura 3.1).

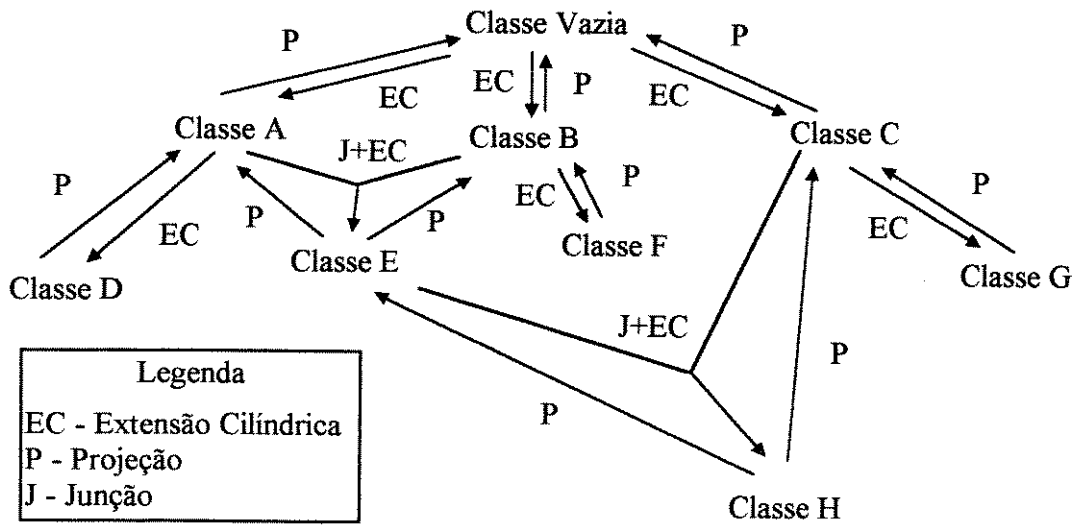


Figura 3.1 - Exemplo de Hierarquia de Classes

DEFINIÇÃO 3.18 - Sub-objeto

Seja c um objeto de uma classe C e d um objeto de uma classe D, que é uma superclasse de C. Se para todos os instantes n, as instâncias de d corresponderem às sub-ênuplas das instâncias de c, d é chamado de um **sub-objeto** de c.

Matematicamente, d corresponde à projeção livre de c em $N \times D$, isto é, $d = c \downarrow N \times D$.

DEFINIÇÃO 3.19 - Objetos Ativos e Passivos.

Um objeto c de uma classe C é chamado de um **objeto ativo** se na definição de C, $m > 0$. Se $m = 0$, c é chamado de **objeto passivo**.

DEFINIÇÃO 3.20 - Interface de Entrada

Sejam c um objeto ativo de uma classe C e I uma superclasse de C, definida por:

$$I = \bigtimes_i V_i \ , \forall i \in \{1,\ldots,n\} \text{ tal que} \begin{cases} \exists f_j, 1 \le j \le m \text{ onde } i \in P_j \\ \forall f_l, 1 \le l \le m \ i \notin Q_l \end{cases}$$

Define-se a interface de entrada i do objeto c, como o objeto passivo gerado pela projeção livre de c em $N \times I$, ou seja, $i = c \downarrow N \times I$.

DEFINIÇÃO 3.21 - Interfaces de Entrada Específicas a Função

Sejam c um objeto ativo de uma classe C, i a interface de entrada de c e I^j uma superclasse de I e de C, tal que:

$$I^j = \bigtimes_i V_i \ , \forall i \in \{1,\ldots,n\} \text{ tal que p/ } f_j \quad i \in P_j \text{ e } \forall l \in \{1,\ldots,m\}, \ i \notin Q_l$$

Define-se a interface de entrada específica à função j de c, i^j como a projeção livre de c em $N \times I^j$. Observe que $i^j = c \downarrow N \times I^j = i \downarrow N \times I^j$ e que para um objeto c de uma classe C, tendo a classe C m funções, existem m interfaces de entrada específicas a função. Cada i^j é um sub-objeto de i e de c.

DEFINIÇÃO 3.22 - Interface de Saída

Sejam c um objeto de uma classe ativa C e O uma superclasse de C, definida por:

$$O = \bigtimes_i V_i \ , \forall i \in \{1,\ldots,n\} \text{ tal que} \begin{cases} \exists f_j, 1 \le j \le m \text{ onde } i \in Q_j \\ \forall f_l, 1 \le l \le m \ i \notin P_l \end{cases}$$

Define-se a interface de saída o do objeto c, como o objeto passivo gerado pela projeção livre de c em $N \times O$, ou seja, $o = c \downarrow N \times O$.

DEFINIÇÃO 3.23 - Interfaces de Saída Específicas a Função

Sejam c um objeto de uma classe ativa C, o a interface de saída de c e O^j uma superclasse de O e de C tal que:

$$O^j = \bigtimes_i V_i \ , \forall i \in \{1,\ldots,n\} \text{ tal que p/ } f_j \quad i \in Q_j \text{ e } \forall l \in \{1,\ldots,m\}, \ i \notin P_l$$

Define-se a interface de saída específica à função j de c, o^j como a projeção livre de c em $N \times O^j$. Observe que $o^j = c \downarrow N \times O^j = o \downarrow N \times O^j$ e que para um

objeto c de uma classe C, tendo a classe C m funções, existem m interfaces de saída específicas a função. Além disso, cada o^j é um sub-objeto de o e de c.

DEFINIÇÃO 3.24 - Existência de um objeto

Um objeto c é dito existir em um instante n, se a função que mapeia as instâncias de c em C é definida para $n \in N$.

DEFINIÇÃO 3.25 - Geração e Consumo de Objetos

Um objeto é dito **gerado** em um instante n, se ele não existe em n e existe em n+1. Um objeto é **consumido** em n, se ele existe em n e não existe em n+1.

DEFINIÇÃO 3.26 - Escopo Habilitante de uma Função

Sejam:

- um objeto ativo c de uma classe $C = \{ (v_1, v_2, \ldots, v_n, f_1, f_2, \ldots, f_m) \}$.
- uma função f_j, componente dos elementos de C e i^j a interface de entrada específica à função f_j.
- a aridade β das instâncias de i^j.
- uma **função de indexação de entrada** gi para a função f_j, $gi : \{1, \ldots, \beta\} \rightarrow \{1, \ldots, n\}$ que mapeia cada componente das instâncias da interface de entrada específica à f_j em uma componente nas instâncias de c.
- $B = \{0,1\}$.

Um **escopo habilitante** para esta função será um conjunto de ênuplas $H = \{(h_t, b_t)\}$, $t = 1, \ldots, \beta$, onde h_t é um objeto de classe $V_{gi(t)}$ e $b_t \in B$ é um valor indicando se o objeto h_t deve ($b_t = 1$) ou não ($b_t = 0$) ser consumido no disparo de c.

DEFINIÇÃO 3.27 - Escopo gerativo de uma função

Sejam:

- um objeto ativo c de uma classe $C = \{ (v_1, v_2, \ldots, v_n, f_1, f_2, \ldots, f_m) \}$.
- uma função f_j, componente dos elementos de C e o^j a interface de saída específica à função f_j.
- a aridade α das instâncias de o^j.
- uma **função de indexação de saída** go para a função f_j.,$go : \{1, \ldots, \alpha\} \rightarrow \{1, \ldots, n\}$, que mapeia cada componente das instâncias da interface de saída específica à f_j em um componente nas instâncias de c.

Um **escopo gerativo** para esta função será um conjunto de objetos $S = \{s_u\}$, $u = 1, \ldots, \alpha$, onde s_u é um objeto de classe $V_{go(u)}$.

DEFINIÇÃO 3.28 - Habilitação de um Objeto Ativo

Um objeto ativo c de uma classe C é dito habilitado em n, se todos os objetos pertencentes a um escopo habilitante de uma de suas funções f_j existem em n. A função f_j é dita estar habilitada em n.

DEFINIÇÃO 3.29 - Disparo de um Objeto Ativo

Sejam:

- um objeto c de uma classe C.
- a instância de c em n, $c(n) = (v_1(n), \ldots, v_n(n), f_1(n), \ldots, f_m(n))$.

• uma função f_j de c em n, habilitada por um escopo habilitante $H = \{(h_t, b_t)\}$.

• um escopo gerativo $S = \{s_u\}$ para f_j, tal que, se $s \in S$, ou s não existe em n, ou $s \in H$.

• o número de valores p para os quais $k \in P_j$, $k = 1, \ldots, n$.

• uma **função de indexação de domínio** $gd : (1, \ldots, p) \rightarrow \{1, \ldots, n\}$ para a função f_j que mapeia para cada componente do domínio de f_j um componente em c.

• a projeção de f(.) em V_k, $f(.)\downarrow V_k$.

• β, α, gi e go conforme anteriormente.

O disparo do objeto no instante n corresponde a:

• Determinação da instância de c no instante n+1, em função da instância de c no instante n e das instâncias de h_t no instante n:

$$v_i(n+1) = \begin{cases} v_i(n) & \text{se } i \notin P_j \text{ e } i \notin Q_j \\ h_{gi^{-1}(i)}(n) & \text{se } i \in P_j \text{ e } i \notin Q_j \\ f_j(w_1, \ldots, w_b)\downarrow V_i & \text{se } i \in Q_j \end{cases}$$

$$\text{onde } w_r = \begin{cases} h_{gi^{-1}(gd(r))}(n), & \text{se } gd(r) \in P_j \\ v_{gd(r)}(n), & \text{se } gd(r) \notin P_j \end{cases}$$

• O consumo, no instante n, dos objetos h_t, $(h_t, b_t) \in H$ tais que $b_t = 1$.

• A geração, no instante n, dos objetos contidos em S e inexistentes em n.

• A determinação do valor das instâncias dos objetos em S, para o instante n+1, em função da instância de c no instante n+1:

$$s_u(n+1) = v_{go(u)}(n+1)$$

DEFINIÇÃO 3.30 - Sistema de Objetos

Sejam:

• c_i objetos de classe C_i, $i = 1, \ldots, \delta$.

• $\mathcal{C} = \bigcup_i c_i$.

• $\Theta_i = \{0, \ldots, m_i\}$, onde m_i é o número de funções do objeto c_i

• $B = \{0,1\}$.

• γ_i, $0 \le i \le \delta$, $\delta > 0$, **funções de seleção** $\gamma_i : N \rightarrow 2^{\mathcal{C} \times B} \times 2^{\mathcal{C}} \times \Theta_i$ as quais para cada objeto c_i, no instante n, selecionam um escopo habilitante H_i, um escopo gerativo S_i e o índice da função a ser executada pelo objeto, tal que, $\forall (c,b) \in H_i$, se $b = 0$, $(\forall k \neq i)((c,1) \notin H_k)$ e se $b = 1$, $(\forall k \neq i)$ $((c,0) \notin H_k$ e $(c,1) \notin H_k)$, $\forall c \in S_i$, $(\forall k \neq i)(c \notin S_k)$ e $(\forall k)((c,1) \notin H_k)$, e H_i é um escopo habilitante e S_i é um escopo gerativo para a função $\gamma_i(n) \downarrow \Theta_i$. Se c_i é um objeto passivo ou, para um determinado n, $\nexists H_i \neq \varnothing$ ou $\nexists S_i \neq \varnothing$, ou ainda, se $(\exists k \neq i)((c_i, 1) \in H_k)$ $\gamma_i(n) = (\varnothing, \varnothing, 0)$. O terceiro índice 0 indica que nenhuma função será executada. Estas condições correspondem às seguintes. Um objeto presente no escopo habilitante não pode ser consumido neste instante por nenhum outro objeto. Ele pode, entretanto, constar do escopo habilitante de outro objeto, desde que não seja

consumido por este. Se o objeto vai ser consumido, ele não pode constar do escopo habilitante de nenhum outro objeto. Se um objeto consta do escopo gerativo de um objeto ativo, ele não pode constar do escopo gerativo de nenhum outro objeto. Do mesmo modo, ele não pode ser consumido por nenhum outro objeto neste instante. Se o objeto é um objeto passivo ou, sendo ativo, não existe um escopo habilitante ou escopo gerativo para uma de suas funções ou, sendo ativo, está sendo consumido no presente instante, necessariamente sua função de seleção deve ser inócua.

Um **sistema de objetos** Ω é um conjunto de pares $\{\omega_i\}$ onde $\omega_i = (c_i, \gamma_i)$, tal que :

- Os objetos c_i sejam funções definidas em um mesmo N.
- Para n=0, exista pelo menos um ω_i com objeto c_i definido.
- Para n>0, todos os objetos ativos c_i com $\gamma_i(n) \neq (\varnothing, \varnothing, 0)$, ou seja, com $\gamma_i(n) = (H_i, S_i, j)$ sejam disparados, conforme o escopo habilitante H_i, e o escopo gerativo S_i, utilizando sua função interna j.
- Para n>0, todos os objetos c_i existentes em n com suas instâncias (n+1) não afetados pelo ítem anterior sejam regenerados: $c_i(n+1) = c_i(n)$.

Observe que a definição de um sistema de objetos corresponde a uma especificação. Para que um determinado conjunto de objetos possa ser considerado um sistema de objetos, os valores das instâncias dos objetos (e mesmo sua existência em diferentes instantes) devem estar associados uns aos outros de acordo com as leis de disparo e regeneração de objetos. Essas leis, a cada instante, determinam os valores das instâncias (ou sua eventual inexistência), em função das instâncias dos objetos do sistema no instante anterior. A determinação dos objetos que interagirão a cada instante, é feita por uma função de seleção, que define escopos habilitantes e escopos gerativos para cada disparo. Essa interdependência entre os objetos é ilustrada na figura 3.2 a seguir:

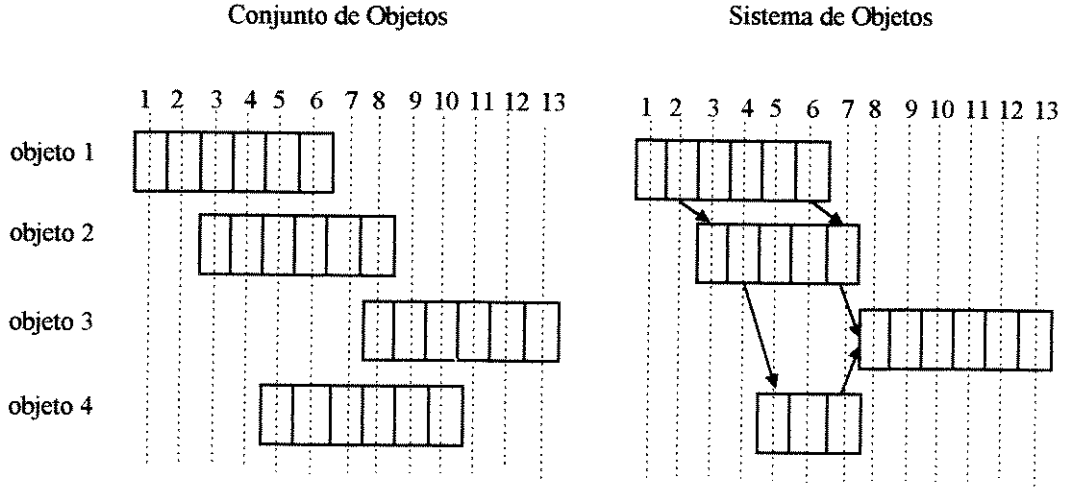


Figura 3.2 - Exemplo de um Sistema de Objetos

Na figura, cada objeto é representado como uma sequência de retângulos, onde cada retângulo corresponde a uma instância temporal do objeto. Para os instantes onde o objeto não é definido, não é desenhado nenhum retângulo. Para o conjunto de objetos, os valores das instâncias não têm nenhuma relação, podendo

ser quaisquer. Para o sistema de objetos, os valores das instâncias nos instantes de tempo sucessivos são determinadas a partir dos valores das instâncias nos instantes anteriores. Um disparo é representado na figura por uma seta. Sendo assim, do instante 1 para o instante 2, não houve disparo, somente regeneração da instância do objeto 1. Do instante 2 para o instante 3, houve o disparo de um objeto (não identificado na figura), utilizando o objeto 1 como escopo habilitante e o objeto 2 como escopo gerativo. O resultado foi a geração do objeto 2, definindo-se o valor de sua instância para o instante 3. Entretanto, o objeto 1 não foi consumido, sendo regenerado para o instante 3. Do instante 3 para o instante 4, também só acontece a regeneração. No instante 5, os objetos 1 e 2 são regenerados e o objeto 4 é gerado, a partir do escopo habilitante formado pelo objeto 2. Do instante 5 para o instante 6 os objetos 1, 2 e 4 são regenerados. Do instante 6 para o instante 7, o objeto 1 é consumido, sendo a instância do objeto 2 determinada pelo disparo de um objeto (não identificado na figura), que tem por escopo habilitante o objeto 1 e como escopo gerativo o objeto 2. Do instante 7 para o instante 8, os objetos 2 e 4 são consumidos e o objeto 3 é gerado. Essas modificaçõe ocorrem devido a um disparo que tem por escopo habilitante os objetos 2 e 4 e como escopo gerativo o objeto 3. Do instante 8 em diante, até o instante 13 (instante final), apenas ocorre a regeneração da instância do objeto 3.

A cada instante, a determinação de quais objetos participarão de quais escopos habilitantes e gerativos, é responsabilidade da função de seleção. Desse modo, a função de seleção tem um papel fundamental na dinâmica de um sistema de objetos. Entretanto, desde que essa função respeite as condições impostas na definição, ela pode ser qualquer uma.

Embora os objetos sejam funções, não se faz, em sua definição, nenhuma restrição adicional de como devem ser estas funções. Assim, em um conjunto de objetos genérico, estas funções podem ser quaisquer. Um sistema de objetos, por outro lado, possui uma natureza claramente recursiva. Essa recursividade não é simples, pois os objetos vistos individualmente podem ser funções parciais, ou seja, não necessitam estar definidos para todo valor em N. Uma propriedade desejável para um sistema de objetos é sua **computabilidade** (Davis, 1958; Epstein & Carnielli, 1989). Uma vez que um sistema de objetos seja computável, pode-se determinar para qualquer instante $n \in N$, o valor das instâncias dos objetos existentes em n, além da inexistência, em n, dos demais objetos do sistema. Entretanto, a natureza recursiva de um sistema de objetos não garante por si só que este seja computável. Para isso, são necessárias algumas condições adicionais, dadas a seguir pelo teorema 3.1.

TEOREMA 3.1 - Computabilidade de um Sistema de Objetos

Seja Ω um sistema de objetos, definido em N.
Se,
- Ω tiver um número finito de elementos ω_i e,
- todas as funções de seleção γ_i de ω_i forem computáveis e,
- todas as funções internas dos objetos c_i de ω_i forem computáveis,

entao Ω será computável.

PROVA: A prova deste teorema ocorre por indução. Suponha que o número de objetos de um sistema de objetos seja P. Considere a função $c_i'(n)$, chamada de objeto estendido, da seguinte forma :

$$c_i'(n) = \begin{cases} c_i(n), & \text{se } c_i \text{ está definida em n.} \\ \varepsilon, & \text{caso contrá rio} \end{cases}$$

Sendo assim, o valor da instância de um objeto estendido c_i' em um determinado instante n é dado por:

$$c_i'(n) = f(c_1'(n-1), c_2'(n-1), \ldots, c_P'(n-1))$$

A função f é determinada ou pelo disparo de algum objeto no instante n-1, ou pela lei de regeneração. Se γ_i é computável e P é finito, é possível determinar-se os escopos habilitantes H_i e gerativos S_i de todos os objetos ativos de Ω para o instante n, bem como a função interna ϕ_i a ser disparada. Se todas as funções internas são computáveis, então ϕ_i é computável. Se (c_i , 1) aparece no escopo habilitante H_j de algum outro objeto c_j , então c_i deve ser consumido, o que significa que ele não existe no instante n. Sendo assim, f(.) = ε. Se c_i aparece no escopo gerativo S_j de algum outro objeto c_j , como a função ϕ_j é computável, então f(.) = ϕ_j (c_{k_1}' , c_{k_2}' ,..., c_{k_m}'), onde c_{k_i}' pertence ao escopo habilitante H_j . Pela definição da função de seleção, cada objeto c_{k_i} pertencente ao escopo habilitante deve existir no instante do disparo, e portanto c_{k_i}' = c_{k_i} e portanto ϕ_j (c_{k_1}' , c_{k_2}' ,..., c_{k_m}') = ϕ_j (c_{k_1} , c_{k_2} ,..., c_{k_m}), Se c_i não aparece nem no escopo habilitante, nem no escopo gerativo de nenhum objeto ativo, então f(.) = c_i (n-1). Deste modo, a função f acima é computável.

Dada uma instância inicial para cada objeto estendido $c_i'(0)$, por indução pode-se calcular recursivamente qualquer instância $c_i'(n)$ destes objetos. Portanto os objetos estendidos c_i' são computáveis. Fazendo-se a associação entre os objetos estendidos c_i' e os objetos c_i , tais que se $c_i'(n) = \varepsilon$ então c_i não está definido para n, então os objetos c_i por sua vez também são computáveis. Como c_i (n) e γ_i (n) podem ser calculados para todo n, caso c_i exista em n, então ω_i também pode ser determinado, o que significa que Ω é computável ■

As condições colocadas pelo teorema 3.1 para garantir a computabilidade de um sistema de objetos são condições suficientes, mas não necessárias. Para ser computável, não necessariamente um sistema de objetos necessita ter um número finito de objetos. Se, para todos os instantes adjacentes n e n+1, o número de objetos existentes tanto em n como n+1 for finito, o sistema é computável. A prova desta afirmativa é análoga à apresentada no teorema 3.1, desconsiderando-se os objetos que não existem nem em n nem em n+1, que não podem nem afetar nem ser afetados pelos disparos em n.

Um sistema de objetos que atenda estas condições pode ser visto como o limite de uma sequência infinita de sistemas de objetos Ω_1 , Ω_2 , ... , onde cada Ω_i tem um número finito de objetos, que estão definidos sobre um domínio N_i finito e incremental, da seguinte forma:

$N_o = \{0\}$
$N_i = N_{i-1} \cup \{i\}$.

Sendo assim, cada sistema de objetos Ω_i é computável, e portanto a sequência é computável. Como o objeto infinito corresponde ao i-ésimo elemento da sequência, onde i é o infinito, por indução tem-se que o sistema de objetos é computável, apesar de infinito.

3.4 Redes de Objetos

Uma rede de objetos é um tipo especial de sistema de objetos, onde algumas restrições são colocadas. Ao contrário de um sistema de objetos, onde qualquer objeto do tipo adequado pode pertencer ao escopo habilitante de um objeto ativo, em uma rede de objetos haverá uma restrição quanto aos objetos que podem fazê-lo. Para implementar estas restrições, algumas condições complementares são definidas. A principal condição é que os objetos estarão associados a lugares. Assim, diz-se que cada objeto, em um instante de tempo, encontra-se em um determinado lugar. Cada objeto possuirá uma lista de lugares, chamados de portas de entrada, que definem os objetos que podem pertencer ao seu escopo habilitante. Mais especificamente, para cada campo da interface de entrada do objeto, haverá um lugar específico associado. Do mesmo modo, cada objeto possuirá uma lista de lugares, chamados de portas de saída, para onde o mesmo deverá enviar os objetos pertencentes ao seu escopo gerativo. Assim como para as portas de entrada, a cada campo da interface de saída haverá um uma porta de saída associada (lugar).

Cada lugar só pode ser ocupado por objetos do mesmo tipo. Assim, para cada lugar está associada uma classe e dois conjuntos de arcos, arcos de entrada e de saída, que conectam os diferentes lugares. Os arcos de entrada, devem vir de lugares cujas classes associadas sejam aquelas correspondentes às classes de cada campo da interface de entrada dos objetos que ocupam o presente lugar. Do mesmo modo, os arcos de saída devem ir para lugares cujas classes associadas sejam aquelas correspondentes às classes de cada campo da interface de saída dos objetos que ocupam o presente lugar. Com isso, quando um objeto é colocado em um lugar, suas portas de entrada corresponderão aos lugares que estão conectados ao lugar onde foi colocado por meio de arcos de entrada. Do mesmo modo, suas portas de saída corresponderão aos lugares que estão conectados ao lugar onde foi colocado por meio de arcos de saída. Assim, objetos ativos somente poderão ter em seu escopo habilitante, objetos que estejam em lugares conectados ao lugar onde se encontram por meio de arcos de entrada. Por sua vez, estes somente poderão enviar os objetos de seu escopo gerativo a lugares que estejam conectados ao lugar onde se encontram por meio de arcos de saída. Fora estas restrições, a dinâmica de uma rede de objetos é semelhante a de um sistema de objetos.

DEFINIÇÃO 3.31 - Rede de Objetos

Sejam:

- um conjunto de classes $\Sigma = \{C_i\}$
- um conjunto de objetos $\mathcal{C} = \{c_i\}$, onde c_i são objetos de classe C_i, $C_i \in \Sigma$, $0 \leq i \leq \delta, \delta > 0$.
- $\Pi = \{\pi_i\}$ um conjunto de lugares π_i
- A um conjunto de arcos $A = \{a_i\}$
- η uma função de nó $\eta : A \rightarrow \Pi \times \Pi$
- ξ uma **função de localização** $\xi : N \times \mathcal{C} \rightarrow \Pi$, que associa para cada objeto $c \in \mathcal{C}$, em um instante n, um lugar π.

•$F(\pi)$ um mapeamento $\Pi \rightarrow 2^{\Pi}$, definido por $F(\pi) = \cup \pi_k$ onde $k \in K$, $K = \{k \mid \exists a_j \in A$ tal que $\eta(a_j) = (\pi_k,\pi) \}$.

• $V(\pi)$ um mapeamento $\Pi \rightarrow 2^{\Pi}$, definido por $V(\pi) = \cup \pi_k$ onde $k \in K$, $K = \{k \mid \exists a_j \in A$ tal que $\eta(a_j) = (\pi,\pi_k) \}$.

• $X(\pi)$ um mapeamento de conexões $\Pi \rightarrow 2^{\Pi}$, tal que $X(\pi) = F(\pi) \cup V(\pi)$.

• $\Xi = \Xi(\pi)$ um mapeamentos de classes $\Pi \rightarrow \Sigma$, de tal forma que $\forall \pi \in \Pi$ para cada campo v_i da interface de entrada de objetos de classe $\Xi(\pi)$, sendo v_i um objeto de classe C, $\exists \pi_k$, $\pi_k \in F(\pi)$, tal que $\Xi(\pi_k) = C$, e para cada campo v_i da interface de saída de objetos de classe $\Xi(\pi)$, sendo v_i um objeto de classe C, $\exists \pi_k$, $\pi_k \in V(\pi)$, tal que $\Xi(\pi_k) = C$.

• i^i a interface de entrada de um objeto de classe $\Xi(\pi_i)$.

• o^i a interface de saída de um objeto de classe $\Xi(\pi_i)$.

• ∂_i o número de campos de i^i e ρ_i o número de campos de o^i .

• a função fpi_i a **função de atribuição de portas de entrada** de objetos que se encontram em um lugar π_i , definida $fpi_i : \{1, \ldots , \partial_i \} \rightarrow A$ e $fpi = \{fpi_i \}$.

• $fpo_i = \{ 1, \ldots , \rho_i \} \rightarrow A$ a **função de atribuição de portas de saída** de objetos que se encontram em um lugar π_i e $fpo = \{fpo_i \}$

• $\Theta_i = \{ 0, \ldots , m_i \}$, onde m_i é o número de funções do objeto c_i

• $\gamma = \{ \gamma_i \}$, $0 \leq i \leq \delta$, $\delta > 0$, cujos elementos são **funções de seleção** $\gamma_i : N \rightarrow 2^{\mathcal{C} \times B} \times 2^{\mathcal{C}} \times \Theta_i$ que para cada objeto c_i , em um instante de tempo n, selecionam um escopo habilitante H_i ,um escopo gerativo S_i e o índice da função a ser executada pelo objeto, tomando-se como restrição que, $\forall (c,b) \in H_i$, $\xi(n,c) = \pi$, $\pi \in F(\xi(n,c_i))$, se $b = 0$, $(\forall k \neq i)((c,1) \notin H_k)$ e se $b = 1$, $(\forall k \neq i)$ $((c,0) \notin H_k$ e $(c,1) \notin H_k)$, $\forall c \in S_i$, $\xi(n,c) = \pi$, $\pi \in V(\xi(n,c_i))$, $(\forall k \neq i)(c \notin S_k)$ e $(\forall k)((c,1) \notin H_k)$. Além disso, H_i deve ser um escopo habilitante e S_i deve ser um escopo gerativo para a função f_k , $k = \gamma_i (n) \downarrow \Theta_i$. Se c_i é um objeto passivo ou, para um determinado n, $\nexists H_i \neq \varnothing$ ou $\nexists S_i \neq \varnothing$, ou ainda, se $(\exists k \neq i)((c_i , 1) \in H_k)$ $\gamma_i (n) = (\varnothing, \varnothing, 0)$. O terceiro índice 0 indica que nenhuma função será executada. Estas condições são análogas às das funções de seleção para sistemas de objetos, acrescentando-se ainda as seguintes condições: Qualquer objeto pertencente ao escopo habilitante de um objeto ativo deve estar em um lugar conectado por um arco de entrada ao lugar onde se encontra o objeto ativo. Qualquer objeto pertencente ao escopo gerativo de um objeto ativo deve ser colocado em um lugar conectado por um arco de saída ao lugar onde se encontra o objeto ativo.

Define-se uma **rede de objetos** $\Re$ como uma ênupla $\Re = (\Sigma, \Pi, \Xi, A, \eta, fpi, fpo, \mathcal{C}, \xi, \gamma)$, tal que:

• um sistema de objetos $\Omega = \{ (c_i , \gamma_i) \}$ seja determinado fazendo-se $c_i \in \mathcal{C}$ e $\gamma_i \in \gamma$, $0 \leq i \leq \delta$, e

• para cada objeto $c_i \in \mathcal{C}$ que tenha uma função f_j disparada no instante n, estando esse objeto no instante n em um lugar $\pi = \xi(n,c_i)$, os objetos s_i^k do escopo gerativo S_i indicado por $\gamma_i (n)$ devem ter uma função de localização definida por:

$$\xi(n+1,s_i^k) = \pi^k$$

onde π^k deve ser tal que $\eta (fpo_\pi (k')) = (\pi,\pi^k)$ e k'é o índice do k-ésimo campo da interface de saída específico à função f_i de c_i referenciado na interface de saída de c_i .

Como no caso de um sistema de objetos, uma rede de objetos pode ser vista como uma especificação, que incorpora toda a trajetória em n dos objetos envolvidos. Assim como para os sistemas de objetos, uma classe importante de redes de objetos corresponde às rede de objetos computáveis. De modo análogo ao desenvolvido para sistemas de objetos, uma rede de objetos computável pode ser determinada iterativamente, a partir de uma sequência de redes de objetos $\Re_0$, $\Re_1$, ... , onde cada $\Re_i$ contém um número finito de objetos, definidos sobre um domínio N_i incremental. Cada rede de objetos da sequência é equivalente à rede de objetos anterior, tendo seus objetos acrescentados de uma instância temporal, de acordo com as leis de disparo e da função de seleção definida para a última instância dos objetos da rede anterior da sequência. Esse procedimento é semelhante ao utilizado para se definir sistemas de objetos infinitos. Por meio desta metodologia, pode-se determinar, para qualquer instante $n \in N$, a instância de qualquer objeto existente em n. A rede de objetos inicial da sequência, $\Re_0$, tem uma denominação especial, sendo chamada de núcleo da rede. Em um núcleo, os objetos e as funções de localização estão definidos somente para a instância n=0 e a função de seleção deve ser computável, sendo descrita por meio de um algoritmo γ'.

DEFINIÇÃO 3.32 - Núcleo de uma Rede de Objetos

Define-se o núcleo de uma rede de objetos como uma rede de objetos $\Re_0 = (\Sigma, \Pi, \Xi, A, \eta , fpi, fpo, \mathcal{C}^0 , \xi^0 , \gamma)$, onde

- Σ, Π, Ξ, A, η , fpi e fpo sejam conforme a definição de rede de objetos e
- $\mathcal{C}^0 = \{c_i' \}$ é um conjunto de objetos definidos apenas para n=0.
- ξ^0 é uma função $\xi^0 : N \times \mathcal{C}^0 \rightarrow \Pi$, definida apenas em n=0.
- γ é uma função de seleção computável, determinada a partir de um algoritmo γ'.

A partir do núcleo de uma rede de objetos, novas redes de objetos, em sequência, vão sendo calculadas. Cada passo do algoritmo, corresponde a gerar um novo elemento na sequência de redes de objetos, que é dado, fundalmentalmente, pela rede de objetos anterior na sequência, acrescentada da definição de novos valores para $\mathcal{C}$, ξ e γ. O novo $\mathcal{C}$ corresponde, basicamente, ao anterior, acrescido dos objetos que devem ser gerados no passo n (o último passo dos objetos no elemento anterior da sequência), de acordo com a função de seleção γ. A função de localização ξ corresponde também à anterior, acrescida de sua definição para o instante n, de acordo com γ. O algoritmo γ' define, incrementalmente, a função γ. Assim, o algoritmo define, passo a passo, novas instâncias para os objetos e novos valores para as funções de localização e seleção, de modo que para cada instante n, obtém-se valores para $\mathcal{C}$, ξ e γ correspondentes a uma nova de rede de objetos finita. Para os casos com n finito, define-se um valor n=final, de tal forma que para n=final, a rede de objetos correspondente ao n-ésimo elemento da sequência é a rede de objetos desejada. Para redes de objeto infinitas (tanto redes com objetos definidos em domínios infinitos como com número de objetos infinito), o mesmo algoritmo pode ser utilizado, de modo a calcular qualquer instância de qualquer objeto da rede. Um exemplo deste método é dado pela figura 3.3 a seguir:

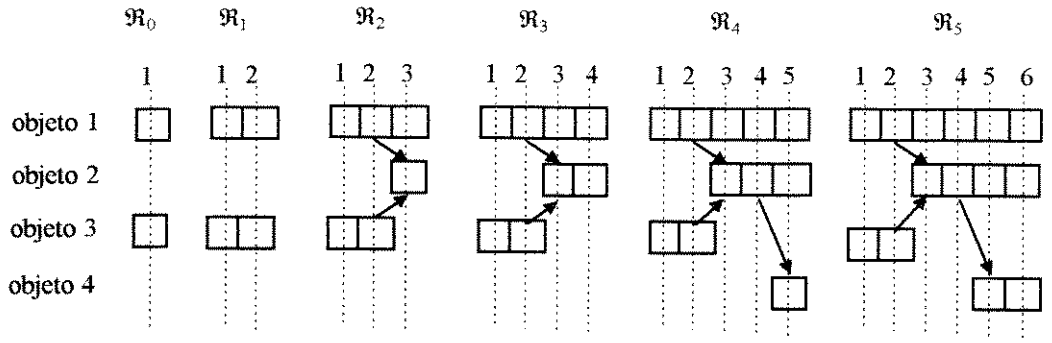


Figura 3.3 - Exemplo de Evolução da Sequência

Exemplos dos principais algoritmos são descritos em pseudo-código, conforme:

procedimento Principal:
{Define-se $\mathcal{C}$, composto pelos objetos c_i dados em $\mathcal{C}^0$.
Define-se a função de localização ξ, composto pelas ênuplas em ξ^0 .
Define-se $\gamma = \varnothing$
Faça n variar de n=0 até n=final
{Aplique γ' para determinar $\gamma(n)$ e atualize γ.
Para todos os objetos ativos c_i existentes no instante n:
{Calcule γ_i (n) = (H_i , S_i , f).
Se $H_i = \varnothing$, vá p/ próximo objeto
Se $H_i \neq \varnothing$:
{execute a função f, gerando uma nova instância c_i (n+1)
atualize a definição de c_i : $c_i = c_i(n) \cup c_i(n+1)$.
Para todo $s_i^k \in S_i$
{Se $s_i^k \notin \mathcal{C}$ gere um novo objeto vazio e acrescente a $\mathcal{C}$
calcule o valor de s_i^k (n+1) a partir de c_i (n+1) e atualize o objeto s_i^k .
determine ξ (n+1,s_i^k) $\in$ V(ξ (n,c_i)) e atualize ξ.
}
}
}
Para todos os objetos c_i tais que (c_i ,1) não consta de nenhum escopo habilitante e c_i não consta de nenhum escopo gerativo.
c_i (n+1) = c_i (n).
}
}

Procedimento γ'
{Para cada objeto ativo c_i
{Para cada função f_j do objeto ativo c_i
{Gere um escopo habilitante vazio para a função f_j
Para cada campo k da interface de entrada correspondentes a f_j
{verifique se no arco apontado por fpo(k) existe:
nenhum objeto, um objeto ou mais de um objeto

Se não existir nenhum objeto, destrua o(s) escopo(s) habilitante(s) e vá p/ próxima função
Se existir apenas um objeto, incorpore-o no(s) escopo(s) habilitante(s).
Se existirem mais de um objeto, para cada escopo habilitante, faça tantas cópias deste quanto forem os objetos e incorpore um objeto a cada cópia.
}
Para cada escopo habilitante, calcule um índice de desempenho
}
}
Para cada objeto ativo c_i
{Faça uma lista ordenada pelo índice de desempenho, contendo a função e o escopo habilitante respectivo
}
Para cada objeto ativo c_i
{Escolha o primeiro elemento da lista como a função e escopo habilitante p/ o objeto
Verifique se a escolha não conflita com as escolhas dos outro objetos.
Se houver conflito, use um critério de desempate. O perdedor passa a escolher o próximo de sua lista.
Se o próprio objeto c_i pertencer ao escopo habilitante de outro objeto, cancele seu escopo habilitante e reorganize a lista.
}
Para cada objeto ativo c_i com escopo habilitante diferente de vazio
{Crie um escopo gerativo vazio para c_i
Para cada campo k da interface de saída específica à função f_j escolhida
{Se houver um objeto em $\mathscr{C}$ não definido para n-1 e n da classe desejada, coloque-o no escopo gerativo, caso contrário crie um novo objeto e inclúa-o.
}
}
Retorne p/ cada objeto, o escopo habilitante, o escopo gerativo e a função escolhidas
}

3.5 Exemplos

3.5.1 Rede de Petri

Como um primeiro exemplo, mostraremos como uma rede de objetos pode representar uma rede de Petri. Utilizaremos a estrutura de rede de objetos para representar a rede de Petri mostrada na figura 3.4.

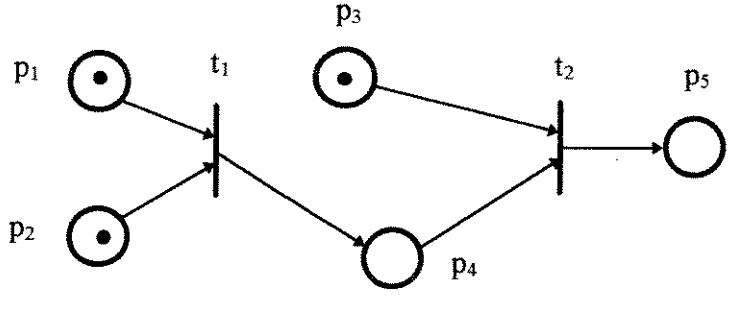


Figura 3.4 - Rede de Petri de Exemplo

Essa rede pode ser entendida como a rede de objetos a seguir. Os lugares contendo objetos ativos são evidenciados por uma circunferência de traço duplo.

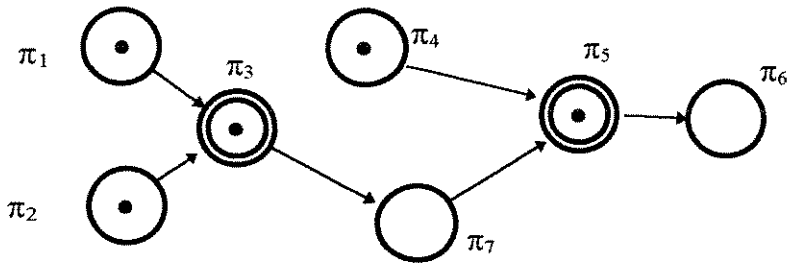


Figura 3.5 - Rede de Objetos correspondente à Rede de Petri

A rede possui 2 objetos ativos, os objetos colocados em π_3 (ou seja, c_1) e π_5 (ou seja, c_2). Os outros objetos da rede são as marcas em π_1 (ou seja c_3), π_2 (ou seja, c_4) e π_4 (ou seja, c_5). Definem-se 2 classes principais. A classe C_1 é a classe dos tokens, $C_1 = \{t\}$. Para o caso deste exemplo, cada objeto da classe token só será caracterizado por uma variável de valor igual a t, constante p/ todo n $\in$ N. A classe C_2 é a classe das transições com duas entradas, e pode assumir objetos do tipo (v_1 , v_2 , v_3 , f_1), onde v_1 e v_2 constituem a interface de entrada, v_3 a interface de saída e $f_1 : C_1 \times C_1 \rightarrow C_1$, f_1 (a,b) = t , ou seja, f_1 é uma função que gera um objeto do tipo token, com valor igual a t.

Nesse caso, o núcleo da rede de objetos pode ser definido:

$\Sigma = \{C_1 , C_2 \}$,
$\Pi = \{ \pi_1 , \pi_2 , \pi_3 , \pi_4 , \pi_5 , \pi_6 , \pi_7 \}$,
$\Xi = \{ (\pi_1 ,C_1), (\pi_2 ,C_1), (\pi_3 ,C_2), (\pi_4 ,C_1), (\pi_5 ,C_2), (\pi_6 ,C_1), (\pi_7 , C_1) \}$
$A = \{ a_1 , a_2 , a_3 , a_4 , a_5 , a_6 \}$,
$\eta = \{ (a_1 , (\pi_1, \pi_3)) , (a_2 , (\pi_2, \pi_3)) , (a_3 , (\pi_3, \pi_7)) ,$
$(a_4 , (\pi_4, \pi_5)) , (a_5 , (\pi_5, \pi_6)) , (a_6 , (\pi_7, \pi_5)) \}$
$fip_{\pi 3} (1) = a_1$
$fip_{\pi 3} (2) = a_2$
$fop_{\pi 3} (1) = a_3$
$fip_{\pi 5} (1) = a_4$
$fip_{\pi 5} (2) = a_6$
$fop_{\pi 5} (1) = a_5$
$\mathcal{C}^0 = \{ c_1 , c_2 , c_3 , c_4 , c_5 \}$,
$c_1 = c_2 = \{ (0, (t,t,t,f_1)) \}$, $c_3 = c_4 = c_5 = \{ (0,t) \}$
$\xi^0 = \{ (0, c_1 , \pi_3), (0, c_2 , \pi_5) , (0, c_3 , \pi_1), (0,c_4 ,\pi_2) , (0, c_5 , \pi_4) \}$
γ' é o algoritmo apresentado.

Para o passo 0, o único objeto ativo habilitado é c_1. De acordo com o procedimento principal e γ' o escopo habilitante H_1 é { (c_3 ,1) , (c_4 , 1) }, e o escopo gerativo S_1 é { c_6 }. Observe que gerou-se um novo objeto c_6 da classe C_1 . A função a ser executada é a função única do objeto, f_1 . Assim, no passo 0, os objetos c_3 e c_4 são consumidos e o objeto c_6 é gerado. No passo 1, o único objeto ativo habilitado é c_2 . De acordo com o procedimento principal e γ', o escopo habilitante H_2 é {(c_5 ,1) , (c_6 , 1) }, e o escopo gerativo é {c_7 }. De novo é gerado

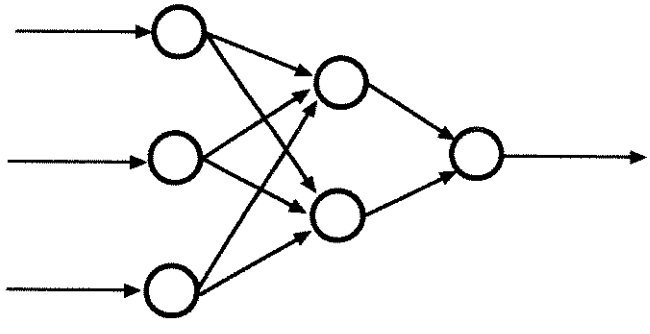
um novo objeto c_7, de classe C_1 . A função a ser executada é a função única do objeto f_1 . Assim, no passo 1, os objetos c_5 e c_6 são consumidos e o objeto c_7 é gerado. Com isso, toda a rede de objetos é determinada:

$c_1(0) = \ldots = c_1(n) = (t,t,t,f_1)$
$c_2(0) = \ldots = c_2(n) = (t,t,t,f_1)$
$c_3(0) = (t)$, $c_3(1) = \ldots = c_3(n) =$ indefinida
$c_4(0) = (t)$, $c_4(1) = \ldots = c_4(n) =$ indefinida
$c_5(0) = c_5(1) = (t)$, $c_5(2) = \ldots = c_5(n) =$ indefinida
$c_6(0) =$ indefinida, $c_6(t) = \ldots = c_6(n) = (t)$
$c_7(0) = c_7(1) =$ indefinida, $c_7(2) = \ldots = c_7(n) = (t)$
$\xi(0, c_1) = \pi_3$
$\xi(0, c_2) = \pi_5$
$\xi(0, c_3) = \pi_1$
$\xi(0, c_4) = \pi_2$
$\xi(0, c_5) = \pi_4$
$\xi(1, c_1) = \pi_3$
$\xi(1, c_2) = \pi_5$
$\xi(1, c_5) = \pi_4$
$\xi(1, c_6) = \pi_7$
$\xi(2, c_1) = \ldots = \xi(n, c_1) = \pi_3$
$\xi(2, c_2) = \ldots = \xi(n, c_2) = \pi_5$
$\xi(2, c_6) = \ldots = \xi(n, c_6) = \pi_7$
$\xi(2, c_7) = \ldots = \xi(n, c_7) = \pi_6$
ξ é indefinido p/ todos os demais valores.
$\gamma_1(0) = (\{(c_3,1), (c_4,1)\}, \{c_6\}, 1)$
$\gamma_2(1) = (\{(c_5,1), (c_6,1)\}, \{c_7\}, 1)$
$\gamma_j(n) = (\varnothing, \varnothing, 0)$ p/ todos os demais valores

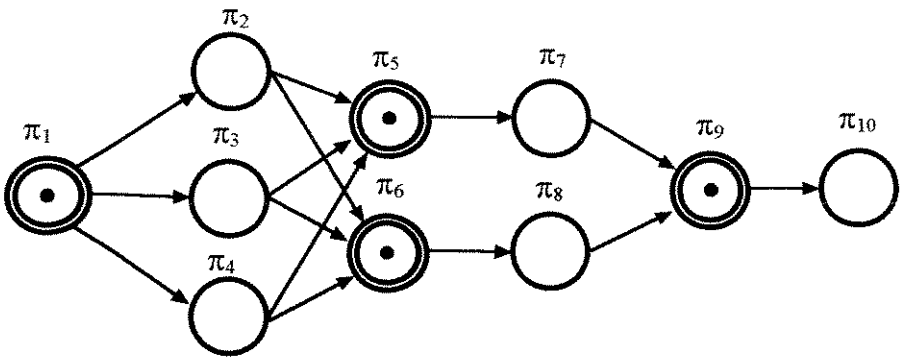
Observe que a partir do núcleo da rede de objetos, e da execução dos algoritmos, obteve-se a definição completa da rede de objetos. Esse exemplo é meramente ilustrativo, mas de um modo geral, qualquer rede de Petri pode ser representada por uma rede de objetos.

3.5.2 Rede Neural

Neste exemplo, mostraremos como utilizar uma rede de objetos para modelar uma rede neural. Seja a rede neural de duas camadas, conforme a seguir:



Para modelar esta rede neural, utilizaremos a rede de objetos a seguir:



As seguintes classes são então definidas:

C_1 - classe dos geradores de amostra. Essa classe contém ênuplas do tipo (v_1 , v_2 , v_3 , v_4 , v_5 , v_6 , v_7 , f_1), onde $v_1 \in \Re$, $\Re$ representando os números reais, é um campo interno, que representa o tempo e $v_2 \in \Re$, $v_3 \in \Re$,$v_4 \in \Re$, $v_5 \in \Re$, $v_6 \in \Re$ e $v_7 \in \Re$ são a interface de saída do objeto. A função f_1 é tal que $f_1 : \Re \rightarrow \Re \times \Re \times \Re \times \Re \times \Re \times \Re \times \Re$ corresponde a uma função que para cada instante de tempo coloca uma entrada diferente na rede neural, e atualiza o campo interno de tempo do objeto. Como dois neurônios irão consumir os objetos concomitantemente, dois objetos iguais são colocados em cada lugar de saída.

C_2 - classe dos números reais = $\Re$

C_3 - classe dos neurônios com três entradas. Essa classe contém ênuplas do tipo (v_1 , v_2 , v_3 , v_4 , v_5 , v_6 , v_7 , v_8 , v_9 , v_{10} , f_1), onde $v_1 \in \Re$, $v_2 \in \Re$ e $v_3 \in \Re$ são os pesos correspondentes à cada entrada do neurônio, e $v_4 \in \Re$ é o offset do neurônio. O campos $v_5 \in \Re$ e $v_6 \in \Re$ são os parâmetros da sigmóide, correspondendo respectivamente à amplitude e à velocidade de subida. Os campos $v_7 \in \Re$, $v_8 \in \Re$ e $v_9 \in \Re$ são a interface de entrada do objeto, representando cada entrada do neurônio. O campo $v_{10} \in \Re$ é a interface de saída do objeto, correspondendo à saída do neurônio. A função $f_1 : \Re \times \Re \times \Re \rightarrow \Re$ pode ser escrita pela seguinte fórmula:

$$f_1(x_1, x_2, x_3) = \frac{A}{1 + \exp\left(m * (w_1 * x_1 + w_2 * x_2 + w_3 * x_3 - \theta)\right)}$$

onde $A = v_5$, $m = v_6$, $w_1 = v_1$, $w_2 = v_2$, $w_3 = v_3$, $\theta = v_4$ são os parâmetros da função.

C_4 - classe dos neurônios com duas entradas. Essa classe contém ênuplas do tipo (v_1 , v_2 , v_3 , v_4 , v_5 , v_6 , v_7 , v_8 , f_1), onde $v_1 \in \Re$ e $v_2 \in \Re$ são os pesos correspondentes à cada entrada do neurônio, e $v_3 \in \Re$ é o offset do neurônio. Os campos $v_4 \in \Re$ e $v_5 \in \Re$ são os parâmetros da sigmóide, correspondendo respectivamente à amplitude e à velocidade de subida. Os campos $v_6 \in \Re$ e $v_7 \in \Re$ são a interface de entrada do objeto, representando cada entrada do neurônio. O campo $v_8 \in \Re$ é a interface de saída do objeto, correspondendo à saída do neurônio. A função $f_1 : \Re \times \Re \rightarrow \Re$ pode ser escrita pela seguinte fórmula:

$$f_1(x_1, x_2) = \frac{A}{1 + \exp\left(m * (w_1 * x_1 + w_2 * x_2 - \theta)\right)}$$

onde $A = v_4$, $m = v_5$, $w_1 = v_1$, $w_2 = v_2$, $\theta = v_3$ são os parâmetros da função.

Definimos então, o núcleo da rede de objetos:

$\Sigma = \{C_1 , C_2 , C_3 , C_4 \}$,

$\Pi = \{ \pi_1 , \pi_2 , \pi_3 , \pi_4 , \pi_5 , \pi_6 , \pi_7 , \pi_8 , \pi_9 , \pi_{10} \}$,

$\Xi = \{ (\pi_1 , C_1), (\pi_2 , C_2), (\pi_3 , C_2), (\pi_4 , C_2), (\pi_5 , C_3),$
$(\pi_6 , C_3), (\pi_7 , C_2), (\pi_8 , C_2), (\pi_9 , C_4), (\pi_{10} , C_2), \}$

$A = \{ a_1 , a_2 , a_3 , a_4 , a_5 , a_6 , a_7 , a_8 , a_9 , a_{10} , a_{11} , a_{12} , a_{13} , a_{14} \}$,

$\eta = \{ (a_1 , (\pi_1, \pi_2)) , (a_2 , (\pi_1, \pi_3)) , (a_3 , (\pi_1, \pi_4)) , (a_4 , (\pi_2, \pi_5)) ,$
$(a_5 , (\pi_2, \pi_6)) , (a_6 , (\pi_3, \pi_5)), (a_7 , (\pi_3, \pi_6)) , (a_8 , (\pi_4, \pi_5)) ,$
$(a_9 , (\pi_4, \pi_6)) , (a_{10} , (\pi_5, \pi_7)) , (a_{11} , (\pi_6, \pi_8)) , (a_{12} , (\pi_7, \pi_9)) ,$
$(a_{13} , (\pi_8, \pi_9)) , (a_{14} , (\pi_9, \pi_{10})) \}$

$fop_{\pi 1} (1) = a_1$
$fop_{\pi 1} (2) = a_2$
$fop_{\pi 1} (3) = a_3$
$fop_{\pi 1} (4) = a_1$
$fop_{\pi 1} (5) = a_2$
$fop_{\pi 1} (6) = a_3$
$fip_{\pi 5} (1) = a_4$
$fip_{\pi 5} (2) = a_6$
$fip_{\pi 5} (3) = a_8$
$fop_{\pi 5} (1) = a_{10}$
$fip_{\pi 6} (1) = a_5$
$fip_{\pi 6} (2) = a_7$
$fip_{\pi 6} (3) = a_9$
$fop_{\pi 6} (1) = a_{11}$
$fip_{\pi 9} (1) = a_{12}$
$fip_{\pi 9} (2) = a_{13}$
$fop_{\pi 9} (1) = a_{14}$

$\mathcal{C}^0 = \{ c_1 , c_2 , c_3 , c_4 \}$,

c_1 é de tipo C_1 : $c_1 = \{ (0, (0,0,0,0,0,0,0,f_1)) \}$

c_2 é de tipo C_3 , $c_2 = \{ (0, (w_{11} , w_{21} , w_{31} , \theta_1 , A_1, m_1 , 0,0,0,0)) \}$

c_3 é de tipo C_3 , $c_3 = \{ (0, (w_{12} , w_{22} , w_{32} , \theta_2 , A_2, m_2 , 0,0,0,0)) \}$

c_4 é de tipo C_4 , $c_4 = \{ (0, (w'_{11} , w'_{21} , \theta_3, A_3, m_3 , 0,0,0)) \}$

$\xi^0 = \{ (0, c_1 , \pi_1) , (0,c_2 , \pi_5) , (0, c_3, \pi_6) , (0, c_4 , \pi_9) \}$

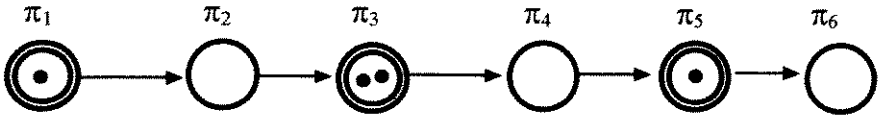
γ' é o algoritmo apresentado.

A partir do núcleo da rede de objetos, obtém-se a definição da rede, por meio da execução dos algoritmos. No passo n=0, somente o objeto c_1 está habilitado (uma vez que não precisa de habilitação, é auto habilitado). Neste instante, criam-se os objetos c_5, c_6, c_7, c_8, c_9 e c_{10} da classe C_2 , por meio da função f_1 de c_1, e atualiza-se o valor interno de c_1 . Observe que dois objetos iguais são colocados nos lugares π_2 , π_3 e π_4 , para que ambos neurônios de 3 entradas possam disparar. No instante n=1, os objetos c_1 , c_2 e c_3 estão habilitados. O objeto c_1 gera uma nova série de amostras, c_{11}, c_{12}, c_{13}, c_{14}, c_{15} e c_{16} . Como os objetos que se encontram nos lugares π_2 , π_3 e π_4 são iguais, utiliza-se como critério de desempate para o algoritmo γ' o índice dos objetos. Assim, o escopo habilitante H_2 torna-se $\{ (c_5 ,1), (c_6 ,1), (c_7 ,1)\}$ e $H_3 = \{ (c_8 ,1), (c_9 ,1) , (c_{10} ,1)\}$. Os objetos c_2 e c_3 são disparados,

gerando dois novos objetos de classe C_2, c_{17} e c_{18} , que são colocados em π_7 e π_8 . No passo n=2, os objetos c_1 , c_2, c_3 e c_4 estão habilitados. Seus escopos habilitantes são então: H_2 = { (c_{11} ,1), (c_{12} ,1), (c_{13} ,1)} , H_3 = { (c_{14} ,1), (c_{15} ,1) , (c_{16} ,1)} e H_4 = { (c_{17} ,1) , (c_{18} ,1)}. Assim, c_1 gera c_{19} e c_{22} em π_2, c_{20} e c_{23} em π_3 e c_{21} e c_{24} em π_4 , c_2 gera c_{25} em π_7, c_3 gera c_{26} em π_8 e c_4 gera c_{27} em π_{10} . Vamos supor que no instante n=3, o objeto c_1 pare de emitir amostras. Com isso, teremos somente os objetos c_2 , c_3 e c_4 habilitados, com escopos habilitantes H_2 = {(c_{19} ,1), (c_{20} ,1), (c_{21} ,1) }, H_3 = {(c_{22} ,1), (c_{23} ,1) ,(c_{24} ,1)} e H_4 = {(c_{25} ,1), (c_{26} ,1)}. Então c_2 gera c_{28} em π_7, c_3 gera c_{29} em π_8 e c_4 gera c_{30} em π_{10} . No instante n=4, somente c_4 está habilitado, com H_4 = {(c_{28} ,1), (c_{29} ,1)}. Então c_4 gera c_{31} em π_{10}, e então nenhum outro objeto fica habilitado daí para frente. Com isso, a rede de objetos fica totalmente determinada.

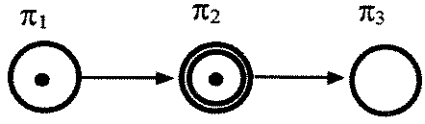
Observe que além de modelar a rede neural, a rede de objetos apresentada ainda implementa o processamento da rede neural em uma estrutura do tipo *pipeline*, aproveitando-se do processamento paralelo de suas fileiras, o que nem sempre é feito nas arquiteturas neurais utilizadas normalmente.

Essa mesma rede neural poderia ser modelada por uma rede de objetos mais sofisticada, como a apresentada a seguir:



Para utilizar esta rede, entretanto, a estrutura dos objetos teria de ser reformulada, de modo a inserir alguns campos sincronizantes, de modo que os dois neurônios em π_3 pudessem discriminar quais os objetos de π_2 a serem consumidos. Isso pode ser feito implementando-se um índice de desempenho diferente p/ cada objeto (para o algoritmo γ'), de modo a direcionar os objetos corretos para os objetos corretos. O mesmo precisaria ser feito em π_4.

Poderiamos ainda modelar o mecanismo de aprendizagem da rede neural, incluindo lugares para a entrada de um valor desejado para a saída, o cálculo do erro, a retropropagação do erro e a alteração dos valores dos pesos e *offsets* dos neurônios. Entretanto, isso não será modelado aqui, pois o propósito deste exemplo é apenas ilustrar o uso do modelo, e não efetivamente modelar uma rede neural. Mas isso pode ser feito. Uma versão extrema da rede neural, seria a seguinte:



onde o objeto em π_1 corresponde à entrada da rede, e a rede neural inteira está encapsulada no objeto em π_2 .

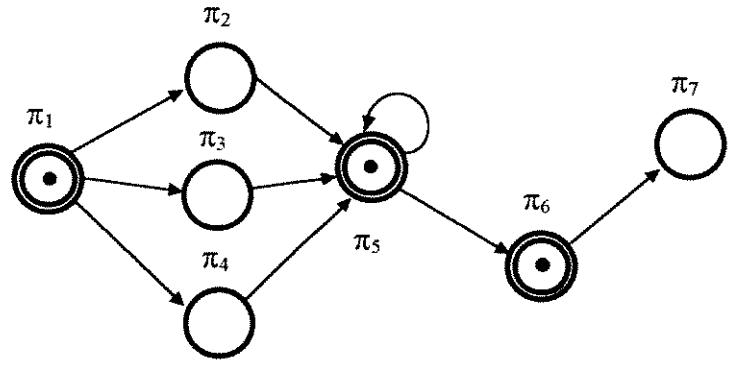
3.5.3 Sistema Adaptativo

Neste tipo de sistema, evidencia-se o poder de representação das redes de objetos sobre outros tipos de modelos. Repare que em todos os exemplos anteriores, utilizou-se uma estrutura de objetos ativos fixa. Ou seja, os lugares com objetos ativos nunca recebiam novos objetos pois os respectivos lugares adjacentes somente continham objetos passivos. Entretanto, a estrutura da rede de objetos

permite a modelagem de sistemas adaptativos, isto é, sistemas cuja estrutura é modificada ao longo do tempo. Ferramentas clássicas de modelagem, tais como a teoria de autômatos finitos e as redes de Petri não são adequadas neste caso. A teoria dos objetos tem um maior poder de representação, permitindo a modelagem de sistemas que modificam sua própria estrutura.

Como exemplo deste tipo de sistema, considere o seguinte problema colocado a título de ilustração.

Seja um tipo de máquina que a partir de um conjunto de peças monta outras máquinas iguais a si próprias, que passam também a montar mais máquinas. Cada máquina, uma vez testada (ou seja, tenha montado uma máquina) é enviada para um buffer de armazenamento, para ser despachada. Esse sistema pode ser modelado pela rede de objetos a seguir:



Esta rede de objetos utiliza as seguintes classes:

C_1 , C_2 , C_3 - Correspondem às classes das peças. $C_1 = \{a\}$, $C_2 = \{b\}$ e $C_3 = \{c\}$. Objetos destas classes funcionam somente como tokens.

C_4 - Classe dos objetos fornecedores de peças. Será utilizado somente um objeto deste tipo, que será colocado em π_1 . Os elementos dessa classe serão ênuplas do tipo (v_1 , v_2 , v_3 , v_4 , f_1), onde $v_1 \in \Re$, $\Re$ representando os números reais, é um campo interno, que representa o tempo e $v_2 \in C_1$, $v_3 \in C_2$ e $v_4 \in C_3$ integram a interface de saída do objeto. A função f_1 é tal que $f_1 : \Re \rightarrow \Re \times C_1 \times C_2 \times C_3$ corresponde a uma função que para cada instante de tempo coloca um conjunto de peças para alimentar as máquinas geradoras de máquinas e atualiza o campo interno de tempo do objeto.

C_5 - Classe das máquinas. Objetos deste tipo terão o comportamento de serem gerados por uma outra máquina, e uma vez existentes, deverão consumir peças e gerar novas máquinas. Por sua vez, serão consumidos quando tiverem gerado sua primeira máquina, sendo armazenados no buffer de saída. Os elementos desta classe serão ênuplas do tipo (v_1 , v_2 , v_3 , v_4 , v_5 , f_1), onde $v_1 \in C_1$, $v_2 \in C_2$ e $v_3 \in C_3$ integram a interface de entrada do objeto, $v_4 \in C_5$ é a interface de saída do objeto, $v_5 \in \{0,1\}$ é um flag interno do objeto indicando se ele já produziu alguma máquina e f_1 corresponde à função $f_1 : C_1 \times C_2 \times C_3 \rightarrow C_5 \times \Re$ de geração do objeto, que a partir das peças monta uma nova máquina e seta um flag interno do objeto indicando que ele já produziu uma máquina.

C_6 - Classe dos objetos despachadores, que verifica as máquinas que já produziram outras máquinas e as armazena em um buffer para serem vendidas. Os elementos desta classe serão ênuplas do tipo (v_1 , v_2 , f_1), onde $v_1 \in C_5$ é a interface de entrada do objeto, $v_2 \in C_5$ é a interface de saída, e $f_1 : C_5 \rightarrow C_5$ é a função de

armazenamento, que transfere os objetos de π_5 para π_7. Observe que, neste caso, a função consome o objeto e o gera novamente, efetuando uma espécie de transporte do objeto, modificando somente o seu lugar.

Definimos então, o núcleo da rede de objetos:

$\Sigma = \{C_1 , C_2 , C_3 , C_4 , C_5 , C_6\}$,

$\Pi = \{ \pi_1 , \pi_2 , \pi_3 , \pi_4 , \pi_5 , \pi_6 , \pi_7 \}$,

$\Xi = \{ (\pi_1 ,C_4), (\pi_2 ,C_1), (\pi_3 ,C_2), (\pi_4 ,C_3), (\pi_5 ,C_5),$
$(\pi_6 ,C_6), (\pi_7 ,C_5) \}$

$A = \{ a_1 , a_2 , a_3 , a_4 , a_5 , a_6 , a_7 , a_8 , a_9 \}$,

$\eta = \{ (a_1 , (\pi_1, \pi_2)) , (a_2 , (\pi_1, \pi_3)) , (a_3 , (\pi_1, \pi_4)) , (a_4 , (\pi_2, \pi_5)) ,$
$(a_5 , (\pi_3, \pi_5)) , (a_6 , (\pi_4, \pi_5)), (a_7 , (\pi_5, \pi_5)) , (a_8 , (\pi_5, \pi_6)) ,$
$(a_9 , (\pi_6, \pi_7)) \}$

$fop_{\pi 1} (1) = a_1$

$fop_{\pi 1} (2) = a_2$

$fop_{\pi 1} (3) = a_3$

$fip_{\pi 5} (1) = a_4$

$fip_{\pi 5} (2) = a_5$

$fip_{\pi 5} (3) = a_6$

$fop_{\pi 5} (1) = a_7$

$fip_{\pi 6} (1) = a_8$

$fop_{\pi 6} (1) = a_9$

$\mathcal{C}^0 = \{ c_1 , c_2 , c_3 \}$, c_1 é de tipo C_4 , c_2 é de tipo C_5 , c_3 é de tipo C_6

$c_1 = \{ (0, (0,0,a,b,c,f_1)) \}$

$c_2 = \{ (0, (a,b,c, NULL ,0,f_1)) \}$

$c_3 = \{ (0, (NULL,NULL,f_1)) \}$

$\xi^0 = \{ (0, c_1 , \pi_1), (0, c_2 , \pi_5) , (0, c_3 , \pi_6) \}$

γ' é o algoritmo apresentado, com o seguinte adendo. Para os objetos da classe C_6, a função de desempenho deve ser nula para os objetos com $v_5 = 0$ no passo corrente. Isso é necessário para desabilitar o consumo de uma máquina que ainda não produziu outra máquina.

A evolução da rede é relativamente simples, podendo ser acompanhada por sua representação gráfica. No instante n=0, apenas o objeto c_1 está habilitado. Ele gera as peças c_4, c_5 e c_6 que serão colocadas em π_2 , π_3 e π_4 . No instante seguinte, essa mesma atividade se repete, mas agora o objeto c_2 também está habilitado, gerando c_7 que é colocado em π_5. Do mesmo modo, c_2 atualiza seu flag interno, indicando que já produziu uma máquina. Com isso, no instante seguinte, além destas atividades (que ocorrem também), o objeto c_3 passa a ser habilitado, transportando c_2 para π_7 . Observe que em π_7 esses objetos passam a ser passivos. A partir daí, o sistema passa a funcionar continuamente. O objeto c_1 gera as peças, os objetos em π_5 montam novas máquinas e o objeto c_3 as transporta para π_7 .

Observe que o sistema descrito aqui é bem atípico, se comparado aos sistemas usualmente encontrados na literatura. O principal produtor, que é o objeto em π_5 é constantemente renovado. Apesar disso, sistemas com esta característica ocorrem frequentemente. A título de exemplo: alunos que se tornam professores e passam a ensinar a novos alunos; programas que geram programas; reações químicas auto-catalíticas; sistemas biológicos onde bactérias geram novas bactérias; vírus atacando células, passando a produzir novos vírus. O tipo de sistema que estamos

particularmente interessados (e que foi a motivação para o desenvolvimento desta teoria), são os sistemas autônomos dotados de aprendizado. Neste caso, a própria estrutura do sistema é alterada de modo a incorporar o conhecimento adquirido em função do aprendizado. Estes sistemas serão modelados posteriormente, utilizando-se as redes de objetos.

3.6 Ferramentas de Análise

O comportamento das redes de objetos pode ser analisado de forma a identificar certas características dos sistemas por elas modelados. Para a descrição das ferramentas de análise disponíveis no momento, algumas definições preliminares são necessárias:

DEFINIÇÃO 3.33 - Classes Ativas e Passivas

Seja C uma classe. A classe C é dita **ativa**, se os elementos de C têm um número de funções m diferente de zero. Se o número de funções m dos elementos de C é igual a zero, a classe é dita **passiva**.

DEFINIÇÃO 3.34 - Lugares Ativos e Passivos

Define-se um **lugar ativo** π, como um lugar tal que $\Xi(\pi)$ é uma classe ativa. Se $\Xi(\pi)$ é uma classe passiva, π é dito um lugar **passivo**.

DEFINIÇÃO 3.35 - Lugares de Parâmetros e Lugares de Estado

Os lugares de uma rede de objetos podem ser divididos **em lugares de parâmetros** e **lugares de estado**. Os lugares de parâmetros são aqueles cujos objetos ocupantes estarão representando parâmetros do sistema sendo modelado. Os lugares de estado são aqueles cujos objetos ocupantes estarão associados a estados do sistema sendo modelado.

DEFINIÇÃO 3.36 - Objetos Móveis e Objetos Imóveis

Seja um objeto c e um lugar π^*.

Se, para todo $n \in N$ em que o objeto c existir, $\xi(n,c) = \pi^*$, então o objeto c é dito um **objeto imóvel**. Caso contrário é chamado de um **objeto móvel**.

DEFINIÇÃO 3.37 - Objetos Constantes

Seja um objeto c de uma classe C. O objeto c é dito constante se, existindo c em um instante n_1, para todo outro instante n em que o objeto exista, $c(n) = c(n_1)$.

Uma primeira avaliação que se faz possível é a análise dos objetos persistentes. Um objeto persistente é um objeto que uma vez gerado, persiste durante toda a trajetória da rede de objetos. A análise dos objetos persistentes é importante, pois estes objetos são elementos característicos da estrutura de um sistema. Mesmo em um sistema adaptativo, onde os objetos ativos podem não ser persistentes, sempre existe um certo número de objetos que são persistentes. São estes objetos que mantém a identidade do sistema, no sentido de que o sistema se modifica, mas uma certa estrutura que caracteriza seu comportamento se mantém. A persistência pode ser de dois tipos. No primeiro destes, chamada de persistência estrutural, a própria estrutura da rede garante a persistência. Este tipo de persistência somente ocorre com objetos ativos. O outro tipo de persistência é

chamado de persistência funcional. Esse tipo de persistência ocorre quando se impõe algum tipo de restrição sobre a função de seleção, de modo que os objetos de um lugar ativo nunca consumam os objetos de um determinado lugar, conectado ao lugar ativo por meio de arcos de entrada.

DEFINIÇÃO 3.38 - Objetos Persistentes

Seja c um objeto de classe C. O objeto c é dito persistente se, caso ele exista em $n_1 \in N$, ele também existe para todo $n > n_1$, $n \in N$.

TEOREMA 3.2 - Teorema Básico da Persistência

Seja c um objeto de classe C. Se $\forall n \in N$ e $\forall c_i \in \mathcal{C}$, $(c,1) \notin H_i(n)$, então o objeto c é persistente.

PROVA: Se (c,1) nunca é membro de um escopo habilitante, então, sendo c uma vez gerado, ele deve sempre existir, pois ele não pode ser consumido por nenhum outro objeto. ■

TEOREMA 3.3 - Teorema da Persistência Estrutural

Seja c um objeto de classe C. Se $\exists n$ tal que $\xi(n,c) = \pi^*$ e $\forall \pi \in V(\pi^*)$, π é um lugar passivo, c é um objeto persistente.

PROVA: Se c se encontra em um instante n em um lugar cujos arcos de saída só apontam para outros lugares com objetos passivos (ou seja, $\forall \pi \in V(\pi^*)$, π é passivo), então não existe nenhum objeto ativo capaz de consumir c. Portanto c é persistente, pois uma vez gerado, ele irá persistir durante toda a trajetória da rede de objetos.■

Uma das primeiras características que se depreende da análise da persistência dos objetos diz respeito ao caráter da adaptabilidade do sistema. Baseado na análise da persistência dos objeto ativos da rede, podemos dizer se se trata de um sistema adaptativo ou não.

DEFINIÇÃO 3.39 - Sistema Adaptativo

Entende-se um **sistema adaptativo** como um sistema que altera seus próprios parâmetros, de modo que seu comportamento seja modificado diante destas alterações. Um sistema que não é adaptativo é dito ser um **sistema não adaptativo**.

TEOREMA 3.4 - Teorema da Adaptabilidade

Seja R uma rede de objetos.

Se todos os objetos c_i da rede que se encontram em lugares de parâmetros são persistentes, imóveis, constantes, e pertencem ao núcleo de R, então o sistema representado por R é **não adaptativo**. Caso contrário, o sistema é **adaptativo**.

PROVA: Se todos os objetos que se encontram em lugares de parâmetros pertencem ao núcleo da rede, são persistentes, imóveis e constantes, então eles já existem no instante n=0, existem $\forall n \in N$, $c_i(n) = c_i(0)$ e $\xi(n,c_i) = \pi^*$ e então nenhum parâmetro da rede é modificado, sendo portanto o sistema não adaptativo. Se algum objeto em um lugar de parâmetros não existir para n=0 e existir para algum $n \neq 0$ (não pertence ao núcleo da rede) ou, se ele existir para n=0, ele deixar de existir para algum outro n (não persistente) ou, for deslocado para outro lugar (móvel) ou, tiver o valor de sua instância modificado (não constante), então os

parâmetros do sistema foram em algum ponto alterados e, portanto, o sistema é adaptativo. ■

TEOREMA 3.5 - 2o. Teorema da Adaptabilidade

Em uma rede de objetos R, se $\nexists$ (i,j) , tal que $(\pi_i , \pi_j) \in \eta$ e π_i e π_j são lugares ativos e, além disso, todos os lugares de parâmetros são lugares ativos, então o sistema é não adaptativo.

PROVA: Se $\nexists$ (i,j) tal que $(\pi_i , \pi_j) \in \eta$ e π_i e π_j são lugares ativos, então $\forall \pi_i$ ativo, $\nexists \pi_j \in V(\pi_i)$, tal que π_j seja um lugar ativo, e portanto $\forall \pi_k \in V(\pi_i)$, π_k é um lugar passivo e portanto, pelo teorema da persistência estrutural, os objetos em π_i são persistentes. Com isso, todos os objetos ativos são persistentes. Do mesmo modo, se $\nexists$ (i,j) tal que $(\pi_i , \pi_j) \in \eta$ e π_i e π_j são lugares ativos, nenhum objeto ativo pode ser gerado durante a evolução da rede. Com isso, os únicos objetos ativos são aqueles que pertencem ao núcleo da rede. Pelo mesmo motivo, eles não podem ser transferidos para outro lugar, sendo portanto imóveis. E também não podem ter os valores de suas instâncias alteradas, sendo portanto constantes. Sendo todos os lugares de parâmetros lugares ativos, pelo teorema 3.4, o sistema é não adaptativo. ■

Observe que no caso do teorema 3.5, o caso contrário não determina que o sistema seja adaptativo. Pode ocorrer, por exemplo, que dois lugares ativos sejam conectados por um arco, mas os objetos que se encontram no vertedouro do arco não possam por algum motivo nem consumir, nem modificar nem mudar de lugar os objetos do outro lugar. Nesse caso, apesar da existência do arco, o sistema é não adaptativo.

Outro fator importante que pode ser analisado diz respeito ao número de objetos que existem a cada instante na rede. Essa análise é importante, pois possibilitará a definição de critérios de estabilidade para os sistemas representados.

DEFINIÇÃO 3.40 - Estado de uma Rede de Objetos

Seja R uma rede de objetos.

Define-se x(n), um vetor onde cada componente $x_i(n)$ corresponde ao número de objetos em um lugar π_i em um determinado passo $n \in N$ como o estado da rede R em n.

DEFINIÇÃO 3.41 - Norma

Define-se a norma de x, $| x |$ como sendo:

$$| x(n) | = \sum_i x_i(n) .$$

Observando a trajetória do vetor x(n), define-se um critério de estabilidade para o sistema representado pela rede R. São identificados 4 tipos de comportamento associados à trajetória x(n):

Trajetória Estável com Ponto de Equilíbrio: Neste caso, a partir de algum n finito, observa-se o seguinte comportamento:

$$x(n+1) = x(n), \quad x_i(n) \neq \infty$$

Trajetória Estável com Ciclo Periódico: Neste caso, observa-se o seguinte comportamento, a partir de algum n finito:

$$x(n+\tau) = x(n) \ , x_i(n) \neq \infty, \ \tau > 1$$

Trajetória Estável Não-Periódica: Nesse caso, não se observa um ponto de equilíbrio nem um ciclo periódico, mas a norma de x se mantém dentro de um intervalo bem definido:

$$a \leq |x(n)| \leq b \ , 0 < a < b < +\infty$$

Trajetória Instável: Nesse caso, a norma de x tende para infinito, quando n tende a infinito:

$$\lim_{n \to \infty} |x(n)| = \infty$$

Todas as trajetórias estáveis correspondem a sistemas que são computáveis a partir de uma quantidade limitada de recursos. Isso significa que é possível estabelecer um programa de computador que, com uma quantidade limitada de recursos de memória, compute qualquer instância temporal finita de um dos objetos da rede. Os sistemas que possuem uma trajetória instável, apesar de computáveis, demandam uma quantidade de recursos constantemente crescente. Portanto, a partir de um determinado instante de tempo, quando os recursos disponíveis se esgotarem, não será possível calcular-se as instâncias posteriores dos objetos da rede. Ou seja, para os sistemas com trajetória instável, os recursos computacionais a disposição irão determinar até que ponto é possível computar uma rede de objetos que modela o sistema. Para instantes de tempo ulteriores a esse limite, as instâncias dos objetos da rede não podem ser determinados.

Observe que para as trajetórias estáveis com ponto de equilíbrio, existem ainda diversos tipos de comportamentos associados. Na definição de estado da rede, utilizou-se apenas o número de objetos por lugar e não os objetos em si e/ou os valores das instâncias dos objetos. Com isso, dependendo do fato do objeto ser persistente ou não persistente, móvel ou imóvel, constante ou não constante, tem-se um diferente tipo de comportamento. Para o caso de objetos persistentes, que se tornem imóveis e constantes, o comportamento é trivial. A partir de um determinado instante de tempo, a atividade cessa na rede. Esse comportamento é usualmente chamado de **deadlock,** ou seja, nenhum objeto se encontra habilitado para disparar, paralisando a atividade da rede. Muitas vezes, tenta-se evitar que esse tipo de comportamento ocorra, embora haja casos em que isso seja esperado. Por exemplo, sistemas cuja atividade tenha um fim em algum ponto. Usualmente, nestes casos, deseja-se que o sistema pare em alguns estados e não em outros. Outro tipo de comportamento, para trajetórias estáveis com ponto de equilíbrio, é quando um ou mais dos objetos nos lugares sejam não-persistentes. Essa situação ocorre quando apesar de, em todos os instantes, o número de objetos em cada lugar permanecer constante, os objetos em si em cada lugar são novos, sendo recém-gerados. Se os objetos forem persistentes, mas móveis, pode-se ter diferentes objetos circulando na rede, sendo que o número de objetos em cada lugar permanece o mesmo. Se eles forem persistentes e imóveis mas não constantes, os valores das instâncias podem estar continuamente sendo alterados. Observe que esses comportamentos, que são parecidos, são completamente diferentes de uma

situação de deadlock. Nestes, a rede não cessa a atividade, pois continuam havendo disparos de objetos ativos, mas simplesmente alcança um tipo de equilíbrio. Nos exemplos apresentados, o primeiro exemplo, da rede de Petri corresponde a um sistema com trajetória estável com ponto de equilíbrio e deadlock, pois a partir de um determinado instante, a atividade do sistema cessa. Os dois outros exemplos (imaginando-se um comportamento diferente para o objeto gerador de entradas no exemplo da rede neural, que não seja finito) correspondem a trajetórias instáveis, uma vez que o numero de objetos no buffer de saída tende a crescer indefinidamente. Esses modelos poderiam ser transformados em sistemas com trajetórias estáveis com ponto de equilíbrio, sem deadlock, simplesmente acrescentando-se um lugar a mais que consumisse os objetos colocados no buffer. Neste caso, após um determinado instante de tempo, o número de objetos em cada lugar torna-se-ía constante, caracterizando uma trajetória estável com ponto de equilíbrio, e sem deadlock, uma vez que a cada passo, um novo conjunto de objetos é gerado. Ainda no caso dos exemplos, os sistemas representados pelos exemplos 1 e 2 são não-adaptativos, uma vez que as condições do teorema 3.4 são garantidas. O sistema esquematizado no exemplo 3, ao contrário, é adaptativo, uma vez que possui objetos ativos não persistentes, de acordo com o teorema 3.3. Um sistema que modelasse uma rede neural, como o exemplo 2, mas incluísse os esquemas de aprendizado da rede, fatalmente teria que alterar os objetos correspondentes aos neurônios, que deixariam de ser persistentes e com isso o sistema seria um sistema adaptativo.

Observe que alguns dos critérios de análise utilizados em redes de Petri (Murata, 1989) são mapeados no critério de estabilidade proposto. Por exemplo, a **limitabilidade** (*boundedness*) está mapeada nas trajetórias estáveis, pois nestas, o número de objetos existentes na rede a cada passo, é limitado à norma do estado da rede. O conceito de **sobrevivência** (*liveness*) também pode ser mapeado nas trajetórias que não tenham ponto de equilíbrio, pois nestas, o estado estará sempre mudando, indicando que pelo menos um objeto ativo da rede foi disparado. Pode também estar mapeado em trajetórias com ponto de equilíbrio, desde que os objetos envolvidos sejam não-persistentes, móveis ou não constantes, quando então, também haverá sempre objetos disparando, apesar do equilíbrio. A **reversibilidade** está mapeada nas trajetórias estáveis com ciclo periódico, pois a partir de um determinado número de passos, retorna-se ao estado inicial. Pode-se ainda abstrair outros conceitos como o da **atingibilidade** (*reachability*), onde se deseja saber se a partir de um estado inicial pode-se atingir um determinado estado por meio de um número finito de disparos de objetos do sistema. O conceito de **seguridade** (*safeness*), está associado à existência de no máximo um objeto em cada lugar da rede, durante toda a trajetória da rede. A **cobertura** (*coverability*) está relacionada com a habilitação de objetos ativos. Um estado está coberto se existe, no conjunto de estados que podem ser atingidos, a partir do estado inicial, um estado onde, para todos os lugares, existe um número maior ou igual de objetos em tais lugares, relativamente ao estado sendo analisado. Assim, se um determinado estado habilita um determinado número de objetos ativos, existe um estado atingível que também habilita estes mesmos objetos ativos. A **não-interferência** ocorre quando para todos os objetos da rede, o disparo de um objeto não desabilita nenhum outro objeto da rede.

Algumas destas propriedades, bem como o comportamento referente ao critério de estabilidade proposto podem, em alguns casos, ser inferidos a partir do núcleo da rede de objetos, não demandando a definição completa da rede.

Por exemplo, redes não adaptativas sem objetos fonte terão sempre uma trajetória estável com ponto de equilíbrio. Redes onde todos os objetos ativos geram um número maior de objetos do que consomem e não ocorre *deadlock*, terão sempre trajetória instável. Redes que para todos os lugares passivos tenham, dentre seus lugares adjacentes de saída (lugares conectados a este por um arco saindo do lugar), somente um lugar ativo, serão sempre não-interferentes. Redes adaptativas onde, para algum lugar ativo, haja um arco realimentando o próprio lugar, e sem nenhum outro lugar ativo adjacente (que possa consumir os objetos gerados), terá uma trajetória instável. Diversos outros exemplos como esses podem ser relacionados, sendo as propriedades inferidas facilmente provadas.

Esses exemplos ilustram a utilidade das ferramentas de análise na classificação e no entendimento do comportamento do sistema, diante do modelo de rede de objeto. As ferramentas de análise apresentadas estão longe de ser completas na caracterização dos sistemas representados por redes de objetos, comparando-se ao modo como é feito com as redes de Petri, sendo colocadas aqui somente a título de ilustração de sua importância e da necessidade de sua definição.

3.7 Discussão e Extensões ao Modelo

Nesse capítulo, apresentaram-se os princípios de uma teoria de objetos, e utilizou-se o conceito de redes de objetos para modelar sistemas. O propósito deste trabalho é duplo. Em primeiro lugar, buscou-se a formalização do conceito de objeto, que é utilizado frequentemente na literatura de ciências da computação, sem uma formalização mais elaborada. Em seguida, com a proposta da rede de objetos, buscou-se uma alternativa para a modelagem de sistemas que exibem um caráter de modificação em sua própria estrutura, o que inviabiliza que sejam modelados por ferramentas clássicas, tais como autômatos finitos e redes de Petri. Neste sentido, o modelo proposto permite que estes sistemas sejam modelados de um modo bem natural.

Apesar do presente modelo estar de acordo com as principais características dos objetos, tais como estes são utilizados em linguagens orientadas a objeto e programação orientada a objeto, existem alguns pequenos pontos onde uma perspectiva mais pragmática poderia impor restrições. Tal como aponta (Wegner, 1995), existe uma grande dificuldade em se encontrar modelos formais para os objetos, pois estes exibem um comportamento que vai além do que pode ser estabelecido em uma máquina de Turing. O problema ocorre quando, e.g. ,um objeto estiver ainda processando uma mensagem e receber outra, que modificaria a resposta da primeira. Isso implicaria em se interromper a execução de uma máquina de Turing, o que é conceitualmente impossível. O modelo apresentado neste capítulo, não apresenta esse problema, por dois motivos. Em primeiro lugar, os objetos não necessariamente são definidos recursivamente. Eles são funções que podem (ou não) ser recursivas. Em princípio, eles não precisam nem ser computáveis. Portanto, não precisam estar teoricamente vinculados a máquinas de Turing. Em segundo, quando definidos recursivamente (em um sistema de objetos, por exemplo), os disparos dos objetos são instantâneos, ou seja, a hipótese de que uma mensagem chegue enquanto outra ainda está sendo processada não é possível. Entretanto, o conceito de disparos instantâneos diminui um pouco o poder de representação do modelo pois, em sistemas reais, os disparos dos objetos não são instantâneos, mas possuem um tempo de latência. Durante esse tempo de latência podem chegar novas mensagens. Para lidar com essa questão, é necessária uma

extensão ao modelo apresentado, que considere na definição de disparo uma temporização dos métodos. Essa extensão pode ser efetuada de duas maneiras possíveis. A extensão mais trivial seria incorporar um número inteiro a cada função que um objeto possui, sendo esse número correspondente ao tempo de latência do disparo dessa função (em instantes de tempo). Observe que nesse caso, pode haver a superposição entre tempos de latência de disparos sucessivos, implicando no problema apontado por Wegner. Entretanto, devido à atemporalidade do modelo, esse problema pode ser facilmente resolvido, desde que alguns critérios de prioridade sejam estipulados, de modo a determinar qual dos disparos deve prevalecer. O problema fica um pouco mais complicado, se for necessária a utilização de tempo contínuo. Nesse caso, todo o modelo necessitaria ser transposto para o tempo contínuo, onde necessariamente os tempos de latência dos disparos teriam uma relevância fundamental. Apesar de não apresentar uma prova formal (que foge um pouco ao escopo desta tese), é sentimento do autor que essas extensões ao modelo não implicariam no problema apresentado por Wegner, ou seja, estas extensões, apesar de não-triviais, seriam um modelo formal de objetos, onde as situações apresentadas por Wegner podem ser modeladas, uma vez que não é usado o conceito de máquina de Turing na formulação do modelo.

Algumas outras questões necessitam de uma maior discussão. Alguns problemas que podem aparecer em aplicações orientadas a objeto, não estão sendo considerados no modelo, pois pretende-se que este se atenha apenas ao aspecto matemático e formal dos objetos. Por exemplo, problemas como os conflitos nos nomes de variáveis e métodos quando ocorre herança múltipla. De acordo com o modelo de herança proposto, esse problema nunca poderia ocorrer, pois os nomes utilizados não são símbolos mnemônicos, mas índices de referência, que são mapeados em novos índices, de acordo com as fórmulas de indução utilizadas nos operadores de junção e extensão cilíndrica. Do mesmo modo, outras questões relacionadas com herança têm uma abordagem muito *sui-generis* no modelo proposto. Por exemplo, a questão da necessidade de se classificar os campos do objeto de acordo com o tipo de herança que se espera destes, tais como a definição de campos e métodos públicos, privados e protegidos, da maneira que é feita no C++. Essa questão não é abordada diretamente no modelo, sendo entretanto factível. Na verdade, a questão da hierarquia não é fundamental para um modelo formal de objetos. Tal como apontado em (Wolczko, 1988), a questão da hierarquia é simplesmente uma conveniência na elaboração das classes, permitindo a reutilização de classes já definidas na definição de novas classes. Entretanto ela não é necessária, do ponto de vista formal. Apesar disso, pode-se facilmente especificar tais características, por meio dos operadores de projeção, extensão cilíndrica e junção. Na verdade, utilizando estes operadores, pode-se inclusive obter uma abstração maior do que a proporcionada pelos conceitos de campos e métodos públicos, privados e protegidos. Pode-se especificar, campo a campo, função a função, quais destes devem ser herdados (utilizando-se uma fórmula de indução conveniente) nas classes filhas. Optou-se portanto, por manter tais questões como extensões ao modelo, uma vez que estas não são fundamentais à formalização da idéia de objeto.

Uma questão mais delicada diz respeito aos objetos compostos. A elaboração de um modelo formal para a composição de objetos (ou seja, dizer-se que um subsistema de objetos pode atuar por si como um objeto) não é um objeto trivial. Em alguns casos (principalmente nos casos apresentados por (Wegner, 1995)), um subsistema de objetos não pode ser formalmente modelado como um único objeto, da maneira como se implementou o modelo. O problema que ocorre, é que

mensagens enviadas às diferentes partes do objeto composto implicariam em mais do que um disparo do objeto composto, em um dado instante de tempo, e no caso dos disparos com tempo de latência, poderiam haver sobreposições de ações. Para contornar esse problema, o modelo atual necessitaria ser extendido, de modo a lidar com tais situações, eventualmente permitindo que um objeto pudesse efetuar mais de um disparo ao mesmo tempo.

Outras questões, poderiam levar a debates conceituais, tais como a questão da destruição dos objetos por consumo. Na maioria das implementações, uma vez que um objeto tenha sido criado, ele não pode ser destruído, a não ser por si próprio. Em nosso caso, isso não ocorre, sendo que o objeto é destruído pelo objeto que o consome. Poderia se dizer que esse fato viola o encapsulamento. Do mesmo modo, o acesso aos valores internos dos campos pelo objeto ativo que está consumindo outro objeto também poderia ser encarado como uma violação ao encapsulamento. De fato, ocorre uma violação ao princípio do encapsulamento, nesses casos, mas essa violação é encarada como uma exceção necessária. Outros autores contornam essa violação não considerando as mensagens como objetos. Uma vez que se considera uma mensagem como um objeto, fica a questão: como destruir uma mensagem sem violar o encapsulamento ? Seria necessário enviar uma mensagem à mensagem pedindo que esta se destrua. E após isso, seria necessária uma mensagem a essa nova mensagem, e assim por diante, gerando uma recursão infinita, o que obviamente é um contrasenso. Entretanto, considerando-se uma mensagem como um elemento diferente de um objeto, perde-se em poder de representação, quando se deseja que qualquer objeto possa ser encarado como uma mensagem, o que é necessário nos sistemas adaptativos que se deseja modelar. Sendo assim, só resta aceitar a exceção no princípio do encapsulamento, nesse caso, ao invés de se considerar as mensagens como exceções aos objetos.

Outra questão polêmica diz respeito às classes primitivas e métodos primitivos. Nesses casos, consideram-se as variáveis numéricas de um modo geral como classes primitivas e os operadores aritméticos e lógicos como métodos primitivos. Observe que, em implementações computacionais, irão existir diferentes tipos de variáveis numéricas, tais como os inteiros de 16 bits, inteiros de 32 bits, números em ponto flutuante de 16 bits, números em ponto flutuante de 32 bits, etc. Apesar de matematicamente considerarmos os operadores aritméticos e lógicos de uma maneira unificada para todas as variáveis numéricas, computacionalmente, estes terão uma implementação completamente diferente, de acordo com o tipo de variável. Em um modelo muito rígido de objetos, os números seriam considerados como objetos, e estes operadores primitivos deveriam ser especificados. Assim, para se efetuar uma soma de dois números a e b, corresponderia a acionar o método de soma do objeto a, e enviar o objeto b como mensagem, gerando um terceiro número (objeto) c. Para os objetos das classes primitivas, entretanto, o tratamento desses casos (com métodos primitivos), nas linguagens orientadas a objeto, não é implementado desta forma. Não vale a pena associar uma porção de funções à estrutura dos objetos primitivos. Apesar disso, o modelo apresentado não faz distinção para classes e métodos primitivos, tal como o fazem as linguagens de programação.

Outro fator específico das implementações dos objetos nas linguagens também não é considerado: A questão do mecanismo de atendimento às mensagens, tais como os métodos síncronos e assíncronos. Algumas linguagens utilizam o método síncrono, outras utilizam o método assíncrono. É sentimento do autor que tal característica é puramente de implementação das linguagens, nada tendo a ver com um modelo formal de objetos, não sendo portanto considerada.

Alguns fatores importantes devem ser destacados no modelo. Devido a sua abordagem, incorporando conceitos tradicionais da programação orientada a objetos, modelos utilizando redes de objetos podem ser facilmente implementados em linguagens de computação orientada a objetos, o que facilita a implementação computacional dos modelos dos sistemas que se deseja estudar. Do mesmo modo a idéia intuitiva do que é um objeto (no sentido cognitivo humano) facilita a elicitação de modelos que poderiam ser de difícil elaboração utilizando-se outro tipo de ferramental matemático.

É necessário ressaltar a semelhança entre as redes de objetos e as redes de Petri. Em princípio, o leitor poderia se sentir impelido a dizer que uma rede de objetos é apenas uma rede de Petri mais elaborada, ou uma extensão de uma rede de Petri (Murata, 1989). De fato, a estrutura de uma rede de objetos é bem semelhante a uma rede de Petri, mais notadamente se examinarmos as redes de Petri de Alto Nível, tais como as redes Predicado-Transição (Genrich & Lautenbach, 1981), as redes Coloridas (Jensen, 1981; Jensen 1990) e as redes de Petri orientadas a objeto (Baldassari & Bruno, 1988; Battiston et. al. 1988; Cardoso et.al. 1990). Entretanto, uma diferença fundamental entre as redes de Petri e as redes de objetos, é que as primeiras não permitem uma estrutura variável. Assim, em uma rede de Petri, somente a marcação pode se modificar com o tempo, mas a estrutura das transições e lugares deve ser fixa. Nas redes de objetos, apesar de tal conceito ser possível, (o que faz com que possamos implementar uma rede de Petri por meio de uma rede de objetos), a maleabilidade com que o conceito de objeto é utilizado é o grande trunfo que permite a representação de sistemas adaptativos e com aprendizado. Nas redes de objetos, não somente os objetos passivos (tokens) podem ser modificados pela dinâmica da rede mas, também, os objetos ativos (o que nas redes de Petri corresponderia à modificação das próprias transições durante a execução da rede). Para formalizar esse conceito, houve a necessidade de se separar os conceitos de rede de objetos e núcleo da rede de objetos. Com isso, a criação e destruição de elementos da estrutura da rede tornou-se possível, o que não é permitido no caso das redes de Petri. Uma comparação com as redes de Petri de *Design Adaptativo* (Zhou & Dicesare, 1989) também não é cabível, neste caso, pois estas na verdade não constituem uma rede que tem seus parâmetros alterados, como no caso das redes de objetos, mas a uma modificação interativa de redes de Petri de modo a melhor representar o sistema sendo modelado. Observe que neste caso, o caráter adaptativo do sistema não é modelado. O que ocorre é simplesmente um *design adaptativo* de uma rede de Petri simples. A primeira tentativa real de se criar redes com caráter adaptativo, vem com as redes auto-modificáveis (Valk, 1978; Valk, 1981). Nestas redes, apesar da estrutura fixa, os parâmetros dos arcos podem ser dependentes do número de tokens em um determinado lugar, fazendo com que o número de tokens consumidos ou gerados por uma transição dependam, no momento da transição, do número de tokens que se encontram em algum lugar da rede. Apesar de ser uma rede adaptativa, este tipo de rede ainda pode ser classificada como uma rede de Petri, pois a estrutura básica da rede, ou seja, os lugares e as transições não são modificados, mas apenas os parâmetros associados aos arcos. Nas redes de objetos, essa idéia de dependência é explorada em toda a sua potencialidade. Ao invés de se ter um parâmetro dependente do número de tokens em um determinado lugar, e esse parâmetro regular o comportamento de uma transição, na rede de objetos o token em um determinado lugar "é " a própria transição. Com isso, modificando-se a estrutura da rede, uma rede de objetos não pode mais ser classificada como uma rede de Petri. Vale a pena ainda mencionar os autômatos adaptativos (José Neto, 1993), que efetivamente efetuam uma

modificação em sua estrutura. Entretanto, os autômatos adaptativos são uma extensão apenas aos autômatos de estados finitos, não permitindo a modelagem de processos com paralelismo e concorrência tal qual as redes de Petri. Nesse ponto, as redes de objetos são ferramentas mais poderosas para a modelagem de sistemas.

Algumas ferramentas de análise foram sugeridas, colocadas a título de exemplo da importância da definição de critérios para a obtenção de características dos sistemas modelados por redes de objetos. Estas ferramentas permitem a obtenção de características dos sistemas, a partir de seu modelo. Com sua definição, as redes de objeto, além de constituirem um modelo formal de sistemas, criam a possibilidade da análise de seu comportamento a partir de seu modelo. Eventualmente, métodos mais rigorosos de análise poderão ser elaborados no futuro, adaptando-se os métodos tradicionais em redes de Petri, tais como a árvore de cobertura (*coverability tree*), e os métodos algébricos (Murata, 1989). Uma possibilidade promissora seria uma adaptação do método dos invariantes (Jensen, 1981; Jensen, 1981b; Genrich & Lautenbach, 1981) à estrutura das redes de objetos. Essa adaptação absolutamente não é trivial, fugindo ao escopo do presente trabalho, mas aparentemente é factível.

Vários outros fatores poderiam ser discutidos nessa seção. Como se disse no começo do capítulo, o propósito aqui é apenas introduzir fundamentos para uma teoria dos objetos, e não estabelecer uma teoria definitiva.

3.8 Resumo

Neste capítulo, introduziram-se os fundamentos para uma teoria dos objetos. Iniciou-se por uma definição formal para o conceito de objeto, a partir de características encontradas nas diferentes aplicações orientadas a objeto disponíveis na literatura. Em seguida, foram definidos os sistemas de objetos e propôs-se um modelo para um tipo especial de sistema de objetos chamado de rede de objetos.

Apresentaram-se alguns exemplos e propuseram-se algumas ferramentas de análise para tais tipos de sistemas. Por fim, efetuou-se uma discussão sobre o modelo proposto, fundamentada em excertos extraídos da literatura.

No próximo capitulo, as redes de objetos serão utilizadas na formalização de modelos de representação para os diferentes tipos de conhecimento apresentados no capítulo 2.

4. Modelo Formal das Estruturas de Representação de Conhecimento

4.1 Introdução

Neste capítulo, apresenta-se uma formalização para os conceitos introduzidos informalmente no capítulo 2, utilizando como base o modelo formal de objetos e redes de objetos.

No capítulo 2, mencionou-se, repetidas vezes, o termo "estrutura", ou ainda "estruturas matemáticas", sem uma descrição mais detalhada do que seriam tais estruturas. A estrutura matemática básica que será utilizada para representar o conhecimento será o objeto matemático desenvolvido no capítulo 3, e em alguns casos as redes de objetos. Para cada objeto matemático, corresponderá um determinado conhecimento, codificado em termos dos parâmetros do objeto. Os conhecimentos remáticos específicos e dicentes específicos serão codificados como objetos passivos. Os conhecimentos argumentativos serão objetos ativos, utilizados para gerar novos conhecimentos no sistema. Sendo assim, a partir de um objeto inicial (representando um determinado conhecimento), novos objetos (novos conhecimentos) serão gerados pelos objetos ativos. Para regular a atividade dos objetos ativos, é necessário delimitar seu escopo de atuação. Para tal, utiliza-se da representação de uma rede de objetos, fazendo com que os conhecimentos argumentativos sejam aplicados somente nos lugares adequados.

A hierarquia dos tipo de conhecimento implicará também em uma hierarquia das estruturas de representação. No nível mais baixo da hierarquia, vem a interface de entrada do sistema inteligente, com os dados dos sensores. Esses dados dos sensores, e possivelmente uma memória destes, será armazenada em uma determinada estrutura. Em seguida, sobre essa estrutura, ficam os conhecimentos remáticos sensoriais específicos. Os conhecimentos remáticos sensoriais genéricos serão expressões que agrupam diversos conhecimentos remáticos sensoriais específicos, estando portanto em um nível hierárquico superior. Aparecem então os conhecimentos remáticos de objeto específicos, que são gerados por abdução sobre os conhecimentos remáticos sensoriais específicos. Os conhecimentos remáticos de objeto genéricos são representações que agregam vários conhecimentos remáticos de objeto específicos. Os conhecimentos remáticos de ocorrência específicos são definidos sobre as estruturas que representam os conhecimentos remáticos de objeto específicos. Os conhecimentos remáticos de ocorrência genéricos agrupam diversos conhecimentos remáticos de ocorrência específicos. Os conhecimentos dicentes são gerados então pelo sequenciamento de diversos conhecimentos remáticos, de acordo com uma regra de formação, e tem seu valor verdade computado sobre as estruturas dos conhecimentos remáticos de objeto específicos. Os conhecimentos argumentativos são estruturas bem mais complexas que os conhecimentos remáticos e dicentes, pois devido ao seu caráter ativo, eles devem compreender partes de estruturas tanto de conhecimentos remáticos, como dicentes. Além destes, os conhecimentos argumentativos possuem um caráter transicional, que permite a

transformação dos conhecimentos de suas premissas nos conhecimentos de suas conclusões.

4.2 Representação do Conhecimento Remático

Nesta seção, serão apresentadas as estruturas de representação para os conhecimentos remáticos, o que inclui os conhecimentos sensoriais, de objeto e de ocorrências.

4.2.1 Conhecimento Remático Sensorial Específico

Este é o tipo de conhecimento mais básico, dentre todos os tipos de conhecimento. Ele compreende um modelo da interface de entrada, restrições da interface de entrada a instantes no tempo específicos, projeções livres da interface de entrada em suas sub-dimensões ou projeções seguidas de restrições. Seu modelo direto é o modelo do objeto matemático, ou seja, para um conjunto de instantes discretos N, e um espaço sensorial S dado como $S = S_1 \times S_2 \times ... \times S_k$, o modelo de um conhecimento remático sensorial específico é dado como o objeto passivo o , $o : N \rightarrow S$.

A construção da estrutura "o" em um sistema inteligente, pode ser feita incrementalmente, à medida que os instantes em N vão ocorrendo. Sendo assim, o objeto o está sempre "em construção", contendo as informações dos instantes de tempo anteriores até o instante atual. A definição total da estrutura de o, somente se dará após decorrido o total de instantes em N.

DEFINIÇÃO 4.1 - Restrições Temporais de Objetos

Seja N um conjunto de instantes, S uma classe e $o : N \rightarrow S$ um objeto de tipo S. Seja $N' \subseteq N$. Uma restrição do objeto o a N', denotado por $o \Downarrow N'$, corresponde ao objeto $o' : N' \rightarrow S$, tal que se $(n,s) \in o$ e $n \in N'$, $(n,s) \in o'$. Caso contrário, $(n,s) \notin o'$.

Exemplo: $N = \{ n_1 , n_2 , n_3 \}$, $S = \Re^3$,
$o = \{ (n_1 , (0,0,0)) , (n_2 , (0,1,0)) , (n_3 , (1,2,2)) \}$.

$N' = \{ n_2 , n_3 \} \rightarrow o' = \{ (n_2 , (0,1,0)) , (n_3 , (1,2,2)) \}$.

A seguir, tem-se um exemplo dos diversos tipos de conhecimentos remáticos sensoriais específicos:

Exemplo: Seja um sistema inteligente, com 3 entradas x_1, x_2 e $x_3 \in X = \{0,1,2\}$, sendo o conjunto de instantes $T = \{ t_1 , t_2 , t_3 \}$. Suponha que em t_1 , $x_1 = 0$, $x_2 = 0$ e $x_3 = 0$. Em t_2 , $x_1 = 0$, $x_2 = 1$ e $x_3 = 0$. E em t_3 , $x_1 = 1$, $x_2 = 2$ e $x_3 = 2$. O conhecimento remático sensorial correspondente à interface de entrada, neste caso pode ser representado pelo objeto $o = \{ (t_1 , (0,0,0)) , (t_2 , (0,1,0)) , (t_3 , (1,2,1)) \}$.

Seja a fórmula de indução $\alpha = [1, (2,3), (2,2)]$.

O objeto $o_{(\alpha)} = \{ (t_1 , (0,0)) , (t_2 , (0,1)) , (t_3 , (1,2)) \}$ é também um conhecimento remático sensorial específico.

O objeto o', dado por o restrito a $T' = \{ t_1 , t_3 \}$, $o' = \{ (t_1 , (0,0,0)) , (t_3 , (1,2,1)) \}$ também é um conhecimento remático sensorial específico, bem como o objeto o'' dado por $o_{(\alpha)}$ restrito a T', $o'' = \{ (t_1 , (0,0)) , (t_3 , (1,2)) \}$.

Um objeto o''' , dado por o restrito a T'' = $\{t_2\}$, o''' = $\{ (t_2 , (0,1,0)) \}$ também o é.

Em muitas situações, objetos do tipo de o''' são utilizados para representar o conhecimento remático sensorial específico oriundo dos sensores. Na realidade, estes são extremamente comuns, uma vez que não carregam um histórico temporal, sendo facilmente armazenáveis. Quando um certo nível de memória for necessário, a partir de argumentos adequados, outros conhecimentos remáticos sensoriais podem ser gerados, mantendo uma memória da interface de entrada a um nível que permita seu armazenamento em estruturas computacionais, sem prejudicar a interpretação de conhecimentos de maior nível.

4.2.2 Conhecimento Remático Sensorial Genérico

Um conhecimento remático sensorial genérico é uma abstração de um conhecimento remático sensorial específico. Ou seja, ele deve ser capaz de representar, ao mesmo tempo, diversos conhecimentos remáticos sensoriais específicos. Para tanto, o conceito de objeto por si só não é suficiente. Para representar esse tipo de conhecimento, introduziremos o conceito matemático de objeto genérico.

DEFINIÇÃO 4.2 - Variável de Conjunto

Sejam:

- N um conjunto enumerável, onde cada n denota um elemento de N.
- $X \subseteq U$ um subconjunto de um conjunto universo U.

Define-se uma **variável de conjunto** x de tipo X como uma função $x : N \rightarrow 2^X$.

Exemplos: Sejam $N = \{1,2,3\}$ e $X = \{1,2,3,4\}$. Um exemplo de uma variável de conjunto x de tipo X pode ser dado por:

$x = \{ (1, \{1,2\}) , (2, \{2,3,4\}) , (3, \{1,3\}) \}$

Supondo agora $X^2 = X \times X$, uma variável de conjunto x' de tipo X^2 pode ser dado por:

$x' = \{ (1, \{ (1,2),(2,3),(2,4),(3,3) \}) , (2, \{(2,3),(4,1),(1,1)\}) , (3, \{(1,3),(2,1) \}) \}$

Observe que neste último exemplo, o valor de x para cada $n \in N$ corresponde a uma relação. Nesse caso, fazendo-se $R_1 = \{ (1,2),(2,3),(2,4),(3,3) \}$, $R_2 = \{(2,3),(4,1),(1,1)\}$ e $R_3 = \{(1,3),(2,1) \}$, relações em $X \times X$, pode-se escrever:

$x' = \{ (1,R_1) , (2,R_2), (3, R_3) \}$

DEFINIÇÃO 4.3 - Objeto Genérico

Seja C uma classe não vazia. Seja c uma variável de conjunto de tipo C. A variável c é chamada então de um **objeto genérico** da classe C.

DEFINIÇÃO 4.4 - Caso de um Objeto Genérico

Seja c um objeto genérico de uma classe C. Um objeto c' de tipo C é dito um caso do objeto genérico c, se $\forall n \in N$, $c'(n) \in c(n)$.

DEFINIÇÃO 4.5 - Objeto Fuzzy

Sejam:

- N um conjunto enumerável, onde cada n denota um elemento de N.
- X uma classe.
- $\widetilde{X}$ um conjunto fuzzy definido sobre X.
- $2^{\widetilde{X}}$ o conjunto de todos os conjunto fuzzy definidos sobre X.

Define-se um **objeto fuzzy** x de tipo X como uma função $x : N \rightarrow 2^{\widetilde{X}}$.

Se X for uma classe passiva, $X = X_1 \times ... \times X_m$, $\widetilde{X}$ será uma relação fuzzy m-ária. Em alguns casos, será interessante utilizar uma representação que não considere diretamente uma relação fuzzy genérica, mas sim relações fuzzy formadas pelo produto cartesiano de conjuntos fuzzy. Neste caso, $\widetilde{X}$ poderá ser representado por uma ênupla de m conjuntos fuzzy. Quando X for uma classe ativa, $\widetilde{X}$ considerará como conjuntos fuzzy somente os campos da ênupla que não sejam funções. A relação fuzzy m-ária nesse caso será representada pelo produto cartesiano fuzzy de todos os elementos que não sejam funções.

Observe-se que qualquer objeto $o = \{ (n,x) \mid \forall n \in N, x \in X\}$ pode ser descrito em termos de um objeto fuzzy, tomando-se para cada $n \in N$ como valor um conjunto fuzzy dado por um conjunto unitário (*singleton*) em $x \in X$.

Como um objeto fuzzy passivo corresponde a uma relação fuzzy, é possível definir operações envolvendo objetos fuzzy. O mesmo ocorre com objetos fuzzy ativos, desde que se considere as operações somente nos campos que não sejam funções.

DEFINIÇÃO 4.6 - União de Objetos Fuzzy

Sejam x' e x'' dois objetos fuzzy de tipo X, definidos em N, tal que $\forall n \in N$, se x'(n) é definido, x''(n) também é definido, e vice-versa.

Define-se a união de x de x' e x'' como um objeto fuzzy tal que $\forall n \in N$,

$$x(n) = x'(n) \, \delta \, x''(n)$$

onde δ é um operador matricial que aplica uma co-norma triangular elemento a elemento nas matrizes m-árias. O operador δ terá validade somente nos campos das ênuplas que não sejam funções.

DEFINIÇÃO 4.7 - Interseção de Objetos Fuzzy

Sejam x' e x'' dois objetos fuzzy de tipo X, definidos em N, tal que $\forall n \in N$, se x'(n) é definido, x''(n) também é definido, e vice-versa.

Define-se a interseção de x de x' e x'' como um objeto fuzzy tal que $\forall n \in N$,

$$x(n) = x'(n) \, \mathcal{J} \, x''(n)$$

onde $\mathcal{J}$ é um operador matricial que aplica uma norma triangular elemento a elemento nas matrizes m-árias. O operador $\mathcal{J}$ terá validade somente nos campos das ênuplas que não sejam funções.

A estrutura básica de representação de um conhecimento remático sensorial genérico é o objeto genérico passivo ou , de um modo mais abrangente, o objeto fuzzy passivo. Observe que, dados diversos conhecimentos remáticos sensoriais específicos, é possível especificar um objeto genérico tal que estes conhecimentos remáticos sensoriais específicos sejam casos do objeto genérico. Se estes casos forem ponderados por valores entre 0 e 1, tem-se então um objeto fuzzy.

Observe que existem diversas maneiras diferentes de se especificar uma relação, ou seja, existem diferentes maneiras de se especificar um objeto genérico. Por exemplo, se o espaço da relação for um espaço métrico, é possível especificar-se uma das ênuplas da relação, e por meio de uma função de distância, estabelecer-se um critério que especifique a relação, como o conjunto de todas as ênuplas que se encontram até uma distância "d" da ênupla de referência. Esse método é conhecido na literatura da ciência da cognição como o método do protótipo (Cohen & Murphy, 1984). Um conhecimento remático sensorial específico é utilizado como protótipo e, a partir deste e de um critério de distância, determina-se um conhecimento remático sensorial genérico. Outro método é o de uma função discriminante que, dado um ponto, determina se esse ponto pertence ou não à relação. Esse é o método utilizado por exemplo nas redes neurais, nas arquiteturas denominadas *Perceptrons* (Lipmann, 1987) utilizando uma função não-linear do tipo *hard-limiter*. As redes neurais do tipo *Perceptron* emulam uma função discriminante, que pode ser tanto mais precisa quanto for o número de neurônios e de camadas na rede. A vantagem da função discriminante sobre o método do protótipo, é que neste, apenas regiões hiperesféricas podem ser definidas, ao passo que uma função discriminante pode englobar regiões de topologias genéricas. Especialmente no caso dos *Perceptrons* com 3 camadas, provou-se que estas redes podem discriminar espaços totalmente genéricos, com a precisão desta discriminação determinada, basicamente, pelo número de neurônios nas camadas intermediárias da rede. De um modo geral, um *Perceptron* com 3 camadas é um aproximador genérico para funções discriminantes contínuas em domínios compactos. Se a função não-linear do Perceptron for contínua (como em um exemplo típico, uma função sigmóide), a função discriminante obtida pelo *Perceptron* caracteriza uma relação fuzzy ao invés de uma relação no sentido clássico. Nesse caso, ao invés de um objeto genérico, obtém-se a especificação de um objeto fuzzy.

É interessante notar, neste caso, um paralelo entre estas observações e descobertas relativamente recentes da morfologia neural do cérebro humano (Verschure, 1993; Edelman & Mountcastle, 1978; Edelman, 1987). De acordo com estas descobertas, as estruturas básicas encontradas no córtex humano são as chamadas "colunas de neurônios", também conhecidas como "grupos neurais". Um grupo neural corresponde a um grupo de neurônios fortemente interconectados entre si, que são ativados conjuntamente na presença de um determinado estímulo. Os neurônios pertencentes ao grupo, que podem estar arranjados em múltiplas camadas, reagem conjuntamente a um estímulo, ao contrário de neurônios vizinhos, que não pertencem ao grupo. Um grupo pode ser representado, em termos estruturais, por uma rede *Perceptron* multi-camadas. Se associarmos essa estrutura à definição de conhecimento remático sensorial genérico, pode-se sugerir que estas estruturas podem estar funcionando no cérebro humano como um meio de representação de conhecimento remático sensorial genérico, ou como veremos mais a frente, como conhecimento remático de objeto genérico. Portanto, o modelo formal apresentado para a representação do conhecimento remático sensorial genérico não é um modelo a mais, para esse tipo de conhecimento, mas um modelo

formal que engloba diversos meios utilizados na representação do conhecimento, incluindo-se modelos matemáticos usados na IA (redes neurais do tipo *Perceptron*) como, eventualmente, estruturas de representação encontradas em seres vivos (grupos neurais).

Em um nível de implementação, é muito importante ressaltar, entretanto, que tanto o objeto genérico como o objeto fuzzy podem ser transformados em objetos. Supondo que os valores das instâncias sejam representados por funções discriminantes (caso mais geral), onde tais funções podem corresponder a redes neurais ou outras quaisquer, pode-se gerar um objeto contendo somente os parâmetros destas funções discriminantes. Entretanto, nesse caso, os argumentos que estiverem utilizando esse tipo de conhecimento devem estar também devidamente modificados para tratar os objetos genéricos ou fuzzy, colocados desta maneira.

4.2.3 Conhecimento Remático de Objeto Específico

O conhecimento remático sensorial pode ser obtido diretamente a partir dos sensores do sistema inteligente. O conhecimento remático de objeto é uma abstração dos sistemas inteligentes, capaz de concentrar um número maior de informações do que um conhecimento remático sensorial. O conhecimento remático sensorial genérico já é uma abstração sobre o conhecimento remático sensorial específico, mas tem ainda sua estrutura muito ligada com este último. Basicamente, o conhecimento remático de objeto é uma abdução feita sobre os conhecimentos sensoriais, quando se propõe que existe um objeto, e que este objeto é o causador dos estímulos sensoriais percebidos. É óbvio que esta tem de ser uma abdução consistente, ou seja, o conhecimento remático de objeto (o objeto que se presume existir) deve estar consistente com os conhecimentos sensoriais obtidos até então pelo sistema. Como deve ser feita esta abdução ? Basicamente, o espaço dos sensores deve ser recodificado por meio de transformações que sejam invariantes para os objetos abduzidos. Assim, a partir de um determinado conhecimento remático sensorial, pode-se identificar a que objeto esse conhecimento está relacionado. Essas transformações devem mapear pontos no espaço dos sensores em um novo espaço, chamado espaço do objeto. Este, deve possuir uma métrica tal que, estímulos diferentes no espaço dos sensores relacionados com um mesmo objeto, sejam próximos no espaço de representação do objeto.

Apesar de sua origem diferente, o conhecimento remático de objeto específico, é estruturalmente similar ao conhecimento remático sensorial específico, podendo ser representado formalmente como um objeto matemático passivo. A única diferença está nas classes dos objetos. Ao invés de serem definidas diretamente sobre o espaço dos sensores, as classes que representarão o conhecimento remático de objeto específico podem ser quaisquer (desde que sejam passivas). Isso ocorre, porque a estrutura destas classes depende da transformação que faz o mapeamento do espaço dos sensores para o espaço dos objetos.

4.2.4 Conhecimento Remático de Objeto Genérico

Do mesmo modo que o conhecimento remático de objeto específico pode ser representado por um objeto matemático passivo, o conhecimento remático de objeto genérico pode ser representado por um objeto genérico passivo, ou por um objeto fuzzy passivo. A classe do objeto genérico (ou objeto fuzzy) também não necessita, em princípio, ter nenhum vínculo com as classes encontradas nos conhecimentos rematicos sensoriais. De novo, isso ocorre porque estas classes são

geradas por uma transformação genérica sobre as classes dos conhecimentos sensoriais, sendo que não se impõe nenhuma restrição quanto ao contradomínio desta transformação. Do mesmo modo que no caso dos conhecimentos remáticos sensoriais genéricos, os objetos genéricos (ou fuzzy) podem ser transformados em objetos matemáticos, para efeito de implementação.

4.2.5 Conhecimento Remático de Ocorrências

O conhecimento remático de ocorrências é utilizado para descrever conceitualmente trechos do histórico de um ou mais objetos. A trajetória temporal de um objeto pode ser vista como uma sequência de instâncias da classe a que o objeto pertence. Esta sequência pode conter sub-sequências que aparecem uma única vez, ou se repetem mais de uma vez na sequência. O conhecimento remático de ocorrências corresponde exatamente à identificação e representação destas sub-sequências, que são chamadas de ocorrências. Entretanto, as instâncias dos objetos podem conter diversos campos, sendo que apenas alguns deles podem ser significativos, para efeito de caracterização da ocorrência. Com isso, o modelo formal para ocorrências deve não somente destacar as sub-sequências, mas permitir o mascaramento dos campos das instâncias que são relevantes para sua determinação. Considera-se, ainda, que as sub-sequências não necessitam ser formadas por elementos contíguos da sequência original. Deste modo, permite-se que se trate como ocorrência, qualquer sequência de instâncias extraídas da trajetória original do objeto. Em alguns casos, deseja-se correlacionar ocorrências em dois ou mais objetos distintos. O modelo formal deve portanto permitir esta correlação. Deste modo, pode-se representar ocorrências que afetam a mais de um objeto. Para modelar matematicamente as estruturas que representam o conhecimento remático de ocorrências, incorporando todos esses requisitos, faremos uma extensão do modelo formal de objeto, definindo o chamado meta-objeto.

DEFINIÇÃO 4.8 - Meta-Objeto

Sejam:

- N um conjunto enumerável, onde cada n denota um elemento de N.
- V um conjunto enumerável, onde cada $v \in V$ é uma variável de tipo N, definida sobre um espaço de ocorrências T, $v : T \rightarrow N$.
- R um conjunto de restrições sobre os valores das variáveis de V (possivelmente vazio).
- X uma classe.

Define-se um **meta-objeto** x de tipo X como uma função $x : V \rightarrow X$.

Exemplos : Sejam T = { 1,2, ... } , N = {1,2,3,4,5,6}, V = { v_1 , v_2 , v_3 }, sendo as variáveis v_1 , v_2 , v_3 : $T \rightarrow N$, $R = \varnothing$, e $X = X_1 \times X_2$, $X_1 = \{1,2,3,4\}$, $X_2 = \{a,b,c\}$.

Um exemplo de meta-objeto x' de tipo X_1 seria: $x' = \{ (v_1 , 1) , (v_2 , 3) \}$.

Outro exemplo de meta-objeto x'' de tipo X seria: $x'' = \{ (v_1 , (1,a)) , (v_2 , (3,a)) \}$.

DEFINIÇÃO 4.9 - Instância de um Meta-Objeto

Seja x um meta-objeto x de tipo X.

Define-se uma **instância** de x como um objeto x' dado pela substituição das variáveis em x, pelos valores dados por instâncias específicas destas variáveis no espaço de ocorrências.

Exemplos: Supondo os meta-objetos x' e x'' no exemplo anterior,

Uma instância de x', fazendo-se $v_1 = 1$ e $v_2 = 2$ é dada por x''' = { (1 , 1) , (2 , 3) }.

Outro exemplo, fazendo-se $v_1 = 2$ e $v_2 = 5$ é dado por x''' = { (2 , 1) , (5 , 3) }.

Uma instância de x'', fazendo-se $v_1 = 1$ e $v_2 = 4$ é dada por x''' = {(1 , (1,a)), (4 , (3,a))}.

DEFINIÇÃO 4.10 - Ocorrência de um Meta-Objeto em um Objeto

Seja um objeto o de uma classe X, e um meta-objeto o' de uma classe X', Uma **ocorrência** o'' do meta-objeto o' em o é dada por um objeto o'' tal que o'' é ao mesmo tempo um sub-objeto de uma instância de o' e uma restrição temporal de um sub-objeto de o.

Exemplos: Utilizando-se os dados do exemplo anterior, considere um objeto x de tipo X, x = { (1,(1,a)) , (2,(3,b)) , (3,(3,a)), (4,(1,c)) , (5,(2,b)), (6,(3,a)). Fazendo-se as atribuições : $v_1 = 1$ e $v_2 = 2$, tem-se uma instância x''' = { (1,1), (2,3) } que é uma restrição temporal do sub-objeto $x \downarrow X_1$ de x a N = {1,2}. Esta é uma ocorrência de x' em x. Outras ocorrências para este caso existem, fazendo-se $v = (v_1 , v_2) = (1,3)$, v = (1,6), v = (4,6). Outras, não tão óbvias ocorrem para v = (4,2) e v = (4,3). O meta-objeto x'' também ocorre em x. Entretanto, nesse caso, existem apenas duas ocorrências, fazendo-se v = (1,3) e v = (1,6).

Quando o conjunto de restrições R no meta-objeto é diferente de vazio, as variáveis do domínio apresentam restrições em seus valores. Uma maneira de denotar tais restrições é por meio de equações e/ou inequações algébricas envolvendo as variáveis. Neste caso, as ocorrências do meta-objeto em outros objetos deverá considerar tais restrições para a determinação das possíveis instâncias do meta-objeto.

Exemplo: Considere-se os exemplos anteriores, supondo-se entretanto a seguinte restrição: $v_2 = v_1 + 1$. Nesse caso, existe apenas a ocorrência de x' em x para v = (1,2), pois para os demais casos, a restrição é violada. Observe que nesse caso, o meta-objeto x'' não ocorre em x.

Outro exemplo, envolvendo inequações seria considerar $v_2 > v_1$. Utilizando-se esta inequação como restrição no exemplo da definição 4.6, evita-se, por exemplo, que os casos não intuitivos v = (4,2) e v = (4,3) sejam considerados ocorrências.

Segundo a definição, uma ocorrência não precisa ser, necessariamente, uma instância do meta-objeto, podendo ser uma instância de qualquer sub-objeto deste. Com isso, a partir de um único meta-objeto, pode-se ter diferentes ocorrências em diferentes objetos, onde cada um dos objeto compartilha um campo diferente com o meta-objeto.

DEFINIÇÃO 4.11 - Ocorrência de um Meta-Objeto em um Objeto Genérico

Seja um objeto genérico x de uma classe X, e um meta-objeto x' de uma classe X'. Uma ocorrência x'' do meta-objeto x' em x corresponde a um objeto x'' tal que x'' é uma ocorrência para algum caso de x.

DEFINIÇÃO 4.12 - Ocorrência de um Meta-Objeto em um Objeto Fuzzy

Seja um objeto fuzzy o de uma classe X, e um meta-objeto o' de uma classe X', Uma **ocorrência** o'' do meta-objeto o' em o é dada por um objeto fuzzy o'' tal que o'' corresponde à interseção de um sub-objeto de uma instância de o', descrito como um objeto fuzzy por meio de *singletons*, e uma restrição temporal de um sub-objeto fuzzy de o.

Exemplo: Sejam os conjuntos fuzzy
$a_1 = \{1/0.2, 2/0.8, 3/0.6\}$, $a_2 = \{1/0.1, 2/0.2, 3/0.9\}$, $a_3 = \{1/0, 2/0.15, 3/0.3\}$,
$b_1 = \{5/0.3, 6/0.4, 7/0.1\}$, $b_2 = \{5/0.4, 6/0.4, 7/0.8\}$, $b_3 = \{5/0.1, 6/0.9, 7/0.8\}$,
$c_1 = \{15/0.2, 18/0.9\}$, $c_2 = \{15/0.3, 18/0.8\}$, $c_3 = \{15/0.7, 18/0.1\}$
e o objeto fuzzy $x = \{(1,(a_1,b_1,c_1)), (2,(a_2,b_2,c_2)), (3,(a_3,b_3,c_3))$, e o meta-objeto
$x' = \{(v_1, (2,5,15)), (v_2, (3,7,18))\}$.
Fazendo-se $v_1 = 1$ e $v_2 = 3$, tem-se uma instância do meta-objeto x', dada por
$x'' = \{(1, (2,5,15)), (3, (3,7,18))\}$.
Utilizando-se então
$a'_1 = \{1/0, 2/1, 3/0\}$, $b'_1 = \{5/1, 6/0, 7/0\}$ e $c'_1 = \{15/1, 18/0\}$,
$a'_2 = \{1/0, 2/0, 3/1\}$, $b'_2 = \{5/0, 6/0, 7/1\}$ e $c'_2 = \{15/0, 18/1\}$ tem-se a representação de x'' por meio de um objeto fuzzy, dado por

$x''' = \{(1, (a'_1, b'_1, c'_1)), (3, (a'_2, b'_2, c'_2))\}$

Uma ocorrência de x' em x, nesse caso, pode ser calculada fazendo-se

$x'''' = (x \Downarrow \{1,3\}) \mathcal{J} x'''$

$x'''' = \{(1, (a''_1, b''_1, c''_1)), (3, (a''_2, b''_2, c''_2))\}$, onde, utilizando-se o mínimo como norma triangular tem-se:

$a''_1 = a_1 \mathcal{J} a'_1 = \{1/0, 2/0.8, 3/0\}$
$b''_1 = b_1 \mathcal{J} b'_1 = \{5/0.3, 6/0, 7/0\}$
$c''_1 = c_1 \mathcal{J} c'_1 = \{15/0.2, 18/0\}$
$a''_2 = a_3 \mathcal{J} a'_2 = \{1/0, 2/0, 3/0.3\}$
$b''_2 = b_3 \mathcal{J} b'_2 = \{5/0, 6/0, 7/0.8\}$
$c''_2 = c_3 \mathcal{J} c'_2 = \{15/0, 18/0.1\}$

DEFINIÇÃO 4.13 - Meta-Objeto Genérico

Sejam:

- N um conjunto enumerável, onde cada n denota um elemento de N,
- V um conjunto enumerável, onde cada $v \in V$ é uma variável de tipo N,
- R um conjunto de restrições sobre as variáveis de V (possivelmente vazio),
- X uma classe.

Define-se um **meta-objeto genérico** x de tipo X como uma função $x : V \rightarrow 2^X$.

DEFINIÇÃO 4.14 - Caso de um Meta-Objeto Genérico

Seja x um meta-objeto genérico de uma classe X. Um meta-objeto x' de tipo X é dito um **caso** do objeto genérico x, se $\forall v \in V$, $x'(v) \in x(v)$.

DEFINIÇÃO 4.15 - Ocorrência de um Meta-Objeto Genérico em um Objeto

Sejam x um meta-objeto genérico de tipo X e x' um objeto de tipo X'. Uma **ocorrência** x'' de x em x' é dada por um objeto x'', tal que x'' é uma ocorrência de algum caso de x em x'.

DEFINIÇÃO 4.16 - Ocorrência de um Meta-Objeto Genérico em um Objeto Genérico

Sejam x um meta-objeto genérico de tipo X e x' um objeto genérico de tipo X'. Uma **ocorrência** x'' de x em x' é dada por um objeto genérico x'', tal que x'' é dado pela união de todas as ocorrências de algum caso de x em casos de x'.

DEFINIÇÃO 4.17 - Ocorrência de um Meta-Objeto Genérico em um Objeto Fuzzy

Sejam x um meta-objeto genérico de tipo X e x' um objeto fuzzy de tipo X'. Uma **ocorrência** x'' de x em x' é dada por um objeto fuzzy x'', tal que x'' é dado pela interseção de um sub-objeto de x, descrito como um objeto fuzzy, e uma restrição temporal de um sub-objeto de x'.

Exemplo: Sejam os conjuntos fuzzy
$a_1 = \{1/0.2, 2/0.8, 3/0.6\}$, $a_2 = \{1/0.1, 2/0.2, 3/0.9\}$, $a_3 = \{1/0.1, 2/0.15, 3/0.3\}$,
$b_1 = \{5/0.3, 6/0.4, 7/0.1\}$, $b_2 = \{5/0.4, 6/0.4, 7/0.8\}$, $b_3 = \{5/0.1, 6/0.9, 7/0.8\}$,
$c_1 = \{15/0.2, 18/0.9\}$, $c_2 = \{15/0.3, 18/0.8\}$, $c_3 = \{15/0.7, 18/0.1\}$
e o objeto fuzzy $x = \{(1,(a_1,b_1,c_1)), (2,(a_2,b_2,c_2)), (3,(a_3,b_3,c_3))$, e o meta-objeto genérico $x' = \{(v_1, ([2,3],[5,6],15)), (v_2, ([1,2],[6,7],18))\}$.
Fazendo-se $v_1 = 1$ e $v_2 = 3$, têm-se uma instância do meta-objeto genérico x', dada por
$x'' = \{(1, ([2,3],[5,6],15)), (3, ([1,2],[6,7],18))\}$.
Utilizando-se então
$a'_1 = \{1/0, 2/1, 3/1\}$, $b'_1 = \{5/1, 6/1, 7/0\}$ e $c'_1 = \{15/1, 18/0\}$,
$a'_2 = \{1/1, 2/1, 3/0\}$, $b'_2 = \{5/0, 6/1, 7/1\}$ e $c'_2 = \{15/0, 18/1\}$ tem-se a representação de x'' por meio de um objeto fuzzy, dado por

$x''' = \{(1, (a'_1, b'_1, c'_1)), (3, (a'_2, b'_2, c'_2))\}$

Uma ocorrência de x' em x, nesse caso, pode ser calculada fazendo-se

$x'''' = (x \Downarrow \{1,3\}) \mathcal{J} x'''$

$x'''' = \{(1, (a''_1, b''_1, c''_1)), (3, (a''_2, b''_2, c''_2))\}$, onde, utilizando-se o mínimo como norma triangular tem-se:

$a''_1 = a_1 \mathcal{J} a'_1 = \{1/0, 2/0.8, 3/0.6\}$
$b''_1 = b_1 \mathcal{J} b'_1 = \{5/0.3, 6/0.4, 7/0\}$
$c''_1 = c_1 \mathcal{J} c'_1 = \{15/0.2, 18/0\}$
$a''_2 = a_3 \mathcal{J} a'_2 = \{1/0.1, 2/0.15, 3/0\}$
$b''_2 = b_3 \mathcal{J} b'_2 = \{5/0, 6/0.9, 7/0.8\}$
$c''_2 = c_3 \mathcal{J} c'_2 = \{15/0, 18/0.1\}$

DEFINIÇÃO 4.18 - Meta-Objeto Fuzzy

Sejam:

- N um conjunto enumerável, onde cada n denota um elemento de N.
- V um conjunto enumerável, onde cada $v \in V$ é uma variável de tipo N.
- R um conjunto de restrições sobre as variáveis de V (possivelmente vazio).
- X uma classe.
- $\tilde{X}$ um conjunto fuzzy definido sobre X.
- $2^{\tilde{X}}$ o conjunto de todos os conjunto fuzzy definidos sobre X.

Define-se um **meta-objeto fuzzy** x de tipo X como uma função $x : V \rightarrow 2^{\tilde{X}}$.

DEFINIÇÃO 4.19 - Ocorrência de um Meta-Objeto Fuzzy em um Objeto

Sejam x um meta-objeto fuzzy de tipo X e x' um objeto de tipo X'. Uma **ocorrência** x'' de x em x' é dada por um objeto fuzzy x'', tal que x'' é dado pela interseção de um sub-objeto de x, e uma restrição temporal de um sub-objeto de x' descrito como um objeto fuzzy.

DEFINIÇÃO 4.20 - Ocorrência de um Meta-Objeto Fuzzy em um Objeto Genérico

Sejam x um meta-objeto fuzzy de tipo X e x' um objeto genérico de tipo X'. Uma **ocorrência** x'' de x em x' é dada por um objeto fuzzy x'', tal que x'' é dado pela interseção de um sub-objeto de x, e uma restrição temporal de um sub-objeto de x' descrito como um objeto fuzzy.

DEFINIÇÃO 4.21 - Ocorrência de um Meta-Objeto Fuzzy em um Objeto Fuzzy

Sejam x um meta-objeto fuzzy de tipo X e x' um objeto fuzzy de tipo X'. Uma **ocorrência** x'' de x em x' é dada por um objeto fuzzy x'', tal que x'' é dado pela interseção de um sub-objeto de x, e uma restrição temporal de um sub-objeto de x'.

Observe que no caso do meta-objeto fuzzy, a definição de ocorrência é praticamente a mesma, sendo a ocorrência em um objeto, em um objeto genérico ou em um objeto fuzzy. Em todos os casos, o que se faz é converter a representação do objeto ou objeto genérico para um objeto fuzzy, e utilizar a definição de ocorrência em um objeto fuzzy.

O conhecimento remático de ocorrências é representado matematicamente por meio de meta-objetos. Mais especificamente, essa representação pode se dar por meio de meta-objetos, meta-objetos genéricos ou meta-objetos fuzzy. O conhecimento remático de ocorrências específico é representado por meta-objetos, meta-objetos genéricos ou meta-objetos fuzzy, tendo-se como restrição R às variáveis $v \in V$, equações algébricas de atribuição, onde explicitamente, se atribui para cada v_i um valor específico em N, fixando uma instância temporal, ou seja, o conhecimento remático de ocorrências específico é representado por instâncias de meta-objetos, meta-objetos genéricos ou meta-objetos fuzzy. No caso do conhecimento remático de ocorrências genérico, isso não é necessário. O uso de meta-objetos, meta-objetos genéricos ou meta-objetos fuzzy se dará em função do conhecimento que se deseja representar, havendo ou não restrições. Observe que a idéia de meta-objeto é uma idéia matemática. Como os conhecimentos rematicos

sensoriais e de objeto são representados por objetos (no sentido matemático), as ocorrências representadas por meta-objetos tanto podem dizer respeito a ocorrências sensoriais como ocorrências em objetos. Outra coisa importante é que apesar de exemplificarmos sempre com meta-objetos contendo dois elementos, isso de modo algum é único. O uso de meta-objetos contendo dois elementos é interessante particularmente para representar eventos, ou seja, mudanças abruptas entre um estado anterior e um estado presente. Entretanto, outros tipos de ocorrências podem ser representadas, como por exemplo a ocorrência unitária, que basicamente identifica um estado. Por meio das ocorrências unitárias, pode-se obter informações a respeito de particularidades do objeto, durante sua existência. Meta-objetos com mais de 2 elementos podem ser utilizados para representar tendências, e comportamentos, como quando se deseja representar que determinado valor está sempre crescendo, ou sempre decrescendo, ou então comportamentos periódicos. Na verdade o modelo formal de meta-objetos é aberto o suficiente para permitir que diversos tipos diferentes de ocorrências possam ser representados.

O uso dos conhecimentos de ocorrências em redes de objetos é possível, mas não é trivial. O disparo de um objeto ativo permite que as funções do objeto considerem apenas a instância imediatamente anterior no tempo. Portanto, para que o conhecimento de ocorrências possa ser utilizado, a estrutura do objeto deve ser alterada de modo a compor uma memória temporal. Essa metodologia é demonstrada na figura 4.1 a seguir:

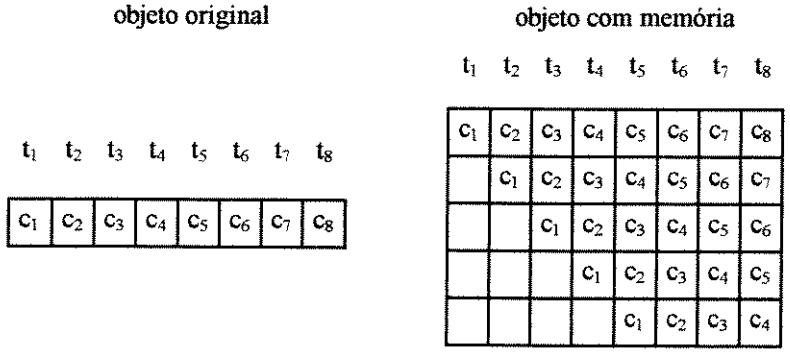


Figura 4.1 - Composição de Memória em Objeto

Uma vez tendo uma memória, é possível o uso dos meta-objetos nas redes de objetos. Para tal, um meta-objeto precisa ser transformado em um objeto matemático, de modo a permitir sua comparação com a instância do objeto modificado em um dado instante. Um exemplo desta transformação é demonstrado na figura 4.2 a seguir:

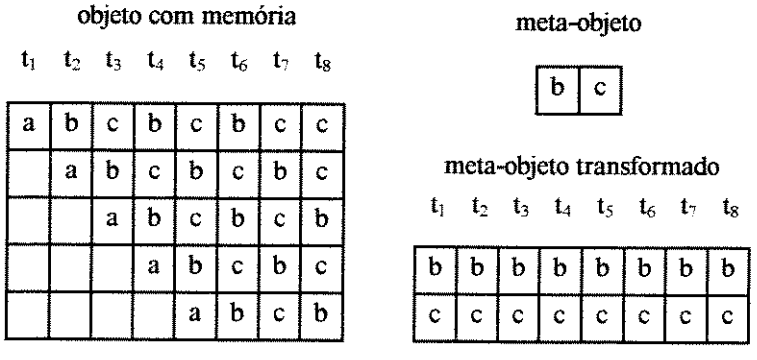


Figura 4.2 - Transformação do meta-objeto em objeto

No exemplo, o meta-objeto $((v_1 , b),(v_2 , c))$ é transformado em um objeto matemático. Com isso, no instante do disparo, ele pode ser comparado com a instância do objeto com memória. Por exemplo, a sequência (b,c) aparece contiguamente pelo menos duas vezes na instância t_8 do objeto com memória, e 6 vezes não contiguamente. Esse esquema permite inclusive que o meta-objeto sofra um processo de aprendizado, sendo redefinido a cada instante no tempo.

Em sistemas reais, a memória do objeto não pode ser infinita. Com isso, existe uma limitação no uso das ocorrências na dinâmica de tais sistemas. Um paliativo para essa restrição é o uso de uma hierarquia de memórias, utilizando-se o conhecimento de ocorrências para gerar uma memória de hierarquia superior. Um exemplo deste método pode ser visto na figura 4.3 a seguir:

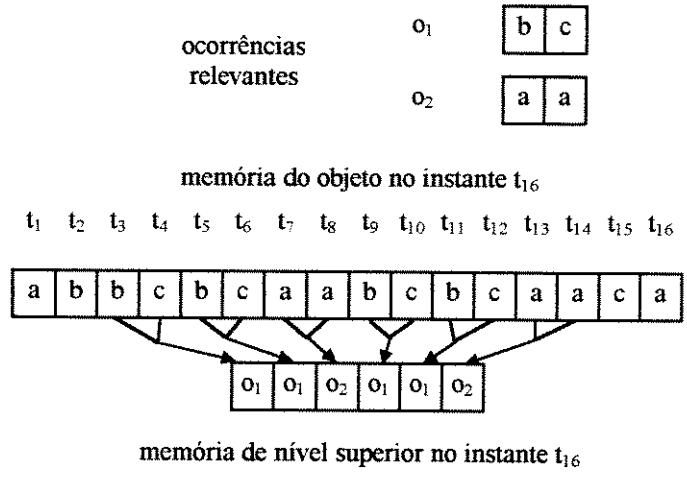


Figura 4.3 - Composição de Memória de Nível Superior

Como pode ser visto no exemplo, a memória de nível superior pode ter um tamanho inferior ao da memória total do objeto. Entretanto, existe alguma perda de informação, pois a memória de nível superior armazena apenas as ocorrências consideradas relevantes, desprezando outras instâncias da memória original do objeto. Essa perda é necessária, de modo que a informação relevante possa ser armazenada. Eventualmente, uma ocorrência pode em um determinado instante deixar de ser relevante, sendo eliminada da memória de nível superior para esse instante. Esse mecanismo de geração de hierarquia pode ser aplicado sucessivamente, obtendo-se uma memória altamente hierarquizada, porém compacta.

No geral, sempre que for necessário utilizar-se um conhecimento remático de ocorrência em uma rede de objetos, é necessária a conversão da variedade de meta-objeto utilizada para objetos matemáticos. Nesse sentido, a definição de meta-objetos e suas variedades serve apenas como fundamento teórico para os conhecimentos rematicos de ocorrência. Pragmaticamente, recorre-se a sua transformação em um objeto matemático, de modo a possibilitar seu uso em uma rede de objetos.

4.3 Representação do Conhecimento Dicente

O conhecimento dicente, conhecido também como conhecimento lógico, ou lógica, é representado por meio de expressões contendo proposições e conectivos lógicos. A característica básica de um conhecimento dicente, é a possibilidade de se atribuir um valor verdade, ou de grau de verdade a uma expressão. O valor verdade de uma proposição pode ser determinado ou a partir do valor verdade de outras proposições, ou por meio de conhecimentos remáticos que são o fundamento da proposição. As proposições podem ser primitivas ou compostas. As proposições primitivas podem ser ainda icônicas ou simbólicas. As proposições icônicas são formadas por expressões contendo conhecimentos remáticos, que fundamentam a proposição. Esses conhecimentos remáticos estão acoplados ao conhecimento dicente, em termos de seu valor verdade. O relacionamento entre os conhecimentos remáticos, e o valor-verdade do conhecimento dicente pode ser estabelecido de duas maneiras. Na primeira, parte-se de conhecimentos remáticos conhecidos, para estabelecer o valor verdade da proposição icônica. Na segunda forma, parte-se de um valor-verdade dado para a proposição icônica, de modo a determinar os conhecimentos remáticos, ou complementar parte dos conhecimentos remáticos que, porventura, estejam indeterminados. O fato é que, nas proposições icônicas, os conhecimentos dicente e remáticos estão acoplados, de modo que, de alguma forma, um deve justificar o outro e vice-versa. Sendo assim, o valor-verdade de uma proposição icônica pode ser estabelecido a partir de seu conhecimento remático, ou a partir do valor-verdade de outras proposições. As proposições simbólicas são um pouco diferente das proposições icônicas. Nelas, não existe o vínculo com conhecimentos remáticos, que fundamentam o valor-verdade da proposição. Com isso, o valor-verdade de proposições simbólicas só pode ser dado a partir do valor-verdade de outras proposições. Em um sistema fechado, é necessário que existam proposições condicionais envolvendo proposições icônicas e proposições simbólicas, de modo que o valor-verdade destas últimas possa ser determinado, a partir do valor-verdade das proposições icônicas, determinados por meio de seus conhecimentos remáticos.

Na inteligência artificial clássica, as proposições icônicas não são consideradas. Com isso, ocorre um desacoplamento entre o conhecimento dicente e o conhecimento remático, levando a um problema usualmente conhecido como a falta de fundamento simbólico (*symbol grounding*). Isso ocorre porque como não existe o fundamento proporcionado pelo conhecimento remático, não se pode determinar o valor-verdade das proposições simbólicas, a menos que se parta do pressuposto que o valor-verdade de algumas proposições simbólicas é previamente conhecido. Assim, assume-se como conhecido o valor verdade de um certo número de proposições simbólicas (chamadas de fatos), sem a fundamentação de conhecimentos remáticos, e a partir de proposições condicionais e de inferências dedutivas, determina-se o valor verdade de outras proposições simbólicas. Observe-se que o acoplamento entre o conhecimento dicente e o conhecimento remático não deixa de existir. O que acontece é que na IA clássica não existe uma metodologia para o tratamento deste acoplamento. Ele ocorre implicitamente, admitindo-se que o valor verdade de um certo número de proposições é previamente conhecido. Isso é equivalente a se dizer que o valor verdade da proposição (ou conhecimento dicente) é determinado por um agente externo ao sistema, ou seja, um ser humano ou outro sistema capaz de efetuar o acoplamento entre o conhecimento remático e o conhecimento dicente, que o fornece como entrada. No modelo aqui apresentado,

existe uma metodologia para o acoplamento entre o conhecimento dicente e o conhecimento remático, colocado sob a forma das proposições icônicas. Assim, não há a necessidade de se assumir (como na IA clássica) que o valor verdade de algumas proposições deve ser conhecido a priori.

DEFINIÇÃO 4.22 Expressão

Define-se uma expressão E como uma sequência de símbolos e_1 , e_2 , ... , e_n .

DEFINIÇÃO 4.23 Proposição Icônica

Sejam:

- S um conjunto de estruturas correspondentes a conhecimentos remáticos sensoriais, $S = \{ s_1 , ... , s_k \}$.
- B um conjunto de estruturas correspondentes a conhecimentos remáticos de objetos, $B = \{ b_1 , ... , b_l \}$.
- O um conjunto de estruturas correspondentes a conhecimentos remáticos de ocorrências, $O = \{ o_1 , ... , o_m \}$.
- N o conjunto dos números naturais.

Define-se uma proposição icônica p como uma expressão formada do seguinte modo:

$$p = a\ (c_1 , ... , c_n\ ) / f , r$$

onde $n \in \{1, 2, 3, ... \}$, $a \in O$, é chamado de **verbo**, $c_i \in S$ ou $c_i \in B$, $i = 1, ... , n$ são chamados de **relatos,** f é uma função $f : N \rightarrow N \times N$ que mapeia cada campo da classe correspondendo ao verbo em um par (x_1, x_2), onde $x_1 \in \{1, ... , n\}$ é um índice correspondente a algum relato, e x_2 é um índice correspondente a um campo da classe correspondendo ao relato definido por x_1 e r é um conjunto de restrições, possivelmente vazio, correspondendo a possíveis restrições nos valores temporais a serem utilizados nas substituições das variáveis referentes ao verbo.

Uma proposição icônica pode ser lida da seguinte maneira: "a ocorre em c_1 , ... , c_n segundo f e r". Isso corresponde a dizer que a ocorrência representada por a, deve ocorrer nos objetos c_1 , ... , c_n , fazendo-se um mapeamento dos campos em a nos campos em c_i em função de f, e a ocorrência deve seguir as restrições colocadas por r.

Exemplo:
Sejam:

- os objetos o_1 e o_2 representando dois conhecimentos remáticos de objeto.
- o meta-objeto m_1 representando um conhecimento remático de ocorrência.
- uma função f dada por $f(1) = (1,2)$ e $f(2) = (2,1)$.
- um conjunto vazio de restrições r, $r = \varnothing$.

Supondo que:

- a classe referente a o_1 é $A = A_1 \times A_2 \times A_3$
- a classe referente a o_2 é $B = B_1 \times B_2 \times B_3$
- a classe referente a m_1 é $C = C_1 \times C_2$
- $A_2 = C_1$ e $B_1 = C_2$
- $o_1 \downarrow A_2 = o'_1$
- $o_2 \downarrow B_1 = o'_2$

- $m_1 (v_1) = (o'_1 (t_6), o'_2 (t_6))$
- $m_1 (v_2) = (o'_1 (t_8), o'_2 (t_8))$
- $m_1 (v_3) = (o'_1 (t_9), o'_2 (t_9))$

Então, um diagrama representando a proposição icônica $m_1 (o_1, o_2) / f, r$ pode ser visualizada na figura 4.4 a seguir.

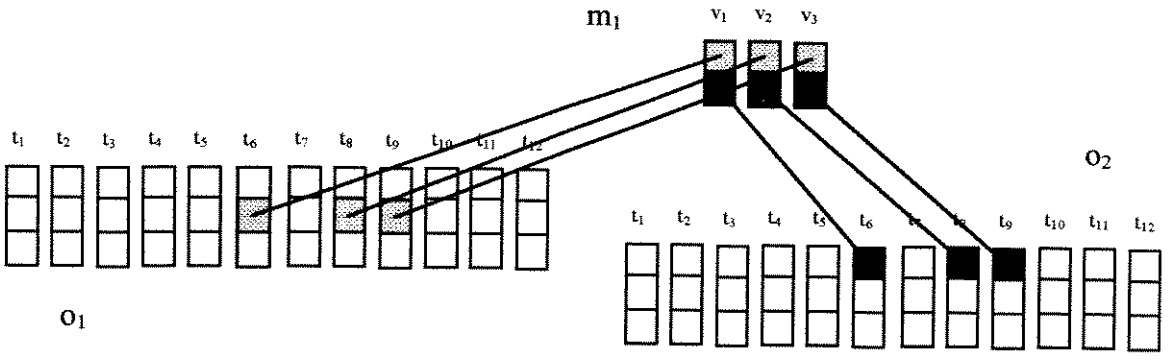


Figura 4.4 - Diagrama Ilustrando Exemplo de Proposição Icônica

Nesse caso, dizemos que o meta-objeto m_1 ocorre em o_1 e o_2 ou, de maneira equivalente, que a proposição icônica $m_1 (o_1, o_2) / f, r$ tem seu valor verdade igual a verdadeiro.

Quando o verbo for representado por um meta-objeto genérico, tem-se o comportamento ilustrado na figura 4.5, a seguir.

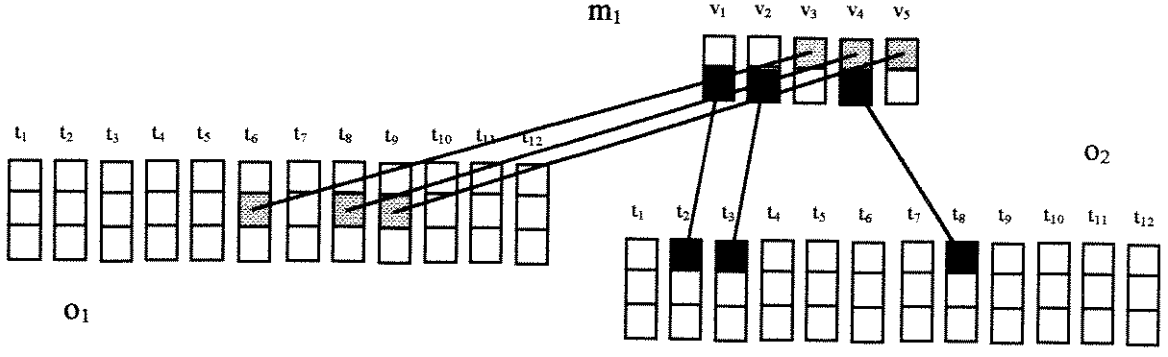


Figura 4.5 - Proposição Icônica Utilizando Meta-Objetos Genéricos

Para o exemplo da figura 4.5, as primeiras componentes de $m_1 (v_1)$ e $m_1 (v_2)$, e as segundas componentes de $m_1 (v_3)$ e $m_1 (v_5)$ são genéricas, podendo assumir qualquer valor (em A_2 no primeiro caso e B_1 no segundo). Com isso, apenas os campos indicados na figura devem coincidir para que o valor verdade da proposição primitiva seja verdadeiro.

DEFINIÇÃO 4.24 - Proposição Simbólica

Define-se uma proposição simbólica como uma expressão contendo um único símbolo, que não corresponde a nenhum conhecimento remático.

DEFINIÇÃO 4.25 - Proposição Primitiva

Define-se uma proposição primitiva como sendo ou uma proposição icônica ou uma proposição simbólica.

DEFINIÇÃO 4.26 - Valor Verdade

Um valor verdade é um valor definido entre 0 e 1, correspondendo ao grau de verdade de uma determinada proposição. Um valor verdade 0 corresponde a uma total falsidade e um valor verdade 1 corresponde a uma total verdade. Do mesmo modo, pode ser feita a associação: valor verdade = 0 $\rightarrow$ falso, valor verdade = 1 $\rightarrow$ verdadeiro. Valores verdade entre 0 e 1 não são nem totalmente verdadeiro nem totalmente falso. Um valor verdade igual a 0.5 corresponde à total indeterminação.

DEFINIÇÃO 4.27 - Determinação do Valor Verdade de uma Proposição Icônica segundo seu Conhecimento Remático

Seja uma proposição icônica $p = a\ (c_1, \ldots, c_n) / f, r$.

O valor-verdade de p, V(p) pode ser calculado a partir do conhecimento remático associado a ela, do seguinte modo:

$$V(p) = v_1 \mathcal{J} \ldots \mathcal{J} v_n$$

onde v_i será igual a:

1. Se existirem m ocorrências o_{im} de a em c_i segundo f e r que são objetos fuzzy:

$$v_i = \sup_m (\sup\ o_{im}),$$

2. Se existir pelo menos uma ocorrência o_i de a em c_i segundo f e r que é um objeto ou objeto genérico:

$$v_i = 1.$$

3. Se não existir nem uma ocorrência de a em c_i ,

$$v_i = 0.$$

DEFINIÇÃO 4.28- Proposição

Sejam:

- um conjunto P de proposições primitivas
- um conjunto L_1 de operadores lógicos unários.
- um conjunto L_2 de operadores lógicos binários.

Uma expressão R é chamada de uma proposição se e somente se:

1. R é uma proposição primitiva, ou
2. R pode ser decomposta em $R_1\ l_2\ R_2$, onde $l_2 \in L_2$ e R_1 e R_2 são proposições, ou
3. R pode ser decomposta em $l_1\ R_1$, onde $l_1 \in L_1$ e R_1 é uma proposição.

Normalmente se utilizam
$L_1 = \{\sim$ (negação) $\}$ e
$L_2 = \{ \wedge$ (conjunção), $\vee$ (disjunção), $\rightarrow$ (implicação) $\}$
Os operadores são utilizados para calcular o valor verdade de uma proposição em função dos valores verdade de suas proposições primitivas. Usualmente utiliza-se:

$$V(\sim a) = 1 - V(a)$$
$$V(a \wedge b) = V(a) \text{ ⅀ } V(b)$$
$$V(a \vee b) = V(a) \text{ ⅄ } V(b)$$
$$V(a \rightarrow b) = V(\sim a \vee b)$$

ou algumas vezes
$$V(a \rightarrow b) = V(a \wedge b)$$

Proposições que utilizem o operador $\rightarrow$ são chamadas de proposições condicionais, estando normalmente associadas a regras.

O conhecimento dicente, como se viu, é representado por meio de expressões, que podem ou não fazer referência a conhecimentos remáticos. Do mesmo modo, verificou-se que sempre se associa a um conhecimento dicente um valor verdade, que no caso de proposições icônicas pode ser calculado por meio do conhecimento remático que o fundamenta ou, no caso geral, por meio do valor verdade de outras proposições (o que é feito por meio de conhecimentos argumentativos, a serem apresentados na próxima seção). Com isso, pode-se encapsular um conhecimento dicente em uma ênupla (E,V), onde E se refere à expressão correspondente ao conhecimento dicente, e V corresponde ao seu valor verdade. Se é necessário representar um caráter temporal, ao invés de simples ênuplas pode-se utilizar objetos que mapeiem o tempo em ênuplas do tipo (E,V).

Na seção a seguir, veremos como os conhecimentos remático e dicente podem ser utilizados, de modo a gerar novos conhecimentos ou modificações no conhecimento existente.

4.4 Representação do Conhecimento Argumentativo

O conhecimento argumentativo é o mais sofisticado dentre todos os tipos de conhecimento. Sua complexidade tem origem em seu caráter transformador, ou seja, sua capacidade de ser em si um conhecimento e, além disso, transformar e criar novos conhecimentos. Os argumentos (ou conhecimentos argumentativos) incorporam o que é usualmente conhecido por inferência, raciocínio, aprendizagem, adaptação, etc.

O estudo dos argumentos na IA não tem sido feito de uma maneira unificada. O argumento dedutivo é normalmente estudado no âmbito da lógica, eventualmente em lógicas mais elaboradas tais como a lógica *fuzzy*. Os argumentos sintéticos, foram estudados, inicialmente, no contexto de aprendizagem de máquina (*machine*

learning) (Bolc, 1987; Michalski et. al. 1983; Michalski et.al. 1986). A área de aprendizagem de máquina enfatiza, principalmente, os argumentos indutivos (Michalski, 1983), e eventualmente combinações de argumentos indutivos com argumentos dedutivos dando origem ao chamado aprendizado por analogia (Carbonell, 1983). Michalski (1987) apresenta uma divisão dos diferentes tipos de aprendizagem:

- Implantação Direta do Conhecimento
- Aprendizagem por Instrução
- Aprendizagem por Dedução
- Aprendizagem por Analogia
- Aprendizagem por meio de Exemplos
- Aprendizagem por meio de Observação e Descobertas

O método básico em aprendizagem de máquina é chamado de aprendizagem conceitual (*conceptual learning*), onde a representação do conhecimento é feita por meio de predicados e extensões na lógica de predicados. Neste caso, existe uma grande lacuna no que diz respeito aos conhecimentos remáticos. Normalmente a ênfase é toda em conhecimentos dicentes. Em termos gerais, a abdução também é pouco citada (Ram & Leake, 1991; Bylander et.al. 1991), principalmente por envolver problemas de complexidade computacional (Bylander et.al. 1991).

Mais recentemente, as contribuições ao estudo dos argumentos tem origem na área conhecida como "inteligência computacional" (*computational intelligence*) (Zurada et.al, 1994), onde três grande áreas se distinguem: a área dos sistemas e lógica fuzzy, as redes neurais e a computação evolutiva. É interessante notar uma similaridade entre as três grandes áreas da inteligência computacional e os três tipos de argumentos. Embora essa comparação não seja totalmente precisa, pode-se associar os sistemas e lógica fuzzy com os argumentos dedutivos, as redes neurais com os argumentos indutivos e a computação evolutiva com os argumentos abdutivos. A computação evolutiva, de um modo especial, trouxe um novo alento ao estudo dos argumentos abdutivos, por tornar tratável, computacionalmente, a sua implementação (dentro de algumas restrições, obviamente).

Dependendo dos conhecimentos e tipos de conhecimento que estão envolvidos no processo de se modificar ou gerar novos conhecimentos vários tipos diferentes de argumentos podem ser identificados. Uma identificação completa de todos os tipos de argumentos está além dos objetivos deste trabalho. Entretanto, apresentaremos nesta seção uma definição formal para os três tipos básicos de argumentos, ou seja, os argumentos dedutivos, indutivos e abdutivos. Alguns exemplos desses argumentos, envolvendo conhecimentos remáticos e dicentes, são desenvolvidos na sequência.

Iniciamos, porém, com algumas considerações sobre características dos conhecimentos remáticos e dicentes que são necessárias para a classificação formal dos tipos de argumento.

4.4.1 Genericidade[1]

Dentro do escopo dos conhecimentos remáticos e dicentes, diferentes conhecimentos de um mesmo tipo podem ser classificados quanto a sua

[1] O termo "genericidade", que não existe naturalmente na língua portuguesa, é aqui introduzido de modo a caracterizar uma "medida" da generalidade.

genericidade, ou seja, quanto um deles é mais genérico que o outro. Em um sentido geral, um conhecimento será tão mais genérico quanto for o número de conhecimentos específicos que ele engloba. Em termos de conhecimentos dicentes, um conhecimento mais genérico é aquele que permite um maior número de interpretações com valor verdade "verdadeiro". Por exemplo, o resultado da conjunção de duas proposições é mais específico que cada uma individualmente, o resultado da disjunção de duas proposições é mais genérico que cada uma individualmente, etc.

Dentro do âmbito da teoria de conjuntos, o conjunto mais genérico é o conjunto universo, sendo que os diferentes conjuntos que são sub-conjuntos do conjunto universo podem ser ordenados por genericidade de acordo com a relação de inclusão. Ou seja, um conjunto que está contido dentro de um outro conjunto é mais específico que este ou, do mesmo modo, menos genérico. Observe, entretanto, que a relação de inclusão não leva a uma ordem total, mas a uma ordem parcial. Diferentes sub-conjuntos do conjunto universo que não compartilhem seus elementos, não podem ser ordenados pela relação de inclusão. Nesse caso, não há sentido em se dizer que um deles é mais genérico que o outro. Lembrando que uma relação pode ser vista como um conjunto, o conceito de genericidade pode ser estendido portanto para relações.

A idéia de genericidade é importante, pois esta leva a uma compactação da informação. Um único conhecimento genérico pode representar diversos conhecimentos específicos e quanto mais genérico este for, mais conhecimentos específicos estará representando. Sendo assim, em determinados casos é mais interessante se ter um único conhecimento genérico do que armazenar diversos conhecimentos específicos.

Em nosso caso, a idéia de genericidade será importante, pois a classificação dos argumentos se baseia na relação de genericidade entre os conhecimentos contidos na premissa e os conhecimentos contidos na conclusão dos argumentos. Argumentos em que o conhecimento contido nas conclusões está também contido nas premissas são chamados de argumentos analíticos ou dedutivos. Argumentos que, na conclusão, contém conhecimentos que não estão contidos nas premissas são chamados de argumentos sintéticos. Os argumentos dedutivos mantém ou explicitam os conhecimentos específicos contidos nas premissas. Os argumentos sintéticos incluem nos conhecimentos das conclusões, conhecimentos que não estão presentes nas premissas. Se essa inclusão é feita de uma maneira construtiva, por exemplo utilizando-se um critério de proximidade, (incluem-se conhecimentos que estão próximos dos conhecimentos das premissas, se é possível definir-se um critério de distância), o argumento é chamado indutivo. Se a inclusão é destrutiva, ou seja, geram-se diversos conhecimentos, eliminando-se aqueles que estão em desacordo com os outros conhecimentos nas premissas, o argumento é chamado abdutivo. Observe que, nesse caso, a inclusão ocorre porque ela não está em desacordo com os outros conhecimentos das premissas, ou seja, é uma inclusão plausível. Muitas vezes, é difícil determinar o que seja uma inclusão plausível. De um modo geral, a plausibilidade pode ser dada por uma função de plausibilidade e um limiar. Se o valor da função de plausibilidade para um dado conhecimento específico for maior que o limiar, este é incluído.

4.4.2 Modelo Formal de um Argumento

Um argumento pode ser definido formalmente como um objeto ativo. Os objetos pertencentes à interface de entrada deste objeto formarão um conjunto

chamado de conjunto de premissas do argumento. Os objetos pertencentes à interface de saída formarão um conjunto chamado de conjunto de conclusões. A função ativa do objeto é uma função que mapeia as premissas nas conclusões:

DEFINIÇÃO 4.29 - Argumento

Sejam:

- uma classe ativa A, contendo ênuplas do tipo $(a_1, \ldots, a_n, f)$.
- um objeto $a : T \rightarrow A$.
- P um conjunto formado pelos elementos pertencentes à interface de entrada de a.
- C um conjunto formado pelos elementos pertencentes à interface de saída de a.

O objeto a é chamado então de um argumento. O conjunto P corresponde ao conjunto de premissas do argumento e o conjunto C corresponde à conclusão do argumento.

Os argumentos podem ser argumentos puros ou argumentos híbridos. Os argumentos puros só possuem as interfaces de entrada e saída como campos não funcionais (não possuem variáveis internas). Os argumentos híbridos possuem variáveis internas, resultado de sua integração com algum conhecimento remático ou dicente. Sendo assim, os argumentos híbridos não são somente argumentos, mas estruturas que representam simultaneamente conhecimentos argumentativos e outros tipos de conhecimento. Em alguns casos pode ser interessante utilizar argumentos puros. Em outros casos, os argumentos híbridos podem ser mais interessantes.

Um objeto ativo pode ter mais de uma função. Um objeto ativo que tenha mais de uma função representa não somente um, mas tantos argumentos quantas sejam as suas funções. Entretanto, somente uma função pode ser disparada em um determinado instante. Isso implica que em um determinado instante, apenas um dos argumentos contidos no objeto ativo será funcional.

DEFINIÇÃO 4.30 - Argumento Dedutivo

Sejam:

- uma classe ativa A, contendo ênuplas do tipo $(a_1, \ldots, a_n, f)$.
- um argumento $a : T \rightarrow A$.
- P o conjunto de premissas de a.
- C o conjunto de conclusões de a.

Se a função f, para cada instância de a, é tal que os conhecimentos contidos em C estão incluídos na união dos conhecimentos contidos em P, então o argumento a é chamado de dedutivo, ou analítico.

DEFINIÇÃO 4.31 - Argumento Sintético

Sejam:

- uma classe ativa A, contendo ênuplas do tipo $(a_1, \ldots, a_n, f)$.
- um argumento $a : T \rightarrow A$.
- P o conjunto de premissas de a.
- C o conjunto de conclusões de a.

Se a função f é tal que nos conhecimentos contidos em C, existe algum conhecimento não contido na união dos conhecimentos contidos em P, então o argumento a é chamado de sintético.

DEFINIÇÃO 4.32 - Argumento Indutivo

Sejam:

- uma classe ativa A, contendo ênuplas do tipo (a_1 , ... , a_n , f).
- um argumento sintético a : T $\rightarrow$ A.
- P o conjunto de premissas de a.
- C o conjunto de conclusões de a.

Se a função f for tal que os conhecimentos contidos em C e não contidos na união dos conhecimentos em P sejam gerados utilizando-se conhecimentos contidos em P, de tal forma que possam ser comparados com conhecimentos contidos em P, em termos de um critério de distância, então o argumento a será dito indutivo.

DEFINIÇÃO 4.33 - Argumento Abdutivo

Sejam:

- uma classe ativa A, contendo ênuplas do tipo (a_1 , ... , a_n , f).
- um argumento sintético a : T $\rightarrow$ A.
- P o conjunto de premissas de a.
- C o conjunto de conclusões de a.

Seja pl uma função de plausibilidade, determinada a partir dos conhecimentos em P, que mede se um determinado conhecimento não está em contradição com os conhecimentos em P.

Se a função f for tal que os conhecimentos contidos em C e não contidos na união dos conhecimentos em P sejam gerados de uma maneira qualquer, e testados por meio da função de plausibilidade pl, então o argumento a será dito abdutivo [2].

Observe que os termos plausibilidade e função de plausibilidade não estão necessariamente vinculados à "medida de plausibilidade" definida na teoria de Dempster-Shafer (Shafer, 1976). Seu significado se dá mais no sentido de (Collins & Michalski, 1989), indicando quão plausível, ou quão "razoável" é o conhecimento gerado pelo argumento.

As definições de argumentos indutivos e abdutivos são bastante genéricas, de modo a abranger diversos tipos diferentes de conhecimentos. De um modo geral, nota-se uma polaridade entre estes tipos de argumentos. O argumento indutivo é um argumento construtivo, no sentido de que ele gera conhecimentos novos que são baseados em conhecimentos existentes nas premissas. Esses conhecimentos novos não precisam ser testados, pois sua proximidade com conhecimentos já existentes, devem (em princípio) garantir sua plausibilidade. O argumento abdutivo, por outro lado, é um argumento destrutivo. Não importa que método é utilizado para a geração destes novos conhecimentos, uma vez que eles serão testados, e os conhecimentos que não forem plausíveis serão eliminados. Observe que o fato de não se importar como esses são gerados permite, por exemplo, que estes sejam gerados indutivamente (ou seja, a partir de conhecimentos nas premissas). Nesse

[2] Em (Bylander et.al., 1991), outra definição de abdução é encontrada. Apesar das definições a princípio parecerem diferentes, a definição aqui apresentada é mais genérica, pois inclui um maior número de casos que podem ser chamados de abdução. O escopo da definição de Bylander é nesse sentido mais reduzido.

caso, um argumento abdutivo pode ter componentes indutivas. De um modo geral, o que caracteriza um argumento indutivo é que os conhecimentos nas premissas são utilizados para gerar os novos conhecimentos incluídos. No caso dos argumentos abdutivos, o conhecimento nas premissas é usado para testar a plausibilidade de conhecimentos que serão inseridos nas conclusões. Se eles forem utilizados também para gerar esses novos conhecimentos, pode-se dizer que o argumento é ao mesmo tempo abdutivo e indutivo. Normalmente, argumentos desse tipo são importantes quando os conhecimentos novos sendo gerados, apesar de gerados a partir de conhecimentos nas premissas, têm com estes uma relação de distância onde essa distância é grande (ou seja, a semelhança entre esses novos conhecimentos e os conhecimentos nas premissas é bem pequena). Nesses casos, apesar de terem origem em conhecimentos das premissas, sua plausibilidade pode ser comprometida. Assim, nada melhor do que testar sua plausibilidade. Nesses casos, têm-se argumentos indutivos-abdutivos.

Em alguns casos, é interessante determinar-se não somente um conhecimento plausível, mas o conhecimento mais plausível. Nesses casos, a função de plausibilidade é utilizada de modo que de diversos conhecimentos plausíveis, somente o mais plausível não será eliminado. Esse é um tipo de abdução extrema, utilizado nos algoritmos genéticos e em (Bylander et.al. 1991).

4.4.3 *Exemplos de Argumentos*

Nesta sub-seção, apresentaremos alguns exemplos dos tipos mais importantes, e mais usuais de argumentos.

EXEMPLO 1: Argumento Indutivo para a Geração de Conhecimento Remático Sensorial Genérico a partir de Conhecimentos Rematicos Sensoriais Específicos, com Exemplos somente Positivos

A meta deste argumento é a partir de um conjunto de conhecimentos rematicos sensoriais específicos, tomados como exemplos positivos, gerar um conhecimento remático sensorial genérico (inicialmente vazio), de tal forma que cada conhecimento específico seja um caso do conhecimento genérico. Nesse exemplo, a construção do objeto genérico que representa o conhecimento remático sensorial genérico será dada pela simples união das instâncias de todos conhecimentos específicos (tomados como *singletons*).

Sejam:

- uma classe passiva $X = X_1 \times \ldots \times X_n$.
- $P = \{p_1 , \ldots , p_m\}$ um conjunto de m objetos p_i do tipo X.
- c um objeto genérico de tipo X (inicialmente vazio).
- uma classe ativa A, formada por elementos $(a_1 , \ldots , a_m , a_{m+1} , f_1)$, onde para $i = 1, \ldots , m$, $a_i \in X$, e $a_{m+1} \subseteq X$. A função $f_1 : X^m \rightarrow 2^X$ pode ser descrita algoritmicamente do seguinte modo:

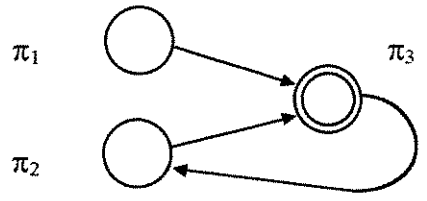
```
f1 (a1 , ... , am , am+1)
am+1 = ∅
Para i = 1 ... m
      am+1 = am+1 ∪ {ai}
```

- a um objeto ativo de tipo A.

Os objetos $p_1 , \ldots , p_m$ correspondem à representação dos m conhecimentos rematicos sensoriais específicos, sendo utilizados como a premissa do argumento. O

objeto c corresponde à representação do conhecimento remático sensorial genérico que será gerado interativamente pelo argumento, sendo portanto sua conclusão.O objeto a corresponde ao argumento indutivo quer irá gerar interativamente o objeto genérico.

Considere uma rede de objetos representada graficamente da seguinte maneira:



onde $\xi(t_k , p_1) = \ldots = \xi(t_k , p_m) = \pi_1$, $\xi(t_k , c) = \pi_2$, $\xi(t_k , a) = \pi_3$.

O disparo do objeto a em t_k corresponde à geração de uma nova instância do objeto c, no instante k+1. Esse disparo corresponde à ação do conhecimento argumentativo representado por a sobre os conhecimentos rematicos sensoriais específicos representados por p_i , atualizando o conhecimento remático sensorial genérico c. Se os conhecimentos p_i por sua vez estiverem também sendo atualizados (por exemplo, por um sensor), então a cada instante, as instâncias de c serão geradas de acordo com as novas instâncias dos p_i .

Observe que no objeto genérico c, os valores das instâncias correspondem a conjuntos formados pela união dos valores das instâncias dos objetos p_i e nada mais. Apesar disso, essa técnica acrescenta genericidade na conclusão, pois os conhecimentos específicos representados na conclusão (ou seja, os casos do objeto genérico) podem ser diferentes dos conhecimentos representados nas premissas. Como esses casos a mais podem ser comparados (em termos de uma medida de distância) aos conhecimentos rematicos nas premissas, esse argumento é indutivo.

EXEMPLO 2: Argumento Abdutivo para a Geração de Conhecimento Remático Sensorial Genérico a partir de Conhecimentos Rematicos Sensoriais Específicos, com Exemplos Positivos e Negativos

A meta deste argumento também é a partir de um conjunto de conhecimentos rematicos sensoriais específicos, tomados como exemplos positivos, e outro conjunto de conhecimentos rematicos sensoriais específicos, tomados como exemplos negativos, gerar interativamente um conhecimento remático sensorial genérico, onde os exemplos positivos estejam incluídos no conhecimento genérico e os exemplos negativos não estejam.

Sejam:

- uma classe passiva $X = X_1 \times \ldots \times X_n$.
- $P = \{p_1 , \ldots , p_m\}$ um conjunto de m objetos p_i do tipo X.
- $N = \{n_1 , \ldots , n_o\}$ um conjunto de o objetos n_i do tipo X.
- c um objeto genérico de tipo X (inicialmente vazio).
- uma classe ativa A, formada por elementos ($a_1 , \ldots , a_m , a_{m+1} , \ldots , a_{m+o} , a_{m+o+1} , f$), onde para i de 1 a m, $a_i \in X$ correspondem a exemplos positivos a serem considerados, para i de m+1 a m+o, $a_i \in X$ correspondem a exemplos negativos a serem considerados e $a_{m+o+1} \subseteq X$. A função $f : X^{2m} \rightarrow 2^X$ pode ser descrita algoritmicamente do seguinte modo:

```
f1 (a1 , ... , am+o , am+o+1)
am+o+1 = ∅
Para i = 1 ... m
      am+o+1 = am+o+1 ∪ {ai}
Para i = 1 ... α
      Gerar aleatoreamente bi ∈ X.
      Se pl(bi , a1 , ... , am+o ) > 0.5
            am+o+1 = am+o+1 ∪ {bi}
```

Observe que esse algoritmo utiliza uma constante α e uma função auxiliar de plausibilidade pl. O valor de α é um parâmetro do sistema. Quanto maior for o valor de α, maior a probabilidade de se incluir um novo conhecimento.

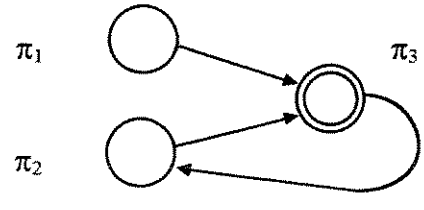
A função de plausibilidade pl verifica a proximidade dos pontos gerados com os exemplos positivos e negativos. Caso o ponto corresponda exatamente a um exemplo positivo, a plausibilidade é igual a 1. Caso corresponda exatamente a um exemplo negativo, a plausibilidade é igual a 0. Caso seja um ponto intermediário, a plausibilidade é calculada com base no exemplo positivo mais próximo e no exemplo negativo mais próximo. A função pl pode ser dada algoritmicamente do seguinte modo:

```
pl(b, a1 , ... , am+o )
Se para algum i entre 1 e m, b == ai
     retorne (1);
Se para algum i entre m+1 e m+o, b == ai
     retorne(0);
Senão
     Para i ∈ {1,...,m} encontre β = min{b-ai }
     Para i ∈ {m+1,...,m+o} encontre γ = min{b-ai}
     retorne ( 1 / (1+e^(-(γ-β)/βγ)) )
```

Seja a um objeto ativo de tipo A.

Os objetos p_1 , ... , p_m correspondem à representação dos m conhecimentos remáticos sensoriais específicos, utilizados como exemplos positivos. Os objetos n_1 ,... , n_o correspondem à representação dos o conhecimentos remáticos sensoriais específicos, utilizados como exemplos negativos. O objeto c corresponde à representação do conhecimento remático sensorial genérico que será gerado interativamente pelo argumento, sendo portanto sua conclusão.O objeto a corresponde ao argumento abdutivo quer irá gerar interativamente o objeto genérico.

Seja agora uma rede de objetos representada graficamente da seguinte maneira:



onde $\xi(t_k , p_1) = \ldots = \xi(t_k , p_m) = \xi(t_k , n_1) = \ldots = \xi(t_k , n_o) = \pi_1$,
$\xi(t_k , c) = \pi_2$, $\xi(t_k , a) = \pi_3$.

O disparo do objeto a em t_k corresponde à geração de uma nova instância do objeto c, no instante k+1. Esse disparo corresponde à ação do conhecimento argumentativo representado por a sobre os conhecimentos remáticos sensoriais específicos representados por p_i e n_i, atualizando o conhecimento remático sensorial genérico c. Se os conhecimentos p_i e n_i por sua vez estiverem também sendo atualizados (por exemplo, por um sensor), então a cada instante, as instâncias de c serão geradas de acordo com as novas instâncias dos p_i e n_i .

Observe que no objeto genérico c, os valores das instâncias correspondem a conjuntos formados pela união dos valores das instâncias dos objetos p_i , além de pontos aleatórios que não estão em desacordo com as instâncias dos objetos n_i . Como esses novos pontos são incluídos de acordo com uma função de plausibilidade, trata-se de um argumento abdutivo.

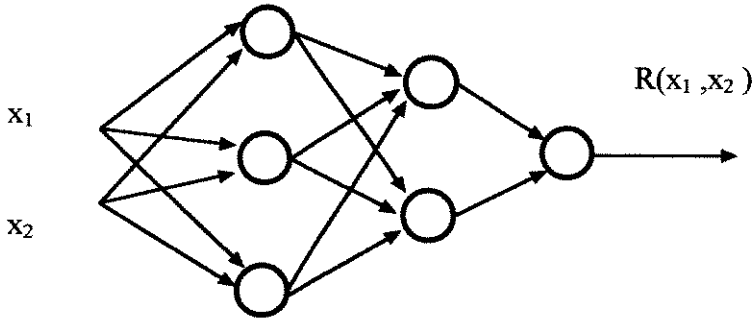
EXEMPLO 3: Argumento Indutivo para a Modificação de Conhecimento Remático Sensorial Genérico a partir de Conhecimento Remático Sensorial Específico (Rede Neural)

Esse argumento implementa o aprendizado indutivo em uma rede neural.

Seja uma classe passiva $X = X_1 \times X_2$.

Seja um conhecimento remático sensorial específico representado por um objeto x de tipo X.

Seja uma classe passiva A formada por ênuplas do tipo $(a_1 , \ldots , a_{20})$, onde cada elemento a_i corresponde a um parâmetro da seguinte rede neural.



Os elementos da ênupla correspondem respectivamente aos seguintes parâmetros da rede neural:

$a_1 = w^1_{11}$, $a_2 = w^1_{12}$, $a_3 = w^1_{13}$, $a_4 = w^1_{21}$, $a_5 = w^1_{22}$, $a_6 = w^1_{23}$,
$a_7 = w^2_{11}$, $a_8 = w^2_{12}$, $a_9 = w^2_{21}$, $a_{10} = w^2_{22}$, $a_{11} = w^2_{31}$, $a_{12} = w^2_{32}$,
$a_{13} = w^3_{11}$, $a_{14} = w^3_{21}$, $a_{15} = \theta^1_1$, $a_{16} = \theta^1_2$, $a_{17} = \theta^1_3$, $a_{18} = \theta^2_1$,
$a_{19} = \theta^2_2$, $a_{20} = \theta^3_1$,

onde w^k_{ij} , corresponde ao peso da i-ésima entrada do j-ésimo neurônio da camada k, e θ^k_i corresponde ao offset do i-ésimo neurônio da camada k.

A saída da rede é calculada em três etapas. Primeiro, para j = 1, ... , 3:

$$x_j^1 = f\left(\sum_{i=1}^{2} w_{ij}^1 . x_i - \theta_j^1\right)$$

e em seguida, para j = 1, 2:

$$x_j^2 = f\left(\sum_{i=1}^{3} w_{ij}^2 . x_i^1 - \theta_j^2\right)$$

e por fim

$$R(x_1, x_2) = f\left(\sum_{i=1}^{2} w_{i1}^3 . x_i^2 - \theta_1^3\right)$$

onde $f(x) = \frac{1}{1+e^{-x}}$

Seja um conhecimento remático sensorial genérico, representado por um objeto a de tipo A. Observe que o objeto a representa não só o instante atual da rede neural, mas toda sua evolução temporal.

Seja uma classe C, dada por ênuplas do tipo (c_1 , c_2 , c_3 , f_1), onde c_1 corresponde a uma ênupla (a_1 , ... , a_{20}), c_2 corresponde a uma ênupla (x_1 , x_2) e c_3 corresponde a uma ênupla (a_1 , ... , a_{20}) . Os elementos c_1 e c_2 correspondem à interface de entrada e c_3 corresponde à interface de saída. A função f_1 é a função de aprendizado, calculando c_3 em função de c_1 e c_2 do seguinte modo::

Primeiro,

$$e_1^3 = R(x_1, x_2).(1 - R(x_1, x_2))^2$$

depois, para j = 1,2:

$$e_j^2 = x_j^2 . (1 - x_j^2) . e_1^3 . w_{j1}^3$$

Em seguida, para j = 1, .. , 3:

$$e_j^1 = x_j^1 . (1 - x_j^1) . \sum_{k=1}^{2} e_k^2 . w_{jk}^2$$

Por fim, faz-se: para i = 1, 2:

$$w_{i1}^3(t+1) = w_{i1}^3(t) + \eta . e_1^3 . x_i^2$$

para i = 1, ... , 3 e j = 1,2:

$$w_{ij}^2(t+1) = w_{ij}^2(t) + \eta . e_j^2 . x_i^1$$

para i = 1,2 e j = 1, ... , 3:

$$w_{ij}^1(t+1) = w_{ij}^1(t) + \eta . e_j^1 . x_i$$

e

$$\theta_1^3(t+1) = \theta_1^3(t) + \eta.e_1^3$$

para j = 1, 2:

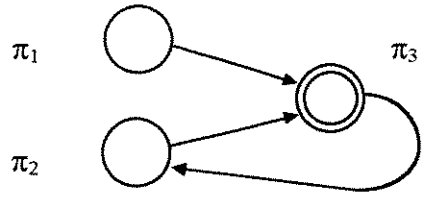
$$\theta_j^2(t+1) = \theta_j^2(t) + \eta.e_j^2$$

para j = 1, ... , 3:

$$\theta_j^1(t+1) = \theta_j^1(t) + \eta.e_j^1$$

Com isso, todos os 20 elementos de c_3 são calculados.

Seja um objeto c de tipo C. O objeto c corresponde ao argumento indutivo que modifica o conhecimento genérico representado em a, utilizando para tal o conhecimento específico colocado em x. Para tanto, tem-se a seguinte rede de objetos:



onde $\xi(t_k , x) = \pi_1$, $\xi(t_k , a) = \pi_2$, $\xi(t_k , c) = \pi_3$.

O disparo do objeto c em t_k corresponde à modificação do objeto a, no instante t_{k+1} , seguido do consumo do objeto x. Esse disparo corresponde à ação do conhecimento argumentativo representado por c sobre o conhecimento remático sensorial específico representados por x , modificando o conhecimento remático sensorial genérico representado por a.

EXEMPLO 4 : Argumento Dedutivo para a Geração de Conhecimento Dicente a partir de Conhecimentos Dicentes (Modus Ponens):

Seja uma classe passiva P, dada por ênuplas do tipo (e, v), onde e ∈ E, onde E é o conjunto de todas as expressões, e v ∈ {0,1}.

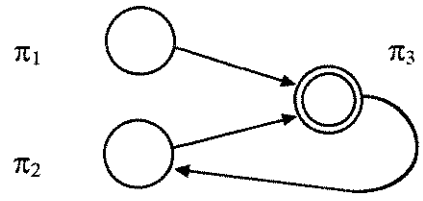
Seja p_1 um objeto da classe P, tal que em um instante t_k , a expressão correspondente a p_1 (t) seja e_1 = "$s_1 \rightarrow s_2$ ", cujo valor verdade seja $v_1 = 1$. Observe que p_1 corresponde à representação de um conhecimento dicente, dado no instante t_k pela ênupla (e_1 , v_1).

Seja p_2 um outro objeto da classe P. Suponha que em um instante t_k , a expressão correspondente a p_2 (t_k) seja e_2 = "s_1 " e que o valor verdade $v_2 = 1$.

Seja uma classe ativa A, dada por ênuplas do tipo (a_1 , a_2 , a_3 , f_1), onde a_1 , a_2 e a_3 pertencem a P, a_1 e a_2 correspondem à interface de entrada, a_3 corresponde à interface de saída e f_1 corresponde a uma função dada pelo seguinte algoritmo:

```
f1 (a1 , a2 , a3)
     if (a1 == "x→y" && a2 == "x")
          a3 = ("y", 1);
     else a3 = ("", 0);
```

Seja a um objeto de tipo A. O objeto a corresponde ao argumento dedutivo que a partir de dois conhecimentos dicentes, representados por p_1 e p_2, gerará um terceiro conhecimento dicente p_3 , que corresponde a uma dedução em função de p_1 e p_2 . Esse procedimento é dado pela rede de objetos a seguir:



onde $\xi(t_k , p_1) = \pi_1$, $\xi(t_k , p_2) = \pi_2$, $\xi (t_k , a) = \pi_3$.

O disparo do objeto a em t_k corresponde à geração de um objeto p_3 , no instante t_{k+1} . Esse disparo corresponde à ação do conhecimento argumentativo representado por a sobre os conhecimentos dicentes representados por p_1 e p_2 , gerando o conhecimento dicente representado por p_3 . Observe que aqui, a função γ, que seleciona quais objetos serão utilizados para o disparo, precisa ser seletiva. Se os objetos p_1 e p_2 não forem compatíveis, o argumento não poderá ser disparado. No caso deste exemplo, os objetos eram compatíveis, pois um deles correspondia ao antecedente do outro. Do mesmo modo, γ deve selecionar um escopo habilitante tal que os objetos p_1 e p_2 não sejam consumidos, pois estas proposições podem ser úteis em algum instante de tempo futuro. Observe também, que o argumento aqui apresentado somente funcionará quando um dos conhecimentos dicentes tiver como expressão um único símbolo, e este for o único consequente de um conhecimento dicente condicional. Para outros tipos de conhecimentos dicentes condicionais, com mais de um antecedente, outros argumentos (que entretanto serão semelhantes a esse) terão que ser utilizados.

4.5 Organização de Sistemas Inteligentes

De acordo com Albus (1991), um sistema inteligente é formado por quatro módulos que se intercomunicam entre si: Percepção Sensorial (PS), Modelo do Ambiente (MA), Julgamento de Valores (JV) e Geração de Comportamento (GC), conforme ilustrado na figura 4.6 a seguir:

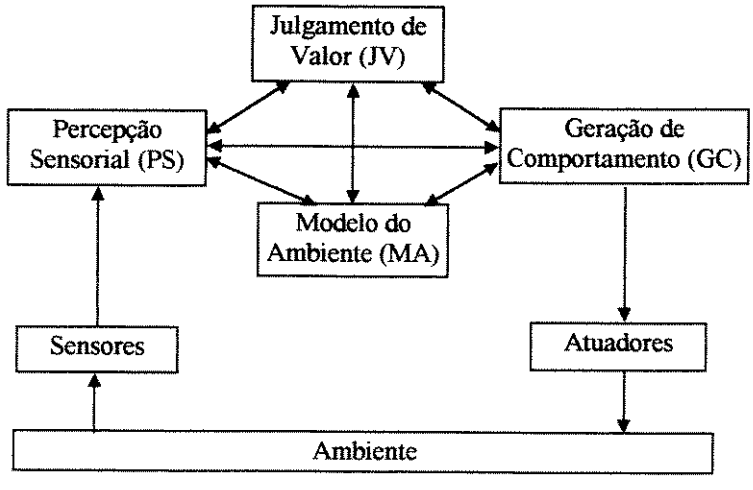


Figura 4.6 - Módulos de um Sistema Inteligente

O módulo PS compara a entrada oriunda dos sensores com expectativas geradas pelo MA. Em PS são integrados em função do tempo e espaço as similaridades e diferenças entre as observações e as expectativas, de modo a detectar eventos e reconhecer padrões, objetos e inter-relacionamentos no ambiente. Os dados oriundos de uma vasta gama de sensores, durante períodos extendidos de tempo, sofrem um processo de fusão, gerando uma percepção consistente e unificada do estado do ambiente. Os algoritmos em PS devem computar a distância, o formato e a orientação, além de características da superfície, e atributos físicos e dinâmicos dos objetos e regiões do espaço.

O módulo MA oferece a cada instante a melhor estimativa do estado do ambiente, vista pelo sistema inteligente. Em MA existe uma base de dados sobre o ambiente, além de um sistema de armazenamento e recuperação de informação, operando sobre esta base de dados. Por meio de um processo de simulação, MA é capaz de gerar expectativas e predições. Sendo assim, MA é capaz de atender a requisições de informações sobre o passado, o presente e o futuro provável do estado do ambiente. Ele provê tais serviços ao módulo GC, de modo que este possa gerar planos e selecionar comportamentos. Também o faz ao módulo PS, de modo que esse possa gerar correlações, comparações com o modelo do ambiente e o reconhecimento de estados, objetos e eventos, baseado no modelo do ambiente. Provê serviços também ao módulo JV, de modo que este possa computar valores como custo, benefício, risco, incerteza, importância, atratividade, etc. O módulo MA é mantido atualizado por meio das informações oriundas de PS.

O módulo JV determina o que é bom ou ruim, recompensa ou punição, importante ou trivial, certo ou improvável. Em JV é feita uma avaliação conjunta do estado atual do ambiente com as predições resultantes dos planos hipotéticos gerados por GC. São computados os fatores custo, risco e benefício, tanto das situações observadas como das atividades planejadas. Às variáveis de estado são atribuídas medidas de probabilidade de exatidão, credibilidade e incerteza. Também são atribuídas medidas de atratividade ou repulsividade aos objetos, eventos e regiões do espaço. Sendo assim, o módulo JV gera subsídios para que decisões sejam tomadas, selecionando-se uma ação em contraposição a outra. Sem um sistema de valores, nenhum sistema inteligente consegue atingir seus objetivos.

O módulo GC deve ser capaz de gerar planos, metas e objetivos, além de executar tarefas. Tarefas são recursivamente decompostas em sub-tarefas, que são sequeciadas de modo a atingir-se os objetivos. Metas são selecionadas e planos gerados por meio da interação cíclica entre os módulos GC, MA e JV. O módulo GC gera planos, o módulo MA prediz os resultados desses planos e o módulo JV avalia esses resultados. Então, GC seleciona os planos com a melhor avaliação e os executa por meio dos atuadores. Além disso, monitora a execução de planos anteriores, modificando-os quando necessário.

Observe que existe uma estreita relação entre as idéias de Albus e as idéias de conhecimento designativo, apraisivo e prescritivo colocadas no capítulo 2.

Os módulos PS e MA podem ser formados por conhecimentos designativos, ou seja, conhecimentos remáticos sensoriais, de objeto ou de ocorrências, bem como conhecimentos dicentes, que são utilizados para representar o estado atual e passado do ambiente, bem como fazer predições para o futuro.

O módulo JV pode ser formado por conhecimentos apraisivos, normalmente conhecimentos remáticos sensoriais ou conhecimentos dicentes, que são utilizados para fazer avaliações ou apreciações sobre o estado atual e predições sobre possíveis estados futuros.

O módulo GC pode ser formado por conhecimentos designativos e prescritivos. Os conhecimentos designativos são os planos e metas a serem atingidos. Os conhecimentos prescritivos são aqueles utilizados para gerar efetivamente um comportamento, enviando dados aos atuadores.

Com isso, a arquitetura de um sistema inteligente pode ser considerada conforme a figura 4.7 a seguir:

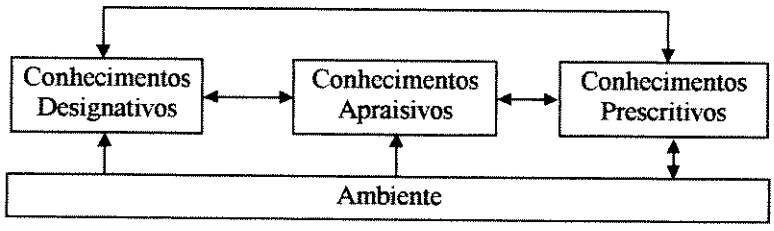


Figura 4.7 - Arquitetura Alternativa para um Sistema Inteligente

Todos esses conhecimentos são representados por meio de objetos, (lembrando-se que objetos podem também representar objetos genéricos, objetos fuzzy, meta-objetos, meta-objetos genéricos ou meta-objetos fuzzy), colocados em lugares apropriados. Para inter-relacionar essas componentes, conhecimentos argumentativos, colocados na forma de objetos ativos, devem ser utilizados. Assim, obtém-se uma rede de objetos que deverá atuar como um sistema inteligente, nos moldes descritos por Albus.

Entretanto, alguns cuidados devem ser tomados. Em muitos casos, é necessário analisar, em um determinado instante de tempo, não só a informação referente a este instante de tempo, mas também a instantes anteriores (e.g., para utilizar conhecimentos de ocorrências, conforme a seção 4.2.5). Nesse caso, os objetos criados para representar tais conhecimentos devem ser dotados de uma memória temporal, ao invés de utilizar somente a informação instantânea. Esse tipo de situação é muito comum, eminentemente com o conhecimento advindo dos sensores. Assim, esquemas alternativos como o mostrado na figura 4.8 a seguir devem ser utilizados:

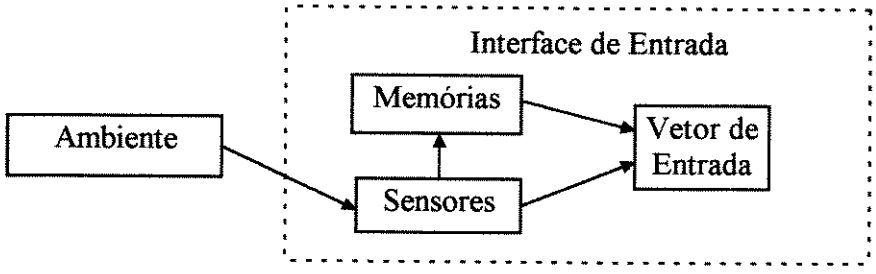
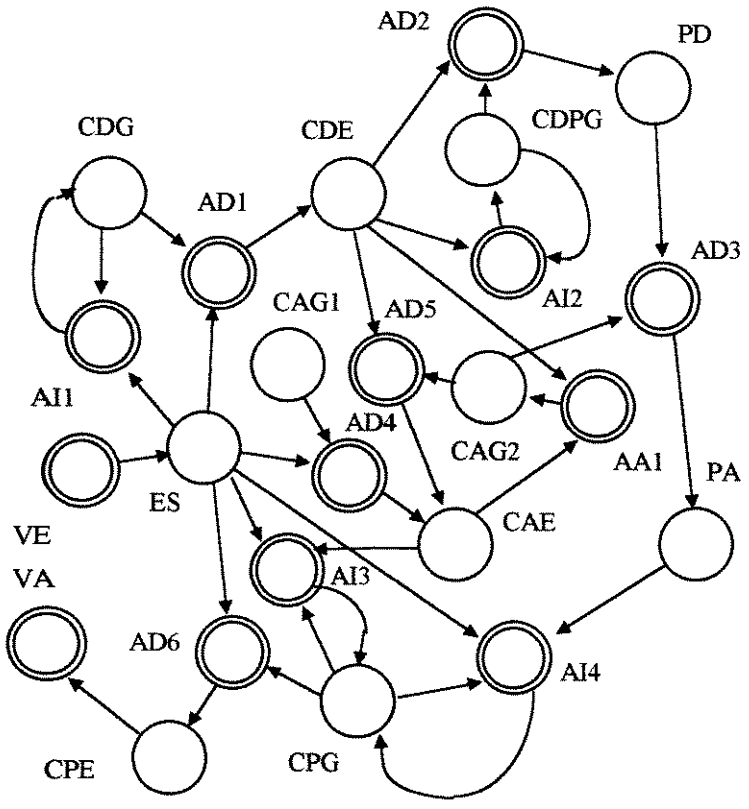


Figura 4.8 - Modelo de Sistema Sensorial

Ao invés de se gerar objetos com instâncias utilizando-se os valores dos sensores, esses são previamente processados gerando o chamado vetor de entrada. O vetor de entrada, além dos dados instantâneos dos sensores incorpora uma memória temporal destes. As medidas dos sensores efetuam um mapeamento entre grandezas físicas do ambiente e uma representação numérica para tais grandezas. Em muitos casos, esse mapeamento poderá envolver uma certa perda de

informação, na medida que os sensores não são ideais, e uma série de distorções podem ser incorporadas na informação, tais como ruídos, erros de quantização, erros de calibração, histereses, deformações limiares (nos limites dos sensores), etc. Com o processamento auxiliar indicado, as informações contidas no vetor de entrada, além de multi-dimensionais tornam-se também multi-temporais, ou seja, passam a conter informações de múltiplas fontes em múltiplos instantes de tempo. Assume-se que a informação contida em um vetor de entrada, em um determinado instante, é a **melhor representação** do ambiente possível neste instante. Sendo assim, o objeto utilizado para representar a interface de entrada do sistema inteligente deve utilizar instâncias do vetor de entrada, e não simplesmente dos sensores.

Um exemplo de um sistema inteligente, organizado utilizando-se diferentes tipos de conhecimentos, e descrito na forma de uma rede de objetos pode ser visualizado na figura 4.9 .



Legenda			
VE	Vetor de Entrada	CDPG	Conhecimento Designativo Preditivo Genérico
VA	Vetor de Atuadores		
ES	Espaço Sensorial	PD	Predição Designativa
ADn	Argumento Dedutivo n	CAGn	Conhec. Apraisivo Genérico n
AIn	Argumento Indutivo n	CAE	Conhec. Apraisivo Específico
AAn	Argumento Abdutivo n	PA	Predição Apraisiva
CDG	Conhec. Designativo Genérico	CPG	Conhec. Prescritivo Genérico
CDE	Conhec. Designativo Específico	CPE	Conhec. Prescritivo Específico

Figura 4.9 - Exemplo de um Sistema Inteligente

Neste exemplo, o objeto fonte em VE (Vetor de Entrada), alimenta o lugar (ES) (Espaço Sensorial), com informações do vetor de entrada. Um argumento indutivo (AI1) utiliza essas informações sensoriais para modificar os objetos em (CDG) (Conhecimento Designativo Genérico). Os objetos em CDG representam objetos genéricos, que armazenam padrões designativos que ocorrem no vetor de entrada. Por meio de um argumento dedutivo (AD1), a entrada sensorial em (ES) é comparada com esse conhecimento genérico, gerando objetos em CDE (Conhecimentos Designativos Específicos), ou seja, os padrões específicos detectados em um determinado momento. Os objetos em CDPG CDPG (Conhecimentos Designativos Preditivos Genéricos) correspondem a objetos genéricos que correlacionam sequências de objetos que ocorrem em CDE, de modo permitir a geração de previsões. Esses objetos são utilizados para determinar-se uma predição designativa (PD), ou seja, uma previsão dos objetos que estão por vir em CDE, por meio do argumento dedutivo AD2. Os objetos em CDPG são gerados e modificados a partir de sequências de objetos em CDE, por meio do argumento indutivo AI2. Esta seção da rede corresponde ao módulo de conhecimento designativo indicado na figura 4.7.

De maneira análoga, a partir da entrada sensorial em ES e de um conhecimento apraisivo genérico (CAG1), o argumento dedutivo (AD4) gera um conhecimento apraisivo específico (CAE). Observe que o conhecimento em CAG1 não é aprendido, mas é inato no sistema inteligente (pois tem a ver com seus objetivos). Entretanto, uma série de conhecimentos apraisivos genéricos (CAG2) pode ser aprendida, a partir da correlação de instâncias em CDE e CAE. Um argumento abdutivo (AA1) observa as instâncias em CDE e CAE e modifica CAG2, de modo que os conhecimentos designativos em CDE possam gerar conhecimentos apraisivos específicos em CAE, por meio do argumento dedutivo AD5. Esses mesmos conhecimentos apraisivos genéricos são utilizados para a partir das predições designativas (PD), gerar predições apraisivas (PA), por meio do argumento dedutivo (AD3). Esta seção da rede corresponde ao módulo de conhecimentos apraisivos indicado na figura 4.7. Os argumentos AD4 e AD5 correspondem ao arco que conecta os módulos de conhecimentos designativos e conhecimentos apraisivos.

A entrada sensorial em ES é utilizada então, junto com um conhecimento prescritivo genérico (CPG), para gerar um conhecimento prescritivo específico (CPE), por meio do argumento dedutivo AD6. Esse conhecimento prescritivo em (CPE) é consumido pelo vetor de atuadores (VA), encerrando o ciclo de controle. O conhecimento prescritivo genérico em CPG é atualizado constantemente utilizando-se a entrada sensorial (ES) e o conhecimento apraisivo específico em CAE, por meio do argumento indutivo (AI3). Do mesmo modo, é atualizado pela entrada sensorial em ES e as predições apraisivas em PA, por meio do argumento indutivo AI4. Essa seção da rede comporta o módulo de conhecimentos prescritivos, indicado na figura 4.7. Os argumentos em AI3 e AI4 correspondem ao arco conectando os módulos de conhecimentos apraisivos e conhecimentos prescritivos.

Como pode ser percebido pelo exemplo, à medida que diferentes tipos de conhecimentos são utilizados, principalmente envolvendo aprendizado de alguns destes, a representação do sistema inteligente como um todo torna-se bem complicada. Neste exemplo, utilizaram-se, somente, conhecimentos remáticos sensoriais (específicos e genéricos), e conhecimentos argumentativos. Do mesmo modo, a comunicação entre conhecimentos designativos e apraisivos ocorre somente em um sentido. O mesmo sucede entre conhecimentos apraisivos e

prescritivos. Não foi considerada a comunicação direta entre conhecimentos designativos e conhecimentos prescritivos. Redes que implementem tipos mais sofisticados de conhecimentos e outros tipos de inter-relacionamento entre eles podem tornar-se bem complicadas.

4.6 Resumo

Nesse capítulo, apresentou-se uma formalização para a representação dos diversos tipos de conhecimentos considerados no segundo capítulo. Iniciou-se com a formalização dos conhecimentos remáticos, passando-se aos conhecimentos dicentes e então aos conhecimentos argumentativos. Por fim, colocou-se um exemplo de como conhecimentos podem ser integrados de modo a resultar um sistema inteligente.

No próximo capítulo, será apresentado um exemplo de aplicação para os conceitos apresentados: a navegação autônoma de veículos.

5. Exemplo de Aplicação - Navegação de Veículo Autônomo

5.1 Introdução

Nesse capítulo, será apresentado um exemplo de aplicação do modelo de representação e processamento do conhecimento colocado no capítulo anterior. Esse exemplo é meramente ilustrativo. Como a teoria é muito abrangente, não é possível utilizar-se em um único exemplo, todos os tipos de conhecimentos descritos pela teoria. O presente exemplo, no entanto, é suficientemente complexo para demonstrar as principais características desta.

O problema escolhido como ilustração é o problema da navegação de um veículo autônomo. Esse problema já é bem conhecido na literatura, mas continua um problema atual, com alguns desafios a serem vencidos. Exemplos na literatura recente incluem (Wang et.al. 1991; Verschure et.a. 1992; Fan & Lui 1994; Spence & Hutchinson, 1995; Krozel & Andrisani II 1995; Beom & Cho, 1995; Rao, 1995; Yen & Pfluger, 1995).

Pode-se descrever resumidamente o problema do seguinte modo: um veículo autônomo se encontra em um determinado ambiente, composto por obstáculos e metas, e o problema consiste em controlar o veículo de modo que o mesmo navegue pelo ambiente evitando os obstáculos e atingindo a(s) meta(s).

As propostas de abordagem deste problema, podem ser classificadas em três categorias. Na primeira (e mais antiga), o problema é resolvido a partir de um modelo prévio do ambiente, descrevendo os obstáculos e a meta. Normalmente, a geometria do veículo é desconsiderada, tomando-se a menor circunferência que o envolve, e acrescentando seu raio à borda dos obstáculos. Com isso o problema fica resumido à trajetória de um ponto, o que é resolvido por meio de algum mecanismo de busca. Um exemplo deste tipo de abordagem pode ser visto em (Krozel & Andrisani II, 1995). Em alguns casos, a geometria do veículo é considerada, de modo a permitir que o mesmo possa navegar por ambientes onde a circunferência que o envolve não conseguiria atravessar determinados trechos, mas o veículo em si consegue (Fan & Lui 1994). Exemplos mais complexos incluem a navegação em ambiente com obstáculos móveis (Spence & Hutchinson, 1995).

A segunda proposta de resolução do problema não utiliza um modelo do ambiente, mas é caracterizada por um comportamento reativo, diante de estímulos advindos de sensores. A proposta de Brooks (1991) é baseada em comportamentos, onde o veículo possui uma série de comportamentos previamente programados, que são ativados seletivamente a partir de estímulos advindos de sensores. Chen & Trivedi (1995) apresentam uma proposta geral para planejamento de trajetória em sistemas baseados em sensores, que inclui também o problema da navegação. As outras propostas, baseadas explicitamente em sensores, não efetuam o planejamento da trajetória, mas proporcionam a combinação de dois comportamentos básicos: evitar obstáculos e aproximação da meta. A partir dos dados dos sensores, uma ação final de controle é determinada efetuando-se a combinação destes dois

comportamentos. Observe a diferença entre essa abordagem e a abordagem de Brooks, onde havia não a combinação de vários comportamentos, mas a seleção de um deles. Exemplos deste tipo de abordagem incluem (Yen & Pfluger, 1995; Beom & Cho, 1995). Estes comportamentos básicos podem ser previamente programados, ou podem ser aprendidos a partir da interação do veículo com o ambiente. Exemplos que incluem este aprendizado podem ser encontrados em (Verschure et.al. 1991; Oliveira et.al. 1994; Fabro 1996).

O terceiro tipo de abordagem corresponde a uma abordagem híbrida entre o primeiro e segundo tipo. Nesse caso, utilizam-se os sensores para a determinação de um modelo incremental do ambiente, onde é efetuado um planejamento da trajetória a ser seguida. Um exemplo deste tipo de abordagem é encontrado em (Rao, 1995).

Todas estas abordagens tem os seus prós e os seus contras. A abordagem de resolução via modelo do ambiente garante que, se existe um caminho para o veículo até a meta, livre dos obstáculos, este será encontrado. Quando existem vários caminhos, pode-se ainda determinar o caminho ótimo, utilizando algum critério de otimalidade, tal como mínima distância, mínimo consumo de energia, mínima periculosidade, etc. O problema com este tipo de abordagem é que ele necessita de um modelo acurado do ambiente, e um modo de determinar a posição do veículo neste. Em problemas resolvidos por meio de simulação, onde se fazem algumas simplificações tais como considerar o ambiente em duas dimensões, somente objetos poligonais, localização precisa do veículo no ambiente, este tipo de abordagem gera resultados muito bons. Quando as simplificações são relaxadas, e tenta-se considerar modelos mais realistas do ambiente a utilização deste tipo de abordagem começa a tornar-se mais problemática.

Os modelos baseados em sensores tendem a ser mais robustos na transição da simulação para o ambiente real, pois os dados utilizados na tomadas de decisão estão muito próximos dos dados reais que são obtidos dos sensores. Alguns efeitos de sensores reais (tais como ruídos e não-linearidades) podem ser incluídos na simulação, tornando-a mais realista. Entretanto, como nesta abordagem não é considerado o modelo do ambiente como um todo, não se garante que o veículo consiga efetivamente se aproximar da meta. Particularmente, em algumas trajetórias mais sofisticadas, onde para atingir a meta o veículo precise inicialmente contrariar os princípios básicos, que são tentar evitar obstáculos e tentar se aproximar da meta, este tipo de abordagem encontra problemas.

Os modelos híbridos tentam se aproveitar das vantagens dos duas abordagens anteriores, de um modo que seus problemas possam ser superados. Entretanto, modelos deste tipo tendem a ser mais complicados.

Existe uma grande celeuma considerando-se as diferentes abordagens, que é o questionamento da necessidade de um modelo do ambiente para que o problema possa ser resolvido a contento. Brooks (1991) argumenta que um modelo do ambiente não é necessário, e que com um certo número de comportamentos adequados, sem modelo do ambiente, o problema pode ser resolvido a contento, somente por meio de comportamento reativo. Nesses casos, existe um comportamento emergente, que ocorre pela integração dos diversos tipos de comportamento, diante dos estímulos do ambiente. Esse comportamento emergente seria em princípio capaz de resolver o problema da navegação. Considerando-se abordagens como a de Verschure (1992), onde não apenas se tem diversos comportamentos, mas estes são ainda aprendidos a partir da interação com o ambiente, e mais de um comportamento podem ser responsáveis simultaneamente pela ação de controle (implementando uma espécie de fusão de comportamentos), surgem cada vez mais evidências de que isso possa ser realmente possível.

Essa polêmica pode ser resumida, basicamente, a dois modelos de tomada de decisão. No primeiro, chamado de modelo lógico, faz-se uma predição das possíveis alternativas, considerando-se não apenas a decisão imediata, mas uma sequência de decisões, que devem por fim satisfazer todos os objetivos. A satisfação dos objetivos é avaliada segundo um modelo do ambiente. No segundo modelo, chamado de modelo intuitivo, a partir de um conjunto de princípios básicos, avalia-se somente a decisão imediata, por meio de uma heurística, e opta-se por aquela que a partir desta heurística seja considerada mais adequada.

Observe que ambos os modelos de tomada de decisão podem ser encarados como buscas. O modelo lógico é uma busca no sentido pleno, onde é criada uma árvore de decisões possíveis, e a melhor sequência de decisões deve ser encontrada. O modelo intuitivo, também pode ser encarado como uma busca, embora em um sentido mais estrito. O modelo intuitivo corresponde a uma busca heurística em profundidade, onde apenas um passo na árvore de busca é utilizado. A busca se dá em tempo real, em função da heurística. Em sistemas mais simples, onde apenas com a heurística, pode-se chegar a uma sequência de decisões que satisfaça os objetivos, o modelo intuitivo pode ser utilizado com êxito. Entretanto, a despeito da qualidade desta heurística, existirão casos onde para se obter êxito, apenas a heurística não é suficiente. Nesses casos, é necessário considerar-se alternativas que violem a heurística, o que somente pode ser feito utilizando-se o modelo lógico. É aqui que entram as abordagens híbridas. Como as abordagens híbridas constroem interativamente um modelo do ambiente, esse pode ser utilizado para determinar uma heurística. Essa heurística será utilizada então não para determinar somente a próxima decisão, como no modelo intuitivo, mas para avaliar toda uma sequência de decisões. Sendo uma sequência de decisões considerada adequada, a primeira decisão desta será utilizada como ação de controle. Um exemplo de uma metodologia análoga a esta, aplicada no controle de sistemas a eventos discretos, pode ser encontrado em (Chung & Lafortune, 1992). Um exemplo de um sistema de controle inteligente seguindo essa filosofia é encontrado em (Gudwin & Gomide, 1994a).

A abordagem que é utilizada no exemplo de aplicação colocado neste capítulo é uma abordagem híbrida. Por meio de seu sistema de sensores, o veículo cria interativamente um modelo do ambiente. Esse modelo é utilizado para a determinação de uma heurística (colocada em termos de conhecimentos apraisivos), que é utilizada na determinação dos conhecimentos prescritivos que correspondem às ações de controle.

5.2 Descrição do Problema Exemplo

Para ilustrar a utilização do modelo de representação e processamento do conhecimento desenvolvido nesta tese, optou-se por elaborar um problema-exemplo que pudesse, na medida do possível, destacar os diferentes tipos de conhecimentos constantes do modelo. Sendo assim, a ênfase no problema exemplo é nos tipos de conhecimento que este abrange. Características de problemas reais da engenharia foram deliberadamente simplificados na elaboração deste problema. Essa simplificação é necessária, de modo a possiblitar a integração, em um mesmo problema, de diferentes aspectos de sistemas reais de navegação que seriam tratados individualmente no caso de um problema real. Com isso, os sensores e atuadores especificados no problema não tem exatamente um compromisso com sua

factiblidade de implementação, servindo apenas para ilustrar, de maneira bem simples, a característica de integração sensorial advinda do modelo.

O problema consiste basicamente, de um veículo autônomo, dotado de um aparato sensorial e motor, que trafega por um ambiente repleto de objetos com características diferenciadas. O veículo autônomo é movido a uma bateria recarregável, que é monitorada pelo seu sistema de sensoreamento. Alguns objetos do ambiente com o qual o veículo pode entrar em contato, podem carregar ou descarregar a bateria. Durante sua trajetória, o veículo deve atingir uma série de objetivos. Esses objetivos serão ponderados a cada instante, sendo que em um determinado instante alguns objetivos serão mais importantes que outros. Os objetivos são basicamente os seguintes:

- Exploração do Ambiente
- Manutenção da Integridade Física do Veículo
- Manutenção da Carga das Baterias em Níveis Operacionais

Para se atingir esses objetivos, o sistema necessita de dois níveis hierárquicos de controle. No nível mais alto, ou nível de decisão, ele determina metas a serem atingidas. Essas metas podem estar relacionadas diretamente à exploração do ambiente, ou seja, visando a determinação de um modelo do ambiente que o represente da melhor maneira possível. Para tal, as metas escolhidas devem conduzir o veículo para além de sua parte conhecida, permitindo a ampliação deste modelo. As metas podem também estar relacionadas à manutenção da carga das baterias. Durante sua trajetória, o veículo deve manter a carga de suas baterias sempre em níveis operacionais. Quando estiverem com os níveis de energia baixos, o veículo precisa entrar em contato com objetos do ambiente que fornecem energia ao veículo. Para tal, é necessário que o sistema descubra tais objetos (e suas propriedades), e saiba sua localização. Nesse caso, as metas escolhidas devem corresponder à posição dos objetos fornecedores de energia mais próximos. Nesse nível hierárquico de decisão, o principal problema é o escalonamento de prioridades, que determinem a cada instante o que é mais importante, explorar o ambiente ou cuidar da carga das baterias. No nível hierárquico mais baixo, ou controle direto, o sistema precisa conduzir o veículo de sua posição corrente até a meta desejada. Nesta trajetória, é necessário que a integridade física do veículo seja mantida, o que ocorre evitando-se que o mesmo colida com objetos sólidos do ambiente, e evitando-se que o veículo entre em contato com objetos que drenam energia. Neste trabalho, abordaremos somente o nível hierárquico mais baixo de controle, ou seja, dada uma meta, como conduzir o veículo de maneira segura até ela.

5.2.1 Descrição do Veículo

O veículo possui uma série de sensores e atuadores que são descritos a seguir. Basicamente, existem três tipos diferentes de sensores, os sensores de informação remota, os sensores de contato e o sensor de carga da bateria. Os atuadores determinam a posição do sensor de informação remota, o ângulo entre as rodas dianteiras e o eixo longitudinal do veículo e a velocidade nominal do veículo. Alguns atributos do estado do veículo são determinados indiretamente a partir dos valores dos sensores e atuadores.

Sensores de Informação Remota

Esses sensores consistem de uma simplificação de um mecanismo de visão. Basicamente é uma matriz de (8 × 8) sensores que cobre uma área retangular, à distância. Esses sensores conseguem tanto detectar as cores como prover informação de posição de objetos do ambiente. Entretanto, eles não conseguem determinar as características de cada objeto, somente sua localização. A área retangular dos sensores pode ser focalizada em qualquer posição, dentro de um raio máximo ao redor do veículo (vide figura 5.1).

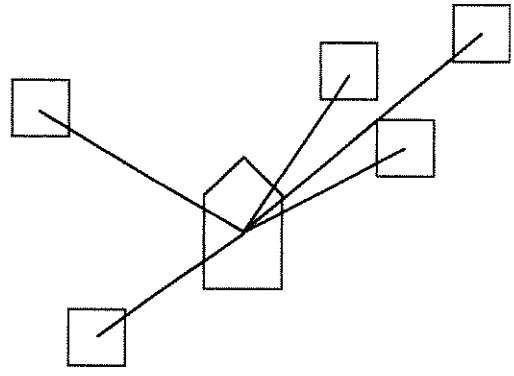


Figura 5.1 - Exemplo de Focalização dos Sensores Remotos

Sensores de Contato

Os sensores de contato consistem de 4 sensores localizados nas extremidades do veículo. Ao contrário dos sensores de informação remota, que podem ser movimentados, os sensores de contato são fixos. Esses sensores são capazes de perceber as características intrínsecas dos objetos do ambiente. Assim, os objetos podem ser categorizados quanto a agradáveis ou desagradáveis, dependendo dos estímulos adquiridos pelos sensores de contato. Esses sensores correspondem a uma simplificação de características do tipo prazer e desprazer, no caso do veículo, associadas a situações favoráveis ou desfavoráveis quanto aos objetivos do sistema.

Sensor de Carga das Baterias

Esse sensor mede a carga das baterias, em termos percentuais. Assim, se as baterias estiverem totalmente descarregadas, os sensores medirão 0. Se estiverem plenamente carregadas, os sensores medirão 100. As baterias têm uma descarga lenta e linear ao longo do tempo. Quando o veículo entra em contato com alguns objetos, entretanto, pode haver tanto uma descarga rápida como uma carga rápida. Esses fatores de carga e descarga rápida, podem ser detectados analisando-se o comportamento temporal do sensor de carga das baterias.

Atuadores de Posição dos Sensores Remotos

A posição dos sensores de Informação Remota, é determinada por meio destes atuadores. Basicamente, são dois, o de ângulo em função da frente do veículo (φ) e o de raio de ação (ρ) (vide figura 5.2).

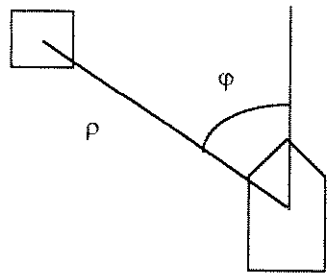


Figura 5.2 - Atuadores de Posição dos Sensores Remotos

Atuadores de Movimentação do Veículo

Os atuadores de movimentação do veículo são basicamente dois. O primeiro é o atuador de tração, que determina uma velocidade nominal v ao veículo. Se v é positivo, o veículo está se movimentando para frente, se negativo, para trás, e se igual a zero, o veículo está parado. O valor de v não corresponde exatamente à velocidade do veículo, mas à velocidade que este estará se o atrito for nulo. Para valores de atrito diferentes de zero, a velocidade real do veículo é menor que a velocidade nominal. O atrito é determinado a partir das propriedades dos objetos por onde o veículo está navegando (vide sub-seção 5.2.2). O outro atuador de movimentação, determina o ângulo θ da posição das rodas em relação ao eixo longitudinal do veículo. Esse atuador determinará a direção que o veículo deverá seguir, quando a velocidade nominal for não-nula. O ângulo das rodas poderá estar entre -45° a +45°, sendo que um ângulo 0 corresponde a seguir em frente.

5.2.2 Descrição do Ambiente

O ambiente corresponde a um retângulo cercado de paredes, com diversos objetos em seu interior. Os objetos podem ser caracterizados por suas propriedades físicas. As propriedades físicas dos objetos serão sua "cor", sua "dureza", seu "gosto" e sua "transferência de energia". Cada objeto pode possuir uma cor diferente, que é detectada pelos sensores de informação remota. A dureza do objeto mede a capacidade do veículo em trafegar sobre os objetos. Diferentes graus de dureza corresponderão a diferentes graus de mobilidade do veículo, facilitando ou dificultando seu trajeto por sobre si. O valor da dureza pode variar entre 0 e 1. Os objetos com dureza 1 serão chamados de intransponíveis, correspondendo a objetos sólidos, que barram o movimento do veículo em sua direção. O "gosto" de um objeto corresponde a um grau de desejo ou repulsa (prazer ou desprazer) proporcionado pelo contato com o objeto. Esse gosto é detectado pelos sensores de contato, sendo que seu valor pode variar de -1 a 1, onde -1 representa total desprazer, 1 representa total prazer e 0 representa indiferença. A transferência de energia de um objeto, corresponde à capacidade de fornecer ou absorver energia do veículo durante o contato. Valores positivos de transferência de energia correspondem a um aumento da carga da bateria do veículo. Valores negativos correspondem à absorção de energia, diminuindo a carga da bateria. Valores neutros (zero) correspondem a uma não modificação da carga (exceto o desgaste normal da mesma no veículo, em função do tempo). O valor da transferência de energia pode variar de -1 a 1. As propriedades de dureza, gosto e transferência de energia têm um vínculo direto com a cor. Assim, determinando-se a cor de um objeto, indiretamente se determinam sua dureza, gosto e transferência de energia.

5.2.3 *Modelo Dinâmico do Veículo*

O modelo dinâmico do veículo é descrito a seguir. As variáveis de interesse correspondem a suas coordenadas x,y,ψ, de acordo com a figura 5.3 a seguir, além do nível de energia das baterias, dado por f.

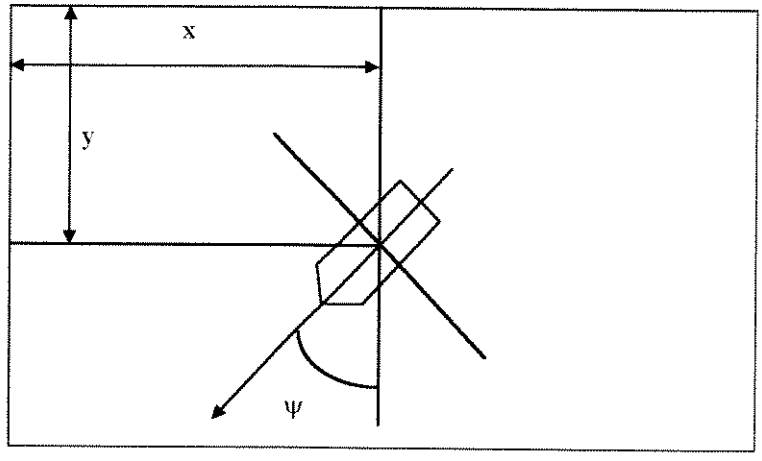


Figura 5.3 - Sistema de Coordenadas do Veículo

Outros parâmetros do sistema são a constante de atrito μ, a velocidade nominal v, a distância entre eixos D e o ângulo das rodas θ em relação ao eixo longitudinal. A distância entre eixos e o ângulo das rodas em relação ao eixo longitudinal do veículo podem ser vistos na figura 5.4 a seguir.

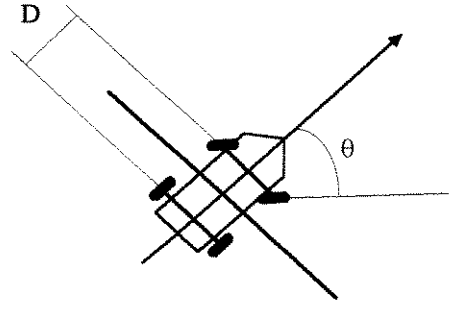


Figura 5.4 - Distância Entre Eixos e Ângulo das Rodas

Para modelar o sistema, dois sistemas de equações são utilizados. O primeiro, para quando θ for igual a zero. O segundo para quando θ for diferente de zero. Para θ igual a zero, tem-se:

$$x(k+1) = x(k) + (1-\mu).v.\cos\left(\psi + \pi/2\right)$$

$$y(k+1) = y(k) + (1-\mu).v.\mathrm{sen}\left(\psi + \pi/2\right)$$

$$\psi(k+1) = \psi(k)$$

$$f(k+1) = f(k) - 10^{-5} + 10^{-2}.\Delta f$$

Para θ diferente de zero, tem-se:

$$x(k+1) = x(k) + \frac{D.\,sen(\Delta\psi)}{sen(\theta)}.\cos\left(\psi + \pi/2\right) - \frac{D.\left(1-\cos(\Delta\psi)\right)}{sen(\theta)}.sen\left(\psi + \pi/2\right)$$

$$y(k+1) = y(k) + \frac{D.\,sen(\Delta\psi)}{sen(\theta)}.sen\left(\psi + \pi/2\right) + \frac{D.\left(1-\cos(\Delta\psi)\right)}{sen(\theta)}.\cos\left(\psi + \pi/2\right)$$

$$\psi(k+1) = \psi(k) + \Delta\psi$$

$$f(k+1) = f(k) - 10^{-5} + 10^{-2}.\,\Delta f$$

onde

$$\Delta\psi = \frac{(1-\mu).v.sen(\theta)}{D}$$

Analisando-se as equações, observa-se que além das variáveis, existem parâmetros e entradas do sistema. O parâmetro D é sempre constante. O parâmetro μ não é constante, sendo que sua determinação a cada instante segue o seguinte procedimento. Se o veículo estiver se movimentando para frente (velocidade nominal positiva), determina-se a dureza dos objetos que se encontram sobre os pontos esquerdo e direito frontal do veículo, tomando-se o máximo valor dentre eles. Isso garante que se a dureza de um deles for igual a 1, o veículo não se movimentará para frente (o que indica uma colisão com um objeto intransponível). Esse procedimento permite que, havendo uma colisão frontal, o veículo possa ser movimentado em marcha ré, saindo da situação de colisão. Se o veículo estiver se movimentando para trás, o procedimento é semelhante, só que considerando-se a dureza dos objetos que se encontram sobre os pontos esquerdo e direito traseiros do veículo, o que detecta também colisões em marcha a ré, e permite ainda que, havendo uma colisão traseira, o veículo possa ser movimentado para frente. Outra variável não-controlável do sistema é Δf, que corresponde à quantidade de energia fornecida ao veículo. Essa variável é determinada de modo semelhante ao parâmetro μ. Nesse caso, considera-se os quatro pontos do retângulo que envolve totalmente o veículo. São obtidos os valores da transferência de energia dos objetos que se encontram nesses 4 pontos, e é determinada uma média desses 4 valores. O valor da média corresponde a Δf. Por fim, v e θ correspondem à velocidade nominal e ao ângulo das rodas, as ações de controle motor do veículo.

5.3 Sistema de Controle Inteligente

Nesta seção, descreve-se o sistema de controle inteligente para o veículo, de modo a satisfazer os objetivos apresentados na seção anterior.

A entrada do controlador é a seguinte ênupla:

$(x,y,\psi,f,v,\theta,\rho,\varphi,c,s)$, onde

x,y,ψ correspondem à posição do veículo

f corresponde à carga das baterias do veículo (em %)

v corresponde à velocidade nominal no instante anterior

θ corresponde ao ângulo das rodas no instante anterior

ρ corresponde à distância do centro do veículo ao sensor remoto no instante anterior

φ corresponde ao ângulo entre o eixo longitudinal do veículo e o sensor remoto no instante anterior

c corresponde aos 4 sensores de contato (c_1 ,c_2 ,c_3 ,c_4), onde c_1 mede o "gosto" na extremidade traseira direita, c_2 na extremidade traseira esquerda, c_3 na extremidade dianteira esquerda e c_4 na extremidade dianteira direita.

s corresponde à uma matriz com 64 sensores ((s_{11} , ... , s_{18}),(s_{21} , ... , s_{28}), ... , (s_{81} , ... , s_{88})), onde s_{11} corresponde ao canto esquerdo superior e s_{88} corresponde ao canto direito inferior do quadrado de lado L correspondente ao campo visual do veículo. Cada sensor está igualmente espaçado de seus vizinhos.

A saída do controlador corresponde à seguinte ênupla:

(v,θ, ρ,φ), onde

v corresponde à velocidade nominal a ser aplicada.

θ corresponde ao ângulo na roda a ser aplicado

ρ corresponde à distância do sensor remoto, a ser aplicada

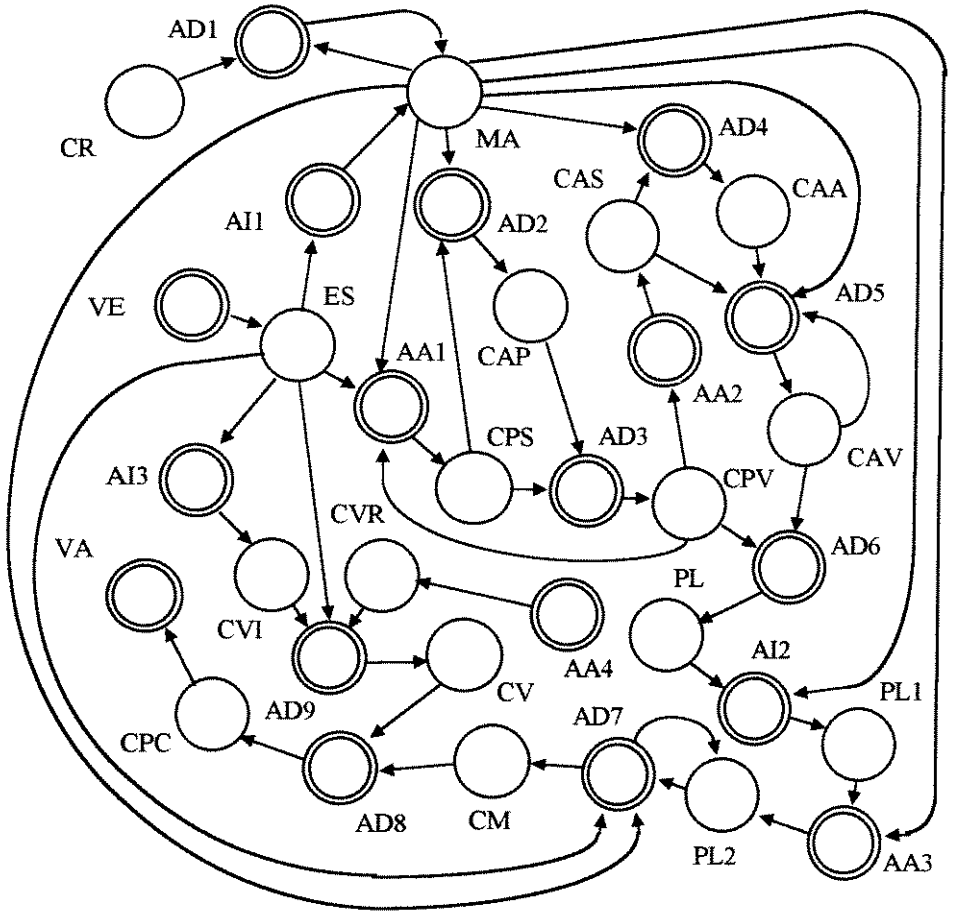
φ corresponde ao ângulo do sensor remoto, a ser aplicado

O sistema de controle é implementado pela rede de objetos apresentada na figura 5.5. A descrição detalhada de cada objeto desta rede será apresentada na próxima seção. A seguir, apresentaremos uma descrição global do funcionamento da rede.

O Vetor de Entrada (VE) corresponde a um argumento que gera as entradas sensoriais, enviando-as ao Espaço Sensorial (ES). Por meio do argumento indutivo 1 (AI1), o sistema induz modelos de objetos que são colocados no lugar dos modelos do ambiente (MA). Utilizando-se um conjunto de regras (CR) de associação, esses modelos são integrados pelo argumento dedutivo 1 (AD1), destruindo-se os modelos antigos, que são substituídos pelo modelo integrado. Estas etapas integram o módulo de percepção e formação de um modelo do ambiente para o sistema inteligente, que fica armazenado em (MA).

Utilizando-se os dados do espaço sensorial (ES) e do modelo do ambiente (MA), o argumento abdutivo 1 (AA1) gera um conjunto de pontos sugestão (CPS), correspondendo a possíveis movimentações para o veículo. O argumento dedutivo 2 (AD2), utilizando-se do modelo do ambiente (MA), avalia se os pontos em (CPS) são adequados para a movimentação do veículo, enviando essa avaliação para (CAP), ou seja o conhecimento apraisivo sobre os pontos. O argumento dedutivo 3 (AD3), utilizando-se o conhecimento apraisivo em (CAP) decide ou não enviar os pontos de (CPS) para o conjunto dos pontos válidos (CPV). Cada ponto em (CPV), por sua vez, passa a gerar novos pontos, por meio do argumento abdutivo 1 (AA1), em um ciclo semelhante ao anterior. Cada conjunto de dois pontos válidos, onde o primeiro deu origem ao segundo, gera um novo arco no conjunto dos arcos sugestão (CAS), por meio do argumento abdutivo 2. Esses arcos são parte de um plano que está sendo elaborado de modo a conduzir o veículo. Cada arco corresponde a uma possível movimentação de um ponto origem a um ponto destino. Essa movimentação é avaliada por meio do argumento dedutivo 4 (AD4), também em função do modelo do ambiente (MA), sendo enviada ao lugar dos conhecimentos apraisivos sobre arcos (CAA). Observe que apesar de um arco ter seus pontos origem e destino como válidos, o arco pode não o ser, pois o

movimento da origem ao destino, pode estar cruzando um objeto indesejável do ambiente. Sendo assim, utilizando-se a avaliação em (CAA), o argumento dedutivo 5 (AD5) resolve incluir ou não o arco em (CAS) no conjunto dos arcos válidos (CAV). A partir do conjunto dos pontos válidos (CPV) e do conjunto dos arcos válidos (CAV), o argumento dedutivo 6 gera um plano, que é enviado a (PL). O argumento indutivo 2 (AI2) tenta otimizar esse plano, substituindo dois arcos por um terceiro, que leve ao mesmo lugar e tenha uma boa avaliação, segundo o



Legenda			
VE	Vetor de Entrada	CAS	Conjunto de Arcos Sugestão
VA	Vetor de Atuadores	CAA	Conh. Apraisivo de Arcos
ES	Espaço Sensorial	CAV	Conjunto de Arcos Válidos
ADn	Argumento Dedutivo n	PLn	Plano n
AIn	Argumento Indutivo n	CM	Controle Motor
AAn	Argumento Abdutivo n	CVR	Controle Visual Randômico
MA	Modelo do Ambiente	CVI	Controle Visual Induzido
CPS	Conjunto de Pontos Sugestão	CV	Controle Visual
CAP	Conh. Apraisivo de Pontos	CPC	Conh. Prescritivo de Controle
CPV	Conjunto de Pontos Válidos	CR	Conjunto de Regras

Figura 5.5 - Rede de Objetos para o Controle de Veículo Autônomo

modelo do ambiente (MA). Esse plano otimizado é mandado então para (PL1). O argumento abdutivo 3 (AA3), gera então, para cada ponto dos arcos do plano em (PL1), um conjunto de novos pontos, próximos ao ponto original, tentando descobrir pontos que sejam ligeiramente mais interessantes que os pontos do plano anterior, de acordo com o modelo do ambiente (MA). Esse argumento dará origem, então, a um novo plano, que é enviado a (PL2). A partir deste plano, e utilizando dados do espaço sensorial (ES) e do modelo do ambiente (MA), o argumento dedutivo 7 (AD7) gera um objeto com as informações de controle motor, de modo que o veículo possa seguir o plano em (PL2). Estas etapam integram o módulo de controle motor do veículo.

Utilizando-se os dados do espaço sensorial (ES), o argumento indutivo 3 (AI3) gera uma sugestão para o controle visual, correspondendo a uma posição do controle visual próxima da anterior, compensando a movimentação do veículo e modificada para focalizar um objeto do sistema. Essa sugestão é enviada para o lugar do controle visual induzido (CVI). Paralelamente, o argumento abdutivo 4 (AA4) gera uma sugestão aleatória, que é enviada para o lugar do controle visual randômico (CVR). Baseado na informação do espaço sensorial (ES), o argumento dedutivo 9 (AD9) decide se o controle visual está focalizando um objeto do ambiente ou está procurando um novo objeto, optando pela sugestão de controle em (CVI) ou em (CVR). Essa opção é enviada então ao lugar de controle visual (CV). Essas etapas integram o módulo de controle visual do veículo.

Por fim, o controle visual (CV) e o control motor são integrados, por meio do argumento dedutivo 8 (AD8), dando origem ao conhecimento prescritivo de controle (CPC). Este é absorvido pelo vetor de atuadores (VA), encerrando o ciclo de controle inteligente.

5.4 Etapas do Sistema de Controle

Nesta seção, apresentamos uma descrição das etapas que integram o sistema de controle inteligente introduzido na seção anterior. De modo a facilitar o entendimento das sucessivas etapas, estas serão representadas por meio de uma linguagem orientada a objeto. Esta representação tem o caráter duplo de facilitar o entendimento do sistema e ilustrar como uma rede de objetos pode ser diretamente traduzida e processada em termos de uma linguagem de programação. A linguagem orientada a objeto escolhida é o C++.

5.4.1 Módulo da Interface de Entrada

Este módulo corresponde à seção da rede de objetos apresentada na figura 5.6 a seguir:

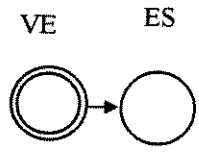


Figura 5.6 - Módulo da Interface de Entrada

A classe associada a ES, tES é a seguinte:

```
class tES
{double x;
 double y;
 double pitch;
 double ax,
 double ay;
 double apitch;
 double fuel;
 double vel;
 double dpitch;
 double scangle;
 double sclength
 int s[8][8];
 double c[4];
 double mx;
 double my;
};
```

onde têm-se a seguinte correlação:

x	x	apitch	ψ(t-1)	sclength	ρ
y	y	fuel	f	s	s
pitch	ψ	vel	v	c	c
ax	x(t-1)	dpitch	θ	mx	meta x
ay	y(t-1)	scangle	φ	my	meta y

Observe que além das variáveis previstas na entrada do sistema inteligente, a classe tES dispõe ainda de uma memória dos valores de x,y e ψ, para o instante de tempo anterior e as coordenadas da meta a ser atingida.

A classe associada a VE, tVE é a seguinte:

```
class tVE
{public:
     tES Gera(void)
     {tES novo;
      novo = CollectDataSensor();
      return(novo);
     }
};
```

A classe tVE é uma classe com uma única função, que retorna um objeto do tipo tES, preenchido com os valores atuais dos sensores. Sendo assim, a cada instante de tempo, um objeto de tipo tVE que se encontra em VE gera um objeto do tipo tES, que é colocado em ES.

5.4.2 *Módulo de Percepção e Modelagem do Ambiente*

Este módulo corresponde à seção da rede de objetos apresentada na figura 5.7 a seguir:

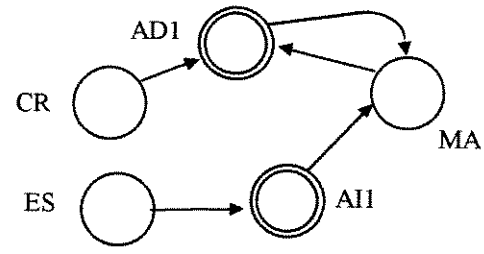


Figura 5.7 - Módulo de Percepção e Modelagem do Ambiente

Neste módulo, é gerado o modelo do ambiente, que fica armazenado em MA e é utilizado intensamente pelos outros módulos do sistema.

A classe associada a MA, tMA é dada a seguir:

```
class tMA
{int x0,y0,x1,y1;
 int color;
 double eval;
};
```

A classe tMA modela os atributos do objeto típico encontrado no ambiente. Nesta classe, x0, y0, x1 e y1 correspondem às coordenadas geométricas de um quadrado, que determinam a posição e tamanho do objeto no ambiente. O campo eval modela a característica "gosto" do objeto, sendo igual a -1 se o objeto é repulsivo, 1 se é atrativo e 0 se é indiferente. As gradações intermediárias entre esses valores correspondem a graus maiores ou menores de atração/repulsividade. O valor do campo eval pode ser conhecido quando os sensores de contato entram efetivamente em contato com o objeto do ambiente, ou quando o sistema já associou uma cor ao gosto do objeto, sendo determinado portanto, por indução. O campo color modela a cor do objeto, que é enxergada por meio dos sensores visuais.

A classe associada a AI1, tAI1 é dada a seguir:

```
class tAI1
{public:
 void FindPatterns(tES es, place MA);
};
```

A classe tAI1 gera objetos com somente uma função, a função FindPatterns, que tem como entrada um objeto do tipo tES e um lugar (MA), onde ela deposita os objetos criados. A função FindPatterns é razoavelmente sofisticada. Ela observa os sensores visuais codificados em *es* e detecta a borda de todos os retângulos que sejam identificáveis a partir da matriz de sensores. Seu funcionamento pode ser ilustrado na figura 5.8 a seguir:

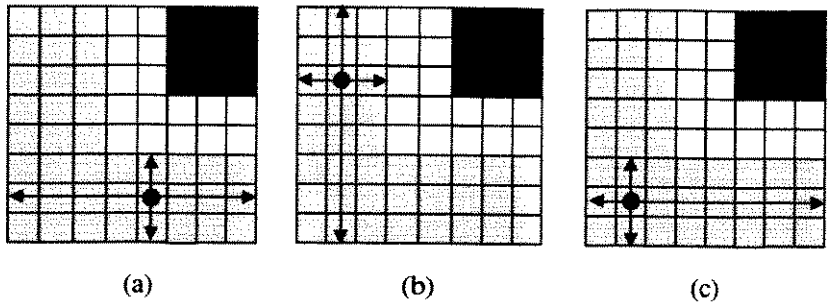


Figura 5.8 - Exemplo de Reconhecimento de Padrão

Para cada ponto da matriz de pontos, é feita uma varredura, primeiro no sentido horizontal, depois no sentido vertical, buscando-se pontos que sejam da mesma cor que o ponto corrente. Nos casos (a) e (b), essa busca é simples. O caso (c) ilustra uma necessidade do algoritmo: a busca no sentido vertical deve ser feita para cada coluna rastreada no sentido horizontal, tomando-se a menor distância encontrada. Se isso não for efetuado, o ponto indicado em (c) retorna o quadrado em toda a matriz, uma vez que o ponto rastreado poderia dar margem a essa interpretação, se a busca fosse feita apenas em sua coluna. Para o caso apresentado, o algoritmo encontra três quadrados, dois cinza (identificados em (a) e (b)) e o terceiro preto.

Os quadrados encontrados pela função geram então objetos do tipo tMA, que são colocados no lugar MA. Observe que o número máximo de quadrados encontrado pela função corresponde a 64 (um para cada ponto), ou seja, quando cada cor for diferente da do sensor vizinho.

Após a geração destes objetos elementares em MA, os objetos em CR e AD1 são responsáveis por sua integração, formando um modelo mais consistente. O lugar do conjunto de regras (CR) contém basicamente três objetos, correspondendo a três regras de integração, que são ilustradas na figura 5.9 a seguir:

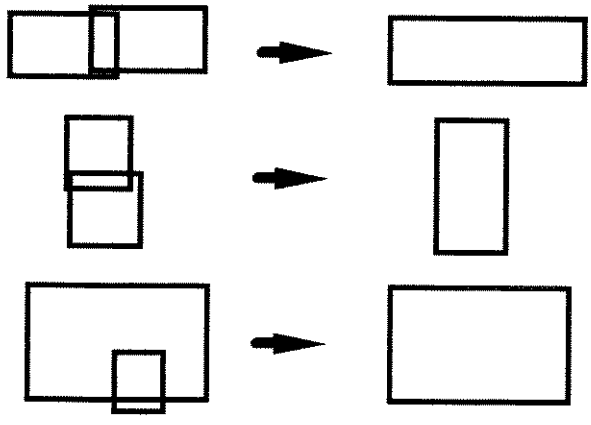


Figura 5.9 - Regras de Integração de Objetos

As regras utilizadas são muito simples. A primeira diz que se dois objetos estão alinhados horizontalmente, eles na verdade correspondem a um único objeto com a dimensão correspondente à sobreposição dos outros dois. A segunda regra diz que se dois objetos estão alinhados verticalmente, então eles correspondem a um

único objeto. A terceira regra diz que se um determinado objeto está integralmente contido em outro objeto, ele na verdade é apenas uma parte desse, sendo descartado. Observe que nas três regras, os objetos não precisam estar perfeitamente alinhados, havendo certa tolerância, tanto no alinhamento quanto na inclusão. Essa tolerância é necessária, pois os sensores visuais não são exatos. Cada ponto do sensor visual não é um ponto real, mas tem uma certa dimensão física, sendo que o resultado sensorial corresponde à integração de todos os pontos reais que se encontram dentro desta dimensão física. Com isso, existe um erro intrínsico na leitura desses sensores. Esse erro é a tolerância admitida na integração de objetos. Observe que o objeto resultado corresponde sempre ao quadrado interno, no caso do alinhamento, devido a um efeito indesejado que ocorre na detecção de quinas, se for utilizado o quadrado externo. Para a regra da inclusão, também não se expande o quadrado maior, mesmo que o quadrado interno extrapole seus limites, para evitar o aparecimento do mesmo efeito indesejado.

A classe associada a CR é tCR:

```
class tCR
{int verb;
 int rel1;
 int rel2;
 int function;
};
```

Os atributos dos objetos do tipo tCR na verdade correspondem a proposições condicionais, onde os termos estão codificados por meio de índices. Esses índices devem ser reconhecidos pelo argumento em AD1. A semântica corresponde a:

SE verb(rel1,rel2) ENTÃO function(rel1,rel2)

As regras utilizadas são as seguintes:
SE halign(rel1,rel2) ENTÃO hadd(rel1,rel2)
SE valign(rel1,rel2) ENTÃO vadd(rel1,rel2)
SE included(rel1,rel2) ENTÃO return(rel1)

A classe associada a AD1, tAD1 é dada por:

```
class tAD1
{public:
  tMA MergeObject(tMA o1, tMA o2, tCR r);
};
```

Um objeto do tipo tAD1 toma dois objetos de MA, o1 e o2, mais uma regra de CR, r, consome os dois objetos em MA e retorna um novo objeto, que é colocado em MA. Na rede de objetos, essa operação é feita quantas vezes for necessário, até que nenhuma das três regras esteja habilitada.

Com isso, implementa-se o módulo de percepção e modelagem do ambiente. O objeto do tipo tES que se encontra em ES é utilizado pelo objeto em AI1 para gerar objetos em MA, correspondendo a modelos para os objetos do ambiente que são percebidos pelo sistema. Como o sistema é capaz de receber apenas informação parcial do ambiente, esses modelos necessitam ser integrados, de modo que através do conhecimento de múltiplas facetas do objeto real, se chegue a um modelo mais próximo deste. Essa tarefa é desempenhada pelas regras em CR e pelo argumento

dedutivo em AD1. Todo esse ciclo é realizado sempre que novas informações chegam por meio do argumento em AI1.

5.4.3 Módulo de Geração de Pontos e Arcos

Esse módulo, na verdade, é um sub-módulo do controle motor. Ele é descrito em separado devido a sua complexidade. Ele corresponde à secção da rede apresentada na figura 5.10 a seguir:

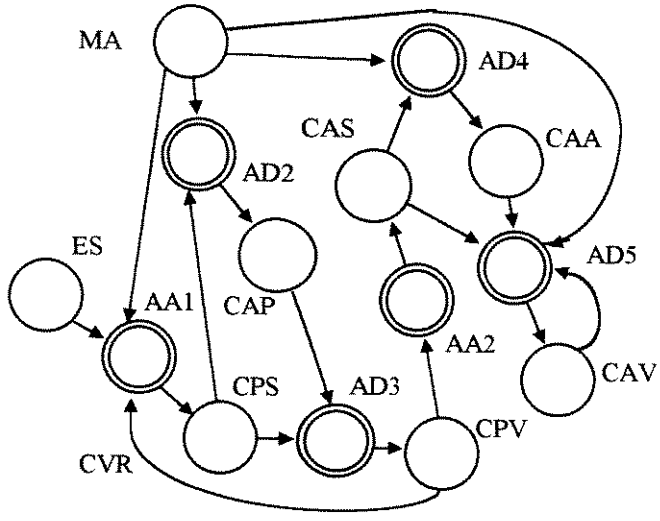


Figura 5.10 - Módulo de Geração de Pontos e Arcos

As classes associadas a CPS e a CPV são equivalentes, sendo chamadas coletivamente de tP, ou classe dos pontos:

```
class tP
{double x;
 double y;
 double p;
 tP *pp;
 int type;
};
```

Os atributos dos objetos dessa classe correspondem basicamente às coordenadas do ponto (x,y,p), onde p é um ângulo, correspondente a um possível movimento do veículo, caso se encontre no ponto (x,y). O atributo pp correspondem ao endereço do ponto-pai, ou seja do ponto que foi utilizado como base para sua geração. O argumento abdutivo em AA1 gera pontos-sugestão, que são enviados a CPS. Essa geração pode se dar de quatro maneiras. A maneira com que um ponto é gerado é identificada pelo atributo type. Na primeira delas, toma-se as coordenadas da meta, que são utilizadas para se gerar o ponto meta. Seu atributo type é 0 e pp é igual a NULL (não é considerado). Na segunda maneira, a posição corrente do veículo é utilizada para gerar um ponto inicial. O atributo pp é igual a NULL e o atributo type é 1. Na terceira maneira, o argumento parte de um ponto válido, colocado em CPV, para gerar um novo ponto. Esse ponto é utilizado para gerar as novas coordenadas x,y,p tem seu endereço armazenado em pp. O atributo type é 2. A quarta maneira utiliza os pontos referentes ao modelo dos objetos do

ambiente. Seu atributo pp é NULL e type é 3. A classe associada a AA1, portanto, têm quatro funções:

```
class tAA1
{public:
 tP GenerateNewPoint0(tES src);
 tP GenerateNewPoint1(tES src);
 tP GenerateNewPoint2(tP src, tMA *ma);
 tP GenerateNewPoint3(tMA *ma);
};
```

A função GenerateNewPoint0 é trivial. Simplesmente extrai de src as coordenadas da meta, atribui NULL a pp e 0 a type.

A função GenerateNewPoint1 também é trivial. Basicamente ela extrai de src os parâmetros x,y e pitch e os coloca em x, y e p. O atributo pp é NULL e type é 1. Isso completa a definição do novo ponto. Esse ponto é enviado então a CPS.

A função GenerateNewPoint2 é um pouco mais sofisticada. A partir do ponto base, indicado em src, o argumento gera aleatoriamente um ângulo entre 0 e 2π. Esse ângulo corresponde a um eixo que será utilizado para a geração do novo ponto, conforme a figura 5.11. A função gera uma reta, na direção do ângulo, e verifica, para cada objeto do modelo do ambiente que seja classificado como repulsivo, se existe uma colisão. Caso a reta não colida com nenhum objeto do ambiente, os limites externos do ambiente são então considerados. Uma vez que se identifique a colisão mais próxima, o ponto que fica a 50% da distância do ponto de colisão é escolhido, conforme pode ser visualizado na figura 5.11.

Somente são considerados os objetos do ambiente que são classificados como repulsivos. Ou seja, somente os objetos do ambiente que tiverem um atributo eval menor que 0 são considerados. Os objetos com atributo eval iguais ou superiores a 0 não são considerados, sendo ignorados pela função.

Por fim, a função determina a coordenada do ponto, identifica o ponto-pai deste (o ponto src), atribuindo-o a pp, e determina um valor 2 para type. Gera então um novo objeto do tipo tP que é enviado a CPS.

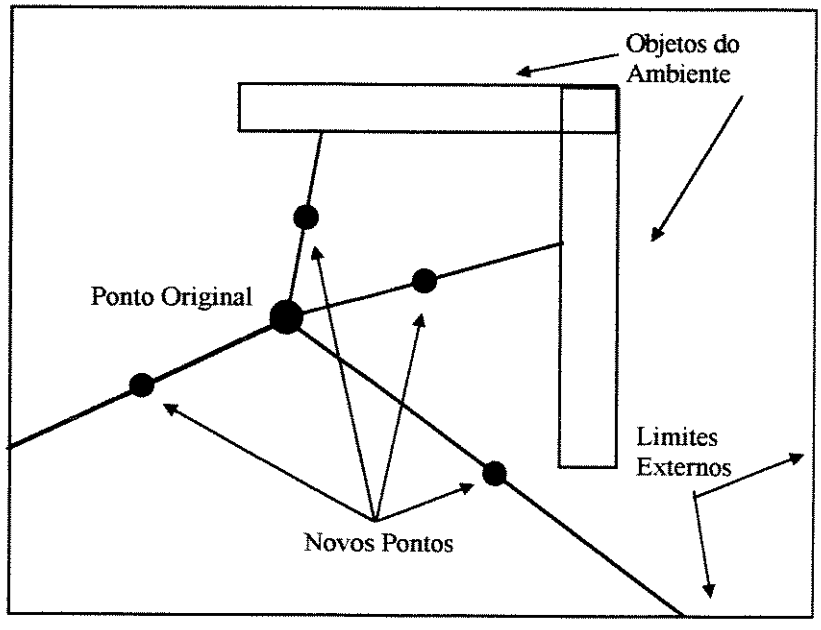


Figura 5.11 - Geração de Novos Pontos

A função GenerateNewPoint3 toma cada objeto em MA, e acrescenta uma borda de segurança sobre os vértices dos quadrados, conforme é visto na figura 5.12. a seguir:

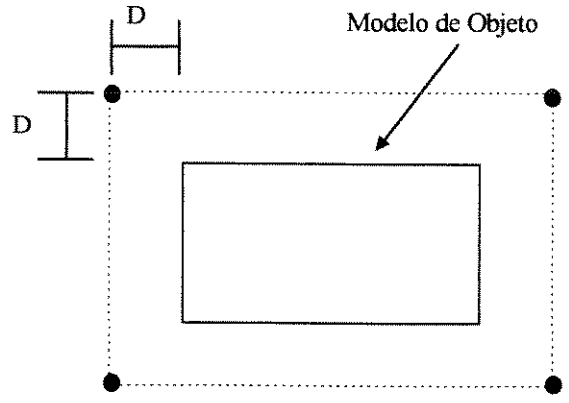


Figura 5.12 - Geração de Novos Pontos segundo Modelo do Ambiente

Essa borda, normalmente, corresponde à largura do veículo, de modo que este possa trafegar com segurança. Esse valor é, entretanto, um parâmetro do sistema, podendo inclusive ser alvo de aprendizado.

A função atribui ainda um valor NULL para pp e 3 para type.

O lugar CAP, correspondendo ao lugar dos conhecimentos apraisivos sobre pontos, tem a classe tEval associada:

```
class tEval
{double eval;
};
```

A classe associada a AD2 é chamada de tEvalPoint, pois sua função básica é avaliar se um determinado ponto é adequado ou não para a movimentação do veículo:

```
class tEvalPoint
{public:
   tEval EvalPoint(tP point, tMA *ma);
};
```

A função EvalPoint toma o ponto point e avalia se este é adequado, gerando um objeto do tipo tEval correspondendo ao resultado da avaliação. Para gerar essa avaliação, a função utiliza os valores dos "gostos" associados aos objetos do ambiente, e as distâncias entre o ponto determinado e os objetos do ambiente. A seguinte fórmula é utilizada:

$$\text{Eval} = \min_i \left(g_i \cdot e^{(-0.4 \cdot \text{dto}(i))} \right)$$

Nesta fórmula, g_i corresponde ao gosto do objeto i, constante do modelo, e dto(i) corresponde à distância até o objeto i.

A distância até o objeto i é calculada conforme a figura 5.13 a seguir.

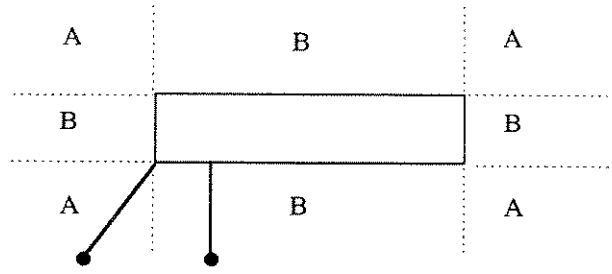


Figura 5.13 - Cálculo da Distância até o Objeto

Se o ponto estiver em algum dos quadrantes do tipo A, então a distância até o objeto corresponde à distância até o vértice mais próximo do quadrado. Caso o ponto esteja em um quadrante do tipo B, então é calculada a distância até o lado mais próximo do quadrado.

A classe associada a AD3, tAD3 é dada por:

```
class tAD3
{double threshold;
 public:
 tP TestPoint(tP point, tEval eval);
};
```

A função TestPoint é muito simples. Ele verifica se a avaliação do ponto é superior a um limiar (o campo threshold), e em caso positivo, libera o ponto para o conjunto dos pontos válidos (CPV). Observe que tanto para a geração da avaliação como para a aceitação do ponto como um ponto válido, foram utilizados parâmetros obtidos de modo empírico. Em um sistema mais sofisticado, estes parâmetros poderiam ainda ser objeto de aprendizado.

A classe associada a CAS (Conjunto dos Arcos-Sugestão) é tA:

```
class tA
{tP *orig;
 tP *dest;
};
```

A classe associada a AA2 é tAA2:

```
class tAA2
{public:
   GenerateNewArc1(tP *p);
   GenerateNewArc2(tP *p1, tP *p2);
};
```

Os objetos dessa classe têm duas funções. A função GenerateNewArc1 tem como parâmetro um ponto do tipo 2, que identifica em si próprio seu ponto pai. O arco é então gerado tomando-se como orig o endereço do ponto pai e como dest o endereço do próprio ponto. A função GenerateNewArc2 pode ter como parâmetro p1, um ponto dos tipos 1, 2 ou 3, e como parâmetro p2, um ponto dos tipos 0 ou 3. O ponto gerado tem como orig o parâmetro p1 e como dest o parâmetro p2.

A classe associada a CAA é tEval;

A classe associada a AD4 é tEvalArc:

```
class tEvalArc
{public:
   tEval EvalArc(tA arc, tMA *ma);
};
```

A função EvalArc analisa o arco arc, e verifica se os pontos origem são adequados, de modo semelhante ao de EvalPoint. Em seguida, verifica se a reta que une o ponto orig ao ponto dest cruza com algum objeto do modelo. Caso haja o cruzamento, a função considera o gosto do objeto sendo cruzado. Esse procedimento é ilustrado na figura 5.14 a seguir:

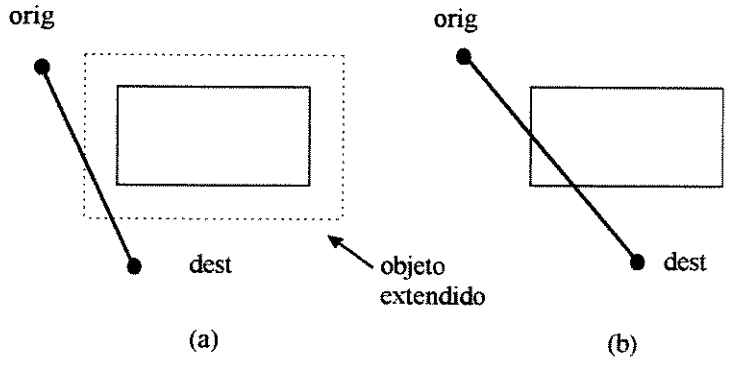


Figura 5.14 - Avaliação de Arco

Quando o ponto destino é do tipo 2, é utilizado o modelo do objeto extendido, conforme o caso (a) da figura. Quando o ponto destino é do tipo 3 ou é a meta, utiliza-se o caso (b) da figura.

O objeto em CAA (conhecimento apraisivo sobre arcos) comporta então uma avaliação de um arco em CAS. Essa avaliação mede o quanto é adequada a movimentação do veículo de orig para dest.

A classe associada a AD5 é tAD5:

```
class tAD5
{double threshold;
 public:
 tA TestArc(tA arc, tEval eval, tA *av, tMA *ma);
};
```

A função TestArc é mais sofisticada que sua dual TestPoint. A função TestArc, além de verificar se a avaliação do arco tem seu valor acima de um limiar (dado por threshold), verfica uma série de outras situações. A partir da lista de arcos válidos av, obtida de CAV, o arco arc é testado, de modo a ver se ele não cruza com nenhum outro arco válido (que já se encontra em CAV), se a distância de seus pontos está maior que uma distância mínima dos pontos dos arcos em CAV e menor que uma distância mínima dos objetos do modelo do ambiente considerados repugnantes. Os arcos que passam por todas estas condições são então enviados a CAV.

Com isso conclui-se a descrição do módulo de Geração de Pontos e Arcos.

5.4.4 Módulo de Geração e Otimização de Planos

O módulo de geração e otimização de planos toma os pontos e arcos válidos, colocados em CPV e CAV, e gera um plano que é otimizado e posteriormente transformado em uma ação de controle. Este módulo, junto com o módulo de geração de pontos e arcos, integram o módulo de controle motor. A secção da rede correspondente ao módulo de Geração e Otimização de Planos é mostrada na figura 5.15 a seguir:

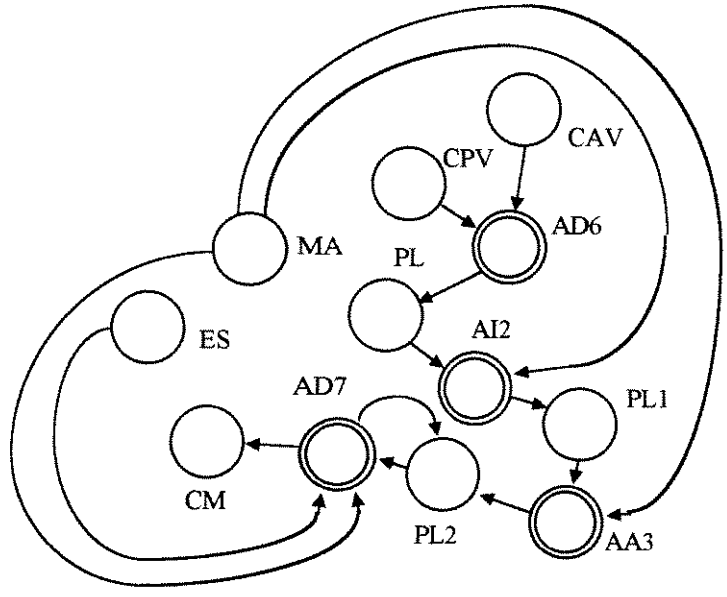


Figura 5.15 - Módulo de Geração e Otimização de Planos

A classe associada a PL, PL1 e PL2 é tPL:

```
class tPL
{tP *pl;
 int npl;
};
```

A classe associada a AD6 é tAD6:

```
class tAD6
{public:
   tPL GeneratePlan(tP *p, tA *a);
};
```

A função GeneratePlan basicamente toma a lista de arcos em CAV, localiza os respectivos pontos em CPV, e gera uma lista de pontos que corresponde a sequência de pontos que o veículo deve seguir para chegar à meta. Para gerar essa lista, ela procura o arco que tem como dest a meta, verifica qual é orig, procura outro arco que tem esse orig como dest, verifica seu orig e assim por diante, até chegar no ponto correspondente ao estado atual do veículo.

O argumento indutivo em AI2 toma esse plano, e faz uma primeira otimização, eliminando pontos que são redundantes, conforme a figura 5.16 ilustra:

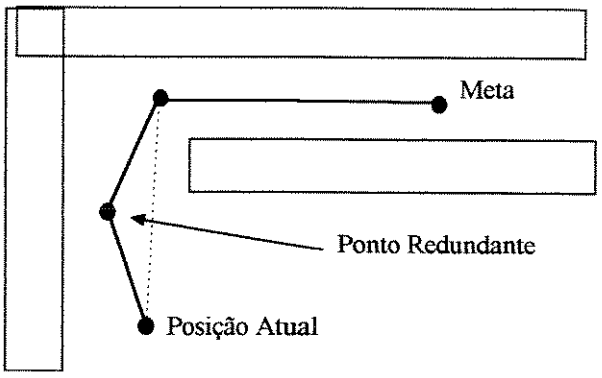


Figura 5.16 - Eliminação de Pontos Redundantes

Os pontos redundantes são aqueles que, se eliminados, continuam permitindo uma movimentação segura do veículo, conforme indicado por uma linha tracejada na figura.

A classe associada a AI2 é tAI2:

```
class tAI2
{public:
  tPL OptimizePlan1(tPL pl, tMA *ma);
};
```

A função toma o plano colocado em PL e gera um novo plano, sem os pontos redundantes, que é colocado em PL1.

O argumento abdutivo AA3 toma o plano em PL1, e o otimiza, alterando ligeiramente os pontos do mesmo, conforme ilustrado na figura 5.17 a seguir:

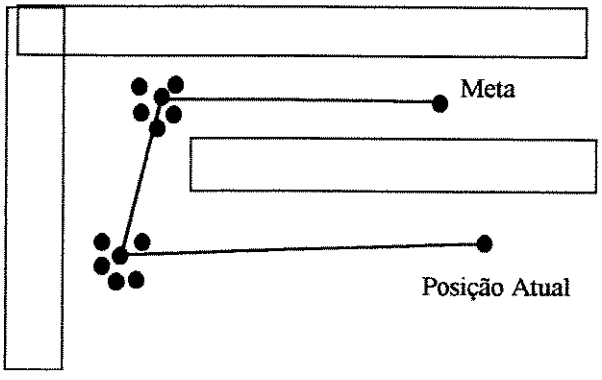


Figura 5.17 - Otimização dos Pontos do Plano

O argumento gera aleatoriamente, para cada ponto do plano diferente da posição atual e da meta, uma série de outros pontos, dentro de um raio máximo limitado. O plano é testado então, por meio do modelo do ambiente, para verificar se é possível melhorar o plano original. Caso um dos pontos abduzidos seja melhor que o original, este é então substituído.

A classe associada a AA3 é tAA3:

```
class tAA3
{public:
   tPL OptimizePlan2(tPL pl, tMA *ma);
};
```

A função OptimizePlan2 toma o plano em PL1, e baseado no modelo ma, gera um novo plano, que é colocado em PL2.

A classe associada a CM é tCM:

```
class tCM
{double vel;
 double dpitch;
};
```

O objeto em CM corresponde à ordem de controle motor que deve ser enviada posteriormente aos atuadores. É composta basicamente pela velocidade o ângulo da roda do veículo. O objeto em CM é gerado pelo argumento dedutivo em AD7.

A classe associada a AD7 é tAD7:

```
class tAD7
{public:
  tCM MotorControl(tPL *pl, tES es, tMA *ma);
  tPL AtualizePlan(tPL pl);
};
```

A função MotorControl é razoavelmente sofisticada. Normalmente, para atingir um determinado ponto, a ação de controle motor a ser tomada é muito simples. Basta determinar as coordenadas polares do ponto a ser atingido, em função da posição do veículo, determinar uma velocidade positiva e o ângulo da roda igual a menos o ângulo encontrado (lembrando que o ângulo da roda é determinado no sentido horário, vide figura 5.4). Entretanto, em situações como a da figura 5.18 a seguir, é necessário que o veículo manobre para atingir a meta.

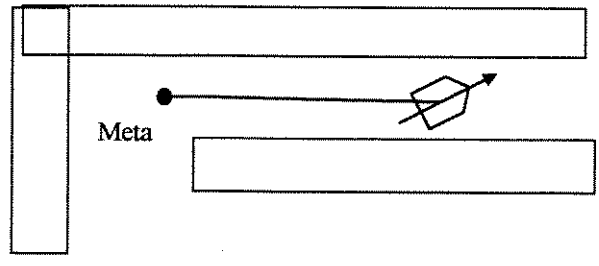


Figura 5.18 - Necessidade de Manobra

Para verificar a necessidade de manobra, o argumento calcula a trajetória ótima para se atingir o ponto-meta. Essa trajetória ótima corresponde a colocar o ângulo da roda em seu máximo (para direita ou para a esquerda, dependendo da posição da meta), e deixar o veículo descrever uma circunferência, até que este direcione-se de frente para a meta. Essa posição está ilustrada na figura 5.19 a seguir:

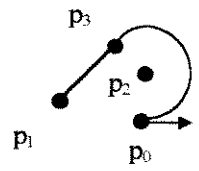


Figura 5.19 - Curva Mínima do Veículo

Na figura o veículo se encontra inicialmente no ponto p_0, trafegando na direção indicada, e deseja chegar a p_1 . O raio da circunferência que descreve a trajetória do veículo, dado o ângulo das rodas é dada por:

$$R = \frac{D}{2\cos\left(\frac{\pi-\theta}{2}\right)}$$

Supondo-se um ângulo máximo de $\pi/4$, têm-se um raio mínimo de:

$$R_{min} = \frac{D}{2\cos\left(\frac{3\pi}{8}\right)}$$

O ponto p_2 é calculado tomando-se uma distância igual a R_{min} a um ângulo de $\pi/2$ a partir da direção indicada. O ponto p_3 é calculado por:

$$p_3^x = p_2^x + R_{min}\cos\xi$$
$$p_3^y = p_2^y + R_{min}\,\mathrm{sen}\,\xi$$

$$\text{onde } \xi = \mathrm{arctg}\left(\frac{p_1^y - p_2^y}{p_1^x - p_2^x}\right) + \frac{R_{min}}{\sqrt{\left(p_1^y - p_2^y\right)^2 + \left(p_1^x - p_2^x\right)^2}} - \frac{\pi}{2}$$

Uma vez calculado o ponto p_3, o sistema supõe que o veículo esteja sobre este ponto, com um *pitch* correspondente à direção em linha reta até p_1 , calcula a posição de cada um dos 4 sensores de contato do veículo, nessa posição, e determina, por meio do modelo dos objetos do ambiente, qual seria o "gosto" sentido pelo veículo nessa posição. Se o mínimo gosto dos 4 sensores for aceitável (alguma coisa como -0.9), isso significa que o veículo não necessita manobrar. Caso contrário, ele entra em estado de manobra. Nesse estado, o veículo verifica qual dos sensores de contato que indica um gosto "pior", e a partir deste, determina uma sequência de movimentos para direita e para a esquerda, alternando marcha a ré e marcha a frente, até que o veículo saia da condição de manobra (ou seja, que o gosto em p_3 seja aceitável). A saída da função MotorControl gera um objeto que é enviado a MC.

A função AtualizePlan, a partir da posição corrente do veículo, verifica se o primeiro ponto do plano foi atingido, retirando-o da lista caso isso aconteça. O plano corrigido é enviado de novo a PL2.

Com isso, conclui-se o módulo de Geração e Otimização de Planos, e por sua vez o Módulo de Controle Motor, que é formado conjuntamente pelo módulo de Geração de Pontos e Arcos e pelo módulo de Geração e Otimização de Planos.

5.4.5 *Módulo de Controle Visual*

O módulo de controle visual é o responsável pela movimentação do sensor visual, que é utilizado pelo módulo de percepção e formação do modelo de ambiente. Apesar de ser um controle em separado, sua ação é fundamental para o perfeito funcionamento do módulo de percepção e geração do modelo do ambiente, sendo imprescindível uma coordenação entre eles de modo que a percepção ocorra a contento.

A secção da rede responsável pelo controle visual está indicada na figura 5.20 a seguir:

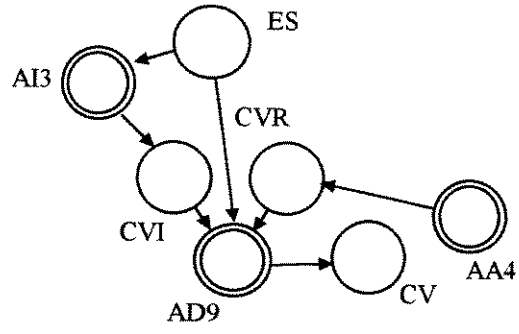


Figura 5.20 - Módulo de Controle Visual

A classe associada tanto a CVI, quanto a CVR e a CV é tCV:

```
class tCV
{double scangle;
 double sclength;
};
```

A classe associada a AI3 é tAI3:

```
class tAI3
{public:
   GenerateInducedVisualControl(tES es);
};
```

A função GenerateInducedVisualControl é razoavelmente sofisticada. Baseada na imagem do sensor visual, ela calcula o centro de massa da imagem, e sugere uma movimentação equivalente a este centro de massa. Esse procedimento está ilustrado na figura 5.21 a seguir:

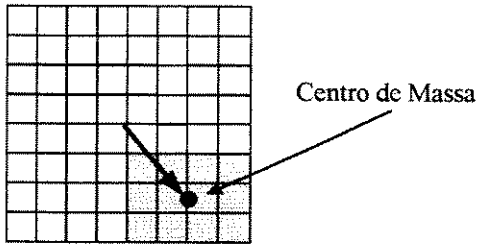


Figura 5.21 - Cálculo do Centro de Massa

Além disso, a função faz uma compensação correspondente à movimentação do veículo. Essa compensação está ilustrada na figura 5.22 a seguir:

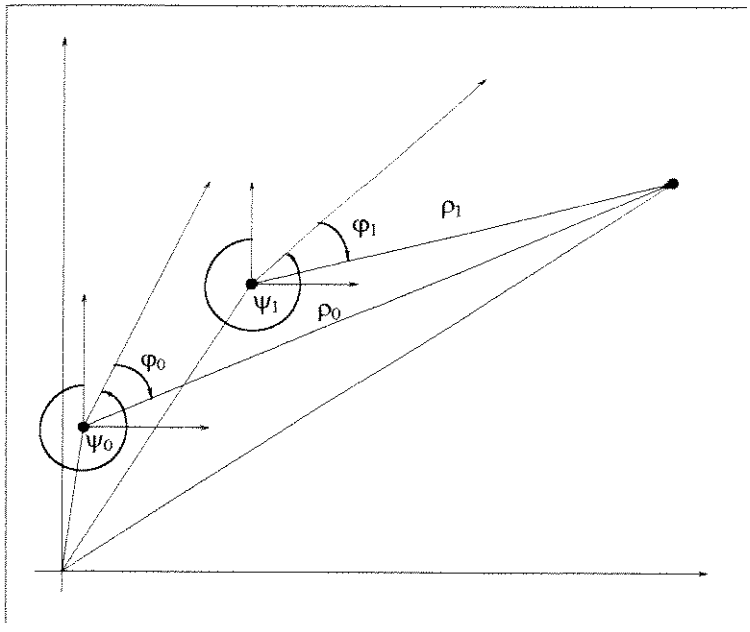


Figura 5.22 - Compensação do Movimento do Veículo

Basicamente o que se deseja é que, estando o veículo na posição (x_0 , y_0 , ψ_0), e tendo coordenadas de controle visual (ρ_0 , φ_0), quando o veículo muda para a posição (x_1 , y_1 , ψ_1), as coordenadas de controle visual (ρ_1 , φ_1), sejam tais que o foco do controle visual aponte para a mesma posição anterior. As equações para (ρ_1 , φ_1), são as seguintes:

$$\varphi_1 = \psi_1 - \operatorname{arctg}\left(\frac{y_0 - y_1 + \rho_0 . \operatorname{sen}(\psi_0 - \varphi_0 - 3\pi/2)}{x_0 - x_1 + \rho_0 . \cos(\psi_0 - \varphi_0 - 3\pi/2)}\right) - \frac{3\pi}{2}$$

$$\rho_1 = \sqrt{\left(y_0 - y_1 + \rho_0 . \operatorname{sen}(\psi_0 - \varphi_0 - 3\pi/2)\right)^2 + \left(x_0 - x_1 + \rho_0 . \cos(\psi_0 - \varphi_0 - 3\pi/2)\right)^2}$$

A saída da função é obtida somando-se o centro de área e a compensação pela movimentação do veículo. O resultado é enviado a CVI.

A classe associada a AA4 é tAA4:

```
class tAA4
{public:
   GenerateAbductedVisualControl(void);
};
```

A função GenerateAbductedVisualControl simplesmente gera um par de coordenadas (ρ_1 , φ_1) aleatório, tomando-se um ρ_1 máximo compatível com os limites do veículo e um φ_1 entre $-\pi/4$ e $\pi/4$. O objeto gerado é enviado a CVR.

O argumento dedutivo em AD9 deve, a partir dos dados dos sensores, decidir se utiliza o objeto em CVI ou em CVR. Caso o sistema esteja focalizando um determinado objeto do ambiente, utiliza-se o objeto em CVI. Se ele estiver procurando por um objeto do ambiente, utiliza o objeto em CVR.

A classe associada a AD9 é tAD9:

```
class tAD9
{public:
   tCV VisualControl(tCV cvi, tCV cvr, tES es);
};
```

A função VisualControl realiza a tarefa de decidir entre as duas sugestões de controle, enviando sua decisão a CV.

Com isso, conclui-se o módulo de controle visual.

5.4.6 Módulo de Interface de Saída

Neste módulo, o controle motor e o controle visual são integrados, resultando um conhecimento prescritivo que é enviado para os atuadores.

A secção da rede correspondente a esse módulo é indicada na figura 5.23 a seguir:

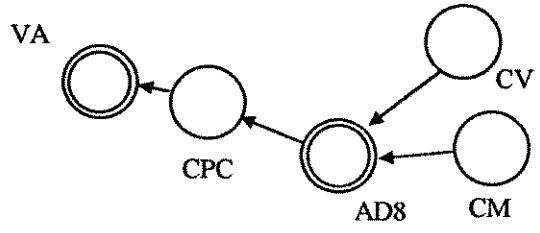


Figura 5.23 - Módulo de Interface de Saída

A classe associada a CPC (conhecimento prescritivo de controle) é tCPC:

```
class tCPC
{double vel;
 double dpitch;
 double sclength;
 double scangle;
};
```

A classe relacionada a AD8 é tAD8:

```
class tAD8
{public:
   tCPC GenerateCPC(tCV cv, tCM cm);
};
```

A função GenerateCPC toma os objetos em CV e CM, correspondendo às determinações parciais de controle visual e controle motor, e integra-as em um único objeto do tipo tCPC. Esse objeto é então enviado a CPC.

A classe associada a VA é tVA:

```
class tVA
{public:
   void SendToActuators(tCPC cpc);
}
```

O argumento dedutivo em VA simplesmente toma o objeto referente ao conhecimento prescritivo de controle em CPC e o envia para os atuadores do sistema.

Com isso conclui-se o módulo de interface de saída, e a descrição das etapas do sistema de controle inteligente.

5.5 Coordenação dos Objetos do Sistema

Para a especificação completa da rede de objetos, não basta a especificação das classes dos objetos que a compõem. É também necessária a determinação das funções de seleção para cada objeto. As funções de seleção é que determinam a coordenação entre os diferentes objetos da rede, de modo que o comportamento inteligente possa emergir.

A coordenação dos objetos do sistema é descrita a seguir. Os primeiros objetos a serem disparados são os objetos do controle visual. Inicialmente, o sistema utiliza o controle visual randômico, até que uma imagem contendo um objeto do ambiente apareça nos sensores visuais. Nesse instante, o sistema passa a uma etapa de focalização, quando tenta identificar a melhor posição para efetuar a percepção do objeto. Durante as etapas de controle aleatório e focalização, o sistema perceptivo permanece inabilitado. Quando o controle visual termina a focalização (i.e. o centro de massa da imagem se encontra a uma distância menor que um limiar), o sistema perceptivo passa a atuar. Então, o argumento em AI1 gera um modelo do objeto focalizado, o argumento em AD1 tenta integrá-lo aos outros modelos já existentes. Uma vez que a percepção tenha se concretizado, o sistema de controle visual retorna ao modo aleatório, repetindo o ciclo. Esta é a coordenação que existe a nível de controle visual e percepção.

Em paralelo, age o controle motor, que tem uma coordenação própria. Inicialmente, a partir da entrada sensorial, o argumento em AA1 gera os pontos origem e meta. Esses pontos são verificados pelos argumentos em AD2 e AD3, mas em princípio estes pontos devem ir direto para CPV. Em seguida, o argumento em AA2 tenta gerar um arco da origem à meta. Se ele consegue, um plano é gerado em PL. Se ele não consegue, o argumento em AA1 gera novos pontos, a partir do ponto origem, primeiro por meio de GenerateNewPoint2, depois por GenerateNewPoint3. O argumento em AA2 tenta então gerar um arco a partir de cada um destes pontos e a meta. Caso ele consiga, o plano é gerado em PL por meio do argumento em AD6. Caso ele não consiga, cada um dos pontos dá origem

a um novo conjunto de pontos, e assim sucessivamente, até que seja possível gerar um arco de um ponto até a meta. Nesse instante, a atividade no módulo de geração de pontos e arcos cessa. Em seguida, são disparados os argumentos em AI2 e AA3, otimizando-se o plano, que é enviado a PL2. Observe que todas essas etapas ocorrem de maneira sequencial.

O argumento em AD7 trabalha em paralelo aos demais objetos. Uma vez havendo um plano em PL2, que é modificado periodicamente, o argumento gera o controle motor, que é enviado a CM. O módulo de interface de saída também é independente. A cada ciclo, ele recolhe os objetos em CV e CM e gera o conhecimento prescritivo de controle, que é enviado aos atuadores.

Observe a coordenação entre os diferentes módulos e sub-módulos. Existem quatro secções da rede disparando em paralelo. A primeira corresponde ao controle visual e percepção. A segunda corresponde à geração dos pontos e arcos, e posteriormente o plano, que no fundo pode ser considerado a geração de um novo plano. A terceira corresponde à geração do controle motor, a partir do plano, e a quarta consiste na geração do conhecimento prescritivo de controle e seu envio aos atuadores. Cada uma dessas secções, entretanto, atua em uma sequência bem definida, ou seja, as funções de seleção para os objetos nesses lugares deve considerar essa sequência, caso contrário, o funcionamento da rede de objetos como um todo será prejudicado.

5.6 Descrição do Simulador

Para executar os experimentos de simulação, foi construído um simulador, utilizando-se ambiente Unix/X-Windows e linguagem C. O simulador permite que, de maneira bem amigável, o usuário construa diferentes ambientes de teste, bem como altere as características dos objetos do ambiente e do veículo. A tela inicial de apresentação do simulador pode ser vista na figura 5.24 a seguir:

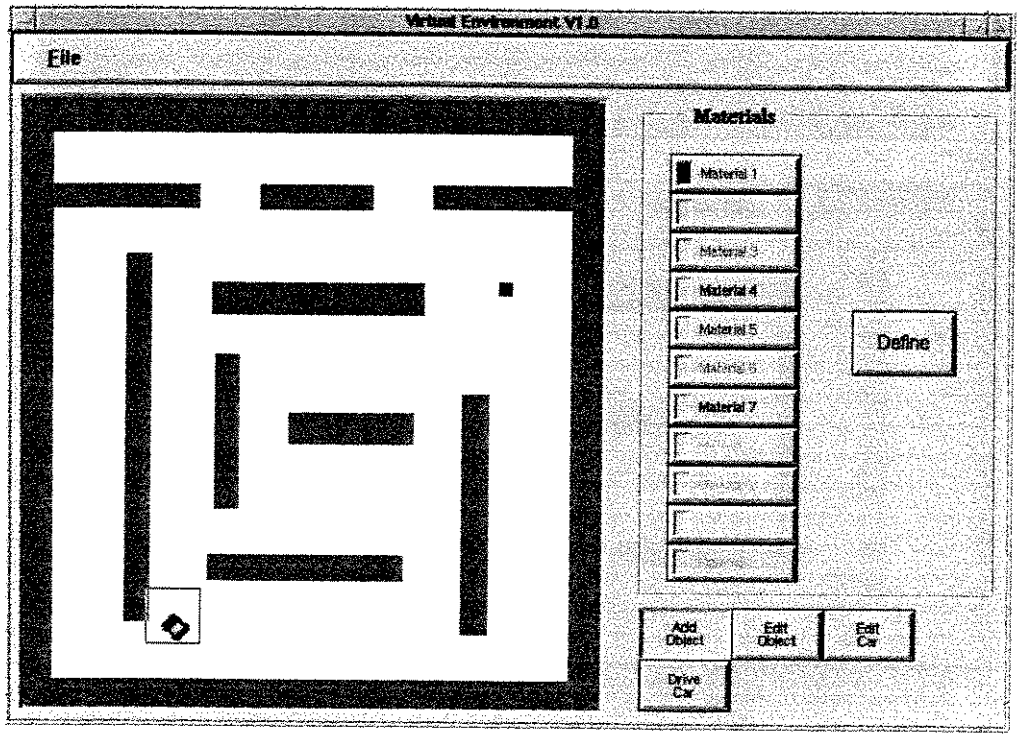


Figura 5.24 - Tela Inicial do Simulador

Nos botões a direita, o usuário pode escolher diferentes materiais e, com o mouse, pode traçar retângulos no *canvas* de visualização do ambiente. Querendo editar algum dos retângulos, o usuário clica o mouse sobre o botão "*Edit Object*", e clicando então o mouse sobre o retângulo desejado, aparecem quatro pontos guias. Com o uso do mouse, o usuário pode então alterar o tamanho e a posição dos retângulos desejados. Os diferentes materiais podem ter regulados sua dureza, gosto e fluxo de energia. Para tal, deve-se clicar sobre o botão "*Define*". Aparece então, uma segunda janela, ilustrada na figura 5.25, que permite que o usuário defina os valores de dureza, gosto e fluxo de energia.

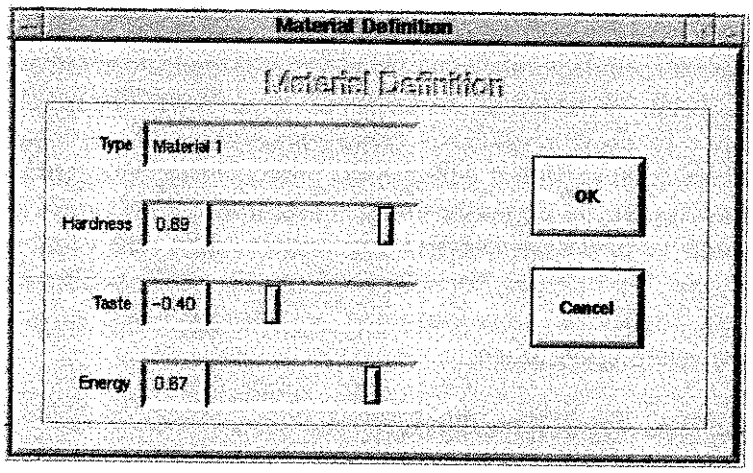


Figura 5.25 - Definição das Características dos Materiais do Ambiente

A partir da tela inicial, clicando-se sobre o botão "Edit Car", o sistema entra em modo de edição do veículo. Os controles na tela mudam conforme mostra a figura 5.26:

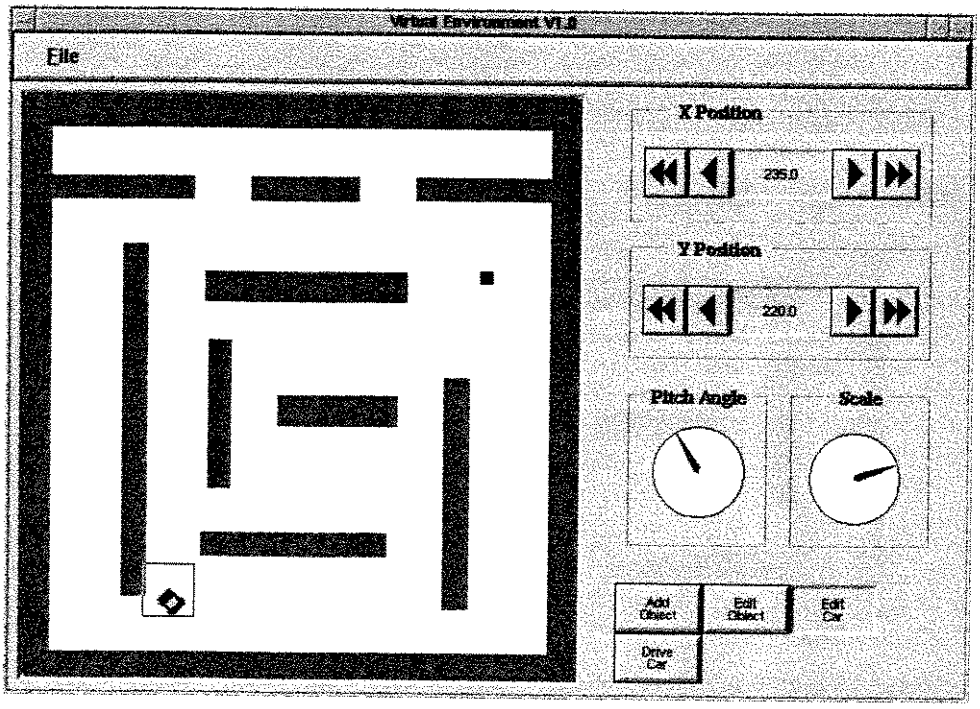


Figura 5.26 - Simulador em Modo de Edição do Veículo

Neste modo, o usuário pode ajustar a posição inicial do veículo, bem como seu ângulo de *pitch* e o próprio tamanho do veículo.

Clicando-se sobre o botão "Drive Car", o simulador entra em modo de simulação. Os controles mudam novamente, conforme mostra a figura 5.27:

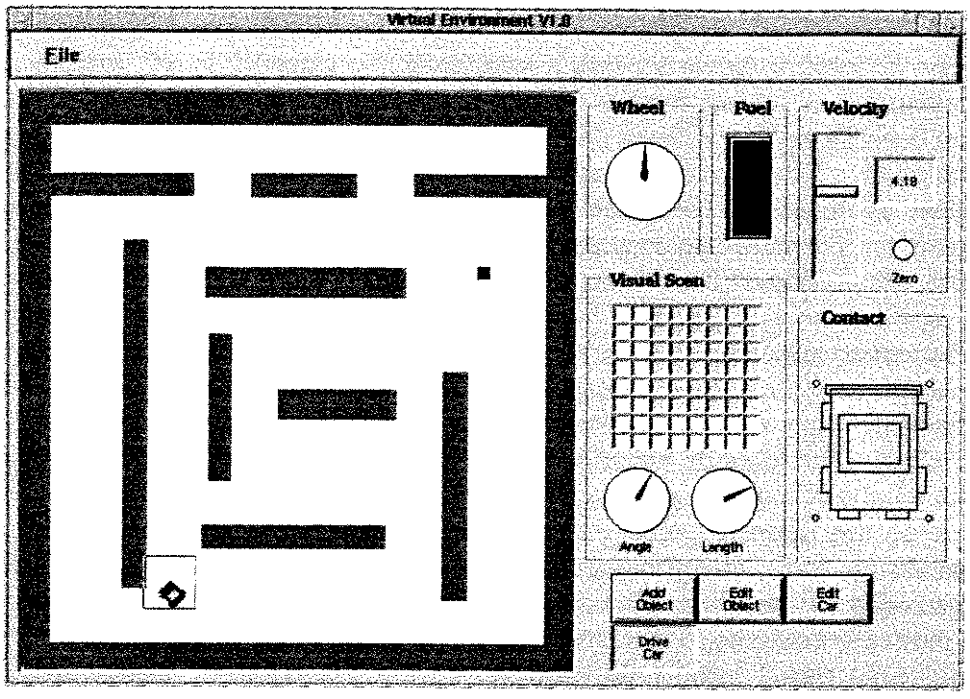


Figura 5.27 - Simulador em Modo de Simulação

Simultaneamente, aparece uma outra janela, com características adicionais de controle, conforme mostra a figura 5.28.

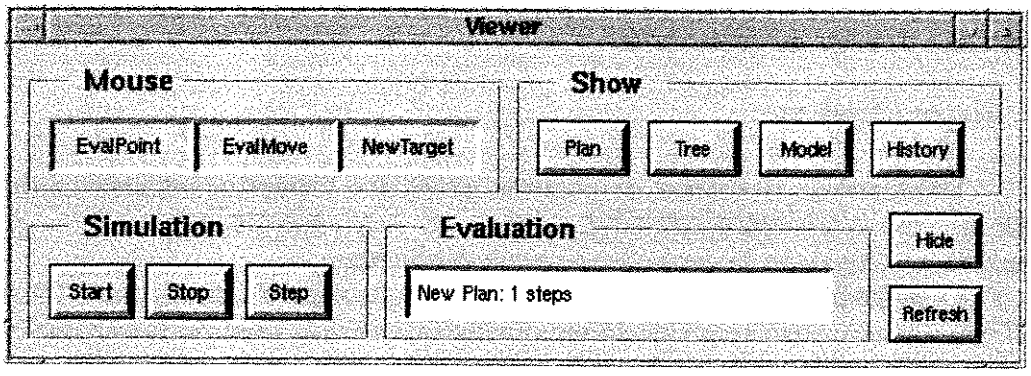


Figura 5.28 - Painel Auxiliar de Controle

Na janela principal, tem-se uma visualização das variáveis de entrada e saída do sistema de controle. O controle "*Wheel*" mostra a direção do ângulo das rodas do veículo. O Controle "*Fuel*" indica o nível das baterias do veículo. O controle "*Velocity*" ao mesmo tempo mostra a velocidade nominal do veículo, e permite que o usuário a altere, durante a simulação. O botão "*Zero*" permite que o usuário pare o veículo, reduzindo imediatamente sua velocidade a zero. O controle "*Visual Scan*" mostra a imagem visualizada pelo sistema de controle, bem como os valores de comprimento e ângulo do foco de atenção. Estando a simulação parada, o

usuário pode modificar esses valores, por meio dos controles e observar a imagem correspondente. O controle "Contact" mostra o estado dos sensores de contato, mostrando a cor do objeto que se encontra em contato com o veículo, em cada uma de suas extremidades.

O painel auxiliar de controle permite que o usuário conduza a simulação. No campo "*Simulation*", encontram-se três botões, permitindo que o usuário comece, termine ou execute passo a passo a simulação. No campo "*Show*", quatro botões permitem que o sistema mostre o plano atual do veículo, a árvore de planos que gerou o plano atual, o modelo do ambiente, e um histórico da posição do carrinho nos últimos 100 passos de simulação. O campo "*Mouse*", tem três botões, que indicam as funções executadas quando se clica o mouse sobre o canvas de simulação. Clicando-se sobre esses botões, muda-se a função atribuída ao botão do mouse, escolhendo-se dentre uma série de funções, a partir de um menu de escolha que aparece sobre o botão. Funções típicas nesse caso são a avaliação do conhecimento apraisivo de um determinado ponto, de um determinado arco, que é desenhado com o mouse, a distância entre dois pontos, a determinação do ponto de alinhamento com a meta utilizando-se curva máxima (conforme figura 5.19), a previsão dos sensores de contato caso o veículo se encontre em um determinado ponto e outras funções auxiliares desse tipo. A saída dessas funções pode se dar na forma de um desenho sobre o canvas de simulação ou, no caso de um valor numérico ou texto, este é mostrado no campo "*Evaluation*". O botão "Refresh" apaga todos os desenhos sobre a tela do canvas, mantendo apenas a tela padrão, que mostra o ambiente, o veículo e a meta. A tecla "Hide" permite fechar a janela referente ao painel auxiliar de controle. Esta janela pode ser aberta novamente teclando-se novamente sobre o botão "Drive Car" na janela principal.

O controle por meio do sistema inteligente se encontra incorporado no próprio código do programa, por meio de uma função chamada CarControl. Entretanto, pode-se testar diferentes tipos de controladores com o mesmo ambiente, simplesmente utilizando-se uma outra função com este mesmo nome, que tenha os mesmos parâmetros sendo passados. Os parâmetros consistem basicamente da entrada e da saída do controlador.

5.7 Resultados

Utilizando-se o sistema de controle inteligente proposto, implementado de acordo com o simulador descrito, efetuou-se uma série de experimentos para verificar a efetividade do controle no veículo autônomo.

O primeiro experimento constou do ambiente descrito na figura 5.29. No primeiro quadro, tem-se o veículo em um instante inicial, antes de tomar qualquer decisão. No segundo quadro, baseado no modelo de ambiente disponível (i.e. vazio, pois o sistema está iniciando), ele escolhe um plano, que no caso é prosseguir diretamente até a meta. Uma vez que as paredes que escondem a meta sejam conhecidas pelo sistema visual, elaborando-se um modelo inicial do ambiente, o veículo procurará caminhos alternativos. A terceira figura mostra o veículo após um certo número de passos, quando o sistema já forma um modelo inicial do ambiente, indicado pelos retângulos na figura com contornos escuros. A partir deste modelo preliminar, o sistema gera um plano alternativo que, entretanto, ainda colide com trechos do ambiente que ele ainda não teve a oportunidade de enxergar. Após um grande número de passos, quando o veículo inclusive chegou até a meta diversas vezes, um modelo bem completo do ambiente já pode ser considerado. Nesse caso,

o sistema encontra um plano facilmente até a meta, sem colidir com nenhum objeto do ambiente.

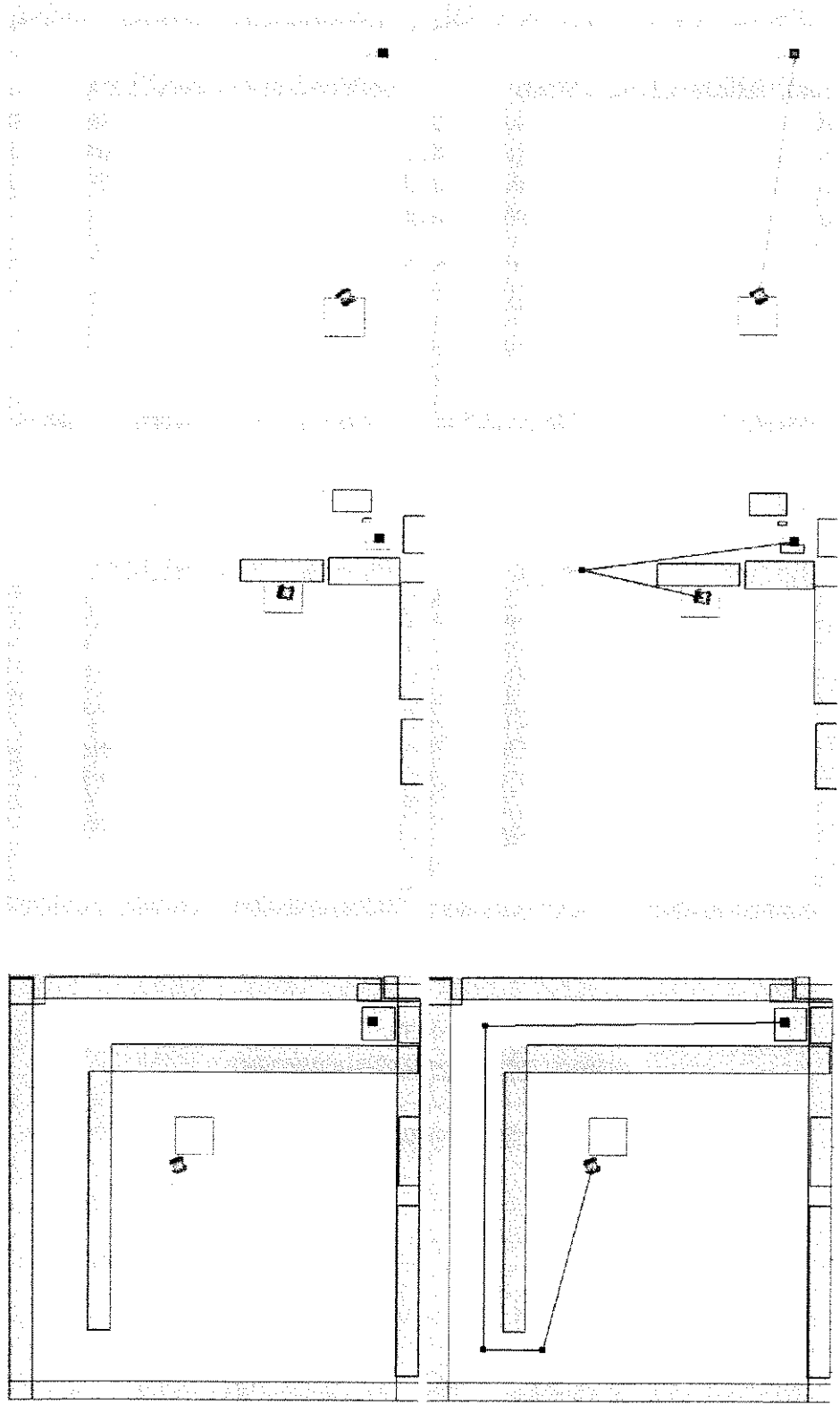


Figura 5.29 - Resultados de Simulação

No ambiente do exemplo em consideração, os objetivos a serem atingidos são antagônicos. Por um lado, o veículo deve se aproximar da meta. Mas, para efetivamente alcançá-la, ele precisa primeiro se afastar consideravelmente dela, o que faz com que sistemas de controle simplesmente reativos, ou seja, que funcionam sem um modelo do ambiente, baseado somente nas informações dos sensores, falhem. Esse cenário foi escolhido justamente para mostrar a superioridade do modelo híbrido sobre o modelo reativo. Conforme pode ser observado na figura 5.29, o sistema de controle por rede de objetos consegue atingir satisfatoriamente a meta. Por outro lado, conforme a figura 5.30, um modelo simplesmente reativo (Fabro, 1996) não consegue atingí-la.

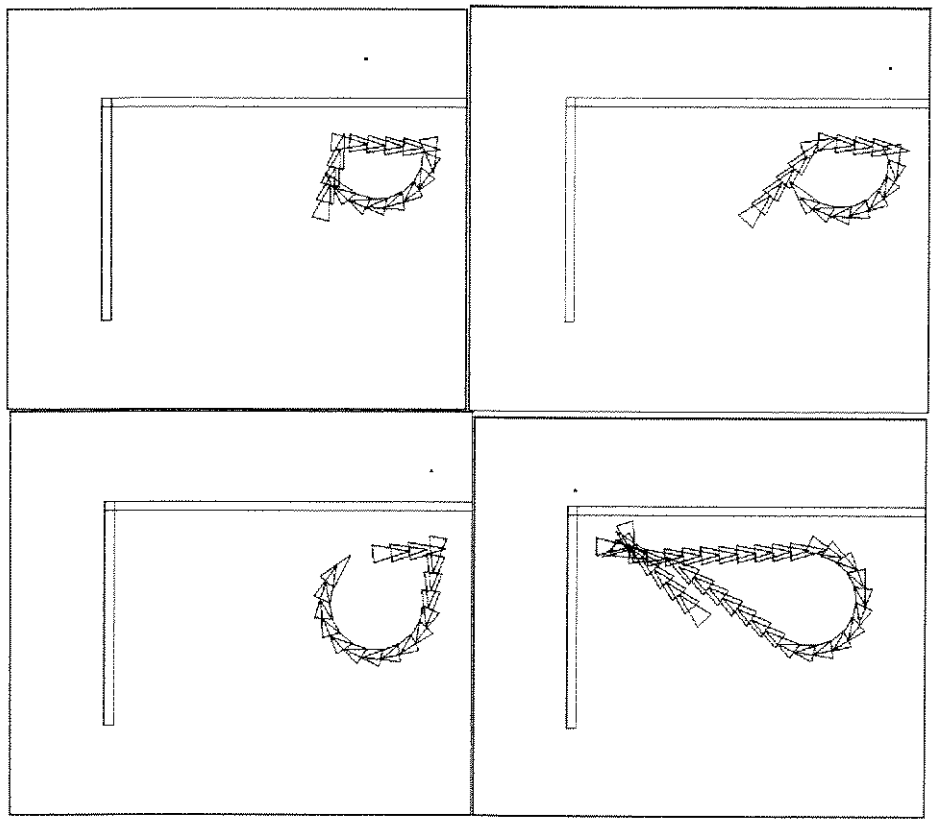


Figura 5.30 - Simulação Equivalente Utilizando Modelo Reativo

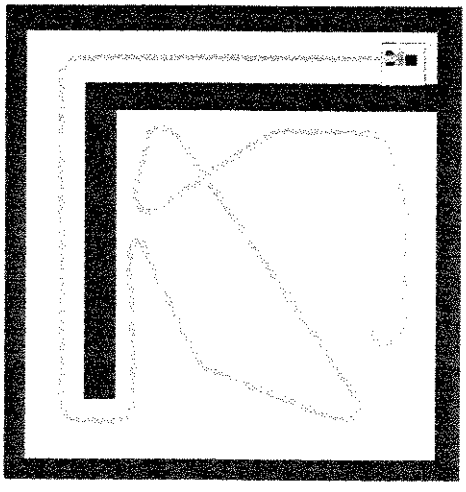


Figura 5.31 - Trajetória do Veículo Controlado por Rede de Objetos

A trajetória completa derivada do controle por meio de rede de objeto, pode ser vista na figura 5.31.

Na figura 5.32, têm-se a árvore de busca para o ambiente testado, e na figura 5.33 para outros dois ambientes analisados.

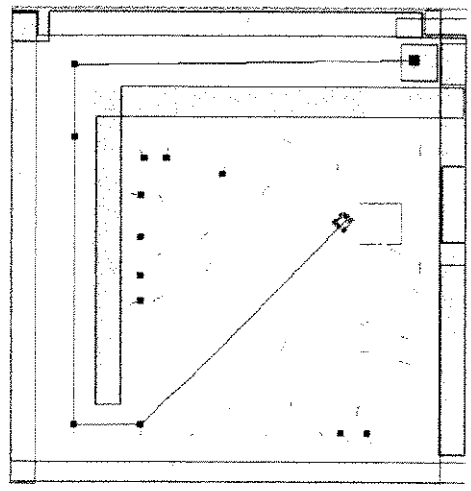


Figura 5.32 - Árvore de Busca para a Formação de um Plano

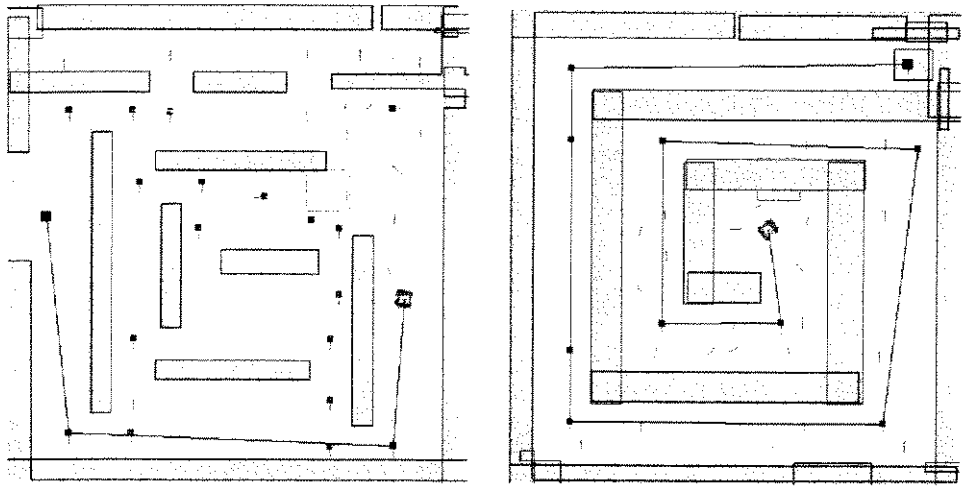


Figura 5.33 - Árvore de Busca em Outros Ambientes

De um modo geral, o controle inteligente via rede de objetos detém uma boa performance em diversos ambientes testados, atingindo a meta em todos eles.

5.8 Resumo

Nesse capítulo desenvolveu-se um exemplo de aplicação das redes de objetos em sistemas de controle inteligente, explorando o problema do controle de um veículo autônomo. Propôs-se uma rede de objetos para implementar o controle inteligente, que foi detalhadamente descrita. Esta rede foi então implementada em um simulador para a verificação de suas propriedades.

No próximo capítulo, têm-se a conclusão da tese, e os trabalhos futuros.

6. Conclusões e Trabalhos Futuros

Neste trabalho, apresentamos os fundamentos básicos necessários para a construção de uma teoria matemática sobre sistemas inteligentes. Para isso, procedeu-se a uma categorização, baseada nas idéias da semiótica, dos diferentes tipos de conhecimento envolvidos no fenômeno da inteligência. Para a formalização adequada dos tipos de conhecimentos catalogados, foi necessário introduzir, preliminarmente, os princípios de uma teoria geral dos objetos. Esta, incorpora o conceito de objeto de uma maneira genérica, de modo a incluir não somente o conceito de objeto, como utilizado na programação orientada a objeto, mas a idéia de um objeto matemático, que é colocado como um modelo para um objeto físico. A partir destes princípios, desenvolveu-se o conceito de redes de objetos, apresentadas como uma ferramenta de modelagem adequada para a descrição de sistemas inteligentes. Então, foi feito um mapeamento dos tipos de conhecimentos catalogados anteriormente, sobre a estrutura formal das redes de objetos, obtendo-se uma representação formal para estes. Por fim, apresentou-se um exemplo de um sistema inteligente, responsável pelo controle de um veículo autônomo, desenvolvido e modelado a partir das redes de objetos. Este exemplo, apesar de ser uma simplificação de um sistema real, serve para ilustrar a integração dos diferentes tipos de conhecimento de modo a resultar um comportamento inteligente emergente.

As principais contribuições desta tese são as seguintes:

- A elaboração de uma taxonomia dos tipos de conhecimento, baseada na semiótica.
- O estabelecimento dos fundamentos matemáticos para uma teoria geral dos objetos.
- O desenvolvimento de uma ferramenta de modelagem, a rede de objetos, que permite a modelagem de sistemas com características de adaptação e aprendizagem.
- A formalização dos diferentes tipos de conhecimentos em termos de redes de objetos.
- A identificação das redes de objetos como ferramentas de modelagem formal adequadas para a representação de sistemas inteligentes.
- Um exemplo de um sistema inteligente para o controle de um veículo autônomo.

Vamos analisar agora o limite e a extensão de cada uma destas contribuições.

Taxonomia dos tipos de conhecimento, baseada na semiótica

A taxonomia dos tipos de conhecimento elaborada, apesar de significativa, é apenas parcial. Peirce, em seus trabalhos, chega a identificar mais de 100 diferentes tipos de signos, que eventualmente podem implicar em diferentes tipos de conhecimentos. Estes, não são discriminados na taxonomia apresentada. Entretanto,

a taxonomia proposta permite a elaboração de sistemas inteligentes razoavelmente sofisticados. Além disso, cria uma organização que não é encontrada usualmente na literatura, quanto aos diferentes tipos de conhecimentos utilizados na construção de sistemas inteligentes.

Fundamentos para uma teoria geral dos objetos.

O formalismo introduzido neste trabalho não chega ao ponto de propor uma teoria geral dos objetos, mas simplesmente propõe fundamentos para que esta elaboração possa ser efetuada. Algumas extensões são realmente necessárias, de modo que o formalismo possa lidar adequadamente com os problemas discutidos no final do capítulo 3, envolvendo o assincronismo na interação entre os objetos. Entretanto, o formalismo, na maneira em que se encontra, é adequado para a representação de sistemas inteligentes, o que foi inicialmente a principal motivação para sua implementação. Neste sentido, trata-se de um passo importante.

Ferramenta de modelagem, a rede de objetos.

As redes de objetos, desenvolvidas sobre o conceito matemático de objetos apresentado, apesar do seu poder de representação, contam ainda com uma série de limitações. Por exemplo, as ferramentas de análise são ainda incipientes, bem aquém das existentes e.g. para redes de Petri. Poucos sistemas foram até agora modelados por meio do formalismo das redes de objetos.

Tipos de conhecimentos em termos de redes de objetos.

A taxonomia dos tipos de conhecimento, como já se levantou, ainda não pode ser considerada completa. Com isso, para os tipos de conhecimentos que não foram especificados, pode ser que o formalismo necessite de algumas modificações.

Redes de objetos: ferramentas de modelagem formal adequadas para a representação de sistemas inteligentes.

Nesse caso, apesar de existir a prerrogativa formal, poucos sistemas inteligentes foram até agora construídos e modelados utilizando-se as redes de objetos. Para uma consolidação das redes de objetos como ferramentas de modelagem para sistemas inteligentes, é necessário que mais problemas sejam resolvidos utilizando-se desta metodologia, o que pode eventualmente ressaltar suas virtudes ou identificar oportunidades para sua extensão.

Sistema inteligente para o controle de um veículo autônomo.

O modelo de veículo autônomo desenvolvido neste trabalho é uma simplificação de um veículo real. Para que se possa considerar a aplicação das técnicas sugeridas aqui, em um sistema real, é necessário que os diferentes módulos sejam replanejados e re-avaliados de modo a se sobrepor a essas simplificações, e realizar sua tarefa dentro das restrições reais encontradas em um problema de engenharia.

Como pode ser depreendido destes fatos, existe um número considerável de trabalhos futuros a serem realizados de modo a dar continuidade na consolidação das contribuições aqui propostas.

Em primeiro lugar, uma análise mais profunda da semiótica, poderá ajudar no enriquecimento da taxonomia dos tipos de conhecimento apresentada neste trabalho, permitindo que o fenômeno da inteligência seja melhor compreendido em toda sua extensão. Esse aumento nos tipos de conhecimentos catalogados demandará então uma representação formal, que necessitará ser investigada.

Apesar dos fundamentos estarem aqui propostos, uma série de pontos precisam ser investigados, de modo que se possa chegar realmente a uma teoria geral dos objetos, que incorpore definitivamente todos os tipos de entidade que podem ser entendidas como objetos. Para tal, modificações são certamente necessárias, por exemplo, analisando-se os objetos em termos de domínios contínuos, ao invés de conjuntos enumeráveis como foi proposto neste trabalho. Essa modificação demandará uma reformulação nos conceitos de sistemas de objetos e redes de objetos, de modo a torná-los conceitos mais amplos. Nesse caso, os sistemas de objetos e redes de objetos, como propostos, serão casos particulares dessas formalizações de âmbito mais geral.

A utilização das redes de objetos como um formalismo na representação de outros tipos de sistemas é outra necessidade. Uma área com enorme potencial de investigação é o uso das redes de objetos como veículo de formalização para as entidades computacionais conhecidas como agentes, ou agentes inteligentes. As características intrínsecas a estas entidades computacionais sugerem que as redes de objeto podem constituir um formalismo extremamente confortável para sua modelagem. A partir disso, uma série de propriedades que se deseja para tais agentes poderiam ser provadas formalmente, permitindo seu uso em situações onde alguns fatores tais como segurança e confiabilidade são críticos.

Particularmente aos sistemas inteligentes, seria necessário investigar sua aplicação a outras classes de problemas. Inicialmente, poderia-se analisar os problemas que são atualmente resolvidos por outros tipos de formalizações, passando-se então a problemas que não se encontram ainda totalmente resolvidos. Essas aplicações são fundamentais na caracterização dos defeitos e virtudes do formalismo apresentado, permitindo sua evolução.

Quanto ao exemplo do veículo autônomo, este poderá continuar como um bom laboratório para evidenciar características desejáveis de um sistema inteligente. Pode-se partir para a implementação do nível decisório de controle, e eventualmente até a experimentos compreendendo múltiplos veículos, comunicando-se entre si, que poderão ressaltar outros aspectos do comportamento inteligente não considerados no presente estudo. Pode-se testar se é possível que algumas características que são consideradas inatas no modelo atual possam emergir a partir da interação entre o veículo e o ambiente, diante de procedimentos adequados de adaptação e aprendizagem, e com isso, ir relaxando as simplificações colocadas no modelo.

Talvez o número de trabalhos futuros necessários suplante o que já foi feito até agora, evidenciando uma característica fundamental desta tese que é a de ser um começo. Ao contrário de muitas teses, que complementam uma teoria já desenvolvida, consolidando-a, neste trabalho o que se fez foi iniciar novas teorias, plantar as sementes para que estas possam ser desenvolvidas. Tanto para uma teoria geral dos objetos como para uma teoria dos sistemas inteligentes, planta-se hoje um fundamento, uma base. Uma pedra fundamental que, espera-se, germine e frutifique, levando o ser humano ao melhor entendimento dos fenômenos que o

cercam, e que permitam um melhor aproveitamento destes fenômenos para o bem de nossa civilização.

7. Referências

(Albus, J.S. 1991) - "Outline for a Theory of Intelligence" - IEEE Transactions on Systems, Man and Cybernetics, vol. 21, n. 3, May/June 1991.

(Anderson, J.R. 1989) - "A Theory of the Origins of Human Knowledge" - Artificial Intelligence 40 (1989) pp. 313-351.

(Baldassari, M.; Bruno, G. 1988) - "An Enviroment for Object Oriented Conceptual Programming Based in PROT Nets" - Lecture Notes in Computer Science 340 - Advances in Petri Nets, pp. 1-19.

(Barr, A. ; Cohen, P. ; Feigenbaum, E.A. 1981) - "The Handbook of Artificial Inteligence", vol. IV, Addison-Wesley Publishing Company, 1981, 2nd edition 1989.

(Barr, A. ; Feigenbaum, E.A. 1981a) - "The Handbook of Artificial Inteligence", vol. I, Addison-Wesley Publishing Company, 1981, 2nd edition 1989.

(Barr, A. ; Feigenbaum, E.A. 1981b) - "The Handbook of Artificial Inteligence", vol. II, Addison-Wesley Publishing Company, 1981, 2nd edition 1989.

(Battiston, E. ; De Cindio, F. Mauri, G. 1988) - "OBJSA Nets : A Class of High Level Nets having Objects as Domains" - Lecture Notes in Computer Science 340 - Advances in Petri Nets, pp. 20-43.

(Beom, H.R.; Cho, H.S. 1995) - "A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning", IEEE Transactions on Systems, Man and Cybernetics, vol. 25, n. 3, March 1995.

(Boden, M. A. 1983) - "As Idéias de Piaget" - tradução de Álvaro Cabral - Editora Cultrix - Editora da Universidade de São Paulo - São Paulo, 1983.

(Bolc, L. 1987) - "Computational Models of Learning" - Springer-Verlag, Berlin

(Booker, L.B. ; Goldberg, D.E. ; Holland, J.H. 1989) - "Classifier Systems and Genetic Algorithms" - Artificial Intelligence 40 (1989) pp. 235-282.

(Brachman, R.J. ; Schmolze, J.G. 1985) - "An Overview of the KL-ONE Knowledge Representation System" - Cognitive Science 9, 171-216 (1985).

(Brooks, R.A. 1991) - "Intelligence Without Reason" - Proceedings of the Twelfth International Conference on Artificial Intelligence, Vol. 1 (1991) 569-595.

(Bylander, T.; Allemang, D.; Tanner, M.C.; Josephson, J.R. 1991) - "The Computational Complexity of Abduction" - Artificial Intelligence 49 (1991) 25-60.

(Carbonell, J.G. 1983) - "Learning by Analogy: Formulating and Generalizing Plans from Past Experience" - in Michalski et.al. *Machine Learning - An Artificial Intelligence Approach* - Morgan Kaufmann Publishers, Inc. pp 83-134.

(Cardoso, J. ; Valette, R. ; Dubois, D. 1990) - "Petri Nets with Uncertain Markings" - Lecture Notes in Computer Science 483 - Advances in Petri Nets, pp. 64-78.

(Chang, C.L. ; Lee, R.C.T. 1973) - "Symbolic Logic and Mechanical Theorem Proving" - Academic Press, New York, 1973.

(Chen, C.X.; Trivedi, M.M. 1995) - "Task Planning and Action Coordination in Integrated Sensor-Based Robots", IEEE Transactions on Systems, Man and Cybernetics, vol. 25, n.4, April 1995.

(Chung, Sheng-Luen ; Lafortune, S. 1992) - "Limited Lookahead Policies in Supervisory Control of Discrete Event Systems"- IEEE Transactions on Automatic Control, vol. 37, n. 12, December 1992.

(Coelho Netto, J.T. 1980) - "Semiótica, Informação e Comunicação" - Coleção Debates - tomo 168 - Editora Perspectiva.

(Cohen, P. ; Feigenbaum, E.A. 1981) - "The Handbook of Artificial Inteligence", vol. III, Addison-Wesley Publishing Company, 1981, 2nd edition 1989.

(Cohen, B.;Murphy,G.L. 1984) - "Models Of Concepts" - Cognitive Science 8 (1984) pp. 27-58

(Collins, A. ; Michalski, R. 1989) - "The Logic of Plausible Reasoning : A Core Theory" - Cognitive Science 13, 1-49 (1989).

(Davis, M. 1958) - "Computability & Unsolvability" - McGraw-Hill Book Company, New York, 1958.

(Eco, U. 1980) - "Tratado Geral de Semiótica" - Coleção Estudos - tomo 73 - Editora Perspectiva.

(Edelman, G. M. ; Mountcastle, V. B. 1978) - "The Mindful Brain : Cortical Organization and the Group-Selective Theory of Higher Brain Function" - Cambridge : MIT, 1978

(Edelman, G. M. 1987)- "Neural Darwinism - The Theory of Neuronal Group Selection" - Basic Books, Inc, 1987

(Enderton, H. 1972) - "A Mathematical Introduction to Logic" - Academic Press, 1972, USA.

(Epstein, R.L.; Carnielli, W.A. 1989) - "Computability : Computable Functions, Logic and the Foundations of Mathematics" - Wadsworth & Brooks/ Cole Advanced Books & Software - Pacific Grove, California, USA.

(Fabro, J.A. 1996) - "Grupos Neurais e Sistemas Nebulosos : Aplicação à Navegação Autônoma" - Tese de Mestrado - DCA-FEE-UNICAMP, Fev. 1996.

(Fan, K.C.;Lui, P.C. 1994) - "Solving Find Path Problem in Mapped Environments Using Modified A* Algorithm" - IEEE Transactions on Systems, Man and Cybernetics - vol. 24, n. 9, September 1994.

(Genrich, H.J.; Lautenbach, K. 1981) - "System Modelling with High Level Petri Nets" - Theoretical Computer Science 13, pp. 109-136.

(Goldberg, D.E. 1989) - "Genetic Algorithms in Search, Optimization, and Machine Learning" - Addison-Wesley Publishing Company, Inc.

(Gudwin, R.R.; Gomide, F.A.C. 1994a) - "Genetic Algorithms and Discrete Event Systems : An Application" - *Proceedings of The First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, 26 de Junho a 2 de Julho de 1994, Orlando, Florida, USA.

(Gudwin, R.R.; Gomide, F.A.C. 1994b) - "Context Adaptation in Fuzzy Processing" - *Proceedings of the Brazil-Japan Joint Symposium on Fuzzy Systems*, 19 a 22 de Julho de 1994, Campinas - SP -Brasil.

(Jensen, K. 1981) - "Coloured Petri Nets and the Invariant Method" - Theoretical Computer Science 14 pp. 317-336.

(Jensen, K. 1981b) - "How to Find Invariants for Coloured Petri Nets" - Lecture Notes in Computer Science 118 - Mathematical Foundations of Computer Science, pp. 327-338.

(Jensen, K. 1990) - "Coloured Petri Nets : A High Level Language for System Design and Analysis" - Lecture Notes in Computer Science 483 - Advances in Petri Nets, pp. 342-416.

(José Neto, J. 1993) - "Contribuições à Metodologia de Construção de Compiladores" - Tese de Livre-Docência - Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica, Universidade de São Paulo.

(Kosko, B. 1992) - "Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence" - Prentice-Hall International, Inc., London, UK.

(Krozel, J.; Andrisani II, D. 1995) - "Intelligent ε-Optimal Path Prediction for Vehicular Travel", IEEE Transactions on Systems, Man and Cybernetics, vol. 25, n.2 - February 1995.

(Laird, J.E. ; Newell, A. ; Rosenbloom, P.S. 1987) - "SOAR: An Architecture for General Intelligence" - Artificial Intelligence 33 (1987) 1-64.

(Lippman, R.P. 1987) - "An Introduction To Computing with Neural Nets" - IEEE ASSP Magazine April 1987

(Michalski, R.S. 1983) - "A Theory and Methodology of Inductive Learning" - in Michalski et.al. *Machine Learning - An Artificial Intelligence Approach* - Morgan Kaufmann Publishers, Inc. pp 83-134.

(Michalski, R.S. 1987) - "Learning Strategies and Automated Knowledge Acquisition - An Overview" in Bolc. R.- *Computational Models of Learning*, Springer Verlag, Berlin.

(Michalski, R.S.; Carbonnell, J.G.; Mitchell, T.M. 1983) - "Machine Learning - An Artificial Intelligence Approach" - Morgan Kaufmann Publishers, Inc.

(Michalski, R.S.; Carbonnell, J.G.; Mitchell, T.M. 1986) - "Machine Learning - An Artificial Intelligence Approach - Volume II" - Morgan Kaufmann Publishers, Inc.

(Morris, C.W. 1947) - "Signs, Language and Behaviour" - New York : Prentice Hall, 1947

(Morris, C. W. 1971) - "Foundation for a Theory of Signs" - in "Writings on the General Theory of Signs" - The Hague : Mouton, 1971

(Morris, C. W. 1964) - "Significant and Significance"

(Murata, T. 1989) - "Petri Nets : Properties, Analysis and Applications" - Proceedings of the IEEE, vol. 77, n. 4, April 1989.

(Newell, A. 1982) - "The Knowledge Level" - Artificial Intelligence 18 (1982) 87-127.

(Nilsson, N. 1980) - "Principles of Artificial Intelligence" - Tioga, Palo Alto, CA, 1980.

(Norman, D.A. 1991) - "Approaches to the Study of Intelligence" - Artificial Intelligence 47 (1991) 327-346.

(Oliveira, M.; Figueiredo, M.; Gomide F. 1994) - "A Neurofuzzy Approach for Autonomous Control" - 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, IIZUKA'94, 1994, Fukuoka.

(Platão, 1991) - Platão - Diálogos - Coleção "Os Pensadores" - Seleção de textos de José Américo Motta Pessanha; tradução e notas de José Cavalcante de Souza, Jorge Paleikat e João Cruz Costa - 5. ed. São Paulo : Editora Nova Cultural - 1991.

(Pedrycz, W. 1989) - "Fuzzy Control and Fuzzy Systems" - John Wiley & Sons Inc. , New York, USA.

(Peirce, C.S. 1975) - "Semiótica e Filosofia" - Textos escolhidos de Charles S. Peirce. Tradução parcial de *The Collected Papers of Charles Sanders Peirce*, em vários volumes, edições e datas diversas. Introdução, seleção e tradução de Octanny Silveira da Mota e Leonidas Hegenberg - 2. ed. - Editora Cultrix, USP - São Paulo, 1975.

(Peirce, C.S. 1990) - "Semiótica" - Coleção Estudos - tomo 46 - - Tradução parcial de *The Collected Papers of Charles Sanders Peirce* / Charles Hartshorne e Paul Weiss, ed. Tradução: José Teixeira Coelho Neto - 2. ed. - Editora Perspectiva - São Paulo, 1990.

(Ram, A. ; Leake, D. 1991) - "Evaluation of Explanatory Hypotheses" - Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society, Chicago, IL, August, 1991.

(Rao, N.S. 1995) - "Robot Navigation in Unknown Generalized Polygonal Terrains Using Vision Sensors", IEEE Transactions on Systems, Man and Cybernetics, vol. 25, n. 6, June, 1995.

(Rendell, L. 1987) - "Conceptual Knowledge Acquisition in Search" - in *Computational Models of Learning* - L. Bolc. (ed.) Springer-Verlag Berlin, Germany, pp. 89-159.

(Shafer, G. 1976) - "A Mathematical Theory of Evidence" - Princeton, N.J. - Princeton University Press, 1976.

(Snyder, A. 1993) - "The Essence of Objects : Concepts and Terms" - IEEE Software, Jan. 1993, pp. 31-42

(Spence, R.; Hutchinson, S. 1995) - "An Integrated Architecture for Robot Motion Planning and Control in the Presence of Obstacles with Unknown Trajectories" - IEEE Transactions on Systems, Man and Cybernetics, vol 25, n. 1 - January 1995.

(Turing, A.M. 1950) - "Computing Machinery and Intelligence" - Mind, 59:433-460, 1950, reproduzido em Collins, A. & Smith, E.E. - Readings in Cognitive Science - A Perspective from Psychology and Artificial Intelligence - Morgan Kaufmann Publishers, Inc. San Mateo, California, 1988

(Valk, R. 1978) - "Self-Modifying Nets - A Natural Extension of Petri Nets" - Lecture Notes in Computer Science 62 - Automata, Languages and Programming, pp. 464-477.

(Valk, R. 1981) - "Generalizations of Petri Nets" - Lecture Notes in Computer Science 118 - Mathematical Foundations of Computer Science, pp. 140-156.

(Verschure, P.F.M.J.; Kröse, B.J.A.; Pfeifer, R. 1992) - "Distributed Adaptive Control : The Self-organization of Structured Behavior" - Robotics and Autonomous Systems 9 (1992) 181-196.

(Verschure, P. 1993) - "Formal Minds and Biological Brains - AI and Edelman's Extended Theory of Neuronal Group Selection" - IEEE Expert, October 1993, pp. 66-75

(Wand, Y. 1989) - "A Proposal for a Formal Model of Objects" - in Object-Oriented Concepts, Databases and Applications, W. Kim and F. Lochovsky, eds., ACM Press, New York, 1989, pp. 537-559

(Wang, F.; Kyriakopoulos, K.; Tsolkas, T.; Saridis, G.N. 1991)- "A Petri Net Coordination Model of Intelligent Mobile Robots" - IEEE Transactions on Systems, Man and Cybernetics - vol. 2, n. 4, July/August 1991.

(Wegner, P. 1995) - "Interactive Foundations of Object-Based Programming" - in Object Technology - A Virtual Roundtable, H. El-Rewini, S. Hamilton, IEEE Computer, vol. 28 n. 10, October 1995, pp. 70-72.

(Wolczko, M. I. 1988) - "Semantics of Object-Oriented Languages" - PhD Thesis - Technical Report Series UMCS-88-6-1 - Department of Computer Science, University of Manchester, Manchester M13 9PL, England.

(Yen, J.; Pfluger, N. 1995) - "A Fuzzy Logic Based Extension to Payton and Rosenblatt's Command Fusion Method for Mobile Robot Navigation", IEEE Transactions on Systems, Man and Cybernetics, vol. 25, n. 6, June, 1995.

(Zhou, M.C. ; Dicesare, F. 1989) - "Adaptative Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems" - IEEE Transactions on System, Man and Cybernetics, vol. 19, n.5 September/October, pp. 963-973.

(Zurada, J.; Marks II, R.J.; Robinson, C.J. 1994) - "Computational Intelligence - Imitating Life" - IEEE Press.