UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SDIP: Um Ambiente Inteligente para a Localização de Informações na Internet

por

Luís Fernando Nunes Fernandez

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

Prof. Liane M. R. Tarouco
Orientador

Porto Alegre, Agosto de 1995

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Fernandez, Luís Fernando Nunes

SDIP: Um Ambiente Inteligente para a Localização de Informações na Internet / Luís Fernando Nunes Fernandez. – Porto Alegre: CPGCC da UFRGS, 1995.

247 p. : il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1995. Orientador: Tarouco, Liane M. R.

Dissertação: Inteligência Artificial, Redes de Computadores Linguagem Natural, Representação do Discurso, Representação do Conhecimento, Internet, Archie, Gopher, WAIS, WWW

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Dr. Hélgio Casses Trindade
Pró-Reitor de Pesquisa e Pós-Graduação: Prof. Dr. Claudio Scherer
Diretor do Instituto de Informática: Prof. Dr. Roberto Tom Price
Coordenador do CPGCC: Prof. Dr. José Palazzo Moreira de Oliveira
Bibliotecária - Chefe do Instituto de Informática: Zita Prates de Oliveira

AGRADECIMENTOS

Com efeito, durante os três anos transcorridos desde o meu ingresso no Curso de Pós-Graduação em Ciências da Computação até o presente, conheci, trabalhei e mantive relações de amizade com diversas pessoas, que muito auxiliaram-me nesta caminhada. Particularmente, gostaria de destacar e agradecer às seguintes pessoas:

- Professora Liane M. R. Tarouco, pela orientação técnica, conselhos e ações práticas efetivadas com o intuito de oferecer as condições materiais para que eu pudesse realizar o meu trabalho, disposição esta demonstrada desde o momento em que eu era seu auxiliar de pesquisa;

- Professora Rosa Maria Viccari e ao doutorando Sérgio Antônio Andrade de Freitas, pelos conselhos e experiências a mim transmitidas na área de inteligência artificial;

- Bolsistas de iniciação científica Ana Carla Martins Vidor e Sandro Rafaelli, os quais me auxiliaram no desenvolvimento do Trabalho Individual, onde foram traçadas as diretrizes básicas que nortearam o trabalho desenvolvido nesta dissertação;

- Maria Teresa Oliveira Severo, bolsista de iniciação científica, pelo auxílio fundamental durante o desenvolvimento do trabalho individual e nos estudos e leituras relacionadas com a dissertação;

- Cristina Melchiors, a quem foi confiada a tarefa de implementar a interface gráfica do protótipo desenvolvido como parte da dissertação, atividade em que ela demonstrou dedicação e competência;

- Rosane Beatriz Oliveira Severo Torres, pela amizade sincera e companheirismo por nós compartilhados ao longo desses três anos em que temos sido colegas e amigos, além de sermos parceiros de estudos e trabalhos;

- Agradeço aos meus familiares: Francisco (pai), Elza (mãe)e José Francisco (irmão), por terem me acompanhado em toda a minha trajetória, não apenas como estudante, mas, principalmente, como homem;

- Gostaria de agradecer a todos os verdadeiros amigos que eu encontrei ao longo da minha vida, que, embora tenham sido poucos, certamente foram sinceros e dedicados;

- Finalmente, gostaria de agradecer àquelas pessoas que tentam obstaculisar-me, motivados por seu preconceito cego e pérfido, àqueles que sob o manto da solidariedade ocultam sentimentos vis, pois eles, com seus pensamentos e atitudes, fazem-me lutar com maior energia e convicção em busca de meus objetivos.

SUMÁRIO

LISTA DE ABREVIATURAS

ADS	> Árvore de Derivação Sintática
APDU	> Application Protocol Data Unit
ARDP	> Asynchronous Reliable Delivery Protocol
DCG	> Descritive Clauses Grammar
DRS	> Discourse Representation Structure
DRT	> Discourse Representation Theory
FS	> front-end SABI
FTP	> File Transfer Protocol
HTTP	> Hypertext Transfer Protocol
IG	> Interface Gráfica
ISO	> International Organization for Standardization
MIG	> Módulo da Interface Gráfica
MIS	> Módulo da Interface SABI
MPA	> Módulo do Processo Archie
MPFTP	> Módulo do Processo FTP
MPIS	> Módulo do Processo Interpretador de Sentenças
NP	> Nome Próprio
OSI	> Open Systems Interconnection
PA	> Processo Archie
PIS	> Processo Interpretador de Sentenças
PN	> Pronome
RPC	> Remote Procedure Call
S	> Sentença
SA	> Servidores Archie
SABI	> Sistema de Automação de Bibliotecas
SCE	> Servidor de Correio Eletrónico
SN	> Sintagma Nominal
SRD	> Sistema de Representação de Diálogos
SV	> Sintagma Verbal
TCP	> Transmission Control Protocol
UDP	> User Datagram Protocol
UDPA	> Unidade de Dados do Protocolo de Aplicação
URL	> Uniform Ressource Locator
V	> Verbo
WAIS	> Wide-Area Information Servers
WWW	> World-Wide Web
XDR	> External Data Representation

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A proposta do trabalho descrito detalhadamente neste texto é implementar um sistema inteligente, que seja capaz de auxiliar os seus usuários na tarefa de localizar e recuperar informações, dentro da rede Internet.

Com o intuito de alcançar o objetivo proposto, construímos um sistema que oferece aos seus usuários duas formas distintas, porém integradas, de interfaces: língua natural e gráfica (baseada em menus, janelas etc). Adicionalmente, a pesquisa das informações é realizada de maneira inteligente, ou seja, baseando-se no conhecimento gerenciado pelo sistema, o qual é construído e estruturado dinamicamente pelo próprio usuário.

Em linhas gerais, o presente trabalho está estruturado logicamente em quatro partes, a saber:

1. Estudo introdutório dos mais difundidos sistemas de pesquisa e recuperação de informações, hoje existentes dentro da Internet.

   Com o crescimento desta rede, aumentaram enormemente a quantidade e a variedade das informações por ela mantidas, e disponibilizadas aos seus usuários. Concomitantemente, diversificaram-se os sistemas que permitem o acesso a este conjunto de informações, distribuídas em centenas de servidores por todo o mundo. Nesse sentido, com o intuito de situar e informar o leitor a respeito do tema, discutimos detidamente os sistemas Archie, gopher, WAIS e WWW;

2. Estudo introdutório a respeito da Discourse Representation Theory (DRT). Em linhas gerais, a DRT é um formalismo para a representação do discurso que faz uso de modelos para a avaliação semântica das estruturas geradas, que o representam. Por se tratar de um estudo introdutório, neste trabalho discutiremos tão somente os aspectos relativos à representação do discurso que são propostos pela teoria, dando ênfase à forma

de se representar sentenças simples, notadamente aquelas de interesse do sistema;

3. Estudo detalhado da implementação, descrevendo cada um dos processos que formam o sistema. Neste estudo são abordados os seguintes módulos:

   - Processo Archie: módulo onde estão implementadas as facilidades que permitem ao sistema interagir com os servidores Archie;
   - Processo FTP: permite ao SDIP recuperar arquivos remotos, utilizando o protocolo padrão da Internet FTP;
   - Front-end e Interface SABI: possibilitam a realização de consultas bibliográficas ao sistema SABI, instalado na Universidade Federal do Rio Grande do Sul;
   - Servidor de Correio Eletrônico: implementa uma interface alternativa para o acesso ao sistema, realizado, neste caso, por intermédio de mensagens do correio eletrônico;
   - Interface Gráfica: oferece aos usuários um ambiente gráfico para a interação com o sistema;
   - Processo Inteligente: Módulo onde está implementada a parte inteligente do sistema, provendo, por exemplo, as facilidades de interpretação de sentenças da língua portuguesa.

4. Finalmente, no epílogo deste trabalho, mostramos exemplos que ilustram a utilização das facilidades oferecidas pelo ambiente gráfico do SDIP.

Descrevendo sucintamente o funcionamento do sistema, os comandos e consultas dos usuários podem ser formuladas de duas maneiras distintas. No primeiro caso, o sistema serve apenas como um intermediário para o acesso aos servidores Archie e SABI, oferecendo aos usuários um ambiente gráfico para a interação com estes dois sistemas. Na segunda modalidade, os usuários formulam as suas consultas ou comandos, utilizando-se de sentenças em língua natural. Neste último

caso, quando se tratar de uma consulta, o sistema, utilizando-se de sua base de conhecimento, procurará aperfeiçoar a consulta efetuada pelo usuário, localizando, desta forma, as informações que melhor atendam às necessidades do mesmo.

PALAVRAS-CHAVE: Linguagem Natural, Representação do Discurso, Representação do Conhecimento, Internet, Archie, Gopher, WAIS, WWW

TITLE: "SDIP: AN INTELLIGENT SYSTEM TO DISCOVER INFORMATION ON THE INTERNET"

ABSTRACT

The proposal of the work describe detailedly in this master dissertation is to implement an intelligent system that will be capable of to help of its users in the task of locate and retrieve informations, inside of the Internet.

With the object of reach this goal, was builded a system that offer to its users two distincts types, however integrated, of interfaces: natural language and graphic (based in menus, windows, etc). Furthermore, the search of the informations is realized of intelligent way, based it in the knowledgement managed by system, which is builded and structured dinamically by the users.

In general lines, the present work are structured logically in four parts, which are listed below:

1. Introdutory study of the most divulgated systems of search and retrieval of informations, today existent inside of the Internet. With growth of this net, increase greatfull the quantity and variety of the informations keeped and published for users by it. Beside it, has appeared to many systems that allow the access to this set of informations, distributed on hundreds of servers in the whole world. In these sense, with the intuit of situate and to inform the reader about the subject, we describe formally the systems archie, gopher, WAIS and WWW , respectively;

2. An Introdutory study of the Discourse Representation Theory (DRT). In this work, the DRT is the formalism utilized for the representation of the discourse that uses models to evaluate semanticly the structures generated, which represent it. In fact, we will discusse in this work so

only the aspects relatives to discourse representation that are purposes by theory, given emphasis for the way to represent simple sentences, notory those recognized and important for the system ;

3. Detailed study of the implementation, describing each of the process that compose the system. In this study are described the following modules :

   - **Archie Process**: Module where are implemented the facilities that allow the system to interact whit the Archie Servers in the Internet;
   - **FTP Process**: it allows the **SDIP** to retrieve remote files, utilizing the standard protocol of the Internet, called FTP (File Transfer Protocol);
   - Front-end and Interface SABI: these components are used by system to realize bibliographic queries to SABI manager, installed at Universidade Federal do Rio Grande do Sul;
   - **Eletronic Mail Server**: it implements an alternative interface to access **SDIP**, realized in this case, throught eletronic mail messages, which transport firstly the user's query and secondly the system's response;
   - **Graphic Interface** : it offers to the users a graphical environment for the interaction with the system ;
   - **Intelligent Process**: module where are implemented the intelligent part of the system, providing, for instance, the facilities for interpretation of sentences wrote in portuguese language.

4. Finally, in the epilogue of this work, we show samples that illustrate the utilization of the facilities implemented at **SDIP**'s graphical environment.

Describing the functionability of the system, the users's commands and queries could be formulated of two disctincts ways. In the first case, the system serves only as the intermediary for the access to Archie servers and SABI, offering for its

users a graphical environment for the interaction with these two others systems. In the second modality, the users formulate their queries or commands, utilizing sentences in natural language. In this last case, when it is a query, the system utilizing its base of knowledgement, will try to refine the user's question, localizing the set of information that better satisfies his needs.

KEYWORDS: Natural Language, Discourse Representation, Knowledge Representation, Internet, Archie, Gopher, WAIS, WWW

1 INTRODUÇÃO

1.1 Contexto

Fazendo-se uma análise do desenvolvimento da tecnologia de processamento da informação, considerada em uma perspectiva histórica, observamos que a informática, desde seu gênese há cinqüenta anos atrás, até os nossos dias, foi uma das áreas do conhecimento humano que mais se desenvolveu. Equipamentos que ontem tomavam uma sala, hoje são substituídos por outros que podem ser colocados sobre uma pequena mesa. Tecnologias que anteriormente tinham um custo de milhões de dólares, ficando restritas a grandes conglomerados industriais, financeiros e comerciais, sofreram uma drástica redução de seus custos, o que permite, até mesmo a uma pessoa física, adquirir e manter um computador pessoal em sua residência. Velocidade de processamento, bem como a capacidade e performance das memórias e dos meios de armazenamento secundários, constituem-se em barreiras que a todo o momento são superadas.

Como uma conseqüência direta da realidade esboçada acima, verificou-se uma rápida difusão da tecnologia de informática em todos os setores da vida social e econômica, atingindo indistintamente universidades, escolas, empresas etc. Adicionalmente, verificou-se uma modificação no perfil das pessoas que utilizam freqüentemente um computador. Assim, um público que anteriormente era constituído apenas por especialistas (engenheiros, físicos, matemáticos etc), presentemente é integrado por usuários não intimamente ligados à área de informática, como é o caso, por exemplo, dos professores primários e secundaristas.

Outro setor que nas últimas décadas, ao lado da informática, experimentou um grande desenvolvimento e expansão foi a tecnologia (hardware e software) de redes de computadores, atingindo fundamentalmente as redes locais e metropolitanas, e, em menor escala, as redes de longa distância.

Este fato fez com que a visão de conceber os computadores como máquinas isoladas umas das outras, constitua-se hoje em um conceito há muito superado. Desta forma, os computadores são interligados por redes locais, metropolitanas ou de longa distância, constituindo-se em usuários e provedores de serviços, cada vez mais conplexos e sofisticados.

Um ambiente real, onde podemos observar a aplicação prática dos conceitos e avanços acima descritos é a rede Internet [COM 88, COM 91, KRO 92, TAN 89]. Esta rede, criada no início da década de setenta, tem se expandido enormemente nos últimos anos, congregando atualmente milhões de usuários em todo o mundo.

Um dos mais importantes aspectos que deve ser enfatizado a respeito da Internet não é, absolutamente, o número de usuários que estão a ela conectados atualmente, mas sim a enorme variedade e quantidade de informações e serviços, a eles disponibilizados por esta rede. Particularmente, no que concerne às informações, centenas de organizações (universidades, institutos de pesquisa, empresas etc) colocam em certos computadores que estejam conectados a Internet, denominados servidores, entre outros, documentos, artigos e pacotes de software, permitindo o acesso público a este conjunto de materiais. Sendo assim, é possível para uma pessoa que tenha acesso a esta rede internacional, recuperar e ler um artigo científico, antes mesmo que este seja apresentado em um congresso ou publicado em um periódico.

Não obstante as vantagens óbvias advindas da existência e disponibilidade deste conjunto de informações em uma rede internacional, a quantidade, variedade e distribuição das mesmas em centenas de servidores constitui-se em um aspecto complicador, e muitas vezes desestimulante, para os usuários que desejam localizá-las e recuperá-las, principalmente àqueles não especializados na área de informática.

Desta forma, os usuários que desejarem localizar e recuperar eficientemente informações, dentro da rede Internet, devem estar familiarizados com as diversas ferramentas de software que foram especialmente desenvolvidas para este fim. Contudo, caso estas ferramentas não forneçam as respostas esperadas, fato que

é bastante comum, os usuários devem estar preparados para trilhar um caminho árduo, que inicia pela determinação dos servidores nos quais residem as informações desejadas.

1.2 O Trabalho Desenvolvido

Como destacamos anteriormente, a rede Internet constitui-se presentemente em um vasto repositório de informações e de serviços, havendo diversas ferramentas de software distintas, desenvolvidas especialmente com o intuito de facilitar, conforme o caso, a utilização, localização ou recuperação de tais recursos[1].

Ao mesmo tempo, um grande número de usuários não especializados em informática têm acesso a Internet. Contudo, estas pessoas, em muitos casos, sentem-se desestimuladas a utilizar efetivamente os recursos providos pela rede, privando-se dos benefícios oferecidos por esta nova tecnologia de difusão da informação. Esta realidade deve-se, fundamentalmente, à dificuldade que estes usuários apresentam para adaptar-se às diversas formas existentes de se localizar o recurso desejado.

Tendo em vista o contexto acima descrito, o protótipo enfocado nesta dissertação insere-se na categoria dos sistemas que auxiliam os usuários a localizar e recuperar informações dentro da rede Internet. No entanto, o *SDIP* (Smart Discovery Information Program) difere fundamentalmente dos demais sistemas existentes, pelo menos em dois aspectos:

Em primeiro lugar, o *SDIP* utiliza os serviços providos por outros sistemas para a realização de sua tarefa, funcionando como um intermediário entre os usuários e as diversas ferramentas hoje existentes.

Finalmente, a pesquisa das informações é realizada inteligentemente, isto é, baseada na sentença em língua natural, formulada pelo usuário, e em bases de

[1] O termo recurso é usado neste trabalho para designar conjuntamente informações e serviços.

informações e de conhecimento, gerenciadas pelo *SDIP* e construídas pelos próprios usuários do sistema.

Em linhas gerais, os usuários podem interagir com o sistema de duas maneiras distintas: ambiente gráfico e mensagens de correio eletrônico.

No primeiro caso, o usuário interage com o sistema, utilizando-se de uma ambiente gráfico, especialmente desenvolvido para este fim. Por intermédio desta interface, podem ser formuladas consultas ou enviados comandos para o sistema, utilizando-se as próprias estruturas existentes no ambiente gráfico (menus, janelas etc), ou através de sentenças em língua natural.

Neste último caso, o sistema interpreta o significado semântico das sentenças recebidas, usando para este fim o formalismo proposto na Discourse Representation Theory (DRT) [KAM 88, KAM 90], e, posteriormente, executa as operações implicitamente expressas na sentença original.

Por outro lado, quando os usuários interagem com o sistema através das demais estruturas presentes no ambiente gráfico, eles estarão usando diretamente os próprios sistemas existentes na Internet, havendo tão somente uma diferença no desenho da interface utilizada em relação às originais.

Destaque-se ainda que o ambiente gráfico permite aos usuários do sistema construir a base de conhecimento, que será utilizada para a localização inteligente das informações.

Finalmente, podem ser enviadas ao sistema consultas bibliográficas, expressas em língua natural, dentro de mensagens do correio eletrônico. Neste caso, após interpretar a sentença, o *SDIP* obtêm os resultados desejados, encaminhando-os aos usuários, também em uma mensagem do correio eletrônico.

1.3 Limitações da Implementação

Visando tornar factível a implementação do protótipo, respeitando-se os prazos impostos para o desenvolvimento deste trabalho, foram fixadas algumas restrições, que melhor deliniam o sistema aqui enfocado. Dentre estas, merecem destaque:

- Não utilização de toda a potencialidade inerente ao formalismo proposto pela DRT [KAM 88, KAM 90];
- O conhecimento gerenciado pelo sistema somente pode ser construído a partir de um conjunto pré-definido de relações, ficando restrito a um domínio e complexidade específicos;
- O protótipo não está capacitado a interagir com todos os sistemas de pesquisa e recuperação de informações, hoje existentes dentro da Internet;
- Não há garantia de que em todos os casos possíveis, o sistema interpretará corretamente uma sentença, podendo, em algumas situações, não realizar as operações desejadas.

1.4 Estrutura do Trabalho

Visando abordar de maneira lógica e seqüencial todos os conceitos importantes, inerentes ao trabalho aqui descrito, estruturou-se esta dissertação da forma que segue:

Como tivemos a oportunidade de destacar anteriormente, dentro da rede Internet existem presentemente uma série de sistemas voltados à pesquisa e recuperações. Nos capítulos de 2 a 5, abordamos quatro destes sistemas, respectivamente: *Archie* [EMT 91, EMT 92, SCH 92, KRO 92], *Gopher* [SCH 92, ANK 93], *WAIS* [KAH 91, DAN 91, SCH 92, KRO 92, FUL 94] e *WWW* [KRO 92, BER 92, BER 92a, BER 93, BER 94, SCH 92], destacando seus conceitos e características

principais. Todavia, o objetivo central desta exposição é fazer com que o leitor compreenda, baseando-se em argumentos concretos, os motivos que nos levaram a escolher um determinado sistema, preterindo outros, para integrar o **SDIP**.

No capítulo 6, descrevemos o formalismo por nós adotado para a representação do significado semântico de sentenças, a Discourse Representation Theory (DRT) [KAM 88, KAM 90], dando ênfase àquelas de particular interesse para o sistema. Os conceitos aqui explorados são fundamentais para o bom entendimento acerca da implementação, sobretudo em um sistema que oferece a seus usuários uma interface em língua natural.

No capítulo 7, descrevemos o protótipo implementado em seus aspectos mais genéricos, abordando-o como um conjunto integrado por vários processos.

A descrição detalhada de cada um dos módulos que formam o *SDIP* é realizada nos capítulos de 8 a 13.

No capítulo 14 está redigido um pequeno manual introdutório ao uso do sistema, destinado principalmente aos usuários iniciantes.

Finalmente, no capítulo 15, tecemos alguns comentários a respeito do trabalho desenvolvido nesta dissertação, procurando destacar as perspectivas de trabalhos e pesquisas futuras por ele engendradas.

2 O SISTEMA ARCHIE

2.1 Introdução

Neste e nos três capítulos subseqüentes, procuraremos familiarizar o leitor com os quatro sistemas de pesquisa e recuperação de informações mais populares junto à comunidade da Internet. Acreditamos que a partir das informações aqui levantadas, o leitor será capaz de discernir claramente as características, particularidades, diferenças e, principalmente, os objetivos que estiveram por trás dos projetos que originaram cada um dos quatro sistemas aqui focalizados.

Adicionalmente, este conjunto de quatro capítulos procura destacar o grande acervo de informações hoje existentes dentro da Internet, distribuído em centenas de computadores em dezenas de países, bem como a tendência mundial hoje verificada de que haja um aumento na quantidade, qualidade e variedade dos acervos e serviços existentes.

Tais perspectivas de expansão tornam evidente a necessidade de que, cada vez mais, sistemas sejam desenvolvidos com o intuito de auxiliar os usuários na tarefa, que a cada dia torna-se mais difícil e complexa, de localizar as informações por ele desejadas, em uma rede similar a Internet.

2.2 Visão Geral

No capítulo introdutório deste trabalho 1, relatamos sucintamente a grande expansão experimentada pela rede Internet [COM 88, TAN 89, COM 91, KRO 92], desde sua implantação em meados da década de setenta, até os nossos dias.

Concomitantemente com a expansão da Internet, ampliaram-se também a variedade e quantidade de serviços e informações colocadas à disposição dos usuários

da rede. Ademais, verificou-se, em muitos casos, que o mesmo serviço ou informação é tornado disponível aos usuários em inúmeros locais, disseminados por toda a rede.

Um exemplo bastante ilustrativo desta classe de serviços que experimentaram um grande impulso dentro da Internet, disseminando-se rapidamente por toda a rede, foi o acesso público a arquivos através de "anonymous file transfer" [DEU 94].

Segundo estimativas levantadas em 1990, havia no mundo cerca de novecentos e cinqüenta servidores que proviam este serviço aos usuários da Internet [EMT 91].

Em virtude da magnitude do número de servidores hoje existentes, o qual se supõe tenha ultrapassado a barreira do primeiro milhar, surgem duas questões fundamentais, cuja resposta adequada tornaria viável ou não uma melhor utilização deste serviço, possibilitando sua expansão, opondo-se à estagnação e provavelmente uma redução que adviria no caso de respondê-las inapropriadamente.

1. Quais os endereços dos servidores que provêm este serviço dentro da Internet?

2. Em qual ou quais servidores, entre tantos hoje existentes, o usuário poderá localizar a informação por ele desejada?

Com o intuito de responder adequadamente a essas questões, desenvolveu-se um sistema, denominado **Archie** [EMT 91, EMT 92, KRO 92, SCH 92], cuja função é prover aos usuários da Internet, uma forma de identificar os servidores FTP que guardam as informações por eles desejadas.

O **Archie** foi um sistema desenvolvido e implementado junto ao Departamento de Ciências da Computação da Universidade McGill no Canadá, a partir de 1991.

O objetivo principal desta aplicação é prover um serviço de pesquisa e localização de servidores FTP públicos e arquivos ou diretórios, capazes de satisfazer a uma consulta formulada por um determinado usuário do sistema. Desta maneira,

o componente do sistema, denominado servidor, gerencia uma base de dados onde estão registrados todos os servidores FTPs públicos conhecidos pelo administrador do sistema, bem como o nome de todos os arquivos e diretórios presentes nos mesmos.

O usuário realiza uma consulta ao sistema por intermédio de um cliente **Archie**, fornecendo-lhe um padrão de busca, representado por uma cadeia de caracteres, a qual é comparada com cada uma das entradas da base de dados anteriormente referida. Sempre que houver o casamento entre o padrão de busca com alguma das entradas da base de dados, o sistema responde ao usuário, informando-lhe o nome do arquivo ou diretório que satisfez a consulta, o nome do servidor FTP público que o contém, bem como a localização ("path") dessas informações dentro do mesmo.

É importante notar que para haver o casamento, o padrão de busca e a entrada da base de dados não precisam ser necessariamente idênticos. Neste particular, o sistema é bastante flexível, oferecendo aos usuários variadas alternativas de operação. Dentre estas, para exemplificar, podemos citar uma modalidade de consulta em que o padrão de busca é representado por uma expressão regular. Neste caso, se usássemos, por exemplo, o padrão de busca ("archie*"), seriam respostas válidas para a nossa consulta, entradas da base de dados que contivessem a cadeia de caracteres ("archie"), sucedida de quaisquer caracteres [1].

A atualização da base de dados, gerenciada por um servidor **Archie**, é realizada automaticamente pelo sistema. Em intervalos fixos de tempo, geralmente mensais, o servidor **Archie** contacta todos os servidores FTPs públicos por ele conhecidos, recuperando os nomes de todos os arquivos e diretórios neles contidos.

Para se ter uma idéia da dimensão do tráfego de dados gerado na Internet, graças à operação de atualização mensal da base de dados de um servidor **Archie**, basta considerarmos que a metade do tráfego de informações, entre To-

[1] Neste exemplo, o caractere '*' pode ser entendido como um símbolo genérico, associável a qualquer cadeia de caracteres.

ronto no Canadá com o resto da Internet, é gerado pelo servidor **Archie** instalado na Universidade McGill, quando ele realiza esta operação [EMT 91].

Presentemente, os usuários podem realizar consultas aos sistemas **Archie**, utilizando-se de três tipos distintos de interfaces:

1. Interativa;
2. Correio eletrônico;
3. Cliente **Archie** local.

Na primeira modalidade, o usuário conecta-se diretamente ao servidor **Archie**, usando o programa Telnet. A operação, após o estabelecimento da conexão, dá-se através de comandos interativos.

Na segunda forma, o usuário submete ao sistema comandos idênticos àqueles utilizados na operação interativa, diferindo desta no fato de que os comandos são enviados ao servidor no corpo de uma mensagem do correio eletrônico. Similarmente aos comandos, as respostas provenientes do sistema, também são encaminhadas ao usuário através do correio eletrônico.

Finalmente, na terceira modalidade de interação, o cliente **Archie** local submete a consulta do usuário a um servidor Prospero [NEU 89, NEU 89a, NEU 92, NEU 92a, NEU 92b], o qual realiza o trabalho de pesquisa sobre a base de dados, operação essa executada pelo próprio servidor Archie nas duas formas de interação, anteriormente citadas.

Esta última forma de operação é de particular interesse para o presente trabalho, uma vez que o protótipo aqui descrito interage com os servidores **Archie** através de um cliente, protocolo e servidores Prospero, especialmente adaptados para este fim.

O sistema Prospero [NEU 89, NEU 89a, NEU 92, NEU 92a, NEU 92b]foi um protótipo implementado como parte do trabalho de doutoramento de Clifford Neuman.

Era objetivo do pesquisador, propor o modelo e construir um protótipo, capaz de solucionar os problemas inerentes a um sistema de arquivos global. Em tais sistemas, os arquivos e diretórios são mantidos em diversos computadores, distribuídos em diferentes organizações. Desta forma, torna-se inviável a construção de um sistema de arquivos global, onde não ocorram, entre outros problemas, conflitos na forma de organizar e estruturar diretórios, multiplicidade de arquivos que apresentam a mesma denominação, diferindo, porém, em seu conteúdo de informação.

A solução proposta resolve estes e outros problemas associados à realidade acima descrita, criando um sistema de arquivos virtual, construído a partir de uma visão particularizada que cada usuário tem do sistema real. Desta maneira, cada usuário tem a possibilidade de constituir o seu próprio sistema de arquivos virtual personalizado, tendo como ponto de partida um sistema real, que existe fisicamente.

O sistema Prospero é formado por um conjunto de componentes, descritos brevemente a seguir:

- O conjunto de processos que implementam as funções de interface com o usuário do sistema;
- Protocolo de comunicação, também denominado Prospero, que permite a cooperação entre os processos clientes e os servidores;
- Processo servidor, responsável pela gerência dos vários sistemas de arquivo virtuais, mapeando-os para o sistema físico real.

Por motivos de performance, integraram-se servidores Prospero ao sistema Archie, sendo construídos clientes que implementam um subconjunto do protocolo original, basicamente com funções de pesquisa em diretórios.

Neste trabalho, não abordaremos detalhadamente o sistema Prospero. Uma descrição do subconjunto de funções do protocolo Prospero, usadas na interação dos clientes **Archie** com os servidores, será visto no capítulo referente à

implementação do Processo Archie 8, um dos componentes do protótipo desenvolvido como parte desta dissertação.

Para maiores informações a cerca do sistema Prospero, o leitor interessado deve consultar [NEU 89, NEU 89a, NEU 92, NEU 92a, NEU 92b].

Na figura 2.1, mostramos um exemplo das informações enviadas por um servidor **Archie**, como resposta a uma consulta formulada pelo usuário. O leitor pode verificar que,a partir das informações ali contidas, torna-se trivial o trabalho de localizar e recuperar informações armazenadas em servidores FTP públicos dentro da rede Internet.

Host ctron.com Location: /pub/General.Info/release.notes DIRECTORY drwxr-xr-x 512 Feb 11 20:39 fddi Location: /pub/apple.dni DIRECTORY drwxrwxr-x 512 Jan 14 18:16 fddi
Host madhaus.utcs.utoronto.ca Location: /doc/ietf DIRECTORY drwxr-xr-x 512 Mar 27 02:17 fddi
Host nic.ddn.mil Location: /ietf DIRECTORY drwxr-xr-x 512 Oct 16 02:58 fddi
Host pop.lbl.gov Location: / DIRECTORY drwxr-xr-x 512 Jun 2 1993 fddi

Figura 2.1 Exemplo de respostas enviadas por um servidor Archie

3 O SISTEMA GOPHER

3.1 Introdução

Dando prosseguimento ao nosso estudo a respeito dos mais populares sistemas de localização e recuperação de informações existentes na Internet, abordaremos neste capítulo o sistema **Gopher** [ANK 93, KRO 92], focalizando sua concepção de funcionamento e, fundamentalmente, o protocolo empregado na comunicação entre os processos cliente e sevidores.

O **Gopher** [KRO 92, SCH 92, ANK 93] foi um sistema inteiramente concebido e inicialmente implementado junto à Universidade do Estado de Minesota nos Estados Unidos. As primeiras versões do "software" – cliente e servidor –, apareceram na Internet a partir do início da década de noventa. Posteriormente, sucessivas versões foram sendo desenvolvidas, destinadas a diferentes plataformas computacionais, desde microcomputadores pessoais, até mesmo computadores de grande porte.

Logo de início, o sistema **Gopher** já obteve uma grande aceitação junto à comunidade de usuários da Internet. Em decorrência, o sistema rapidamente difundiu-se por todas as partes do mundo, verificando-se uma grande expansão no seu uso, com a instalação de centenas de servidores.

Conceitualmente, o **Gopher** foi projetado e implementado com o intuito de construir-se um sistema capaz de prover serviços distribuídos de entrega de documentos, os quais residem em servidores autônomos, dispersos por toda a rede.

Consideramos importante destacar que, neste sistema, o conceito de documento é bastante genérico, englobando também, por exemplo, a possibilidade de realizar-se pesquisas em textos indexados. Neste caso, é permitido ao usuário informar ao sistema uma ou mais palavras chave e este procurará documentos ou textos que, de fato, contenham estas palavras.

Adicionalmente, buscou-se simplificar ao máximo o protocolo de comunicação entre os clientes e os servidores, com o objetivo de que, até mesmo departamentos de uma universidade onde, em geral, encontram-se apenas computadores de pequeno porte, possam instalar e manter seus próprios servidores **Gopher**. Desta forma, facilitou-se ainda mais a disseminação desse sistema , não apenas em organizações, encaradas sob o ponto de vista macroscópico, mas também dentro das mesmas, atingindo as suas repartições e departamentos.

Não obstante os documentos e serviços estejam distribuídos em inúmeros servidores, tais informações não são mostradas aos usuários desta forma, poder-se-ia até mesmo dizer desordenadas, e sim, seguindo um modelo hierárquico, bastante semelhante a um sistema de arquivos convencional.

Segundo esta modelagem, cada entrada em um diretório pode ser um outro diretório, um arquivo (texto, imagem etc), ou então um item especial, como, por exemplo, pesquisa em textos indexados, anteriormente mencionada.

Para clarificar melhor a idéia que está por trás deste modelo informacional, consideremos o seguinte exemplo prático:

Em uma universidade, os departamentos de Ciências da Computação, Física e Matemática instalaram seus próprios servidores **Gopher**.

Neste momento, surge o problema de como ordenar estes e, possivelmente, outros servidores **Gopher**, de forma a apresentar aos usuários informações organizadas coerentemente.

Uma boa resposta a essa questão seria estruturar-se os servidores hierarquicamente. Neste exemplo, seria apropriado instalar-se um servidor **Gopher** principal, denominado um servidor raiz, que contivesse elos que possibilitassem aos usuários atingir a cada departamento individualmente.

Seguindo este modelo, os usuários que se conectassem ao servidor raiz, perceberiam três níveis hierárquicos, a saber:

1. Servidor principal;

2. Servidores instalados em cada departamento;

3. As informações propriamente ditas, constituindo-se nas folhas da árvore invertida.

É importante enfatizar que a estruturação seguindo o modelo de uma árvore invertida não é obrigatória. Nada impede, por exemplo, que um servidor secundário, instalado no Departamento de Física, contenha um elo que conduza a um servidor secundário de outro departamento. Desse modo, constata-se que este modelo informacional é melhor representado por um grafo não dirigido, ao contrário de uma árvore invertida.

A opção por organizar-se as informações seguindo um modelo semelhante a um sistema de arquivos convencional, deveu-se, segundo [ANK 93], a quatro fatores principais:

1. Muitos usuários estão familiarizados com estruturas arranjadas hierarquicamente. Diretórios contendo ítens, tais como, documentos, servidores e subdiretórios, são representações largamente usadas em muitos sistemas, como, por exemplo, "eletronic bulletin board" e nos demais "campus-wide information systems";

2. O tipo de estrutura hierárquica adotada em um sistema de arquivos, pode ser facilmente representável no que concerne a sua sintaxe. O modelo usado no sistema **Gopher** é de fácil compreensão, sendo projetado com o intuito de facilitar a depuração dos processos cliente e servidores. Pode-se, como veremos mais adiante, usar o programa TeoNet para simular um cliente **Gopher**, verificando as respostas enviadas pelos servidores;

3. À medida em que o projeto **Gopher** foi desenvolvido em um ambiente universitário, um dos objetivos do sistema foi oferecer aos departamentos

a possibilidade de publicar informações, usando até mesmo seus computadores de pequeno porte. Deste modo, a adoção de uma organização da informação seguindo o modelo proposto em um sistema de arquivos, minimiza o problema de desenvolver-se rapidamente servidores que residirão em plataformas computacionais de pequeno porte;

4. Os conceitos neste tipo de abordagem são facilmente extensíveis, bastando para tanto, associar-se a cada entrada de um diretório um determinado tipo, que definiria as características e comportamento daquele objeto. Esclarecendo melhor este conceito, no sistema **Gopher**, uma entrada de um diretório pode ser um objeto da classe arquivo, pesquisa indexada de documentos, entre outros. Esta característica é definida por um tipo associado a cada uma das entradas do diretório. No primeiro caso, o cliente sabendo que o objeto é um arquivo, apenas leria os "bytes", até recuperá-lo integralmente. No entanto, no segundo caso, o cliente deverá primeiramente submeter a consulta do usuário a um servidor para, somente depois, estar apto a recuperar os objetos de tipo arquivo, que são os resultados da consulta prévia, caso o usuário assim o solicite.

As figuras 3.1 e 3.2 ilustram a forma adotada pelo sistema **Gopher** para organizar as informações mostradas aos usuários. É importante notar que o "layout" da informação (diretórios, arquivos, imagens etc), ao contrário da estrutura, diferirá de computador para computador, uma vez que cada equipamento apresenta características de "hardware" e "software" singulares.

Na figura 3.1, mostramos o diretório principal do servidor **Gopher** instalado no Centro Nacional de Supercomputação da Universidade Federal do Rio Grande do Sul.

Já na figura 3.2, mostramos o diretório principal de outro servidor **Gopher** instalado na mesma universidade, o qual é ativado selecionando-se o item do menu ("UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL (UFRGS)"), apresentado no diretório raiz mostrado na figura 3.1.

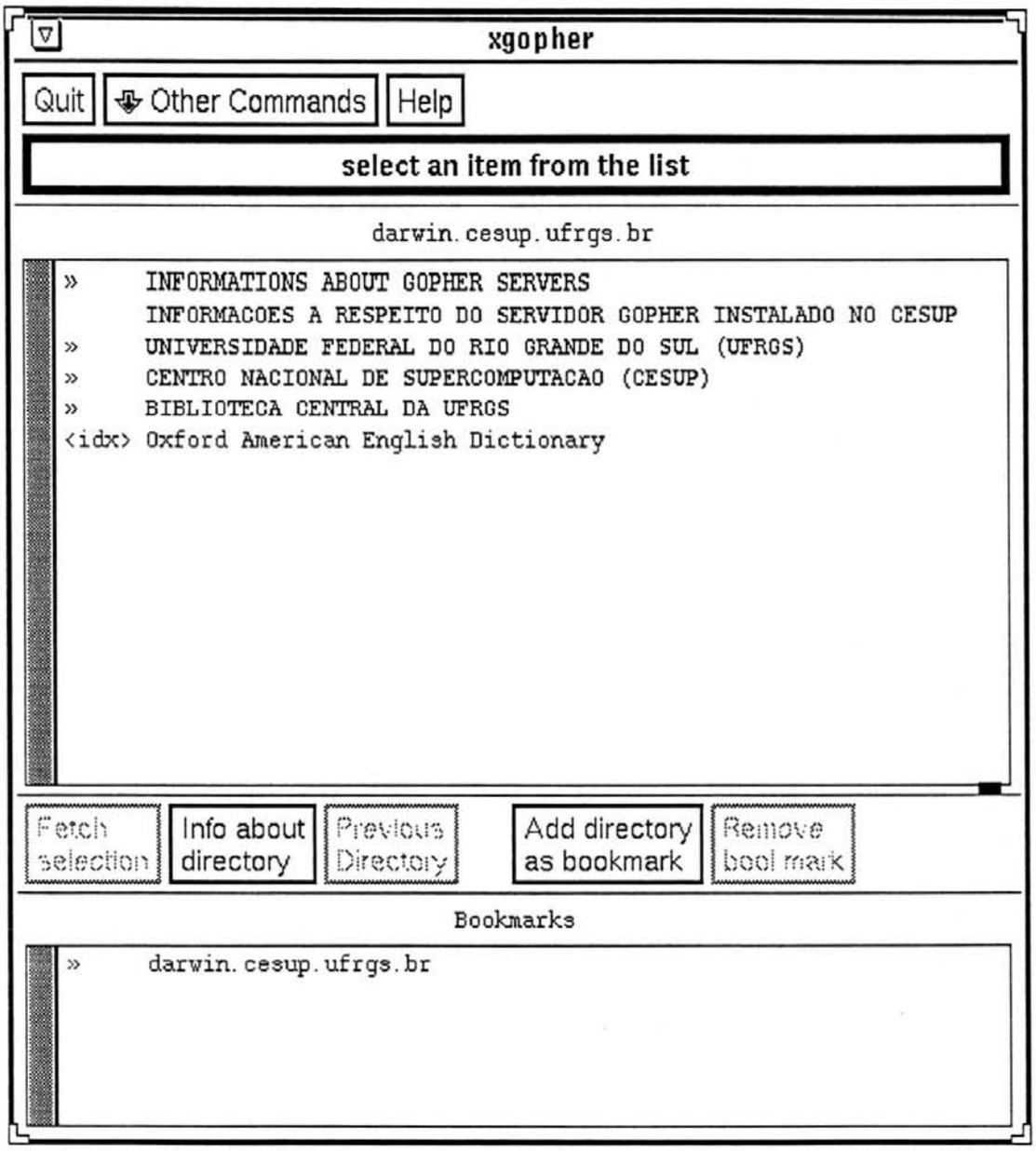

Figura 3.1 *Diretório principal do servidor gopher instalado no CESUP*

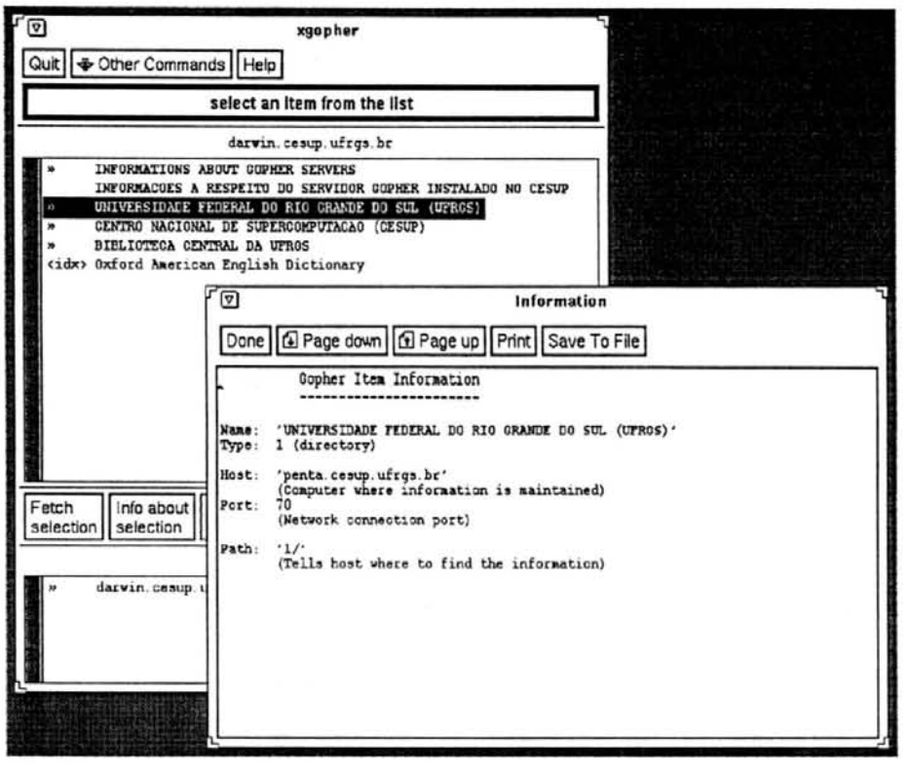

Figura 3.2 *Diretório raiz de outro servidor gopher instalado na UFRGS*

3.2 Protocolo de Comunicação

Nesta seção, descreveremos informalmente o protocolo de comunicação usado na interação entre os clientes e servidores **Gopher**. Para tanto, lançaremos mão de exemplos, onde mostraremos o conjunto de dados trocados entre os processos que se comunicam. Mais precisamente, apresentaremos a interação ocorrida entre o cliente e os servidores que está ilustrada nas figuras 3.1 e 3.2, anteriormente referidas.

Para uma descrição formal completa do protocolo **Gopher**, o leitor deve consultar [ANK 93].

O protocolo usado para troca de informações entre o processo cliente e os servidores **Gopher** é bastante simples, em consonância com os objetivos do sistema.

Essencialmente, neste protocolo, o cliente envia um seletor, podendo este ser vazio, ao processo servidor. Este, por sua vez, responde ao cliente, enviando-lhe

um diretório, documento ou ainda um conjunto de ítens, constituindo um diretório, resultante de uma pesquisa sobre documentos indexados.

O servidor **Gopher** não retém informações a respeito do andamento das transações executadas pelos clientes, caracterizando o protocolo de comunicação **Gopher** como "stateless". Deste modo, fica a cargo dos processos clientes manterem informações que lhes permitam, por exemplo, subir um nível ou mais na estrutura hierárquica da informação, retornar ao servidor principal etc.

Um documento pode ser um arquivo texto, binário, uma imagem, etc. Por outro lado, um diretório é composto por um conjunto repetitivo de linhas de texto, finalizadas por <CR>¡LF¡. O término de um diretório é marcado por uma linha de texto que contém apenas o caractere '.', seguido de um <CR>¡LF¡.

Os clientes **Gopher** devem conectar-se aos servidores utilizando-se da porta de número setenta (70), atualmente padronizada dentro da Internet.

Adicionalmente, toda a comunicação do sistema **Gopher** pressupõe a existência de um protocolo de transporte confiável[1], ou seja, o TCP (Transmission Control Protocol) [COM 88, TAN 89, COM 91], dentro da rede Internet.

Finalmente, resta-nos ainda detalhar a estrutura lógica de um diretório **Gopher**. Dissemos anteriormente, que um diretório é composto por um conjunto repetitivo de linhas de texto. Cada uma destas é, de fato, uma estrutura lógica composta de cinco campos. Os três campos intermediários, por serem de tamanho variável, são delimitados por um caractere <TAB>.

A figura 3.3 mostra a estrutura lógica de cada linha do diretório.

A seguir, detalhamos o significado de cada um dos campos:

[1]Simplificadamente, um protocolo de transporte confiável garante, aos níveis superiores, uma transmissão correta dos dados recebidos. Quando não for possível à entidade de transporte oferecer esta qualidade de serviço, o transmissor é informado através de um código de erro.

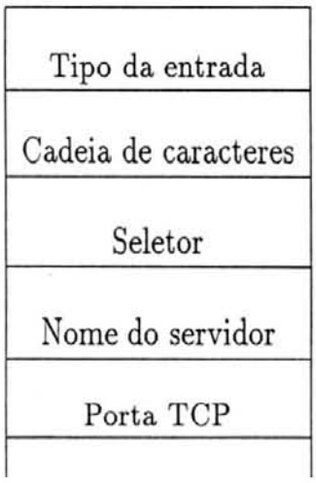

Figura 3.3 Estrutura lógica de uma linha do diretório Gopher..

- Tipo -> Qualifica a entrada. Por exemplo, caso este campo contenha o caractere '1', o cliente saberá que esta entrada corresponde a outro diretório **Gopher**.

  Na tabela 3.1 estão listados todos os valores possíveis para este campo, com os respectivos significados;

- Cadeia de caracteres visíveis -> Representa a informação que é mostrada aos usuários, constituindo-se em cada uma das opções que aparecem no menu;

- Seletor -> Informação que identifica univocamente um objeto dentro do servidor;

- Servidor -> Servidor **Gopher** associado à entrada. Para acessar o objeto definido pelo seletor, o cliente **Gopher** deverá conectar-se ao computador, cujo endereço Internet está indicado neste campo;

- Porta -> Número da porta onde o servidor associado à entrada aguarda as conexões.

A partir desse momento, passaremos aos nossos exemplos ilustrativos.

Tabela 3.1 Tipos de entradas permitidas em um diretório Gopher

Caracter	tipo	Observação
0	Arquivo	—
1	Item é um diretório	—
2	CSO (qi)	Phone-Book Server
3	Erro	—
4	Arquivo BinHexed	—
—	do Macintosh	—
5	Arquivo binário DOS	término da transmissão
—	—	marcado pelo fechamento da
—	—	conexão
—	—	por parte do servidor
6	Arquivo Unix	—
—	uuencoded	—
7	Seletor aponta para um	—
—	Index-Search server	—
8	item corresponde a uma	—
—	sessão TelNet	—
9	Arquivo binário	Final da transmissão
—	—	assinalado pelo fechamento da
—	—	conexão
-	—	por parte do servidor
+	Servidor reduntante	—
T	seção tn3270	—
g	Arquivo no formato GIF	—
I	Item é algum tipo de imagem	Cabe ao cliente definir
—	— a forma de exibí-la.	

Para obtermos as situações representadas nas figuras 3.1 e 3.2, são necessárias duas transações. Na primeira, o cliente contacta o servidor **Gopher** do Centro Nacional de Supercomputação, recebendo deste, o diretório principal. Em um segundo momento, após ter sido selecionada a opção ("UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL (UFRGS)"), o cliente utilizando-se das informações contidas nesta entrada do diretório, contacta o respectivo servidor, usando o número de porta indicado, enviando-lhe o seletor de objeto apropriado. Finalmente, o servidor contactado, responde ao cliente, transmitindo-lhe um novo diretório.

Na figura 3.4 está representada a primeira transação, onde o cliente genérico contacta o servidor **Gopher** do Centro Nacional de Supercomputação.

Na figura 3.5, mostramos a segunda transação, envolvendo um servidor **Gopher** alternativo, associado à entrada do diretório selecionada.

Finalmente, é importante destacar que o protocolo **Gopher** é "stateless", ou seja, os servidores não retêm nenhuma informação a respeito das transações após sua conclusão. Assim, cabe ao cliente gerenciar informações que permitam, por exemplo, ao usuário retroceder aos diretórios superiores.

- **Cliente estabelece uma conexão com (*darwin.cesup.ufrgs.br*), usando a porta setenta (70);**
- **Servidor aceita a conexão, porém aguarda a transmissão inicial do cliente;**
- **Cliente envia ao servidor uma linha vazia, solicitando o seu diretório principal;**
- **Servidor envia o diretório requerido;**

```
1INFORMATIONS ABOUT GOPHER SERVERS (TAB) 1/Information About Gopher
(TAB) gopher.tc.umn.edu (TAB) 70
0INFORMACOES A RESPEITO DO SERVIDOR GOPHER INSTALADO NO CESUP (TAB) 0/cesup-gopher
(TAB) darwin.cesup.ufrgs.br (TAB) 70
1UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL (UFRGS) (TAB)
(TAB) penta.ufrgs.br (TAB) 70
1CENTRO NACIONAL DE SUPERCOMPUTACAO (CESUP) (TAB) 1/cesup
(TAB) darwin.cesup.ufrgs.br (TAB) 70
1BIBLIOTECA CENTRAL DA UFRGS (TAB) 1/biblioteca
(TAB) darwin.cesup.ufrgs.br (TAB) 70
7Oxford American English Dictionary (TAB) 7/gopherservices/enquire.english
(TAB) uts.mcc.ac.uk (TAB) 70
.
```

- **Servidor e cliente fecham a conexão.**

Figura 3.4 Eventos ocorridos na primeira interação de um cliente com um servidor Gopher

- Cliente estabelece uma conexão com (*penta.ufrgs.br*), usando a porta setenta (70);
- Servidor aceita a conexão, todavia aguarda uma ação por parte do cliente;
- Cliente envia um seletor vazio, solicitando ao servidor o envio do diretório principal;
- Servidor envia o seu diretório principal;

1O Instituto de Informatica (TAB) 1/instituto_de_informatica (TAB) penta.ufrgs.br (TAB) 70
1Biblioteca (TAB) 1/biblioteca (TAB) 143.54.1.20 (TAB) 70
1Chasque (TAB) 1/chasque (TAB) penta.ufrgs.br (TAB) 70
1Gopher da Informatica (TAB) (TAB) caracol.inf.ufrgs.br (TAB) 70
1Gopher da RNP (TAB) (TAB) gopher.rnp.br (TAB) 70
1Gopher do CNPq (TAB) (TAB) rnpdf2.cnpq.br (TAB) 70
1Rede TCHE (TAB) 1/redetche (TAB) penta.ufrgs.br (TAB) 70
1Secr.Ciencia.Tecnologia (TAB) 1/Secr.Ciencia.Tecnologia (TAB) penta.ufrgs.br (TAB) 70
1UFRGS (TAB) 1/UFRGS (TAB) penta.ufrgs.br (TAB) 70
.

- Cliente e servidor fecham a conexão.

Figura 3.5 Eventos ocorridos na segunda interação do cliente com um servidor

4 WIDE-AREA INFORMATION SERVER

4.1 Introdução

O sistema **WAIS** ("Wide-Area Information Servers") [KAH 91, DAN 91, SCH 92, KRO 92, FUL 94] foi desenvolvido com o intuito de prover um serviço capaz de auxiliar os usuários a localizar informações em um ambiente de grandes redes de computadores, onde este tipo de serviço torna-se quase que imprescindível em nossos dias.

Na concepção do projeto, fixaram-se quatro princípios ou linhas fundamentais, as quais o sistema deveria seguir. Adicionalmente, em todo o desenvolvimento posterior do sistema, buscou-se manter a fidelidade a tais preceitos. São eles:

1. Construir-se um sistema para pesquisa, visualização e difusão de informações, fundamentalmente capaz de operar sobre uma WAN ("Wide-Area Network");
2. Sistema baseado em padrões conhecidos;
3. Fácil de utilizar-se;
4. Flexibilidade e estar voltado à expansão.

O desenvolvimento do sistema principiou em outubro de 1989, com a primeira versão para a Internet surgindo somente em abril de 1991.

Desde o início do desenvolvimento do projeto **WAIS**, pensou-se em criar um sistema de arquitetura aberta, o qual usaria um protocolo padronizado para prover os serviços de comunicação entre os processos clientes e servidores. Desta maneira, optou-se pela adoção de um protocolo baseado na versão de 1988 do ("Z39.50 Information Retrieval Service Definition and Protocol Specifications") [NAT 88],

padrão nacional nos Estados Unidos, desenvolvido pela NISO (("National Information Standards Organization"). Ainda hoje, esta versão inicial do protocolo de comunicação é largamente usada dentro da Internet, pelos sistemas **WAIS** nela instalados. Recentes levantamentos dão conta de que, aproximadamente cinqüenta mil (50.000) usuários dentro da Internet, possuem em suas instalações clientes **WAIS** que implementam a referida versão inicial do protocolo de comunicação Z39.50.

A opção por usar-se um protocolo de comunicação, apenas baseado padrão Z39.50, não adotando-se integralmente sua especificação, deu-se em razão de que este protocolo foi desenvolvido fundamentalmente para serviços de pesquisa e recuperação de dados bibliográficos, o que, indubitavelmente, não satisfazia as necessidades de caráter mais genéricas inerentes ao sistema **WAIS**. Sendo assim, adotou-se um subconjunto do protocolo Z39.50-1988 original, agregando-lhe algumas características que vêm ao encontro das necessidades bem próprias do sistema **WAIS** [FUL 94], permitindo a pesquisa e recuperação de informações genéricas e não particularizadas.

Não obstante, estas características adicionais foram sendo incorporadas às versões posteriores do Z39.50, tornando-o um protocolo de comunicação aplicável genericamente à pesquisa e recuperação de informações de diversas categorias, não ficando restrito tão somente a dados bibliográficos.

4.2 Conceitos Básicos

A arquitetura do sistema **WAIS** é constituída por quatro componentes principais [KAH 91], a saber:

1. Cliente;
2. Servidor;
3. Base de dados;

4. Protocolo de comunicação.

O programa cliente é responsável pela interação com os usuários, procurando, em cooperação com os servidores, localizar e recuperar as informações por eles desejadas. De fato, todo o trabalho de pesquisa dentro do sistema **WAIS** é realizado pelo processo servidor, no qual reside uma ou mais bases de dados. O processo cliente funciona, tão somente, como um intermediário entre o processo servidor e o usuário do sistema, fornecendo a ele uma interface adequada a suas necessidades.

Finalmente, o protocolo baseado no Z39.50-1988 é usado para prover os serviços de comunicação entre o processo cliente e o servidor.

O sistema **WAIS**, como dissemos anteriormente, implementa duas operações básicas: Pesquisa e recuperação de informações.

Por outro lado, analisando-as de outra forma, cada uma delas desdobra-se em duas transações sucessivas.

Em um primeiro momento, o cliente envia uma requisição ao servidor, solicitando-lhe a realização de uma dada tarefa. Este, por sua vez, em um segundo momento, responde ao cliente, informando-lhe os resultados da operação requisitada e já concluída.

Na primitiva de solicitação de pesquisa ("Search Request"), o cliente informa ao servidor o nome das bases de dados sobre as quais ele deverá realizar a pesquisa, com a respectiva consulta formulada pelo usuário, a qual pode conter uma expressão em língua natural ou booleana e um conjunto de identificadores de documentos. Baseado neste conjunto de documentos relevantes para o usuário, caso eles tenham sido especificados, o servidor **WAIS** implementa a denominada facilidade de realimentação ("Feed-Back Facility"). Este serviço permite aos usuários, tendo selecionado um conjunto de documentos, solicitar ao sistema que localize outros documentos, similares em conteúdo, àqueles informados na requisição.

A resposta a uma solicitação de pesquisa ("Search Response") é composta por um conjunto de citações, onde cada uma é constituída de um pequeno resumo descritivo do documento a ela associado. Cada citação contém informações suficientes para que o usuário seja capaz de determinar claramente se é ou não interessante solicitar a recuperação do respectivo documento.

A requisição de recuperação ("Retrieval Request") contém basicamente um identificador de documento, designando inequivocamente seu nome e localização, permitindo, desta forma, que ele seja enviado ao usuário.

Finalmente, a resposta à solicitação de recuperação ("Retrieval Response"), constitui-se no próprio documento requisitado anteriormente.

4.3 Objetivos do Projeto

Nesta seção, aprofundaremos os conceitos envolvidos na implementação original do sistema **WAIS**, bem como o uso que este faz da versão de 1988 do protocolo de comunicação Z39.50. Para tanto, descreveremos os objetivos históricos do projeto, tecendo, para cada um deles, breves considerações concernentes às restrições e necessidades que devem ser supridas pelo protocolo Z39.50. Tais considerações serão melhor detalhadas na próxima seção, evitando-se, desta maneira, a confusão de conceitos.

1. Proporcionar aos usuários o acesso a informações bibliográficas e não-bibliográficas, inclusive textos e imagens.

   Em virtude do Z39.50-1988 ser orientado à pesquisa e recuperação de dados bibliográficos, tornou-se mister agregar-lhe alguns conceitos adicionais, capacitando-o a trabalhar com informações de caráter não-bibliográficas, nos mais variados formatos;

2. As formas de pesquisa devem ser simples, não sofrendo influências de alterações ocasionadas por modificações na funcionalidade dos servidores.

Clientes que implementam o protocolo Z39.50-1988 convertem a consulta do usuário, utilizando a notação da polonesa reversa, para uma consulta de tipo um (1)("Type-1 query"), associando a cada termo da expressão um conjunto de atributos bibliográficos. Particularmente para o sistema **WAIS**, foi implementado um novo tipo de consulta, denominada consulta de tipo três (3) ("Type-3 query"), na qual o cliente deixa de analisar e transformar a estrutura sintática da consulta do usuário. Adicionalmente, o cliente passa a não ter a responsabilidade de conhecer quais são os atributos bibliográficos reconhecidos pelos servidores;

3. O sistema deve prover facilidade de relevante "feed-back".

   Caso o usuário, no momento da formulação da consulta, especifique o conjunto de documentos relevantes, estes serão encaminhados ao servidor como parte integrante da mesma. Com este objetivo, incorporou-se esta facilidade a classe três (3) de consulta;

4. Os servidores **WAIS** devem ser capazes de operar no modo "stateless".

   No modelo proposto pelo protocolo Z39.50-1988, os servidores mantinham os resultados de uma consulta para um possível uso posterior por parte do cliente. Por conseguinte, todas as operações de recuperação seriam efetuadas sobre estes resultados. Por essa razão, para o sistema **WAIS**, foi necessário desenvolver-se uma outra alternativa, capaz de suportar o modo "stateless" de operação;

5. O sistema deve prover as facilidades necessárias para que o cliente possa recuperar seções ou partes de documentos.

   Em virtude do protocolo Z39.50-1988 implementar várias formas de recuperar partes de documentos, fez-se, dentro do modelo **WAIS**, uma série de restrições com o intuito de delimitar adequadamente a abrangência do protocolo de comunicação. A capacidade de trabalhar com partes de documentos foi estendida, adicionalmente, à facilidade de "feed-back" re-

levante. Neste caso, o conjunto de documentos relevantes para o usuário incorpora este novo tipo de informação;

6. Usar como protocolo de transporte de dados o TCP [COM 88, TAN 89, COM 91].

   O protocolo padrão Z39.50-1988 foi projetado para residir na camada de aplicação do modelo de referência OSI ("Open Systems Interconnection"), proposto pela ISO ("International Organization For Standardization"), usando os serviços de apresentação da camada inferior.

   Devido à popularidade dos protocolos TCP-IP e da Internet, o sistema **WAIS** foi projetado para operar sobre o TCP. Uma completa descrição do uso do protocolo Z39.50-1988 sobre o protocolo de transporte TCP, pode ser encontrada em [FUL 94].

4.4 O Protocolo de Comunicação do Sistema WAIS

O protocolo de comunicação usado no sistema **WAIS**, denominado (WAIS Protocol), implementa os serviços de inicialização e de pesquisa, propostos no padrão Z39.50-1988, onde cada um deles é composto por uma requisição, advinda do cliente, seguida de uma resposta, proveniente do processo servidor.

Com o intuito de tornar possível a construção de servidores que operam no modo "stateless", ambas as funções, consulta e recuperação, são implementadas usando-se apenas o serviço de pesquisa proposto no Z39.50-1988. Sendo assim, os servidores não precisam manter os resultados de uma consulta, para que seja possível realizar-se a posterior recuperação dos documentos por parte dos processos clientes.

Em virtude da requisição de serviço de pesquisa ("Search Service Request") do Z39.50-1988 conter somente a consulta, com seu respectivo tipo, o sis-

tema **WAIS** utiliza justamente a classe da consulta para diferenciar o serviço de pesquisa do serviço de recuperação de informações.

O serviço de pesquisa de informações é implementado usando-se uma consulta de tipo três (3) ("Type-3 query"), adicionada ao protocolo original.

Por outro lado, o serviço de recuperação de informações é implementado utilizando-se a consulta de tipo um (1) ("Type-1 query"), definida no próprio protocolo Z39.50-1988 original.

Uma pesquisa dentro do sistema **WAIS** é iniciada pelo processo cliente, com o envio de uma UDPA (Unidade de Dados do Protocolo de Aplicação) Z39.50-1988 de requisição de serviço de pesquisa ("Z39.50-1988 Search Service Request APDU ("Application Protocol Data Unit")"), usando uma consulta de tipo três (3) ("Type-3 query").

A consulta, basicamente, compõe-se de dois campos.

O primeiro constitui-se no conjunto de palavras ou expressões informadas pelo usuário. O segundo campo é composto por um conjunto de documentos ou partes de documentos, usados na facilidade de "feed-back" de informações relevantes para o usuário.

Por sua vez, cada uma das entradas associadas a um documento é composta por um identificador do documento, tipo, código de controle e ponto de demarcação de início e término.

O identificador e o tipo do documento designam, respectivamente, a localização e o formato do documento correspondente.

O código de controle denota a unidade de medida a ser considerada para os pontos de demarcação de início e término do documento. Exemplos de códigos de controle usados no sistema **WAIS**, incluem: byte, linha, parágrafo e documento completo. No último caso, os campos de delimitação de início e término são ignorados, posto que o objeto representa um documento integral.

Posteriormente, o processo servidor responde a consulta requisitada pelo cliente, enviando-lhe uma UDPA de resposta do serviço de pesquisa ("Search Service Response APDU"), contendo uma lista de registros ou citações **WAIS**.

Cada citação ou registro refere-se a um documento presente no servidor. Estas estruturas de informação estão organizadas nos seguintes campos:

- Um pequeno texto que sumariza o conteúdo do documento, permitindo ao usuário identificar o assunto principal nele abordado;
- O escore numérico do documento baseado em sua relevância para a consulta especificada pelo usuário. Este escore é normalizado, podendo alcançar o valor numérico máximo de mil (1000);
- Lista dos formatos nos quais pode-se encontrar o documento, como, por exemplo, Postscript, TIF, GIF, texto, etc;
- Identificador do documento;
- Tamanho do documento, expresso em número de bytes.

O número de citações **WAIS**, retornadas como resposta a uma consulta, será limitado pelo tamanho da mensagem, negociado pelos processos cliente e servidor, quando do serviço de inicialização.

A recuperação de um documento é iniciada por intermédio do envio de uma UDPA de requisição de serviço de pesquisa ("Search Service Request APDU"), usando uma consulta de tipo um (1) ("Type-1 query").

Este tipo de consulta pode conter um máximo de quatro campos, a saber:

1. Identificador do documento;
2. Formato do documento;
3. Indicador de posição inicial;

4. Indicador de posição final.

O identificador do documento é obtido a partir do conjunto de citações ou registros **WAIS**, recebidos pelo processo cliente como resposta a uma pesquisa ou consulta realizada previamente.

O formato do documento é selecionado pelo usuário, dentre aqueles indicados na respectiva citação.

Finalmente, os indicadores de marcação de início e término são usados para a recuperação parcial de documentos, delimitando-se os pontos de início e término da seção desejada. Estes valores, como dissemos anteriormente, podem ser expressos em termos de bytes, linhas, parágrafos, etc.

O servidor responde ao cliente, enviando-lhe o documento requisitado, por intermédio de uma UDPA de resposta do serviço de pesquisa ("Search Service Response APDU"). Neste caso, um único registro é retornado, contendo a informação desejada, no formato especificado pelo usuário.

Nesta seção descrevemos informalmente o protocolo envolvido na comunicação entre clientes e servidores **WAIS**. O leitor interessado em uma abordagem formal a respeito dos conceitos aqui apresentados, com uma especificação dos protocolos **WAIS** e Z39.50-1988, deve consultar [NAT 88, SCH 90, LYN 91, AME 92].

4.5 O Futuro do Sistema WAIS

Desde a primeira versão do sistema **WAIS**, a popularidade do protocolo padrão Z39.50-1988 [NAT 88] tem aumentado sensivelmente, como é evidenciado pela sua maior aceitação, seja nacional ou internacionalmente. Ademais, a representação junto ao comitê do Z39.50 foi expandida, englobando não apenas a comunidade dos bibliotecários, mas, também, agências governamentais, instituições de educação e o setor empresarial foram comtemplados.

Adicionalmente, surgiram uma série de padrões que vêm ao encontro das necessidades inerentes ao sistema **WAIS** e projetos semelhantes. Assim, este conjunto de padrões emergentes servem para complementar o Z39.50, eliminando as suas eventuais carências e lacunas. Para exemplificar este fato, poderíamos usar o padrão emergente URI ("Universal ressource Identifiers") [BER 94a] para representar os identificadores de documentos.

Este conjunto de novos padrões devem ser adotados rapidamente, para que sistemas de pesquisa e recuperação de informações, que operam em uma WAN, sejam usados cada vez mais, principalmente junto ao mundo empresarial, no qual algumas necessidades bem específicas, tais como segurança e contabilização de custos, devem ser plenamente satisfeitas.

Presentemente, está se trabalhando no desenvolvimento da próxima versão do sistema **WAIS**,totalmente baseado neste conjunto de novos padrões Emergentes.

A nova versão do sistema adotará, como base de seu protocolo de comunicação, o padrão Z39.50-1992 [AME 92], abandonando a versão aprovada em 1988 do mesmo protocolo. Esta nova versão do protocolo Z39.50 incorpora funcionalidades adicionais, implementadas no próprio protocolo **WAIS**.

Contudo, é importante destacar que os trabalhos em curso no desenvolvimento de um novo sistema **WAIS**, mantêm-se fiéis aos preceitos que nortearam o projeto original, os quais foram abordados na seção introdutória deste capítulo 4.1.

5 WORLD-WIDE WEB

Neste capítulo, apresentaremos o último sistema voltado à pesquisa e recuperação de informações, existente na rede Internet, estudado durante o período de desenvolvimento do presente trabalho. Adicionalmente, descreveremos informalmente o protocolo empregado na comunicação entre os processos clientes e servidores que constituem o sistema estudado.

5.1 Visão Geral

O **World-Wide Web**, ou simplesmente (**WWW**) [KRO 92, SCH 92, BER 92, BER 92a, BER 93, BER 94], foi o primeiro sistema que trouxe para a pesquisa e recuperação de informações as enormes e inegáveis vantagens advindas do uso extensivo de técnicas de hiperdocumentos. Na verdade, o pioneirismo desse sistema não reside propriamente na adoção destas técnicas de organizar as informações, uma vez que, antes dele, pelo menos um sistema as empregou, mas sim no fato de que o **WWW** trabalha sobre uma rede de longa distância, ao contrário dos demais que tinham como base de comunicação redes locais.

O **WWW** organiza as informações em um hipertexto distribuído, onde cada nodo pode estar associado à adversos tipos de objetos, tais como: textos, imagens, diretório de objetos, denominados "cover pages" e índices. Além disso, em alguns objetos estão implementadas funções que permitam a pesquisa dos documentos indexados, assemelhando-se, neste caso, ao sistema WAIS [KRO 92].

Analogamente aos demais sistemas discutidos anteriormente neste trabalho, a arquitetura do **WWW** segue o modelo cliente-servidor. Nesta arquitetura, todas as facilidades de interface com o usuário são implementadas pelo processo cliente. Ao servidor, no entanto, atribui-se a tarefa única e específica de gerenciar as informações e serviços que serão requisitados pelos processos clientes, verdadeiros intermediários entre os usuários do sistema e os servidores.

A comunicação entre estes dois componentes do sistema é realizada através de um protocolo especialmente desenvolvido, denominado HTTP ("Hypertext Transfer Protocol") [BER 93, BER 94], o qual necessita obrigatoriamente de um serviço de transporte de dados confiável, como é o caso do TCP [COM 88, TAN 89, COM 91].

Os clientes **WWW**, além de seu protocolo de comunicação próprio, também são capazes de reconhecer o protocolo de transferência de arquivo FTP ("File Transfer Protocol") [POS 85], bem como o empregado no sistema NEWS ("Network News Transfer Protocol (NNTP")) [KAN 86].

O protocolo FTP é usado com o intuito de acessar os servidores de arquivos públicos na Internet. Neste caso, os diretórios de arquivos são mostrados aos usuários como objetos do hipertexto.

Por outro lado, o reconhecimento do NNTP permite aos usuários do sistema acessar os grupos e ler artigos enviados ao sistema NEWS.

As informações dentro do sistema **WWW** estão organizadas segundo o modelo que as classifica em três agrupamentos lógicos distintos.

No primeiro grupo, as informações estão organizadas por assuntos. Nesta abordagem, uma entrada no diretório principal do sistema conduz o usuário à árvore de assuntos. Deslocando-se pelos nodos da árvore, é possível encontrar-se informações a respeito de diversas áreas do conhecimento humano, tais como: Astronomia, Ciências Biológicas etc. À medida que novos assuntos surgem, estes são inseridos em locais apropriados dentro da estrutura hierárquica, de forma que os usuários possam encontrá-las facilmente.

Nesse tipo de organização não é importante a classe do servidor onde reside a informação, mas sim em que assunto estas se enquadram. Desta forma, por exemplo, em um determinado nível da estrutura hierárquica, podem-se encontrar elos que conduzem a diferentes tipos de servidores de informação, no entanto podemos estar certos de que eles mantêm informações relativas ao mesmo assunto.

Em um segundo grupo, as informações estão classificadas pelo tipo de servidor aos quais elas são vinculadas. O diretório de objetos correspondente a esta classificação contém uma lista de todos os servidores com os quais o sistema **WWW** é capaz de estabelecer um diálogo. Assim, existem entradas para os sistemas Gopher [ANK 93], WAIS, News e o próprio **WWW**. Adicionalmente, nesse mesmo diretório existe uma entrada associada aos servidores de arquivos públicos, os quais são pesquisados com o auxílio do sistema Archie [KRO 92], discutido no capítulo 2 deste trabalho.

Finalmente, no terceiro agrupamento, as informações estão classificadas pelo tipo de organização que as mantêm. Contudo, esta é uma forma de classificação pouco usada dentro do sistema e, por esta razão, não será vista em maiores detalhes neste trabalho.

A utilização do sistema por parte dos usuários é bastante simples. Primeiramente, o usuário deve conectar-se a algum servidor, utilizando um cliente local. No momento em que for estabelecida a comunicação, o usuário visualizará a denominada página principal do servidor. A partir deste ponto, o uso do sistema consiste tão somente em selecionar elos que o conduzirão à informação desejada.

Quando, por exemplo, for necessário que o sistema localize documentos, de maneira similar ao WAIS, o **WWW** solicitará ao usuário que informe a chave de busca. Tão logo um ou mais documentos sejam localizados, o usuário terá a oportunidade de recuperá-los ou visualizá-los, bastando para tanto, selecionar o elo que conduz ao documento desejado.

Nas figuras 5.1 e 5.2, ilustramos o funcionamento do sistema **WWW**.

Na primeira figura 5.1, mostramos a página principal de um servidor **WWW** instalado no Centro Nacional de Supercomputação da Universidade Federal do Rio Grande do Sul. Para visualizar a mesma página principal, os usuários devem fazer com que seus clientes conectem-se a este servidor primário.

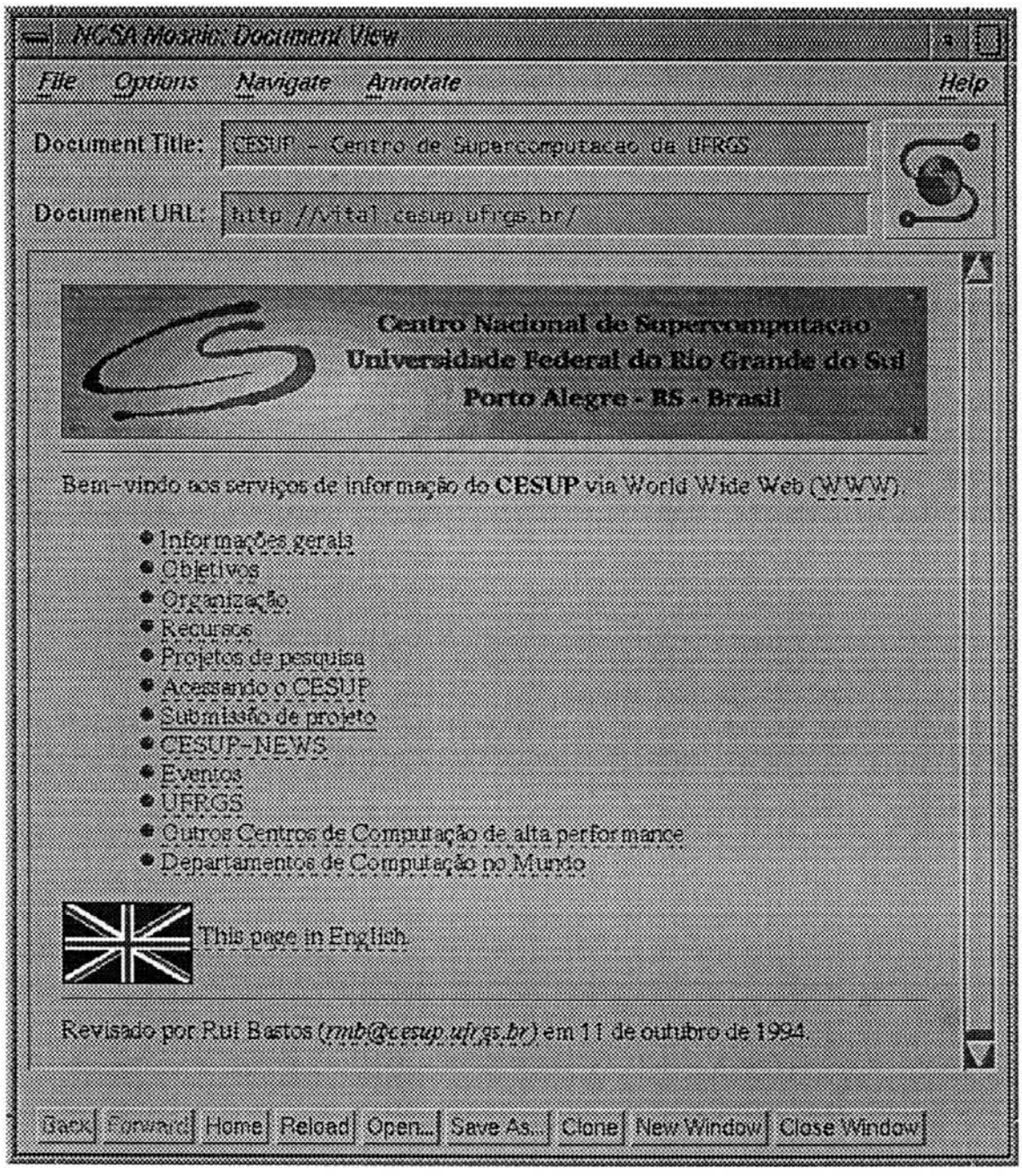

Figura 5.1 *Página principal do servidor* **WWW** *do CESUP*

A figura 5.2 mostra o resultado observado por um usuário que seleciona o elo (*Recursos*) e, posteriormente, o elo (*Equipe*), mostrando-nos a versatilidade deste sistema, visto que, além de textos, o **WWW** também é capaz de recuperar e reproduzir imagens de diversos formatos, dependendo evidentemente do tipo de cliente usado.

Figura 5.2 *Parte da equipe profissional do CESUP*

5.2 Protocolo de Comunicação

Para finalizarmos o nosso estudo a respeito do **World-Wide Web** e, conseqüentemente, concluirmos a análise dos mais difundidos sistemas de informação da Internet, resta-nos, ainda, explicar sucintamente as características do protocolo de comunicação que possibilita a implementação do sistema ora em estudo.

O HTTP ("Hypertext Transfer Protocol") [BER 93, BER 94]foi o protocolo especialmente desenvolvido para prover os serviços e facilidades requeridas por um sistema com as características e necessidades do **WWW**.

Não obstante o protocolo HTTP ser largamente usado nas atuais implementações do sistema, ainda hoje desenvolvem-se trabalhos e pesquisa objetivando aperfeiçoar a versão atual do protocolo, tendo-se, contudo, sempre em mente a necessidade de haver compatibilidade entre a nova e as antigas versões.

Para ser capaz de atender aos requerimentos inerentes às funcionalidades implementadas pelo sistema **WWW**, este protocolo de comunicação apresenta algumas características bastante singulares:

- Leveza e rapidez na transmissão de dados. Na seção anterior, dissemos que o **WWW** é o primeiro sistema que implementa técnicas de hipertexto sobre uma rede de longa distância. Neste particular, torna-se fundamental o uso de um protocolo que nos forneça excelentes taxas de transmissão, o que diminui a sobrecarga da rede e o tempo de espera do usuário, ao se deslocar de um documento para outro. Fundamentalmente, o HTTP é um protocolo que apresenta boa velocidade de transmissão, sendo, inclusive, mais eficiente que o próprio FTP.

- Orientação a objeto. Sistemas de informações reais devem implementar não apenas a capacidade de recuperar documentos, mas também funcionalidades adicionais que lhes permita maior versatilidade e flexibilidade, estendendo seu uso a áreas alternativas como, por exemplo, pesquisa de informações. O protocolo HTTP implementa essas funcionalidades por intermédio de um conjunto de métodos ou operações que são executadas sobre objetos identificados por um endereço, denominado URL ("Uniform Ressource Locator"). Adicionalmente, novos métodos podem ser criados e incluídos dentro do conjunto de métodos já existentes, ampliando-se, desta forma, a versatilidade e potencialidade do sistema.

Resumidamente, uma URL identifica univocamente um objeto dentro da Internet, designando:

- Tipo do servidor (Gopher, FTP, **WWW** etc) no qual reside o objeto;
- Endereço do servidor dentro da rede Internet;
- A localização do objeto dentro do servidor;
- O número da porta TCP que deve ser usada para contactar o servidor, caso este use uma porta não padronizada e, conseqüentemente, desconhecida pelo cliente.

O HTTP é fundamentalmente um protocolo "stateless", ou seja, o servidor não retém informações referentes a uma operação já finalizada, vedando-lhe a possibilidade de conhecer o estágio atual do diálogo mantido com o cliente.

Cada transação é constituída dos seguintes eventos:

1. Conexão. O cliente estabelece uma conexão TCP com o servidor, usando a porta de número oitenta (80), já padronizada na Internet. Contudo, outras portas não reservadas podem ser usadas, tendo-se então de especificá-las como uma das componentes presentes na URL;
2. Requisição. Envio, por parte do cliente, de uma mensagem de requisição;
3. Resposta. Envio, por parte do servidor, de uma mensagem de resposta à requisição do cliente;
4. Desconexão. Fechamento da conexão de transporte por ambos os processos participantes.

Como pode-se verificar a partir da transação acima descrita, no protocolo de comunicação HTTP existem tão somente dois tipos distintos de mensagem:

- Uma mensagem de requisição, onde o cliente solicita ao servidor que efetue uma determinada tarefa;
- Uma mensagem de resposta, enviada pelo servidor ao cliente, contendo os resultados efetivos da execução de uma dada operação.

Em ambos os casos, porém, as mensagens possuem o mesmo formato lógico, sendo compostas por um cabeçalho e um corpo.

O cabeçalho da mensagem é constituído por um conjunto de campos, organizados em uma seqüência de linhas de texto, finalizadas por dois caracteres("return" e "linefeed").

Por outro lado, o corpo da mensagem onde está contido o objeto é opcional, ou seja, sua presença ou ausência, na mensagem de requisição e/ou de resposta, dependerá da operação que for solicitada pelo cliente.

Assim, por exemplo, se o cliente desejar recuperar um dado objeto armazenado em um servidor, aplicando sobre ele o método (get), o objeto estará presente no corpo da mensagem de resposta e não no corpo da mensagem de requisição.

Todavia, caso o cliente queira armazenar um dado objeto em um determinado servidor, ele poderá fazê-lo, enviando este objeto como parte integrante de sua mensagem de requisição, solicitando ao servidor a aplicação do método (put) sobre o mesmo.

Dentre os campos presentes e um cabeçalho de uma mensagem de requisição, podemos destacar:

- Versão do protocolo HTTP que está sendo usada pelo cliente;
- Método a ser aplicado sobre o objeto identificado por uma URL;
- Identificação do objeto sobre o qual deve ser executada a operação;
- Informações utilizadas para a determinação dos custos financeiros referentes a aplicação do método solicitado.

O cabeçalho de uma mensagem de resposta contém campos que informam ao cliente os resultados obtidos com a execução da operação requisitada, no tocante a seu sucesso ou fracasso. No caso de insucesso, estas informações permitem ao cliente conhecer as razões que levaram a este resultado indesejado.

Além dessas informações, o cabeçalho de uma mensagem de resposta pode conter, entre outros, os seguintes campos:

- Conjunto adicional de requisições que são permitidas ao usuário realizar em relação ao objeto contido na mensagem de requisição original;
- Conjunto dos métodos aplicáveis ao objeto identificado na requisição original, cuja utilização pode ser solicitada por qualquer usuário;
- Data da criação do objeto;
- Formas de representação em que pode-se encontrar o objeto identificado na requisição original.

Na verdade, tanto na mensagem de requisição, quanto na de resposta, o número de campos existentes em ambos os cabeçalhos, excedem em muito os que foram aqui mostrados. No entanto, parece-nos que uma abordagem detalhada deste tema, além de alongar demasiadamente o capítulo que focaliza o sistema **WWW**, também estaria em desacordo com a idéia de fazer-se uma apresentação informal dos sistemas estudados, dispensando-se particularidades de implementação. Ademais, o sistema ora em estudo não está integrado ao protótipo desenvolvido neste trabalho, tornando desnecessário ater-nos a descrições pormenorizadas do protocolo de comunicação. Certamente, caso usássemos o sistema **WWW** para acessarmos informações dentro da Internet, teríamos de descrevê-lo detalhadamente, enfocando desde suas características mais gerais, até suas particularidades e idiossincrasias.

O leitor interessado em conhecer detalhadamente a versão atual do protocolo de comunicação HTTP, pode consultar [BER 94].

6 A TEORIA DA REPRESENTAÇÃO DO DISCURSO

6.1 Introdução

Nos capítulos precedentes, fizemos uma breve descrição a respeito dos mais importantes e populares sistemas de pesquisa e recuperação de informações, hoje existentes dentro da Internet. Acreditamos que, desta forma, conseguimos situar o leitor dentro do contexto no qual se insere o trabalho aqui apresentado.

No entanto, resta-nos ainda abordar um assunto crucial, sobretudo para um sistema que se propõe a oferecer aos usuários uma interface alternativa em língua natural, juntamente com uma interface gráfica convencional, qual seja, como representar e avaliar o significado semântico de uma ou mais sentenças da língua portuguesa que formam um determinado discurso.

Por conseguinte, devemos adotar um formalismo que nos permita interpretar o significado de um discurso, para posteriormente realizar ou não as ações nele expressas. Desta maneira, pode-se observar que somente algumas interpretações de sentenças terão relevância para o SDIP, ou seja, aquelas para as quais se possam associar uma ou mais ações exeqüíveis pelo sistema.

6.2 A DRT

A DRT ("Discourse Representation Theory"), proposta inicialmente por Hans Kamp [KAM 88, KAM 90], foi a metodologia formal por nós adotada no presente trabalho para a representação do significado semântico do discurso – sendo este composto por uma ou mais sentenças correlacionadas e coerentes – e como formalismo para a avaliação da representação gerada.

Segundo Freitas [FRE 93], não se encontra na literatura da área uma definição precisa a respeito do que vem a ser a DRT. Isto deve-se, ainda segundo Freitas [FRE 93], ao vulto que esta teoria tomou junto ao meio acadêmico sem a devida formalização de todos os seus conceitos, ou seja, devido a não formalização de todos os tipos de sentenças e operadores por ela utilizados.

Não obstante à ausência de uma definição precisa, a DRT foi o formalismo adotado no presente trabalho, uma vez que, para o conjunto de sentenças relevantes para o sistema, seus conceitos e definições estão claramente delineados.

Corroborando a posição defendida por Freitas [FRE 93], o próprio Hans Kamp, em seu trabalho [KAM 90], não define claramente o que vem a ser a DRT, situando-a entre dois extremos:

- De um lado, a DRT é caracterizada como uma teoria que faz previsões definidas a cerca das possibilidades das anáforas nominais e sobre as condições de verdade de certas classes de sentenças, nas quais os sintagmas nominais desempenham um papel fundamental;
- No outro extremo, Kamp concebe a DRT como sendo uma proposta geral para o significado lingüístico, na qual as características e requisitos das teorias baseadas em modelos da semântica formal são combinados com um conceito procedural, de modo a proporcionar um significado lingüístico correlacionado com a forma lingüística.

Adicionalmente, usaremos na continuidade deste capítulo, similarmente a Kamp, as DRSs ("Discourse Representation Structures") para a representação da estrutura semântica das sentenças.

A seguir, mostramos alguns exemplos de sentenças para as quais pode-se encontrar uma representação dentro da DRT:

Texto 6.1 Aline tem um bonito cabelo. Ela o trata carinhosamente.

Frase 6.2 João morreu.

Texto 6.3 Cláudia comprou um carro. Ela o lava nos finais de semana.

Frase 6.4 Ana Paula comprou um Santana e Helena comprou um Pointer.

Frase 6.5 Marcos, Paulo ou Miguel foi à praia

6.3 A DRS

As DRSs ("Discourse Representation Structures"), como dissemos na seção anterior, são as estruturas utilizadas pela DRT com o intuito de representar o discurso.

Uma DRS é composta unicamente de dois elementos [KAM 90]:

- Um conjunto de referentes do discurso, denominado (drefs), compondo o chamado universo da DRS, os quais sempre aparecem no topo do diagrama que a representa;
- Um conjunto de condições, sempre mostradas abaixo do universo da DRS.

Na figura 6.1, mostramos o diagrama que representa a DRS construída a partir da sentença 6.6. Neste exemplo, x e y correspondem aos referentes do discurso, enquanto *Luís(x), Aline(y) e ama(x, y)* são as condições da DRS.

Frase 6.6 Luís ama Aline.

Além destas, outras estruturas podem aparecer como condição em uma DRS, inclusive uma outra DRS. Neste caso, a DRS que aparece como condição é denominada subDRS, enquanto a estrutura que a engloba é denominada DRS principal.

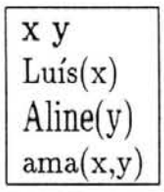

Figura 6.1 DRS associada à sentença "Luís ama Aline".

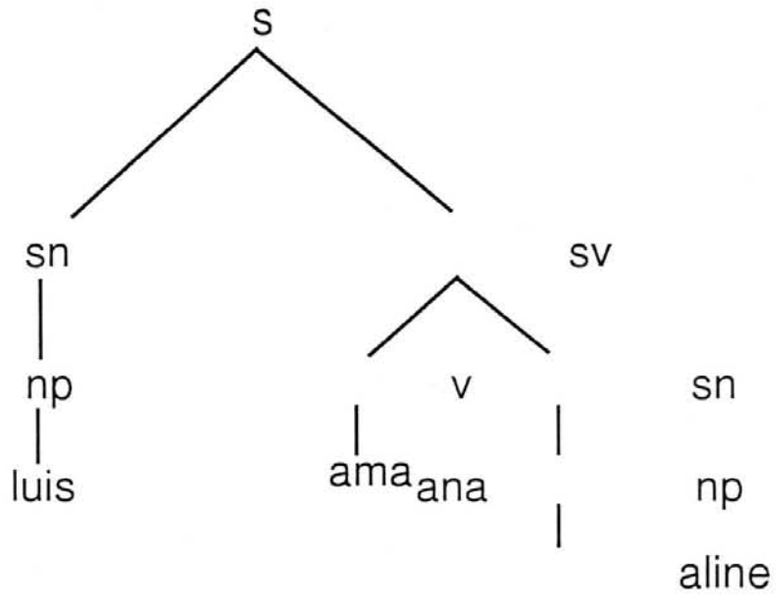

Figura 6.2 *Árvore sintática associada à sentença "Luís ama Aline".*

Na figura 6.2, mostramos a árvore sintática representativa da sentença 6.6 ora em estudo. Nas figuras 6.3 e 6.4 estão representados os estágios de evolução da árvore sintática mostrada na figura 6.2, conforme vai se construindo passo a passo a DRS mostrada na figura 6.1.

Analisando este conjunto de figuras, o leitor pode verificar que a redução da árvore sintática inicial, mostrada na figura 6.2, obedece às seguintes regras gerais [KAM 90]:

1. Para sintagmas nominais de nomes próprios que contiverem um novo indivíduo, cria-se e introduz-se um novo referente no universo da DRS.

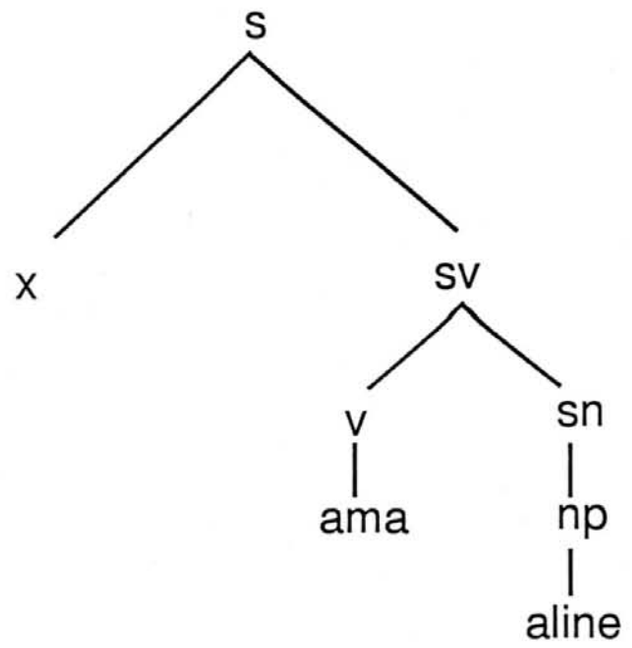

Figura 6.3 *Árvore Sintática após o processamento do sintagma nominal Luís.*

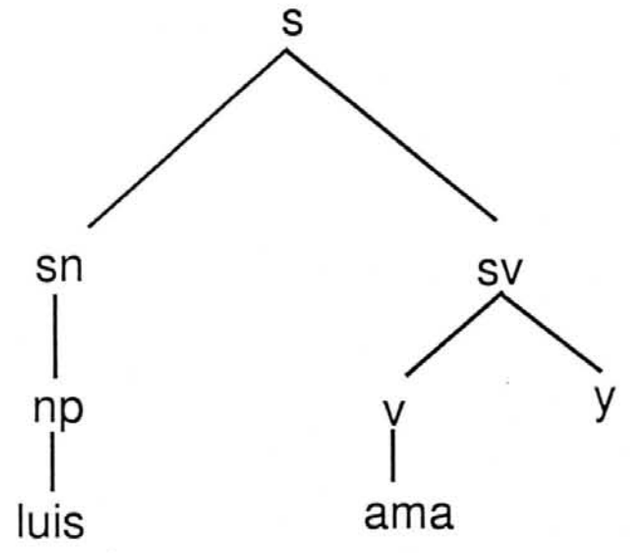

Figura 6.4 *Árvore sintática após o processamento do sintagma nominal Aline.*

Desta forma, podemos definir um referente como sendo um elo de ligação entre os indivíduos e as entidades do conhecimento de senso comum;

2. A redução da subárvore associada ao sintagma nominal, referido no item anterior, cria uma nova condição formada pelo nome próprio sucedido pelo referente colocado dentro de parênteses;

3. O novo referente criado substitui a respectiva subárvore na árvore de derivação sintática obtida a partir da sentença original.

O significado lingüístico em uma DRS é determinado pelo conjunto de condições e referentes do discurso, enquanto a avaliação semântica da mesma dá-se dentro de um modelo.

A construção de uma DRS não se constitui em uma atividade empírica, mas sim dirigida por um conjunto de passos e regras formais, denominadas genericamente de regras de construção de uma DRS. Em virtude da complexidade envolvida na construção de uma DRS e a sua relevância para o presente trabalho, abordaremos este tópico separadamente, dedicando-lhe uma seção especial.

6.4 Construção das DRSs

A construção de uma DRS pode ser entendida como sendo um processo que envolve essencialmente, um conjunto de regras, que transformam uma subárvore resultante da análise sintática de uma sentença, e um algoritmo que une a representação anterior do discurso e a sentença que vai ser processada.

Este caráter incremental do algoritmo de construção de uma DRS é particularmente importante se considerarmos que as DRSs não servem apenas para representar sentenças únicas e simples, mas também conjuntos de sentenças que compõem unidades maiores e mais complexas, tais como, parágrafos e textos.

Para elucidar o que dissemos acima, considere o pequeno texto mostrado a seguir (6.7):

Texto 6.7 Marcos ama Maria. Ela o despreza.

Se tomássemos isoladamente a primeira sentença, seria bastante simples encontrar uma representação que a descrevesse semanticamente, baseando-nos apenas em sua árvore de derivação sintática. Contudo, se quisermos criar uma representação onde também esteja expressa a segunda sentença, seria inviável sabermos a quem se referem os pronomes anafóricos (*ela* e *o)* se desprezarmos a representação construída para a primeira sentença. Desta maneira, a dependência de uma sentença em relação à representação que contém informações obtidas a partir de sentenças previamente processadas, torna imprescindível a existência de um algoritmo que manipule, não apenas a árvore de derivação sintática de uma sentença, mas também a representação do discurso expressa através das DRSs. Passemos então ao algoritmo de construção das DRSs [KAM 90].

Considere um discurso coerente e coeso, representado por um conjunto finito de sentenças, denotado pela expressão: D = S_1, S_2, ..., S_i, S_{i+1}, ..., S_n. Considerando ainda, como ponto inicial, a DRS K0 (DRS principal), sem referentes e, tão pouco, restrições ou condições sobre estes referentes. Sob estas condições, demonstra-se o seguinte algoritmo:

Repetir para $i = 1, 2, \ldots, n$

1. *Juntar a análise sintática da sentença* $S_i[O_i]$ *às condições de* K_{i-1}. *Designar esta DRS por* K_i^*.

2. *Dado um conjunto de condições redutíveis de* K_i^*, *aplicar as regras de reescrita às condições redutíveis – sub-árvores que ainda existem na ADS, e que estão na forma gramatical – de* K_i^* *até que a DRS obtida* K_i, *não contenha mais condições redutíveis.*

Para uma melhor compreensão por parte do leitor, mostraremos o algoritmo de construção de uma DRS aplicando-o a exemplos práticos, os quais serão vistos nas seções subseqüentes. Antes, porém, devemos descrever formalmente a maneira de reduzir os componentes de uma sentença simples, possibilitando-nos construir a DRS a ela associada.

6.5 Sentenças Simples

Nesta seção, descreveremos os tipos de sentenças que geram uma representação semântica dentro da DRT e que podem ainda ser consideradas como uma representação de primeira ordem, sendo estes os tipos de sentenças reconhecidas e tratadas na implementação. Adicionalmente, ilustraremos cada regra de redução através de um exemplo, tornando, desta forma, os conceitos claros para o leitor.

6.5.1 Nomes Próprios

A seguir, relacionamos os passos requeridos para a redução de um nome próprio em uma sentença [KAM 90]:

- Cria-se um novo referente,
- Criação de uma condição $\beta(x)$, onde β é o nome próprio e x o referente
- Inserção de ambos, referente e condição, na DRS principal.

Para ilustrarmos a aplicação das regras acima citadas, considere a sentença 6.8, cuja árvore de derivação sintática está representada na figura 6.5.

Frase 6.8 Carlos odeia Patrícia.

Aplicando as regras de redução ao nome próprio (*Carlos*), obtemos a DRS representada na figura 6.7. A árvore sintática resultante é mostrada na figura 6.6,

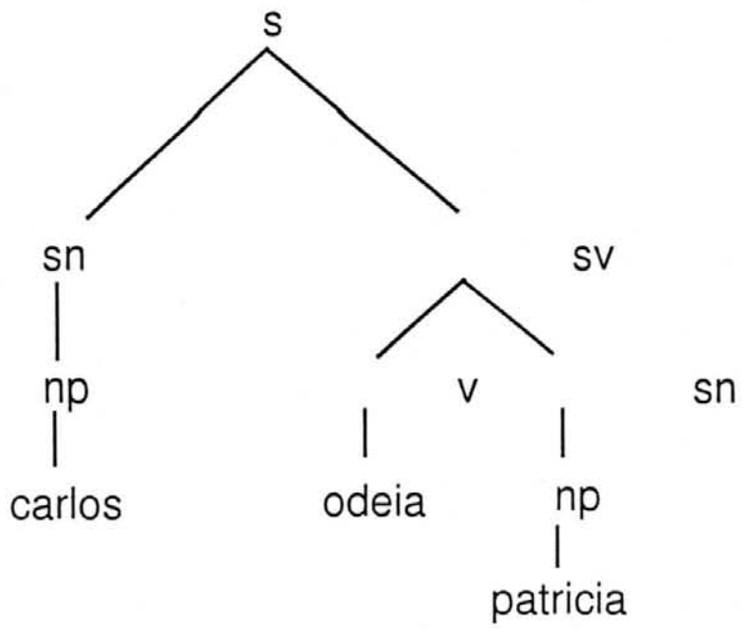

Figura 6.5 *Árvore de derivação sintática associada à sentença "Carlos odeia Patrícia".*

onde observa-se a modificação estrutural por ela sofrida com a aplicação das regras de redução. Esta evolução na representação sintática é obtida substituindo-se a subárvore correspondente ao sintagma nominal, pelo novo referente introduzido no universo da DRS, associado ao nome próprio (*Carlos*).

Prosseguindo no processamento da sentença 6.8, observamos que existe ainda o nome próprio (*Patrícia*), presente no predicado, que ainda pode ser reduzido. Assim, aplicando novamente o algoritmo apropriado para a redução de um nome próprio, chegamos à forma irredutível para a sentença considerada 6.8. Nas figuras 6.8 e 6.9, mostramos respectivamente a árvore sintática e DRS finais, resultantes do processamento da sentença 6.8.

6.5.2 Pronomes

Neste trabalho, faremos apenas um breve estudo a cerca das regras de construção das DRSs que representam sentenças pronominais. Ademais, limitaremos

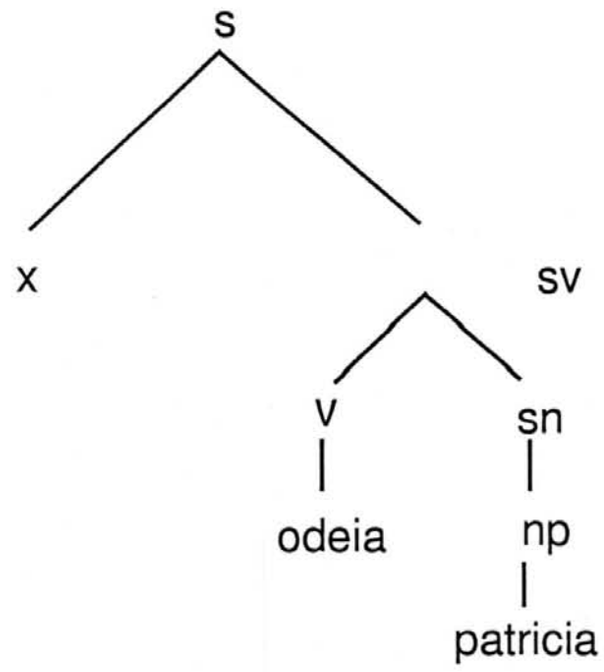

Figura 6.6 *ADS resultante após a redução do nome próprio Carlos.*

x
Carlos(x)
x odeia Patrícia

Figura 6.7 DRS produzida após o processamento do nome próprio *Carlos*

o nosso estudo aos pronomes anafóricos, ou seja, aqueles que se referem a algum item ou objeto mencionado anteriormente dentro do discurso.

A brevidade e limitação na abrangência do estudo feito a respeito desta classe de sentença, deve-se basicamente ao fato de que este tipo de construção (sentença, árvore sintática e DRS), não é necessária aos propósitos do sistema, não sendo, por conseguinte, implementado no protótipo.

Considere o discurso mostrado a seguir 6.9:

Texto 6.9 Marcos gosta de Lúcia. Ela odeia-o.

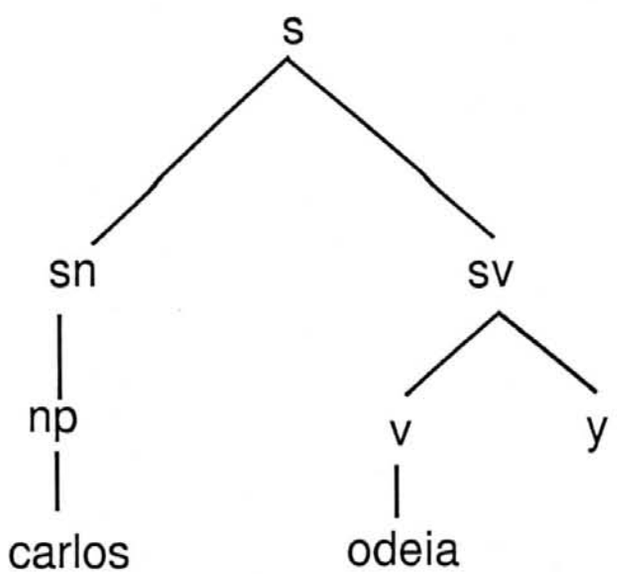

Figura 6.8 *ADS final irredutível gerada a partir da sentença considerada.*

x y
Carlos(x)
Patrícia(y)
odeia(x,y)

Figura 6.9 DRS final produzida a partir da sentença "Carlos odeia a Patrícia".

No exemplo acima, o processamento da primeira sentença ("Marcos gosta de Lúcia") é idêntico ao que já foi mostrado na seção anterior 6.5.1, onde descrevemos o algoritmo usado para a redução de nomes próprios.

Nas figuras 6.11 e 6.10, mostramos respectivamente a árvore de derivação sintática e a DRS resultantes do processamento da primeira sentença.

Agora, observemos a segunda sentença ("Ela odeia-o"), onde aparecem dois pronomes anafóricos. O primeiro (*Ela*) refere-se obviamente ao nome próprio feminino (*Lúcia*). O segundo (*o*), por sua vez, está relacionado ao nome próprio masculino (*Marcos*).

Em ambos os casos, deve haver na DRS resultante, duas condições que associem a cada um dos pronomes, os respectivos objetos.

x y
Marcos(x)
Lúcia(y)
gosta(x,y)
Ela odeia-o.

Figura 6.10 DRS construída a partir da primeira sentença do texto considerado.

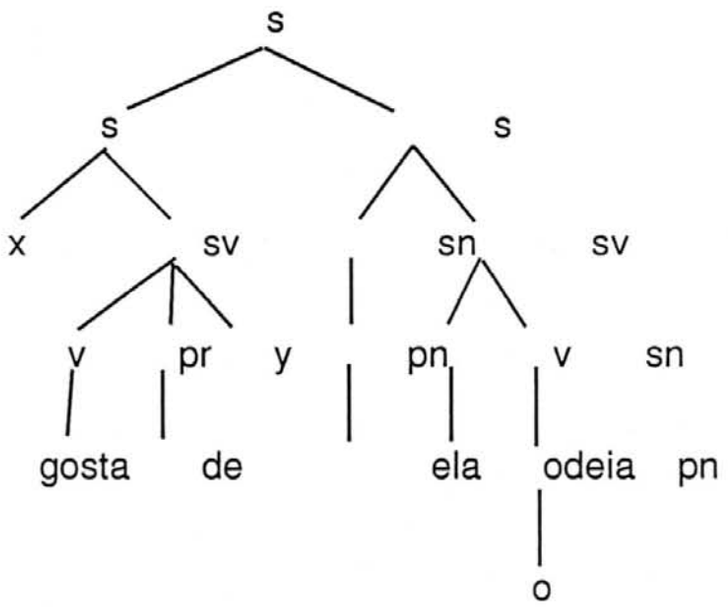

Figura 6.11 *ADS gerada com a redução da primeira sentença.*

Dentro da DRT, o problema do relacionamento de pronomes anafóricos com seus antecedentes é resolvido através de regras puramente sintáticas, ou seja, baseadas nos atributos dos objetos (gênero, número, grau etc), o que, sem dúvida, a limita bastante neste particular.

Em outros trabalhos, como, por exemplo, [FRE 93, FRE 93a], são adotados algoritmos e metodologias externas a DRT com o intuito de resolver o problema de ligação introduzido pelos pronomes anafóricos.

Na figura 6.12 está mostrada a DRS que representa o significado semântico da sentença 6.9, enquanto na figura 6.13 está ilustrada a árvore sintática

irredutível, associada à mesma sentença. É importante notar que o símbolo (=), colocado dentro da DRS, não significa igualdade entre os referentes (x, w) e (y, z). A interpretação correta informa-nos que (w) está associado a (x) e que, similarmente, o referente (z) está relacionado ao referente (y).

x y z w
Marcos(x)
Lúcia(y)
gosta(x,y)
z=y
w=z
odeia(z,w)

Figura 6.12 DRS gerada pelo processamento completo do texto "Marcos gosta de Lúcia. Ela odeia-o".

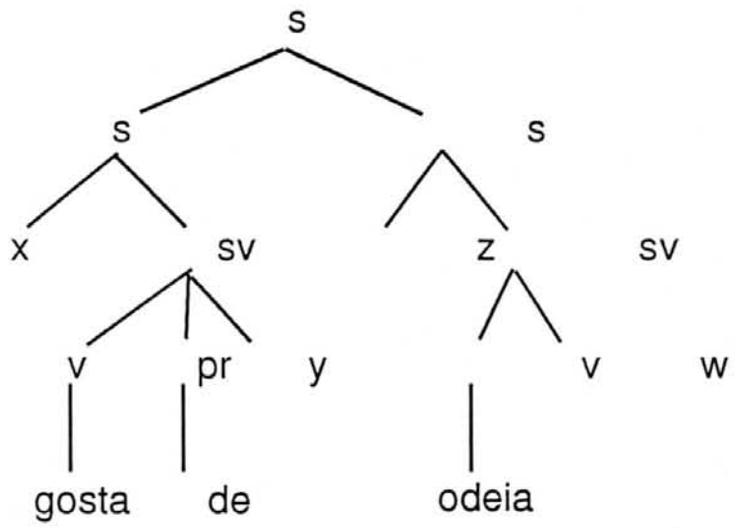

Figura 6.13 *Árvore sintática irredutível, associada à sentença do exemplo.*

A seguir, mostramos o algoritmo de construção da DRS que representa semanticamente a classe de sentenças estudadas nesta seção [KAM 90]:

1. Cria-se um novo referente do discurso correspondente ao pronome anafórico;

2. Insere-se o novo referente no universo da DRS;

3. Determina-se a que objeto está se referindo o pronome anafórico;

4. Cria-se uma nova condição da forma $\alpha = \beta$, onde α corresponde ao novo referente do discurso e β designa um referente alternativo, já inserido no universo da DRS, associado a α;

5. Insira esta nova condição no conjunto de condições da DRS.

6.5.3 Sintagmas Nominais Indefinidos

Sintagmas nominais indefinidos são aqueles em que junto a um objeto aparece um artigo indefinido, por exemplo, (um, uma, etc), atribuindo-lhe um caráter de generalidade.

Deste modo, torna-se necessário que a interpretação para este tipo de objeto seja diferenciada da adotada para objetos únicos e individualizados, pertencentes ao conhecimento de senso comum dos agentes.

Por exemplo, considere a sentença mostrada a seguir 6.10:

Frase 6.10 Ana Paula tem um namorado.

No caso acima, a expressão (*um namorado*) não representa uma entidade específica, tangível no universo do ouvinte, mas sim um objeto não individualizado ou genérico, dentre vários outros de conhecimento do agente, possuindo todos as mesmas características fundamentais. Desta forma, a representação adotada para a DRS deve capturar a diferença existente entre objetos individualizados, como, por exemplo, nomes próprios e objetos indefinidos, como ilustrado na sentença 6.10.

Assim, uma representação aceitável para a sentença anterior é mostrada na figura 6.14. Note-se, todavia, que a condição inserida na DRS pelo nome próprio é diferenciada daquela condição introduzida pela existência de um sintagma nominal indefinido. Conseqüentemente, quando de sua avaliação em um modelo, a DRS refletirá a diferença semântica existente entre os dois indivíduos considerados.

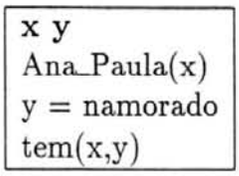

Figura 6.14 DRS com a representação de um objeto indefinido

6.5.4 Sentenças Imperativas

Nesta subseção, veremos a metodologia por nós adotada para representar sentenças imperativas, ou seja, aquelas em que o agente ordena, pede, ou ainda, exorta o ouvinte a realizar uma ação, executar uma tarefa etc.

A representação desta classe de sentenças é de particular interesse para o presente trabalho, visto que o centro do diálogo mantido entre o sistema e o usuário, será composto por sentenças predominantemente imperativas, nas quais este solicitará ao primeiro que execute uma determinada ação ou tarefa.

Na língua portuguesa, as sentenças imperativas podem aparecer sob diversas formas, dependendo de como o agente deseja expressar a sua vontade, podendo variar desde uma formulação polida, onde o agente quase que faz um convite ao ouvinte, até uma forma de comando, na qual o agente dá uma ordem direta ao ouvinte.

Neste trabalho, trataremos apenas de sentenças que se assemelhem às mostradas a seguir 6.11 e 6.12. Em todos os casos ilustrados abaixo, é possível, com o que foi visto até o momento, encontrar-se as DRSs que representam o significado semântico destas frases.

Frase 6.11 Ana, saia do carro.

Frase 6.12 Saia do carro!

Agora, tomemos como exemplo prático a frase mostrada abaixo 6.13:

Frase 6.13 Compra um chocolate.

Nesta sentença, o agente está ordenando ao ouvinte, expresso implicitamente, que adquira um chocolate.

Com o objetivo de representar o significado semântico deste tipo de sentença, tornamos explícito na DRS resultante, o ouvinte ao qual o agente se dirige.

Supondo-se que na sentença do exemplo 6.13 e em todas as outras sentenças similares o ouvinte seja sempre o mesmo, por exemplo, o indivíduo Cláudia. Sob esssas condições, a DRS que representa a sentença do exemplo 6.13 é mostrada na figura 6.15.

No que concerne à implementação, o ouvinte que explicitaremos e inseriremos no universo da DRS, sempre que tivermos este tipo de situação para representar, será um indivíduo, ou melhor, o nome próprio Sistema.

x y
Cláudia(x)
y=chocolate
compra(x,y)

Figura 6.15 Representação DRS adotada para sentenças imperativas

6.6 Modelo

A avaliação de uma DRS é feita dentro de um modelo. O modelo é uma estrutura de informação com respeito a qual é possível avaliar-se expressões de alguma linguagem e, em particular, avaliar sentenças desta linguagem no que concerne a sua falsidade ou veracidade. Um determinado modelo deve ser capaz de caracterizar os fatores abaixo relacionados [KAM 90]:

- Os indivíduos existentes;

- As propriedades destes indivíduos;
- As relações que estes indivíduos mantêm entre si;
- E, finalmente, os nomes pelos quais eles são designados.

Para uma linguagem L, um modelo M, dentro do qual as expressões dessa linguagem são avaliadas, é definido através da tripla ordenada (U_M, $Nome_M$, $Pred_M$), onde:

- U_M é um conjunto não vazio, constituído de todos os indivíduos existentes, denominado universo de M.
- $Nome_M$ é um conjunto cujos membros são pares da forma ⟨A, i_A⟩onde A é um nome de L e i_A um indivíduo de U_M com esse nome.
- $Pred_M$ é o conjunto formado por todos os pares ⟨P, P_M⟩, sendo P um predicado de L e P_M sua extensão em M.

6.7 Prova de uma DRS

Seja K uma DRS confinada a V e a R, seja γ uma condição DRS simples, e seja f uma função, possivelmente parcial, de interpretação de R em M, por exemplo, uma função cujo domínio é um subconjunto de R e cujo contradomínio está contido em U_M.

- f verifica a DRS K em M SSE f verifica cada uma das condições pertencentes à $Cond_K$ em M
- f verifica a condição γ em M sse:
  1. γ é da forma X = Y e f mapeia X e Y no mesmo elemento de U_M.
  2. γ é da forma π(X) e f mapeia X no elemento a de U_M tal que ⟨π,a ⟩ ∈
     $Nome_M$.

3. γ é da forma η(X) e f mapeia X no elemento a de U_M tal que a $\in Pred_M(\eta)$.

4. γ é da forma ζ(X) e f mapeia X no elemento a de U_M tal que a $\in Pred_M(\zeta)$.

5. γ é da forma ξ(X,Y) e f mapeia X e Y nos elementos a e b de U_M tais que (a,b) $\in Pred_M(\xi)$ [KAM 90]

6.8 Veracidade de uma DRS em um Modelo

Seja K uma DRS confinada a V e a R e seja M um modelo definido para V.

Diz-se que K é verdadeira em M sse existir uma função de interpretação f de R em M tal que:

- Dom(f) = U_K
- f verifica K em M.

Com o intuito de clarificar os conceitos relacionados a esse tópico, mostraremos ao leitor, através de um exemplo prático, como provar a veracidade de uma DRS dentro de um modelo. Considere o texto mostrado abaixo 6.14:

Texto 6.14 Carlos jantou com a Paula. Ele deu a ela uma flor. Marcos viajou com a Márcia. Ele a levou à praia.

e sua respectiva representação DRS 6.16:

Um modelo mínimo que verifica o discurso ou texto 6.14 é:

$M_1 = \{\langle U_{M1}, Nome_{M1}, Pred_{M1}\rangle\}$ onde:

- U_{M1} = { 1,2,3,4,5,6}

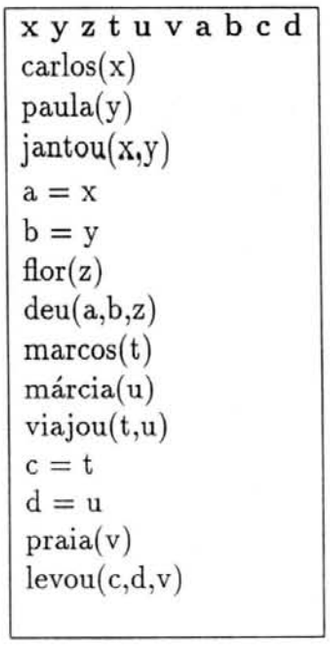

Figura 6.16 DRS gerada a partir do texto do exemplo considerado.

- $Nome_{M1}$ = {⟨carlos, 1⟩, ⟨paula, 2⟩, ⟨marcos, 3⟩, ⟨márcia, 4⟩}
- $Pred_{M1}$ = {⟨flor,$flor_{M1}$⟩, ⟨praia,$praia_{M1}$⟩,
  ⟨jantou, $jantar_{M1}$⟩, ⟨deu, dar_{M1}⟩, ⟨saiu, $sair_{M1}$⟩, ⟨levou, $levar_{M1}$⟩}

  sendo:

- $jantar_{M1}$ = { ⟨1,2⟩}
- dar_{M1} = { ⟨1,2,5⟩}
- $viajar_{M1}$ = {⟨3, 4⟩}
- $levar_{M1}$ = { ⟨3,4,6⟩}
- $flor_{M1}$ = { 5 }
- $praia_{M1}$ = { 6 }

Existe uma interpretação f que verifica a DRS 6.16 no modelo M_1:

- f(x) = 1, f(y) = 2, f(z) = 5 $\Rightarrow$dar(1,2,5)
- f(t) = 3, f(u) = 4, f(v) = 6 $\Rightarrow$levar(3,4,6)

Não existe outra interpretação **f** que verifique a DRS no modelo M_1.

6.9 Outras Classes de Sentenças

Nesta seção, faremos um estudo simplificado a respeito de outros tipos de sentenças exploradas por Kamp em [KAM 90].

Apesar destas classes de sentenças não terem sido implementadas nesta versão do protótipo, resolvemos incluí-las neste texto com o intuito de que o leitor comece a familiarizar-se com estruturas mais complexas, representáveis dentro da DRP através das DRSs.

6.9.1 Negação

Segundo Kamp [KAM 90], a negação é usada na linguagem humana de diversas formas, todavia sempre com o intuito de negar algum fato do mundo interno do agente.

Alguns exemplos, onde são comumente aplicados estes tipos de construção, aparecem a seguir:

- O falante não concorda com alguma informação transmitida por outro agente;
- O falante expressa a inexistência de um determinado indivíduo em seu mundo interno;
- Finalmente, o agente expressa a não veracidade de uma sentença anteriormente formulada.

Neste trabalho, no entanto, usaremos apenas sentenças negativas, onde busca-se afirmar a inexistência de referentes e condições (do discurso, representado por intermédio das DRSs), dentro de um modelo criado ou concebido pelo ouvinte.

Sendo assim, considere como exemplo ilustrativo a sentença negativa 6.15, mostrada a seguir:

Frase 6.15 Cláudia não gosta de praia.

A figura 6.17 mostra a DRS associada à sentença do exemplo 6.15. Neste caso, a DRS deve representar para o agente a existência do indivíduo (*Cláudia*) e negar a veracidade, a partir do símbolo (*não*), do restante da sentença, o qual afirma que (*Cláudia*) gosta de praia.

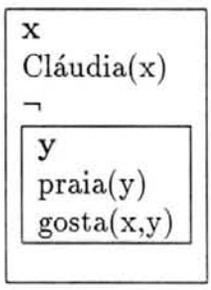

Figura 6.17 DRS que representa a sentença negativa estudada.

6.9.2 Condicionais

Desta subseção, abordaremos sentenças condicionais simples, onde os pronomes anafóricos, caso existam, aparecem somente na sentença que corresponde à assertiva da condição. Uma abordagem detalhada a respeito da construção das DRSs associadas a diversos tipos de sentenças condicionais, pode ser encontrada em [KAM 90].

Considere a seguinte sentença ilustrativa 6.16:

Frase 6.16 Se Marina gosta de Marcos então ela o namora.

Neste exemplo, a sentença ("*se Marina gosta de Marcos*") constitui o chamado antecedente ou condição. Por outro lado, ("*ela o namora*") constitui o denominado conseqüente ou assertiva hipotética.

As figuras 6.18 e 6.19 mostram respectivamente as DRSs, preliminar e final, que representam a sentença do exemplo 6.16.

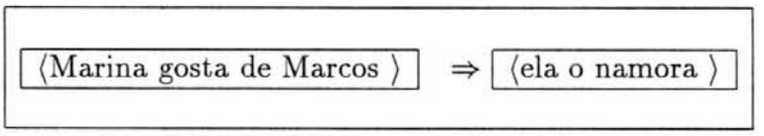

Figura 6.18 Representação DRS preliminar do exemplo de uma sentença condicional.

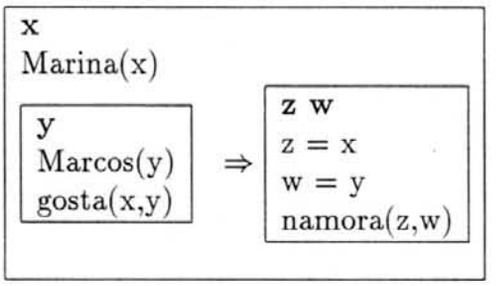

Figura 6.19 DRS final associada à sentença condicional.

Neste sentido, observa-se que as DRSs obtidas capturam perfeitamente a idéia de que, supondo-se que *Marina* goste de *Marcos*, certamente *ela o* namora.

6.9.3 Quantificadores Universais

Nesta subseção, mostraremos como são representadas, através de DRSs, sentenças, onde são introduzidas palavras indicativas de quantificações, formando um sintagma nominal, cujo núcleo passa a ser quantificado.

Considere a sentença mostrada a seguir 6.17:

Frase 6.17 Todo o homem que possui um carro bonito tem um ar preocupado

Todavia, se considerarmos a sentença sob uma outra forma, ou seja, se a estruturarmos diferentemente, como está mostrada a seguir 6.18:

Frase 6.18 Se um homem possui um carro bonito então ele tem um ar preocupado.

Se compararmos ambas as sentenças, embora elas sejam evidentemente estruturalmente diferentes, observamos que as duas possuem a mesma interpretação semântica. De fato, quantificadores universais atribuem um caráter condicional às sentenças. Conseqüentemente, as sentenças condicionais e com quantificadores universais serão representadas adotando-se os mesmos princípios para a construção das DRSs.

Na figura 6.20, mostramos a DRS gerada a partir da sentença 6.17.

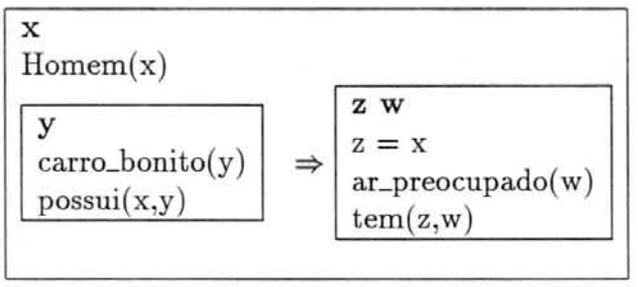

Figura 6.20 DRS gerada a partir da sentença do exemplo, quantificada universalmente.

6.9.4 Sentenças Disjuntivas e Conjuntivas

Como último tópico desta seção, onde abordamos a construção de DRSs para sentenças complexas, mostraremos a forma de se representar semanticamente sentenças disjuntivas, ou seja, aquelas onde aparece a conjunção (*ou*) e sentenças conjuntivas, caracterizadas pela presença da conjunção (*e*).

Na verdade, abordaremos apenas a representação de sentenças semelhantes a 6.19 (disjuntiva) e 6.20 (conjuntiva). Para uma discussão mais aprofundada

a respeito do tema, com a definição dos algoritmos de obtenção das DRSs, considerações de aplicabilidade e acessibilidade, o leitor deve consultar o trabalho de Hans Kamp [KAM 90]. Neste mesmo trabalho, Kamp mostra outras classes adicionais de sentenças conjuntivas e disjuntivas, aprofundando bastante os aspectos inerentes à sua representação.

Texto 6.19 Ana ou Cláudia ama Marcelo.

Texto 6.20 Adriana gosta de cachorros e Márcia gosta de pássaros.

Considerando a sentença disjuntiva 6.19. Neste exemplo, as figuras 6.21 e 6.22 mostram respectivamente a DRS parcial e final que representam semanticamente a sentença estudada. O exemplo denota que pelo menos um dos indivíduos (*Ana* ou *Cláudia*) ama o outro indivíduo (*Marcelo*). Similarmente, a veracidade da DRS principal estará na dependência da veracidade de pelo menos uma das duas subDRSs.

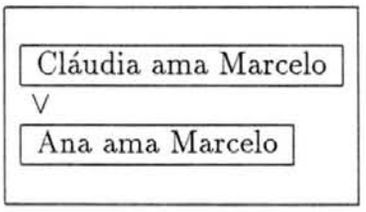

Figura 6.21 DRS preliminar que representa uma sentença disjuntiva.

Por outro lado, as figuras 6.23 e 6.24 atribuem uma interpretação diferenciada para a segunda sentença 6.20, visto que esta é caracterizadamente conjuntiva. Neste caso particular, para que a DRS principal seja verdadeira dentro do modelo M, representando semanticamente a sentença 6.20, é mister que ambas as subDRSs sejam igualmente verificadas dentro do mesmo modelo.

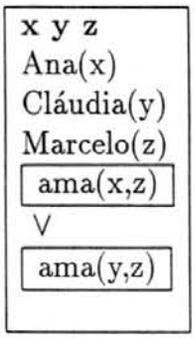

Figura 6.22 DRS final, representativa da mesma sentença disjuntiva.

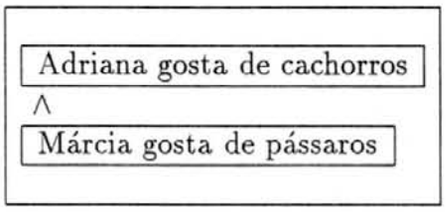

Figura 6.23 DRS preliminar de uma sentença conjuntiva.

6.10 Formalização da linguagem DRS

Nas seções e subseções precedentes deste capítulo, estivemos dedicados basicamente à construção das DRSs, mostrando os algoritmos que devem ser empregados em cada situação prática. Assim, obtivemos diversos tipos de DRSs que representam, desde sentenças simples, até construções mais complexas, como, por exemplo, sentenças conjuntivas. A este conjunto formado pelos diversos tipos de DRSs, Kamp denomina linguagem DRS [KAM 90]. Esta é formada basicamente pelos diversos tipos de condições geradas a partir da representação de sentencas, usando-se DRSs.

Definindo a linguagem DRS para uma DRS K finita, constituída pelo par: $\langle U_K, Cond_K \rangle$, onde U_K é um conjunto finito de referentes e $Cond_K$ um conjunto finito de condições DRS, existem os seguintes tipos de condições definidas:

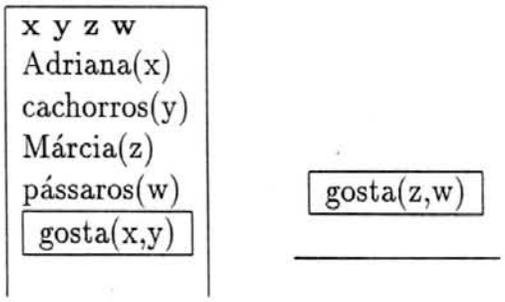

Figura 6.24 DRS final, representativa da sentença conjuntiva, anteriormente referida.

1. x=y, onde x e y são referentes. Este tipo de condição aplica-se a sentenças que contenham pronomes anafóricos;

2. P($x_1, x_2, \ldots, x_n$), onde $x_1, x_2, \ldots, x_n$ são referentes e P é o nome do predicado. Esta condição é usada, por exemplo, na representação de verbos transitivos, diretos ou indiretos;

3. $\neg\ K$, onde K é uma DRS finita. Esta condição representa uma negação dentro de uma sentença;

4. $K_1 \Rightarrow K_2$, onde K_1 e K_2 são DRSes finitas, sendo aplicadas a sentenças condicionais e com quantificadores universais;

5. $K_1, K_2, \ldots, \vee K_n$ onde $K_1, K_2 \ldots, K_n$ são DRSs finitas. Usadas para representar sentenças disjuntivas;

6. $K_1, k_2, \ldots, \wedge K_n$, onde $K_1, K_2, \ldots, K_n$ são DRSs finitas. Estas condições são usadas na representação de sentenças conjuntivas.

6.11 Algumas Considerações Finais

No presente capítulo, procuramos familiarizar o leitor não especializado com a teoria e formalismo inerentes à representação do significado semântico de sentenças.

Para tanto, apresentamos a DRT, descrevendo brevemente seus conceitos, e as DRSs, usadas para representar semanticamente as sentenças consideradas.

Esta abordagem, até certo ponto informal, foi adotada com o intuito de tornar a leitura do texto mais dinâmica, facilitando a compreensão por parte dos leitores. Ademais, parece-nos sem sentido aprofundarmo-nos em conceitos que não serão explorados na continuidade deste trabalho, principalmente no que concerne à forma de representar-se sentenças da língua portuguesa não usadas na implementação do protótipo. Neste particular, o leitor constatará que, por exemplo, sentenças com quantificadores universais não são reconhecidas e, conseqüentemente, tratadas pelo sistema.

Todavia, pessoas que estejam interessadas em aprofundar seus conhecimentos a respeito da DRT, podem consultar, em ordem crescente de complexidade, respectivamente, [KAM 88, FRE 92a, FRE 93, KAM 90].

7 O AMBIENTE SDIP

Nos capítulos precedentes deste trabalho, estudamos detalhadamente os sistemas de pesquisa e recuperação de informações mais usados presentemente dentro da Internet (capítulos 2, 3, 4 e 5), bem como a teoria necessária para a representação semântica de sentenças (capítulo 6), assuntos de vital importância para que o leitor consiga situar contextualmente o trabalho aqui descrito e, fundamentalmente, compreenda o formalismo que nos possibilitou construir um sistema capaz de interpretar sentenças, ainda que bastante simples, da língua portuguesa.

A partir deste momento, portanto, ingressaremos no ponto central do presente trabalho, qual seja, a descrição detalhada do sistema desenvolvido e implementado durante o período de andamento da dissertação.

Em virtude da complexidade do protótipo desenvolvido, o qual é composto por diversos processos, estudaremos sua estrutura, funcionamento, modo de uso, processos constituintes etc, em diversos capítulos, esperando, desta forma, facilitar a compreensão por parte do leitor.

No presente capítulo, faremos uma exposição geral a respeito do protótipo, enfatizando seus processos constituintes, bem como os formatos das mensagens empregadas na comunicação entre cada um deles.

Nos seis capítulos subseqüentes (capítulos 8, 9, 10, 11, 12 e 13), abordaremos separadamente cada um dos processos que compõem o sistema em estudo, destacando seu funcionamento e estrutura.

Posteriormente, no capítulo 14, ofererecemos aos leitores um manual do usuário para o **SDIP**, com exemplos de utilização do sistema, abordando desde sua interface gráfica até o interfaceamento por intermédio de sentenças da língua portuguesa.

7.1 Visão Geral

O protótipo desenvolvido e implementado, denominado **SDIP** (*Smart Discovery Information Program*), é composto por um conjunto de processos onde, cada um deles, é responsável pela realização de tarefas bastante específicas. Assim, por exemplo, existe dentro do sistema um processo responsável tão somente pela implementação do protocolo de transferência de arquivos FTP [POS 85], o qual permite ao sistema recuperar arquivos armazenados em servidores FTP de acesso público em qualquer parte do mundo.

Desde o início do desenvolvimento do protótipo ora em estudo, buscou-se projetar não um sistema monolítico, mas sim um ambiente orientado à expansão, ou seja, um conjunto integrado de ferramentas ao qual é possível, sem grande esforço de programação, agregar outros módulos ou então adicionar novas funcionalidades aos já existentes.

A construção de um ambiente composto por vários processos, além de responder satisfatoriamente ao objetivo acima citado, também apresenta uma série de vantagens no que concerne ao seu desenvolvimento, implementação e uso. Dentre estas, merecem destaque:

- Possibilidade de projetar, implementar e testar individualmente cada um dos módulos do sistema, o que, sem dúvida, diminui tanto o tempo quanto a complexidade envolvida na integração final dos diversos módulos constituintes;

- Pouca incidência de erros quando da integração dos módulos em virtude dos testes preliminares realizados em cada um deles;

- Esta arquitetura possibilita o uso mais adequado dos recursos computacionais e de comunicações disponíveis, permitindo que cada módulo do sistema resida em computadores distintos;

- A substituição de um módulo do sistema por outro com as mesmas funcionalidades é trivial e transparente para o restante do sistema, desde que mantida a mesma interface;

- Nesta arquitetura os processos que compõem o ambiente são relativamente pequenos se comparados com o processo que resultaria da construção de um sistema monolítico, composto por um único módulo.

Ao lado de todas as vantagens acima citadas, a adoção deste tipo de arquitetura de sistema também apresenta algumas desvantagens, basicamente relacionadas com o tempo de programação adicional necessário para desenvolver o conjunto de bibliotecas que implementam as funções de interface entre os processos que formam o protótipo, bem como uma pequena diminuição na performance geral do sistema em virtude de que a comunicação entre os processos componentes dá-se através de mensagens que fluem, por exemplo, em uma rede local, o que, evidentemente, é mais moroso do que a passagem de parâmetros em uma chamada de função , realizada através da pilha do sistema, método usado em um protótipo composto por um único processo. Mesmo assim, acreditamos que as vantagens advindas da adoção deste tipo de arquitetura superam em muito as desvantagens, principalmente no que concerne o tempo de desenvolvimento e a modularidade do sistema final.

Na figura 7.1, mostramos um diagrama representativo do ambiente **SDIP**. Cada um dos retângulos de pontas arredondadas representa um processo componente do sistema. As setas dirigidas representam o sentido no qual sempre é iniciada uma operação, ou seja, identifica qual o processo que funciona como cliente e/ou servidor em uma comunicação.

Assim, por exemplo, na relação existente entre a **Interface Gráfica** e o **Cliente FTP**, aquele executa as funções do processo cliente, enquanto este é sempre o processo servidor.

Por outro lado, na relação existente entre a **Interface Gráfica** e o **Processo Inteligente**, o comportamento de ambos dependerá do tipo de operação em

curso. Portanto, em um dado momento, a **Interface Gráfica** será o processo cliente, enquanto o **Processo Inteligente** será o servidor. Em outras ocasiões, todavia, as funcionalidades de ambos os componentes serão invertidas.

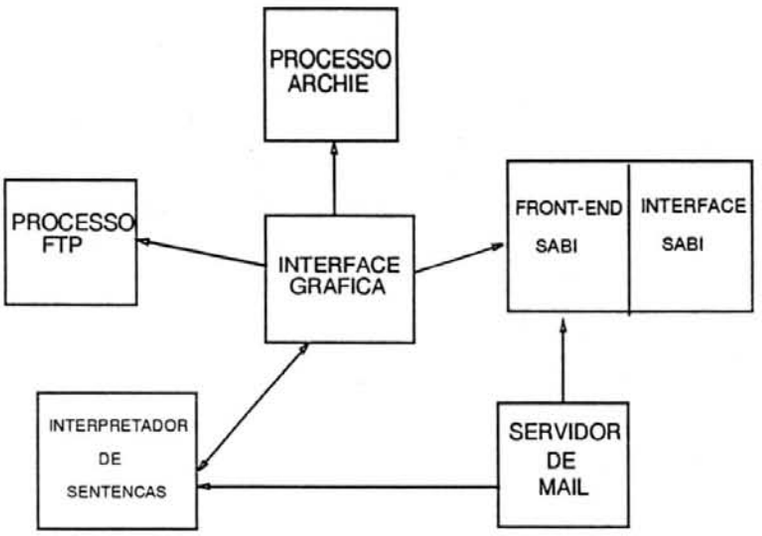

Figura 7.1 *Diagrama Representativo do Sistema SDIP*

Nas tabelas 7.1, 7.2 e 7.3, mostramos, respectivamente, uma especificação em SDL-PR [INT 88, BEL 91], que oferece ao leitor uma macrovisão do sistema **SDIP**. Posteriormente, mostraremos as especificações que descrevem o comportamento particular de cada um dos processos que fazem parte do ambiente.

A seguir, descrevemos brevemente cada um dos processos componentes do sistema **SDIP**. Uma abordagem detalhada a respeito de cada um dos componentes será vista nos capítulos subseqüentes do presente trabalho.

1. **Processo Archie** – > Implementa as funções que permitem ao sistema dialogar com os servidores Archie [KRO 92] da Internet;

2. **Processo FTP** – > Implementa o protocolo de transferência de arquivos FTP, permitindo ao sistema, entre outras coisas, recuperar arquivos armazenados nos servidores FTP públicos da Internet;

3. **Front-end** e **Interface SABI** – > Permitem ao **SDIP** dialogar com o sistema SABI, possibilitando-lhe recuperar informações bibliográficas

Tabela 7.1 Macroespecificação do SDIP em SDL-PR.

SYSTEM	SDIP;	/*SISTEMA SDIP*/
SIGNAL	S_ARC_IG,	/*CONDUZ DADOS
	S_IG_ARC,	ENTRE O MPA E MIG*/
	S_ARC_ENV,	/*COMUNICAÇÃO
	S_ENV_ARC;	COM AMBIENTE*/
SIGNAL	S_FTP_IG,	/*COMUNICAÇÃO ENTRE
	S_IG_FTP,	MPFTP E MIG*/
	S_FTP_ENV_1,	
	S_FTP_ENV_2,	
	S_ENV_FTP_1,	/*COMUNICAÇÃO DO
	S_ENV_FTP_2;	MPFTP COM O AMBIENTE*/
SIGNAL	S_SCE_ENV,	/*COMUNICAÇÃO DO SCE
	S_ENV_SCE,	COM O AMBIENTE*/
	S_SCE_PIS,	/*COMUNICAÇÃO DO SCE
	S_PIS_SCE,	COM O MPIS*/
	S_SCE_FS,	/*COMUNICAÇÃO DO SCE
	S_FS_SCE;	COM O MIS*/
SIGNAL	S_FS_IG,	/*COMUNICAÇÃO DO MIG
	S_IG_FS;	COM MIS*/
SIGNAL	S_IS_ENV,	/*COMUNICAÇÃO DO MIS
	S_ENV_IS;	COM O AMBIENTE*/
SIGNAL	S_IG_ENV,	/*COMUNICAÇÃO DO MIG
	S_ENV_IG,	COM O AMBIENTE*/
	S_IG_PIS_1,	/*COMUNICAÇÃO ENTRE O
	S_PIS_IG_1,	MIG E O MPIS*/
	S_IG_PIS_2,	/*IDEM AO
	S_PIS_IG_2;	DESCRITO ACIMA*/
SIGNAL	S_CR;	/*CRIAÇÃO DE
		PROCESSOS*/

Tabela 7.2 Macroespecificação do SDIP em SDL-PR.

		/*DEFINIÇÃO DOS CANAIS USADOS PELOS BLOCOS*/
CHANNEL	C_ARC_ENV;	
FROM B_ARC	TO ENV	WITH S_ARC_ENV;
FROM ENV	TO B_ARC	WITH S_ENV_ARC;
ENDCHANNEL	C_ARC_ENV;	
CHANNEL	C_ARC_IG;	
FROM B_ARC	TO B_IG	WITH S_ARC_IG;
FROM B_IG	TO B_ARC	WITH S_IG_ARC,S_CR;
ENDCHANNEL	C_ARC_IG;	
CHANNEL	C_FTP_ENV;	
FROM B_FTP	TO ENV	WITH S_FTP_ENV_1,S_FTP_ENV_2;
FROM ENV	TO B_FTP	WITH S_ENV_FTP_1,S_ENV_FTP_2;
ENDCHANNEL	C_FTP_ENV;	
CHANNEL	C_FTP_IG;	
FROM B_FTP	TO B_IG	WITH S_FTP_IG;
FROM B_IG	TO B_FTP	WITH S_IG_FTP,S_CR;
ENDCHANNEL	C_FTP_IG;	
CHANNEL	C_SCE_ENV;	
FROM B_SCE	TO ENV	WITH S_SCE_ENV;
FROM ENV	TO B_SCE	WITH S_ENV_SCE;
ENDCHANNEL	C_SCE_ENV;	
CHANNEL	C_SCE_PIS;	
FROM B_SCE	TO B_PIS	WITH S_SCE_PIS,S_CR;
FROM B_PIS	TO B_SCE	WITH S_PIS_SCE;
ENDCHANNEL	C_SCE_PIS;	
CHANNEL	C_SCE_SABI;	
FROM B_SCE	TO B_SABI	WITH S_SCE_FS,S_CR;
FROM B_SABI	TO B_SCE	WITH S_FS_SCE;
ENDCHANNEL	C_SCE_SABI;	
CHANNEL	C_SABI_IG;	
FROM B_SABI	TO B_IG	WITH S_FS_IG;
FROM B_IG	TO B_SABI	WITH S_IG_FS,S_CR;
ENDCHANNEL	C_SABI_IG;	
CHANNEL	C_SABI_ENV;	
FROM B_SABI	TO ENV	WITH S_IS_ENV;
FROM ENV	TO B_SABI	WITH S_ENV_IS;

Tabela 7.3 Macroespecificação do SDIP em SDL-PR.

		/*DEFINIÇÃO DOS BLOCOS QUE FORMAMO SDIP*/
BLOCK	B_ARC REFERENCED;	/*BLOCO DO MPA*/
BLOCK	B_FTP REFERENCED;	/*BLOCO DO MPFTP*/
BLOCK	B_SCE REFERENCED;	/*BLOCO DO SCE*/
BLOCK	B_SABI REFERENCED;	/*BLOCO DO MIS*/
BLOCK	B_IG REFERENCED;	/*BLOCO DO MIG*/
BLOCK	B_PIS REFERENCED;	/*BLOCO DO MPIS*/
ENDSYSTEM	SDIP;	

relativas ao acervo mantido nas bibliotecas setoriais e central da Universidade Federal do Rio Grande do Sul;

4. **Servidor de EMail SABI** – > Componente do **SDIP** que implementa uma interface em língua natural para o sistema SABI, orientada aos usuários do correio eletrônico;

5. **Interface Gráfica** – > Processo através do qual os usuários interagem com o sistema utilizando um ambiente gráfico;

6. **Processo Inteligente** ou **Interpretador de Sentenças** – > Componente onde estão implementadas todas as funções inteligentes do sistema **SDIP**, tais como: interpretação de sentenças, gerenciamento da base de conhecimento, gerenciamento do dicionário de palavras, etc.

7.2 Comunicação entre os componentes do SDIP

Como vimos na seção anterior (7.1), o **SDIP** é um ambiente constituído por um conjunto de seis processos, que devem necessariamente trocar informações entre si para que o sistema possa operar efetivamente.

De maneira análoga ao que ocorreu no projeto global do **SDIP**, onde buscou-se construir um sistema modular, também no desenvolvimento das funções que provêm as facilidades de comunicação foi possível adotar-se esta mesma filosofia de projeto. Neste particular, foram construídas cinco bibliotecas de funções , uma para cada processo constituinte do sistema, onde estão presentes as rotinas que implementam todas as operações exeqüíveis por cada um deles.

Note-se que somente o processo **Servidor de E-Mail SABI** não possui uma biblioteca de funções de comunicação a ele associada, uma vez que este é o único processo no sistema que em nenhum momento funciona como servidor. Ao contrário dos demais, este processo recebe requisições e envia as suas respostas, única e exclusivamente, através do correio eletrônico.

Todas as funções de requisição e resposta de operação utilizam como serviço básico de comunicação o RPC (Remote Procedure Call) [SUN 88a, SUN 88b, SUN 90].

A opção por este tipo de serviço de comunicação, embora apresente um maior overhead se comparado, por exemplo, ao uso de sockets [COM 88], deveu-se principalmente ao fato de que o RPC, através do formato XDR (External Data Representation) [SUN 87, SUN 90], resolve o problema da diferença de representação de dados existente entre computadores de arquiteturas e fabricantes distintos. Por outro lado, caso optássemos pelo uso de sockets, a dissolução deste problema ficaria sob a responsabilidade do protótipo, o que complicaria desnecessariamente a implementação do mesmo.

Somente as trocas de informações verificadas entre o **Front-end SABI** e a **Interface SABI** ocorrem através de um arquivo convencional. Esta alternativa foi adotada, posto que esses dois processos devem obrigatoriamente residir na mesma estação de trabalho. Sendo assim, inexiste o problema de incompatibilidade na representação de dados, aliado ao fato de que há uma maior eficiência e rapidez na comunicação quando usamos o sistema de arquivos convencional se comparado à

utilização de uma rede local comum, tornando desnecessário o emprego do RPC e do formato XDR.

Vimos anteriormente que dentro do **SDIP** sempre que dois processos se comunicam, um deles funciona como cliente, enquanto o outro processo realiza o papel do servidor. Conseqüentemente, uma transação ou operação básica dentro do **SDIP** é composta por dois estágios, a saber:

1. Em um primeiro momento, o processo cliente envia uma requisição ao processo servidor, solicitando-lhe a realização de uma dada terefa que pode ser desde uma operação bastante simples como, por exemplo, a atribuição de um determinado valor a uma variável, até uma operação complexa como a recuperação de um arquivo;

2. Em um segundo momento, o processo servidor responde ao cliente, informando-lhe os resultados da operação previamente requisitada.

Para cada um desses dois estágios que formam uma transação completa, existe um formato de mensagem específico, embora elas apresentem algumas características estruturais similares. Mais precisamente, tanto nas mensagens de requisição, quanto nas de resposta, os cabeçalhos, além de serem obrigatórios, sempre apresentam as mesmas estruturas. O corpo, ao contrário, poderá ou não estar presente, diferindo em sua estrutura, dependendo da operação que estiver sendo requisitada ou respondida.

Na figura 7.2, mostramos o formato geral de uma mensagem de requisição. O significado de cada um dos campos que aparecem na figura 7.2 está explicado a seguir:

- *Processo* $->$ Código que identifica o processo que está requerendo a execução de uma dada tarefa;

- *Operação* $->$ Código que identifica univocamente a operação a ser executada pelo processo servidor;

- *Corpo* – > Campo opcional que contém os parâmetros para a execução da operação requisitada, sempre que eles forem necessários.

Figura 7.2 Formato geral de uma mensagem de requisição, enviada por um processo cliente a um servidor

Na figura 7.3, ilustramos o formato genérico de uma mensagem de resposta, enviada pelo processo servidor ao cliente, quando do término da operação requisitada.

A seguir, detalharemos o significado de cada um dos campos mostrados na figura 7.3:

- *Resultado* – > Código indicativo do resultado da operação, ou seja, denota o seu sucesso ou fracasso;
- *Erro* – > Caso tenha ocorrido um erro na realização da operação requisitada, este campo contém um código que o identifica;
- *Mensagem* – > Este campo está associado ao anterior. No caso de ocorrência de um erro, ele contém uma mensagem inteligível, descrevendo o problema ocorrido;
- *Corpo* – > A presença deste campo está na dependência da operação requerida. Quando presente, contém informações obtidas a partir da realização efetiva da operação requisitada.

Nas figuras 7.4 e 7.5, mostramos as mensagens de requisição e resposta, associadas à operação de estabelecimento de conexão com um servidor FTP público.

Resultado
Erro
Mensagem
Corpo

Figura 7.3 Formato geral de uma mensagem de resposta enviada pelo servidor ao cliente

No exemplo, observe-se que o nome do servidor com o qual se deseja estabelecer a conexão está colocado dentro do corpo da mensagem de requisição. Por outro lado, observamos que na mensagem de resposta não há a presença do corpo, visto que o processo cliente necessita saber tão somente se a operação transcorreu normalmente ou não, sendo dispensáveis outras informações adicionais.

Processo
0
penta.ufrgs.br

Figura 7.4 Mensagem enviada pelo cliente solicitando o estabelecimento de uma conexão FTP

Onde:

Processo	=	Número que identifica o processo cliente.
0	=	Código que identifica a operação desejada.
penta.ufrgs.br	=	Nome do servidor FTP público que deve ser contactado.

Onde:

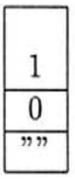

Figura 7.5 Mensagem de resposta informando o resultado da operação de estabelecimento de conexão.

1 = Operação transcorreu normalmente.

0 = Código de erro indeterminado.

"" = Não existe mensagem de erro.

Finalmente, devemos ainda dizer que não mostraremos neste capítulo as mensagens de requisição e de resposta, associadas às demais operações exeqüíveis por cada um dos processos componentes do sistema, por considerarmos que esta abordagem, em nível tão detalhado, foge dos objetivos gerais do presente texto. Ademais, a única estrutura variável morfologicamente, tanto nas mensagens de requisição, quanto nas de resposta é o corpo das mesmas.

Neste particular, o corpo de uma mensagem de requisição, em linhas gerais, sempre conterá os parâmetros presentes na função associada à operação requisitada, implementada em cada uma das bibliotecas de rotinas de interface.

Similarmente, o corpo de uma mensagem de resposta será formado, em geral, pelo resultado devolvido pela função relacionada à operação solicitada.

Assim, acreditamos que o leitor terá uma boa visão a respeito dos formatos de mensagens enviadas e recebidas em cada operação, analisando as sinopses das funções de biblioteca, presentes em todos os módulos que formam os sistema, sendo, por conseguinte, desnecessário prolongar-nos mais na discussão a respeito deste tema.

8 PROCESSO ARCHIE

8.1 Introdução

Começaremos, neste capítulo, a estudar cada um dos processos que unidos formam o conjunto denominado **SDIP**.

Como dissemos no capítulo introdutório do presente trabalho, uma das facilidades que deveria ser implementada como parte do sistema proposto nesta dissertação seria a pesquisa e localização de informações dentro da rede Internet.

Para tanto, deveríamos trabalhar no sentido de construir uma interface que possibilitasse ao ambiente **SDIP** usar os serviços providos por pelo menos um dos servidores de informação hoje em funcionamento na rede considerada.

Durante o transcorrer do trabalho, foram estudados minuciosamente os quatro servidores de informação mais populares junto à comunidade da Internet, os quais, sem dúvida, ainda experimentarão futuramente um grande incremento em sua aceitação e uso.

Evidentemente, durante o desenvolvimento do sistema, teríamos de determinar qual dentre os quatro servidores estudados adequar-se-ia melhor, pelo menos em um primeiro momento, aos objetivos principais do **SDIP**.

Sendo assim, a nossa escolha recaiu sobre o sistema Archie [EMT 91, EMT 92, SCH 92, KRO 92]. A opção por este servidor de informação deveu-se principalmente aos seguintes motivos:

- Simplicidade do protocolo empregado na comunicação entre o cliente e os servidores Archie, em comparação aos demais sistemas estudados, o que facilitou a implementação do protótipo;

- As informações estão concentradas em um pequeno número de servidores em oposição, por exemplo, ao sistema Gopher, no qual existem inúmeros servidores, cada um deles armazenando poucas informações;

- O sistema Archie trabalha sobre uma base de dados onde estão registrados inúmeros servidores FTP públicos, com seus arquivos e diretórios, nos quais residem a maior parte das informações hoje disponíveis dentro da Internet.

Assim, com o intuito de possibilitar o acesso do ambiente **SDIP** e, conseqüentemente, de seus usuários ao sistema Archie, construímos um processo que implementa o protocolo de comunicação utilizado por estes servidores de informações, permitindo-lhes acessar as bases de dados por eles gerenciadas.

No desenvolvimento do processo que desempenha as funções de um cliente Archie, adotamos uma filosofia de projeto seguida na implementação de todos os demais processos constituintes do ambiente **SDIP**.

Mais precisamente, ao contrário de se implementar o cliente Archie como um conjunto de funções intrínsecas, internas ao ambiente, optou-se por construir-se um processo individual, onde está implementado esse conjunto de procedimentos.

A integração deste processo ao **SDIP** é possível graças a uma API (Application Program Interface), através da qual fluem todos os comandos destinados ao processo Archie, bem como as respostas por ele devolvidas ao sistema.

Adotando-se esta solução, além de resolvermos o problema particular do **SDIP**, também conseguimos desenvolver uma interface que pode constituir-se em uma boa solução para outros pesquisadores que pretendam construir aplicações que usem as facilidades do sistema Archie. Neste caso, não será preciso dispor-se de tempo para o desenvolvimento de um novo cliente Archie, mas, tão somente, estudar a API já implementada e testada.

8.2 Descrição do Processo Archie

O processo onde estão implementados os procedimentos e a API que permitem ao **SDIP** usar os serviços providos pelos servidores Archie, foi codificado integralmente em linguagem C padrão [KER 78, SUN 88, SUN 89a].

Como destacamos anteriormente, a interação do **SDIP** com o **Processo Archie** dá-se através de uma API, sendo indiferente para o sistema o fato de o processo estar sendo executado local ou remotamente.

Dentre os comandos reconhecidos por este processo, podemos destacar:

- Inicialização do processo;
- Término de execução;
- Especificação do servidor Archie a consultar;
- Determinação do número máximo de respostas aceitáveis por parte do processo de interface;
- Atribuição de uma determinada prioridade à consulta;
- Especificação dos padrões de busca usados para a pesquisa de arquivos e diretórios, trabalho realizado pelo servidor Archie consultado;
- Solicitação do tempo transcorrido, desde o início até a conclusão de uma consulta;
- Liberação, por parte do sistema, para o início da consulta a um servidor Archie.

Finalmente, note-se que a efetivação de uma consulta a um servidor Archie é composta, obrigatoriamente, por um conjunto de chamadas de funções ou comandos sucessivos. Assim, primeiramente, deve-se enviar o comando de inicialização

do processo, havendo, neste momento, um ajuste do conteúdo de suas variáveis e estruturas internas para valores pré-determinados. Conseqüentemente, este comando somente deve ser usado uma única vez em cada execução do **Processo Archie**.

Posteriormente, deve-se, pelo menos, informar ao processo os padrões de busca que devem ser usados na pesquisa, caso os demais parâmetros de operação, ajustados anteriormente para valores pré-determinados, estejam de acordo com o esperado pelo usuário. Neste momento, o **SDIP** poderia iniciar uma consulta, enviando um comando de liberação para o processo de interfaceamento.

Todavia, caso o servidor Archie a ser consultado diferisse daquele ajustado no momento da inicialização, teríamos de enviar um comando de especificação de servidor, antes que pudéssemos dar início à consulta. Este é somente um exemplo, entre muitos outros que poderíamos citar, no qual teríamos de modificar algum parâmetro de operação do processo que não estivesse de acordo com o desejado para, somente depois, podermos dar início a consulta.

8.2.1 Protocolo de Comunicação

Conforme tivemos a oportunidade de destacar no capítulo 2, os clientes Archie, executando localmente na estação de trabalho do usuário, interagem com os seus servidores através do protocolo de comunicação **Prospero**, desenvolvido especialmente para atender às necessidades particulares de um sistema de mesmo nome [NEU 89, NEU 89a, NEU 92, NEU 92a, NEU 92b].

Em linhas gerais, o protocolo Prospero foi projetado para ser utilizado em sistemas cuja arquitetura segue o modelo cliente-servidor.

Adicionalmente, cada mensagem de requisição ou de resposta, enviadas respectivamente pelos processos clientes e servidores, estão estruturadas em uma seqüência de linhas de texto, assemelhando-se, neste particular, ao protocolo *Gopher*. Quando se tratar de uma mensagem de requisição, ela poderá conter um ou mais comandos, os quais serão executados seqüencialmente pelo processo servi-

dor. Posteriormente, as respostas obtidas com a execução destes comandos serão encaminhadas ao processo cliente, encapsuladas em uma mensagem de resposta.

Ao contrário dos demais servidores de informações descritos neste trabalho, os quais utilizam o TCP [COM 88, TAN 89, COM 91] para o transporte confiável de dados, o sistema **Prospero**, mais precisamente o seu protocolo de comunicação, usa as facilidades e serviços providos pelo ("User Datagram Protocol" (UDP)) [COM 88, TAN 89, COM 91] para esta finalidade. Por esta razão, fez-se necessária a implementação de uma camada ou nível intermediário situado entre o UDP e o protocolo Prospero, denominado ("Asynchronous Reliable Delivery Protocol" ARDP)), responsável pelo controle de seqüenciamento dos pacotes de informações trocados pelos processos clientes e servidores.

No que concerne à utilização do protocolo Prospero dentro do sistema Archie, tornando possível a comunicação entre os processos clientes e servidores, somente uma pequena parcela dos diversos comandos e respostas implementadas por este protocolo são de fato usadas. Mais precisamente, apenas os comandos associados à pesquisa em diretórios, e as respostas deles decorrentes, são reconhecidos respectivamente pelos servidores e clientes Archie.

O conjunto dos comandos que permitem a realização de uma pesquisa de informações são conduzidos ao servidor encapsulados em uma única unidade de dados, estruturada em três linhas de texto consecutivas, as quais contêm respectivamente as seguintes informações:

1. Especificação da versão do protocolo Prospero reconhecida pelo processo cliente;

2. Informações para autenticação do usuário, a qual, presentemente, é constituída apenas pelo seu ("Username") local;

3. Finalmente, o comando de pesquisa e o padrão de busca que deve ser utilizado para a localização das informações desejadas. Neste mesmo campo aparecem parâmetros adicionais que determinam o número máximo de

respostas requeridas pelo usuário, quantidade de ocorrências válidas que devem ser desconsideradas[1] e a modalidade de pesquisa que deve ser executada[2].

Por outro lado, as respostas produzidas por uma consulta são enviadas ao processo cliente encapsuladas em uma ou mais unidades de dados, igualmente estruturadas em uma seqüência de linhas de texto, as quais contêm informações completas a respeito dos arquivos e/ou diretórios localizados pelo servidor inicialmente consultado.

8.2.2 Especificação do Processo Archie

A seguir, descrevemos o comportamento do **Processo Archie**, utilizando-nos de um formalismo adequado para este fim. Juntamente com a especificação em SDL-PR [INT 88, BEL 91], mostrada nas tabelas 8.1, 8.2 e 8.3, são feitos alguns comentários que ajudarão o leitor a compreender o funcionamento do processo estudado neste capítulo.

[1] Este parâmetro permite a recuperação parcial de informações. Assim, em cada consulta formulada ao servidor, avança-se o contador de respostas que devem ser desconsideradas

[2] Esta informação define o tipo de casamento permitido entre o padrão de busca e as entradas do diretório. Neste caso, podemos especificar, por exemplo, que somente serão consideradas respostas válidas para uma determinada consulta os nomes de arquivos e diretórios que forem idênticos ao padrão de busca fornecido pelo usuário.

Tabela 8.1 Especificação em SDL-PR do Processo Archie.

BLOCK	B_ARC;	/* DEF. DO MÓDULO ARCHIE*/
SIGNALROUTE R1		
FROM ENV	TO P_CREATE	WITH S_CR
FROM P_CREATE	TO ENV	WITH S_ARC_IG;
SIGNALROUTE R2		
FROM ENV	TO P_ARCHIE	WITH S_IG_ARC
FROM P_ARCHIE	TO ENV	WITH S_ARC_IG;
SIGNALROUTE R3		
FROM ENV	TO P_ARCHIE	WITH S_ENV_ARC
FROM P_ARCHIE	TO ENV	WITH S_ARC_ENV;
CONNECT C_ARC_IG	AND	R1,R2;
CONNECT C_ARC_ENV	AND	R3;

Tabela 8.2 Especificação em SDL-PR do Processo Archie.

PROCESS	P_CREATE;	/*INICIALIZA O PROCESSO ARCHIE*/
START;		
STATE	ST_CREATION;	
INPUT	S_CR;	/*SIGNAL USADA CRIAR UM PROC.*/
CREATE	P_ARCHIE;	
OUTPUT	S_ARC_IG;	
STOP;		
ENDSTATE	ST_CREATION;	
ENDPROCESS	P_CREATE;	

Tabela 8.3 Especificação em SDL-PR do Processo Archie.

PROCESS	P_ARCHIE;	/*PROCESSO ARCHIE*/
STATE	ST_MAIN;	
INPUT	S_IG_ARC(CMD);	/*COMANDOS PARA O PROCESSO*/
DECISION	CMD;	
('CONFIGURAÇÃO'):		/* COMANDOS QUE CONFIGURAM O PROCESSO*/
NEXTSTATE	ST_CONFIGURE;	
('CONSULTA'):		/*INICIA UMA CONSULTA*/
OUTPUT	S_ARC_ENV;	/*CONSULTA SERV. ARCHIE*/
NEXTSTATE	ST_QUERY;	
('PARAR'):		
OUTPUT	S_ARC_IG;	
STOP;		
ENDDECISION;		
ENDSTATE	ST_MAIN;	
STATE	ST_CONFIGURE;	/*EXECUTA OS COMANDOS DE CONFIGURAÇÃO*/
...		
OUTPUT	S_ARC_IG;	
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_CONFIGURE;	
STATE	ST_QUERY	
INPUT	S_ENV_ARC;	/*RESPOSTA DO SERV. ARCHIE*/
OUTPUT	S_ARC_IG;	/*RESPONDE
NEXTSTATE	ST_MAIN;	INICIAL*/
ENDSTATE	ST_QUERY;	
ENDPROCESS	P_ARCHIE;	
ENDBLOCK	B_ARC;	

9 RECUPERAÇÃO DE ARQUIVOS

9.1 Introdução

No capítulo onde descrevemos o sistema Archie (capítulo 2), tivemos a oportunidade de destacar a grande quantidade de servidores FTP públicos hoje existentes dentro da Internet, os quais disponibilizam aos usuários da rede um vasto acervo de informações, abrangendo inúmeras áreas do conhecimento humano.

Um dos objetivos principais do ambiente **SDIP** é oferecer aos seus usuários um componente integrado dentro do sistema, que lhes possibilitasse pesquisar os servidores FTP públicos, diretórios e arquivos descobertos pelo sistema Archie, bem como recuperar aqueles arquivos que satisfazem as suas necessidades.

Todavia, para tornar realidade este objetivo, tivemos de superar uma dificuldade que, certamente, será enfrentada por qualquer pesquisador que se proponha a desenvolver um sistema no qual esteja presente esta facilidade. Estamos referindo-nos ao fato de que não há disponível atualmente, dentro da Internet, um sistema que, concomitantemente, implemente o protocolo de transferência de arquivos FTP [POS 85] e uma API que permita a outras aplicações usarem transparentemente este serviço. Os sistemas hoje existentes constituem-se, tão somente, em clientes que interagem diretamente com os usuários, ao mesmo tempo em que o protocolo FTP está implementado como parte integrante do sistema, sendo, por esta razão, inseparável do conjunto.

Desta maneira, para sermos capazes de oferecer esta facilidade aos usuários do sistema SDIP, tivemos de desenvolver e implementar um processo capaz de trabalhar com o protocolo de transferência de arquivos acima referido. No entanto, ao contrário dos demais sistemas que implementam internamente este protocolo, adotamos uma solução similar àquela adotada para o sistema Archie e SABI, ou seja, construímos um processo que implementa o protocolo de transferência de

arquivos, mas interage com o ambiente **SDIP** através de uma API. Desta forma, acreditamos que o processo ora em estudo não servirá apenas para atender às necessidades individuais do SDIP, mas poderá ser uma solução genérica para sistemas que requeiram tais facilidades para a recuperação de arquivos remotos.

9.2 O Processo FTP do SDIP

O processo onde está implementado o protocolo de transferência de arquivo FTP e a API que permite o acesso transparente do **SDIP** a este serviço foi escrito completamente em linguagem C padrão [KER 78, SUN 88, SUN 89a].

Particularmente, no que se refere ao protocolo de transferência de arquivos, implementou-se somente um subconjunto da especificação completa do protocolo FTP [POS 85], basicamente as funções relacionadas com o estabelecimento da conexão com o servidor remoto, abertura de sessão, transferência de arquivos em ambos os sentidos e mudança de diretório, tanto remoto quanto local. As demais funcionalidades como, por exemplo, a criação de um diretório no servidor remoto, deverão ser implementadas nas versões subseqüentes deste processo.

Como vimos na seção introdutória deste capítulo (9.1), toda a interação do ambiente SDIP, mais precisamente da **Interface Gráfica** com o **Processo FTP**, ocorre através de uma API especialmente construída para este propósito.

A seguir, mostramos as principais operações executadas pelo **Processo FTP**, invocadas pela **Interface Gráfica** por intermédio da API existente.

1. Inicialização do processo servidor;
2. Término da execução do processo considerado;
3. Estabelecimento da conexão com o servidor FTP remoto;
4. Abertura de uma sessão de transferência de arquivos;
5. Transferências de arquivos em ambos os sentidos;

6. Especificação do tipo dos dados a serem transmitidos;
7. Transferência múltipla de arquivos em ambos os sentidos;
8. Tempo gasto na transferência dos dados;
9. Quantidade de bytes transmitidos;
10. Estado de operação do processo servidor de interface;
11. Troca de diretório, tanto remoto quanto local;
12. Fechamento de sessão FTP de transferência de arquivo.

9.3 Especificação do Processo FTP

A seguir, mostramos nas tabelas 9.1, 9.2 e 9.3, a representação em SDL-PR [INT 88, BEL 91] do processo estudado neste capítulo, o qual opera como um cliente FTP. Comentários pertinentes a respeito de detalhes da especificação estão inseridos na própria representação SDL-PR, segundo as regras definidas pela linguagem.

Tabela 9.1 Especificação em SDL-PR do Processo FTP.

BLOCK	B_FTP;	/* DEF. DO MÓDULO FTP*/
SIGNALROUTE R1		
FROM ENV	TO P_CREATE	WITH S_CR
FROM P_CREATE	TO ENV	WITH S_FTP_IG;
SIGNALROUTE R2		
FROM ENV	TO P_FTP	WITH S_IG_FTP
FROM P_FTP	TO ENV	WITH S_FTP_IG;
SIGNALROUTE R3		
FROM ENV	TO P_FTP	WITH S_ENV_FTP_1, S_ENV_FTP_2
FROM P_FTP	TO ENV	WITH S_FTP_ENV_1, S_FTP_ENV_2;
CONNECT C_FTP_IG	AND	R1,R2;
CONNECT C_FTP_ENV	AND	R3;

Tabela 9.2 Especificação em SDL-PR do Processo FTP.

PROCESS	P_CREATE;	/*INICIALIZA O PROC. FTP*/
START;		
STATE	ST_CREATION;	
INPUT	S_CR;	/*SOLICITA A CRIAÇÃO DE UM PROCESSO*/
CREATE	P_FTP;	
OUTPUT	S_FTP_IG;	
STOP;		
ENDSTATE	ST_CREATION;	
ENDPROCESS	P_CREATE;	

Tabela 9.3 Especificação em SDL-PR do Processo FTP.

PROCESS	P_FTP;	/*PROCESSO FTP*/
STATE	ST_MAIN;	
INPUT	S_IG_FTP(CMD);	/*SIGNAL TRÁS */
		OS COMANDOS*/
DECISION	CMD;	
('CONFIGURAÇÃO'):		/* COMANDOS QUE
		CONFIGURAM O PROC.*/
OUTPUT	S_FTP_ENV_1;	/*ENVIA COMANDO
		PARA O SERV. FTP*/
NEXTSTATE	ST_CONFIGURE;	
('LS,GET'):		/*RECEBE
		ARQ. OU DIR.*/
OUTPUT	S_FTP_ENV_1;	/*ENVIA COMANDO
		PARA O SERV. FTP*/
NEXTSTATE	ST_TRANSFER;	
('PARAR'):		
OUTPUT	S_FTP_IG;	
STOP;		
ENDDECISION;		
ENDSTATE	ST_MAIN;	
STATE	ST_CONFIGURE;	
INPUT	S_ENV_FTP_1;	/*RECEBE RESP.
		DO SERV. FTP*/
OUTPUT	S_FTP_IG;	
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_CONFIGURE;	
STATE	ST_TRANSFER;	
INPUT	S_ENV_FTP_1;	/*RESPOSTA DO
		SERV. FTP*/
INPUT	S_ENV_FTP_2;	/*RECEBE
		ARQ. OU DIR. */
OUTPUT	S_FTP_IG;	/*RESPONDE AO
		USUÁRIO */
OUTPUT	S_FTP_ENV_2;	/*GRAVA O
		ARQ. LOCAL*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_TRANSFER;	
ENDPROCESS	P_FTP;	
ENDBLOCK	B_FTP;	

10 ACESSANDO O SISTEMA SABI ATRAVÉS DO CORREIO ELETRÔNICO

10.1 Introdução

O ambiente SDIP, além de seu módulo de interface gráfica, destinado a atender aos usuários que desejam utilizar interativa e integralmente as facilidades e serviços oferecidos pelo sistema, também possui uma forma alternativa de interfaceamento através de mensagens de correio eletrônico, trocadas entre os usuários finais e o sistema. Assim, usuários que não têm acesso aos recursos gráficos de uma estação de trabalho, podem fazer uso de algumas facilidades do SDIP, sendo necessário apenas enviar-lhe uma mensagem de E-Mail e aguardar a resposta do sistema, também enviada ao usuário por intermédio do correio eletrônico.

A incorporação deste tipo de interface ao sistema **SDIP**, deveu-se a dois fatores principais:

1. Esta forma alternativa de interface torna o sistema disponível a um grande número de usuários, os quais, em sua esmagadora maioria, não têm acesso a recursos computacionais com as facilidades e serviços gráficos que lhes permitam usar o outro módulo de interface;

2. Existência de um protótipo desenvolvido e implementado em um trabalho anterior [FER 93], cuja concepção e objetivos básicos seguem a mesma filosofia e caminhos propostos neste trabalho.

   Na realidade, aquele protótipo foi o percursor do sistema aqui descrito, uma vez que nele trabalhou-se no uso de sentenças em língua portuguesa, aplicando-as apenas à recuperação de dados bibliográficos.

   Neste trabalho, com efeito, usa-se esta metodologia para recuperar informações genéricas, ampliando-se, portanto, os horizontes anteriormente propostos.

10.2 Funcionamento Básico

Como dissemos na seção anterior 10.1, toda a interação do usuário com este componente do sistema, dá-se única e exclusivamente por intermédio de mensagens trocadas entre os participantes através do correio eletrônico.

Mais precisamente, o usuário envia uma mensagem, contendo em seu corpo uma ou mais sentenças em língua portuguesa, as quais serão recebidas e interpretadas pelo sistema, permitindo-lhe determinar e executar o conjunto de operações nelas expressas implicitamente.

Posteriormente, o sistema enviará ao usuário, também através de uma mensagem de correio eletrônico, os resultados obtidos quando do término das operações anteriormente referidas.

Infelizmente, somente uma parcela das facilidades implementadas junto à interface gráfica são oferecidas aos usuários que se utilizam do correio eletrônico para contactar o **SDIP**, ou seja, apenas estão disponíveis os serviços associados à recuperação de informações bibliográficas junto ao sistema SABI.

Esta limitação deve-se tão somente ao fato de que o uso dos serviços relacionados com os servidores Archie e FTP, requerem um diálogo interativo entre o usuário e o sistema (principalmente no momento em que se deseja pesquisar diretórios e recuperar arquivos), o que é inviável de ser obtido através do correio eletrônico.

Conceitualmente, o funcionamento deste módulo do ambiente **SDIP** é bastante simplificado. Em linhas gerais, este processo opera em um ciclo, no qual, em um primeiro momento, ele recebe as mensagens enviadas pelos usuários, executa as operações por elas expressas e, finalmente, responde aos usuários, enviando-lhes uma mensagem contendo as informações bibliográficas recuperadas.

Graças a sua natureza cíclica e ao fato de que as mensagens de correio eletrônico são caracteristicamente assíncronas, podendo chegar a qualquer momento,

optou-se por ativar o processo de interface em intervalos regulares de tempo (uma hora), ao contrário de executá-lo a cada mensagem que chegasse.

No momento de sua ativação, o processo de interface verifica se existem mensagens de usuários armazenadas no sistema. Na hipótese de que no último intervalo de uma hora não tenha sido recebida nenhuma mensagem, o processo retorna ao seu estado de espera, permanecendo assim até o próximo momento de sua execução.

No entanto, se no último lapso de tempo o sistema houver recebido alguma mensagem, o processo iniciará a execução de uma série de passos que somente estarão concluídos quando for processada a última mensagem armazenada. Neste momento, o processo passa novamente para o estado de espera, até ser novamente despertado pelo sistema.

O algoritmo executado com o intuito de processar as mensagens recebidas está sumarizado a seguir. É importante notar que não foram incluídos neste algoritmo resumido o tratamento de erros que por ventura ocorram em qualquer fase do processamento das mensagens.

1. O processo lê todas as mensagens recebidas e armazenadas pelo sistema, retirando das mesmas as informações relevantes, ou seja, o endereço do remetente e o conteúdo do corpo;

2. Para cada uma das mensagens faça:

   (a) Interpretação do significado das sentenças com o auxílio do processo apropriado;

   (b) Recuperação, através do **processo de interface com o sistema SABI**, das informações bibliográficas solicitadas na sentença em língua portuguesa;

   (c) Envio da resposta ao usuário por intermédio de uma mensagem de correio eletrônico.

10.3 Implementação e Especificação do Processo

No que concerne à implementação deste módulo do ambiente **SDIP**, ele foi desenvolvido agrupando-se duas linguagens de programação: uma de nível médio (linguagem C) [KER 78, SUN 88, SUN 89a] e outra orientada a objeto (C++) [ELL 90, SUN 89]. Adicionalmente, utilizou-se duas ferramentas para a geração de software. Para a construção de um analisador capaz de reconhecer os ítens léxicos por nós definidos, os quais compõe uma mensagem de correio eletrônico, usamos o **Lex** [MAS 90]. Finalmente, para gerar o analisador capaz de reconhecer a estrutura sintática de uma mensagem, utilizamo-nos da ferramenta **YACC** [MAS 90].

A seguir, mostramos a especificação em SDL-PR [INT 88, BEL 91] deste componente do ambiente **SDIP** (tabelas 10.1 e 10.2). Alguns comentários foram inseridos na própria representação com o intuito de auxiliar o leitor a compreender o funcionamento lógico do processo estudado. É importante notar que os programas gerados por ambas as ferramentas (**Lex** e **YACC**) não foram mostrados detalhadamente nesta especificação.

Tabela 10.1 Especificação em SDL-PR do Servidor de Correio Eletrônico.

```
BLOCK                   B_SCE;          /* DEF. DO MÓDULO
                                        SCE*/
SIGNALROUTE R1
FROM P_SCE              TO ENV          WITH S_SCE_ENV
FROM ENV                TO P_SCE        WITH S_ENV_SCE;

SIGNALROUTE R2
FROM P_SCE              TO ENV          WITH S_SCE_FS,S_CR
FROM ENV                TO P_SCE        WITH S_FS_SCE;

SIGNALROUTE R3
FROM P_SCE              TO ENV          WITH S_SCE_PIS,S_CR
FROM ENV                TO P_SCE        WITH S_PIS_SCE;


CONNECT C_SCE_ENV       AND             R1;
CONNECT C_SCE_SABI      AND             R2
CONNECT C_SCE_PIS       AND             R3;
```

Tabela 10.2 Especificação em SDL-PR do Servidor de Correio Eletrônico.

```
PROCESS     P_SCE;              /*PROCESSO SCE*/
TIMER       TIME;               /*RELÓGIO*/
START;
STATE       ST_MAIN;
TASK        SET(3600,TIME);     /*PROCESSO AGUARDA
                                POR UMA HORA*/
OUTPUT      SYSTEM SDIP/        /*ATIVA O PROC.
            BLOCK B_SCE/        DE INTERFACE
            R2 SCR;             COM O SABI*/
OUTPUT      SYSTEM SDIP/        /*ATIVA O PROC.
            BLOCK B_SCE/        INTERPRETADOR
            R3 S_CR;            DE SENTENÇAS*/
NEXTSTATE   ST_READ;
ENDSTATE    ST_MAIN;

STATE       ST_READ;
INPUT       S_ENV_SCE;          /*RECEBE AS
                                MENSAGENS DOS
                                USUÁRIOS*/
...
NEXTSTATE   ST_INTERPRET;
ENDSTATE    ST_READ;

STATE       ST_INTERPRET;
OUTPUT      S_SCE_PIS;          /*ENVIA SENTENÇAS
                                PARA O PIS*/
INPUT       S_PIS_SCE;          /* RECEBE
                                RESPOSTA */
NEXTSTATE   ST_QUERY;
ENDSTATE    ST_INTERPRET;

STATE       ST_QUERY;
OUTPUT      S_SCE_FS;           /*CONSULTA AO SABI*/
INPUT       S_FS_SCE;           /*RECEBE RESPOSTA*/
NEXTSTATE   ST_SEND;
ENDSTATE    ST_QUERY;

STATE       ST_SEND;
OUTPUT      S_SCE_ENV;          /*RESPONDE AOS
                                USUÁRIOS */
OUTPUT      S_SCE_PIS(KILL);    /*MATA O PIS*/
OUTPUT      S_SCE_FS(KILL);     /*MATA O IS*/
NEXTSTATE   ST_MAIN;
ENDSTATE    ST_SEND;
```

11 A INTERFACE COM O SISTEMA SABI

11.1 Introdução

A Universidade Federal do Rio Grande do Sul mantém, junto ao Centro de Processamento de Dados, um sistema para automação de seus serviços de catalogação e recuperação de informações bibliográficas. Este sistema, denominado SABI (Sistema para a Automação de Bibliotecas), permite à comunidade acadêmica, ou pelo menos àquela parcela que têm acesso a um computador conectado à rede da Universidade, realizar pesquisas bibliográficas sobre o acervo literário catalogado no sistema.

Não obstante às enormes vantagens advindas do uso do SABI para a gerência de informações bibliográficas, a versão atual do sistema é bastante limitada no que diz respeito à interface, seja para usuários humanos quanto para outros programas de aplicação.

Sob o ponto de vista do usuário humano, a utilização eficiente de toda a potencialidade do sistema requer o profundo conhecimento da linguagem de consulta própria do SABI (CDS-ISIS), o que, via de regra, é de domínio apenas dos bibliotecários e bibliotecárias, uma vez que esses profissionais usam habitualmente o sistema.

Por outro lado, sob o ponto de vista de outras aplicações, não existe implementado, dentro do sistema SABI, uma API que permite a esses programas acessar tão somente o gerenciador de dados bibliográficos, tornando possível, por exemplo, a construção de interfaces que melhor respondam às necessidades dos usuários não especializados.

Uma das facilidades oferecidas aos usuários pelo ambiente **SDIP** é a possibilidade de recuperar informações bibliográficas de sistemas similares ao SABI, seja pela interface gráfica, bem como através de sentenças em língua portuguesa.

No último caso, as consultas podem ser formuladas interativamente ou através do correio eletrônico, colocando-se as sentenças apropriadas no corpo da mensagem.

Para tornar factível a implementação desse serviço, foi imperativo que, de alguma forma, superássemos as limitações e barreiras acima expostas. Para tanto, construímos um processo que, por um lado, fornece às demais aplicações uma API simples e clara para o uso efetivo do SABI e, por outro lado, comporta-se como um usuário humano na interação com esse sistema, funcionando como um intérprete que traduz comandos da API para a linguagem de consulta CDS-ISIS, além de comandos adicionais de configuração do processo ora em estudo.

A partir desse momento, iniciamos o estudo detalhado dos dois processos nos quais estão implementadas as funcionalidades que permitem ao **SDIP** interagir com o sistema SABI.

11.2 Os Processos de Interface com o SABI

De maneira análoga ao componente do **SDIP**, responsável pelo tratamento das consultas dos usuários encaminhadas ao sistema através de mensagens do correio eletrônico, o processo dentro do qual está implementada uma API que permite a interação com o SABI também constitui-se em uma extensão ao protótipo desenvolvido anteriormente, descrito em [FER 93]

Todavia, a metodologia adotada presentemente para a comunicação com o sistema SABI, difere substancialmente da que foi proposta no trabalho precedente.

Na versão atual, a associação entre o processo de interface e o sistema SABI é iniciada por intermédio da função *rcmd()*, a qual, basicamente, estabelece uma conexão de nível de transporte entre duas entidades, executando, no computa-

dor remoto, o comando "*login*". Adicionalmente, a função *rcmd()* cria um (*socket*[1]) [COM 88] para onde serão direcionadas todas as operações de entrada e saida realizadas pelo processo executado no computador remoto.

No momento em que forem enviados o "*username*" e a "*password*" corretas, os dois processos participantes – componente de interface e sistema SABI –, podem trocar informações através do "*socket*", criado quando do estabelecimento da conexão.

Note-se que nesta abordagem há um melhor aproveitamento de recursos e um aumento de eficiência na comparação com a proposta anterior, uma vez que o uso do protocolo *TelNet* requer maior quantidade de recursos, tanto computacionais, quanto de comunicação para chegar a resultados finais idênticos aos obtidos no presente protótipo.

Por motivos de implementação, os procedimentos de interfaceamento com o sistema SABI são executados por dois processos distintos. No primeiro, denominado (**Front-end SABI**), está implementada a API usada diretamente pelo **SDIP**. O segundo processo, denominado (**Interface SABI**), é, de fato, o componente responsável pela interação com o sistema SABI. Adicionalmente, ambos os processos compartilham a característica comum de terem sido escritos tão somente em linguagem C padrão [KER 78, SUN 88, SUN 89a].

O papel desempenhado por cada um destes dois processos dentro do ambiente **SDIP** está bem delineado, podendo ser facilmente compreendido.

O **Front-end SABI** recebe e trata todos os comandos provenientes da interface gráfica ou do processo servidor de correio eletrônico. Este conjunto de comandos serve para modificar o contexto de operação deste componente do sistema, determinando as características que governam a interação com o SABI.

[1] Estrutura lógica que permite a um processo enviar e receber dados sobre uma rede de comunicação de dados.

A seguir, mostramos todos os parâmetros de configuração modificáveis existentes, bem como todas as operações exeqüíveis por este componente do **SDIP**, sendo, em ambos os casos, acessados através da API implementada.

- Computador remoto onde reside o sistema SABI ou similar;
- Identificação do usuário, a qual permite o acesso público e remoto ao sistema;
- Determinação do tempo máximo permitido em cada transição da máquina de estados que representa os estágios da interação entre os sistemas;[2]
- Especificação do autor, título, assunto e editora, valores estes usados em uma consulta simples;
- Especificação de uma expressão na linguagem CDS-ISIS, usada em uma consulta complexa;
- Nome dos arquivos usados para a comunicação entre os dois processos (**Front-end** e **Interface SABI**);
- Comando de ativação de uma consulta simples ou complexa;[3]
- Comando de inicialização do componente de interface, com o respectivo ajuste para a configuração inicial;
- Aviso de término de execução da interface.

Observando o conjunto de comandos e operações descritas anteriormente, podemos verificar que a geração das expressões na linguagem CDS-ISIS, capazes de satisfazer as consultas formuladas pelos usuários, não é realizada por este componente do sistema. De fato, conforme o tipo de interface utilizada, esta tarefa é

[2]Este parâmetro é usado para garantir um tempo de resposta aceitável na comunicação.

[3]Esta operação somente pode ser executada no momento em que os demais parâmetros de configuração estiverem devidamente ajustados.

executada alternativamente pelo Processo Interpretador de Sentenças ou pela Interface Gráfica.

Por sua vez, a **Interface SABI**, baseada no contexto de operação mantido pelo processo anterior, estabelece a comunicação com o SABI, recupera os dados bibliográficos e os devolve ao **Front-end**, concluindo, neste momento, a sua operação.

Desta forma, o leitor pode constatar que a **Interface SABI** somente é ativada pelo **Front-end**, no momento em que todos os parâmetros de uma consulta estiverem ajustados adequadamente, posto que o funcionamento deste processo depende fundamentalmente da correção e integridade de tais informações.

A comunicação entre esses dois processos é efetivada de uma maneira que difere da solução adotada para os demais componentes do sistema.

O **Front-end** comunica-se com a **Interface SABI**, no momento de sua execução, através da forma normal de passagem de parâmetros entre processos, provido pelo sistema Unix.

Por outro lado, a comunicação no sentido inverso, dá-se através de dois arquivos convencionais. O primeiro contém um código e uma mensagem que indicam o resultado da operação de consulta, no tocante a seu sucesso ou fracasso. No segundo arquivo estão armazenados os dados bibliográficos recuperados, caso a consulta tenha transcorrido normalmente.

11.3 Especificação dos Processos

Nesta seção, mostramos a especificação em SDL-PR [INT 88, BEL 91] de cada um dos processos que formam o componente responsável pela interação com o sistema SABI (tabelas 11.1, 11.2, 11.3 e 11.4).

Tabela 11.1 Especificação em SDL-PR do módulo de interface com o SABI.

BLOCK	B_SABI;	/* DEF. DO MIS*/
SIGNAL	S_RESP;	/*CONDUZ DADOS DO IS PARA O FS*/
SIGNALROUTE R1		
FROM P_IS	TO P_FS	WITH S_RESP;
SIGNALROUTE R2		
FROM P_IS	TO ENV	WITH S_IS_ENV
FROM ENV	TO P_IS	WITH S_ENV_IS;
SIGNALROUTE R3		
FROM P_CREATE	TO ENV	WITH S_FS_IG
FROM ENV	TO P_CREATE	WITH S_CR;
SIGNALROUTE R4		
FROM P_CREATE	TO ENV	WITH S_FS_SCE
FROM ENV	TO P_CREATE	WITH S_CR;
SIGNALROUTE R5		
FROM P_FS	TO ENV	WITH S_FS_IG
FROM ENV	TO P_FS	WITH S_IG_FS;
SIGNALROUTE R6		
FROM P_FS	TO ENV	WITH S_FS_SCE
FROM ENV TO P_FS	WITH S_SCE_FS;	
CONNECT C_SABI_ENV	AND	R2;
CONNECT C_SABI_IG	AND	R3,R5;
CONNECT C_SCE_SABI	AND	R4,R6;

Tabela 11.2 Especificação em SDL-PR do módulo de interface com o SABI.

PROCESS	P_CREATE;	/*INICIALIZA O PROC. FS*/
START;		
STATE	ST_CREATION;	
INPUT	SYSTEM SDIP/ BLOCK B_SABI/ R3 S_CR, SYSTEM SDIP/ BLOCK B_SABI / R4 S_CR;	/*SIGNAL USADA PARA CRIAR UM PROC. */
CREATE	P_FS;	
DECISION	SENDER;	
(P_IG):	OUTPUT S_FS_IG;	
(P_SCE):	OUTPUT S_FS_SCE;	
ENDDECISION;		
STOP;		
ENDSTATE	ST_CREATION;	
ENDPROCESS	P_CREATE;	

Tabela 11.3 Especificação em SDL-PR do módulo de interface com o SABI.

PROCESS	P_FS;	/*PROCESSO FS*/
DCL	SEND;	
STATE	ST_MAIN;	
INPUT	S_IG_FS(CMD),	/*SIGNAL TRÁS
	S_SCE_FS(CMD);	OS COMANDOS
		PARA O PROC.*/
TASK	SEND=SENDER;	
DECISION	CMD;	
('CONFIGURAÇÃO'):		/* COMANDOS QUE
		CONFIGURAM O
		PROCESSO*/
NEXTSTATE	ST_CONFIGURE;	
('CONSULTA'):		
CREATE	P_IS;	/*CRIA O PROC.
		QUE INTERAGE
		COM O SABI*/
NEXTSTATE	ST_QUERY;	
('PARAR'):		
DECISION	SENDER;	
(P_IG):	OUTPUT S_FS_IG;	
(P_SCE):	OUTPUT S_FS_SCE;	
ENDDECISION;		
STOP;		
ENDDECISION;		
ENDSTATE	ST_MAIN;	
STATE	ST_CONFIGURE;	/*CONFIGURA OS
		PARÂMETROS DA
		CONSULTA */
...		
DECISION	SEND;	
(P_IG):	OUTPUT S_FS_IG;	
(P_SCE):	OUTPUT S_FS_SCE;	
ENDDECISION;		
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_CONFIGURE;	
STATE	ST_QUERY;	
INPUT	S_RESP;	/*RECEBE DADOS
		BIBLIOGRÁFICOS*/
DECISION	SEND;	
(P_IG):	OUTPUT S_FS_IG;	
(P_SCE):	OUTPUT S_FS_SCE;	
ENDDECISION		

Tabela 11.4 Especificação em SDL-PR do módulo de interface com o SABI.

PROCESS	P_IS;	/*INTERAGE COM O SABI*/
STATE	ST_MAIN;	
OUTPUT	S_IS_ENV;	/*COMANDOS PARA O SABI*/
INPUT	S_ENV_IS;	/*RECEBE RESPOSTAS*/
OUTPUT	S_RESP;	/*DEVOLVE DADOS BIBLIOGRÁFICOS PARA O FS*/
STOP;		
ENDSTATE	ST_MAIN;	
ENDPROCESS	P_IS;	
ENDBLOCK	B_SABI;	

12 A INTERFACE GRÁFICA

12.1 Introdução

O ambiente **SDIP** constitui-se em um sistema desenvolvido para atender principalmente os usuários não especializados na área de informática.

O projetista que se propõe a construir uma aplicação, cujo público alvo principal é constituído por pessoas com este perfil, deve ter uma atenção redobrada com o desenho e a funcionalidade da interface utilizada pelos usuários do sistema. Em suma, devemos construir um modelo de interface que, além de possibilitar o aproveitamento de todo o potencial do sistema, também seja de fácil utilização e compreensão por parte dos usuários.

Aproveitando-nos de todos os recursos de hardware e software colocados a nossa disposição por uma estação de trabalho sun, construímos uma interface de usuário baseada principalmente em menus, janelas e botões. Desta forma , acreditamos que o modelo de interface adotado no ambiente **SDIP** será fundamental para que os usuários possam explorar e aproveitar todas as facilidades implementadas pelo sistema.

12.2 Descrição do Processo da Interface Gráfica

Ao contrário dos demais processos componentes do **SDIP** que foram codificados diretamente em linguagem C padrão [KER 78, SUN 88, SUN 89a], a **Interface Gráfica do Usuário** foi desenvolvida inicialmente com o auxílio da ferramenta de construção de interfaces *Guide*.

Com o término da fase de planejamento e construção da interface, os programas gerados pelo *Guide* foram complementados com funções codificadas em C padrão, nas quais estão implementados os serviços e facilidades que permitem

o uso efetivo da interface. Esta complementação faz-se necessária, uma vez que a ferramenta *Guide* apenas gera os programas que desenham os menus e janelas que compõem a interface, cabendo ao projetista implementar as funções que permitem o uso prático da mesma.

Facilidades gráficas adicionais requeridas pelo sistema **SDIP**, não implementáveis através da ferramenta *Guide*, foram desenvolvidas com o uso das funções providas pela biblioteca gráfica *XView* [HEL 90].

Como podemos observar no capítulo 7, a **Interface Gráfica** é o processo central dentro do ambiente **SDIP**, posto que este é o único componente que possibilita aos usuários utilizar plenamente as capacidades do sistema, além de constituir-se no único ponto de entrada de conhecimento para o Processo Interpretador de Sentenças.

Adicionalmente, cabe ainda destacar que a **Interface Gráfica**, juntamente com o Processo Interpretador de Sentenças, desempenham funções ímpares dentro do **SDIP**, uma vez que ambos os processos são os únicos componentes do sistema que podem, dependendo da circunstância, funcionar como usuários ou provedores de serviços. No entanto, a **Interface Gráfica** distingue-se pelo fato de que opera como um servidor assíncrono de serviços, ou seja, o processo não fica preso aguardando uma requisição enviada pelo Interpretador de Sentenças. Quando Este processo necessita de um serviço da **Interface Gráfica**, ele a interrompe, assincronamente, enviando-lhe a solicitação apropriada.

A comunicação entre este processo – Interface Gráfica – com os demais componentes do ambiente **SDIP**, é realizada através das AIPs implementadas para cada um desses processos. Similarmente, o Processo Interpretador de Sentenças comunica-se com a **Interface Gráfica**, também através de uma API desenvolvida especialmente para esta finalidade.

A funcionalidade da **Interface Gráfica** é baseada, como dissemos, em um conjunto de janelas, menus e botões. Através dessas estruturas, o usuário envia

comandos, seleciona operações e responde a questionamentos formulados pelo sistema. A seguir, descrevemos e ilustramos por intermédio de figuras, as principais estruturas de diálogo presentes na **Interface Gráfica** do ambiente SDIP.

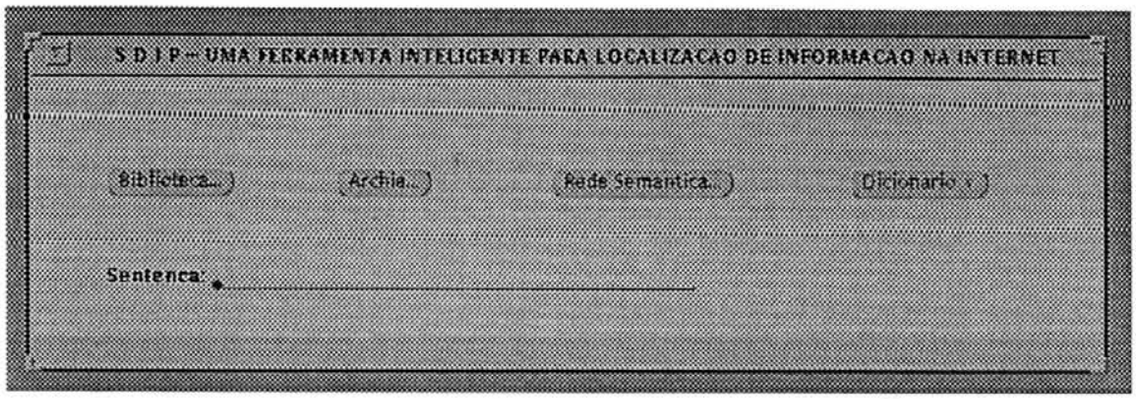

Figura 12.1 *Janela Principal do SDIP.*

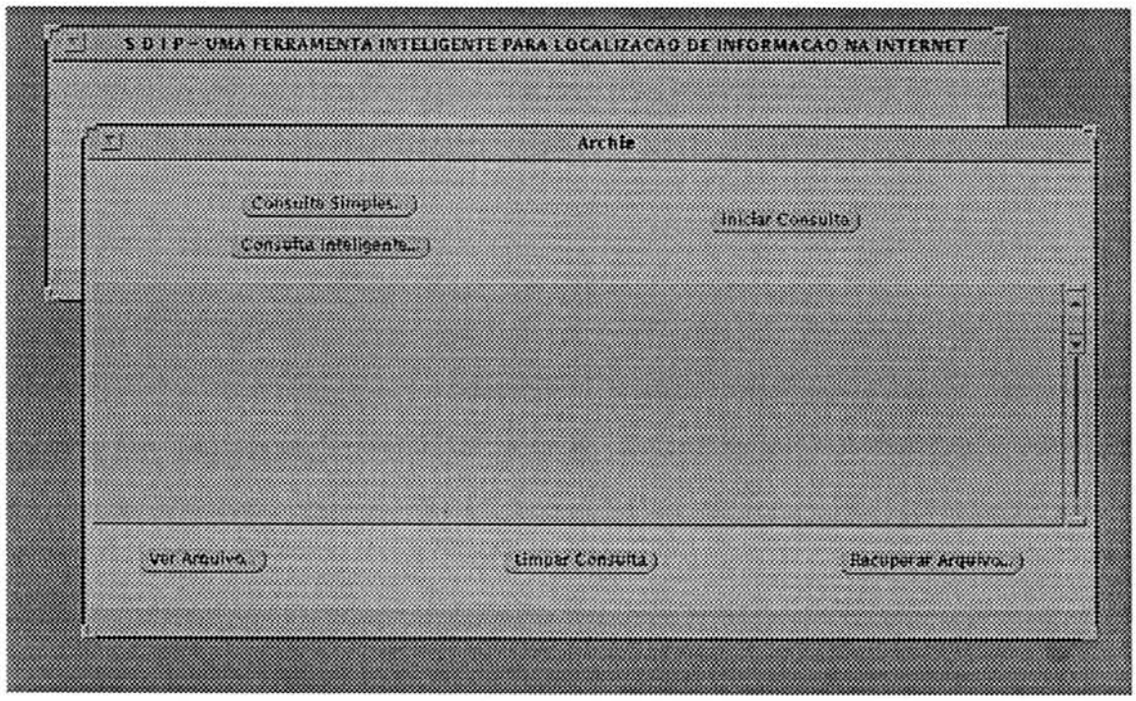

Figura 12.2 *Janela do Sistema Archie.*

1. Janela principal, mostrada na figura 12.1. Possui quatro botões principais, cada um associado a uma subfunção do sistema;

   (a) Janela do sistema Archie, mostrada na figura 12.2. Permite aos usuários a interação com o sistema Archie, possibilitando a realização

de uma consulta simples ou inteligente, visualização e recuperação de arquivos;

(b) Menu de acesso ao dicionário de palavras do sistema, ilustrado na figura 12.3. Usado para a inclusão de novos ítens léxicos no dicionário de palavras do sistema, como detalharemos no capítulo 14;

(c) Campo de frases, ilustrado na figura 12.4, usado para a especificação de uma sentença em língua portuguesa, a qual será posteriormente interpretada pelo sistema com o intuito de determinar os comandos por ela expressos;

(d) Janela para o acesso aos sistemas de consultas bibliográficas 12.5. Permite aos usuários a formulação de uma consulta aos servidores, visualização, salvamento em arquivo e envio dos resultados obtidos através do correio eletrônico, bem como inclusão de novos servidores, que podem ser consultados a qualquer momento;

(e) Janela para a manipulação da base de conhecimento do **Processo Interpretador de Sentenças**, mostrada na figura 12.6. Este conjunto de janelas, menus e botões, permite-nos incluir novas entradas na base de conhecimento, visualizar a rede semântica mantida pelo sistema e, também, excluir entradas da mesma.

2. Finalmente, temos os botões que são ativados assincronamente pelo Processo Interpretador de Sentenças, sempre que este tiver de aprender o significado de um discurso, como está ilustrado na figura 12.7.

Uma discussão detalhada a respeito do funcionamento da interface gráfica do ambiente **SDIP** realizar-se-á em um capítulo posterior deste trabalho, onde ilustraremos com exemplos claros e elucidativos a maneira adequada de utilizar-se este módulo do sistema.

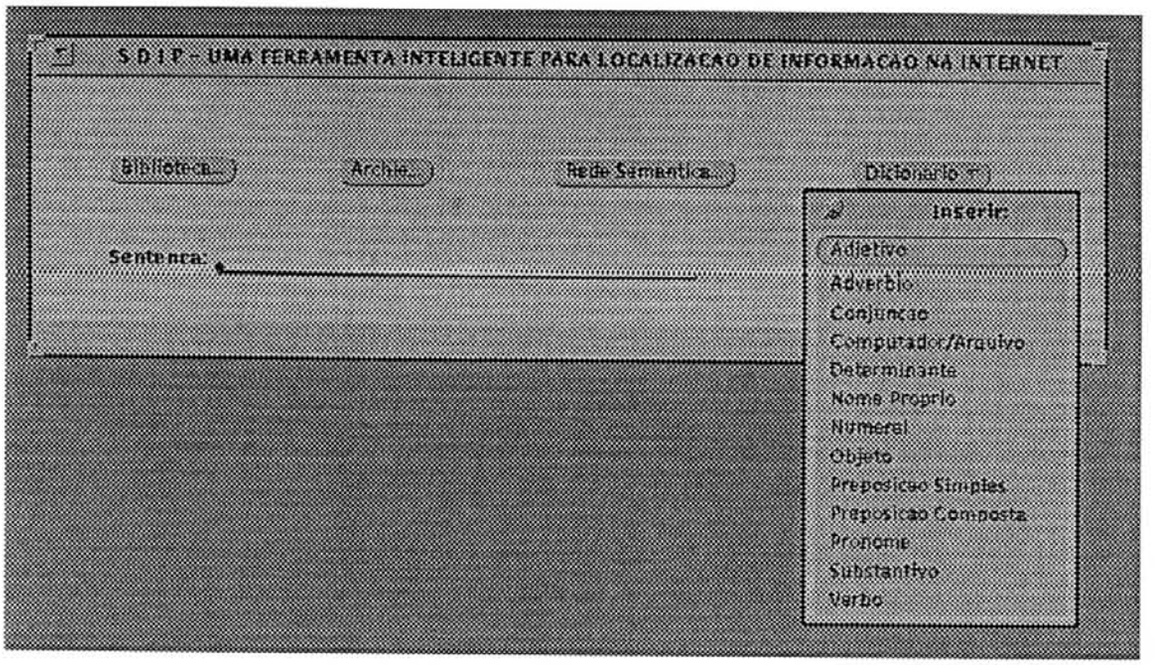

Figura 12.3 *Menu de atualização do Dicionário do Sistema.*

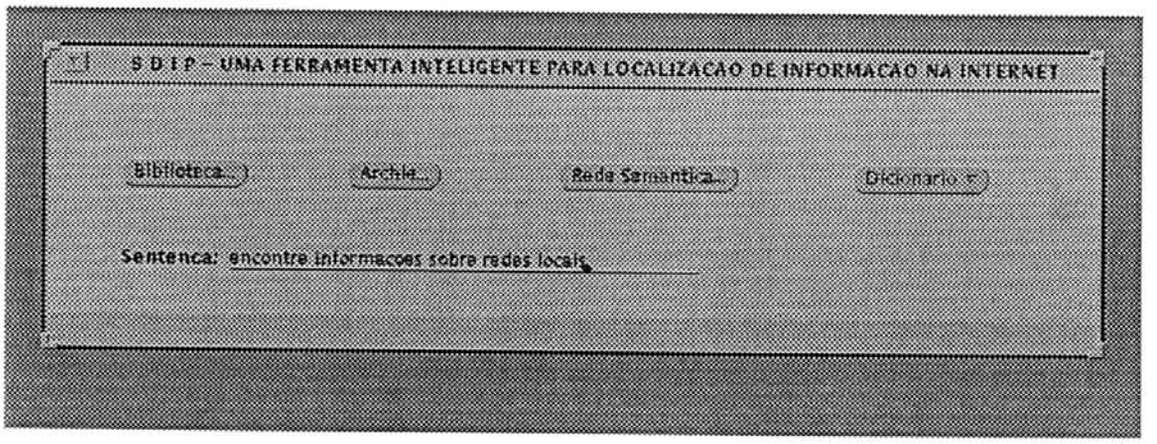

Figura 12.4 *Utilização do SDIP através de sentenças em língua natural.*

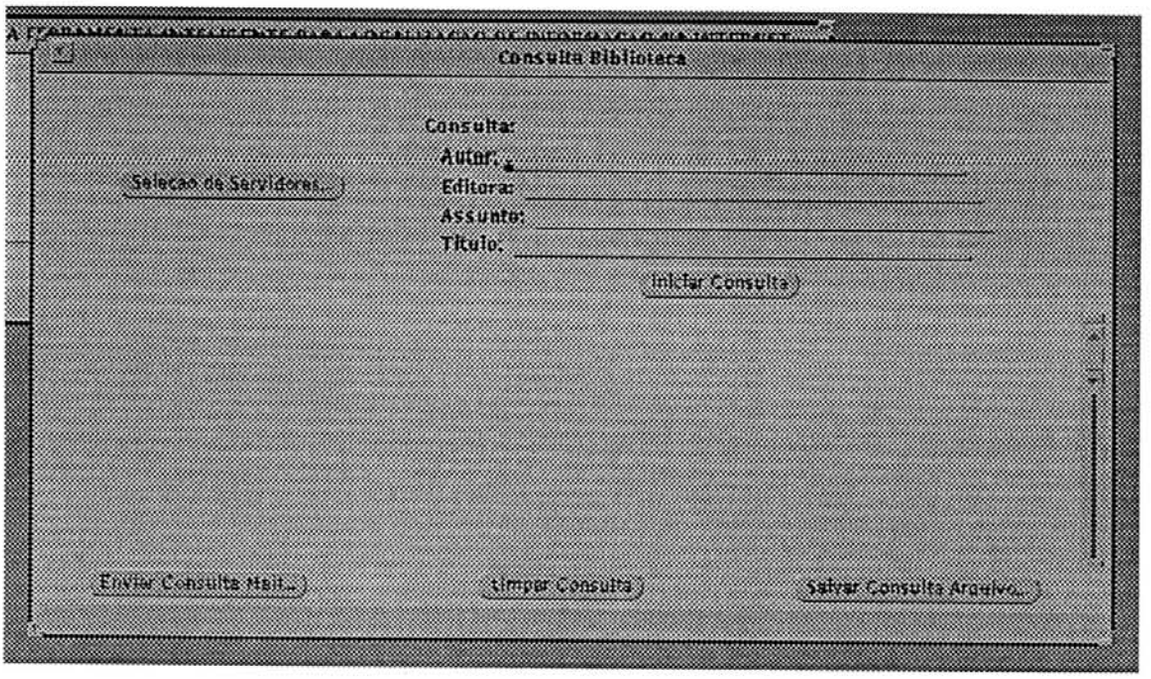

Figura 12.5 *Janela principal de acesso aos sistemas de consultas bibliográficas.*

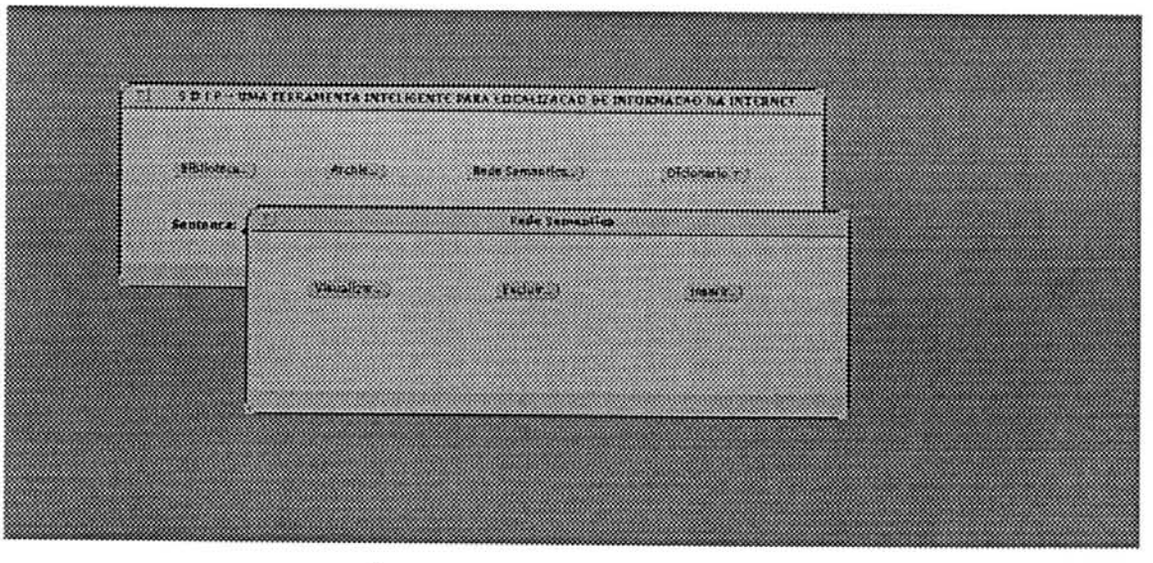

Figura 12.6 *Janela usada para o gerenciamento da base de conhecimento do sistema.*

Tabela 12.1 Especificação em SDL-PR do módulo de interface gráfica.

BLOCK	B_IG;	/*DEF. DO MIG*/
SIGNALROUTE R1		
FROM P_IG	TO ENV	WITH S_IG_ENV
FROM ENV	TO P_IG	WITH S_ENV_IG;
SIGNALROUTE R2		
FROM P_IG	TO ENV	WITH S_IG_ARC,S_CR
FROM ENV	TO P_IG	WITH S_ARC_IG;
SIGNALROUTE R3		
FROM P_IG	TO ENV	WITH S_IG_FTP,S_CR
FROM ENV	TO P_IG	WITH S_FTP_IG;
SIGNALROUTE R4		
FROM P_IG	TO ENV	WITH S_IG_FS,S_CR
FROM ENV	TO P_IG	WITH S_FS_IG;
SIGNALROUTE R5		
FROM P_IG	TO ENV	WITH S_IG_PIS_1,S_IG_PIS_2,S_CR
FROM ENV	TO P_IG	WITH S_PIS_IG_1,S_PIS_IG_2;
CONNECT C_IG_ENV	AND	R1;
CONNECT C_ARC_IG	AND	R2;
CONNECT C_FTP_IG	AND	R3;
CONNECT C_SABI_IG	AND	R4;
CONNECT C_IG_PIS	AND	R5;

12.3 Especificação do Módulo em SDL-PR

Nesta seção, mostraremos a especificação completa em SDL-PR [INT 88, BEL 91] do módulo gráfico de interface do usuário (tabelas 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7 e 12.8). Para tornar a especificação mais compreensível e clara para os leitores, inserimos na própria representação, sempre que julgamos necessário, alguns comentários que descrevem o comportamento do processo, a função de uma determinada variável etc.

Tabela 12.2 Especificação em SDL-PR do módulo de interface gráfica.

PROCESS	P_IG	/*PROCESSO DA INTERFACE GRÁFICA*/
DCL	RESP	BOOLEAN;
START;		
STATE	ST_INIT;	/*CRIA OS PROCESSOS DO SDIP*/
OUTPUT	R2 S_CR;	/*CRIA O PA*/
INPUT	S_ARC_IG;	
OUTPUT	R3 S_CR;	/*CRIA O PFTP*/
INPUT	S_FTP_IG;	
OUTPUT	R4 S_CR;	/*CRIA O FS*/
INPUT	S_FS_IG;	
OUTPUT	R5 S_CR;	/*CRIA O PIS*/
INPUT	S_PIS_IG_1;	
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_INIT;	

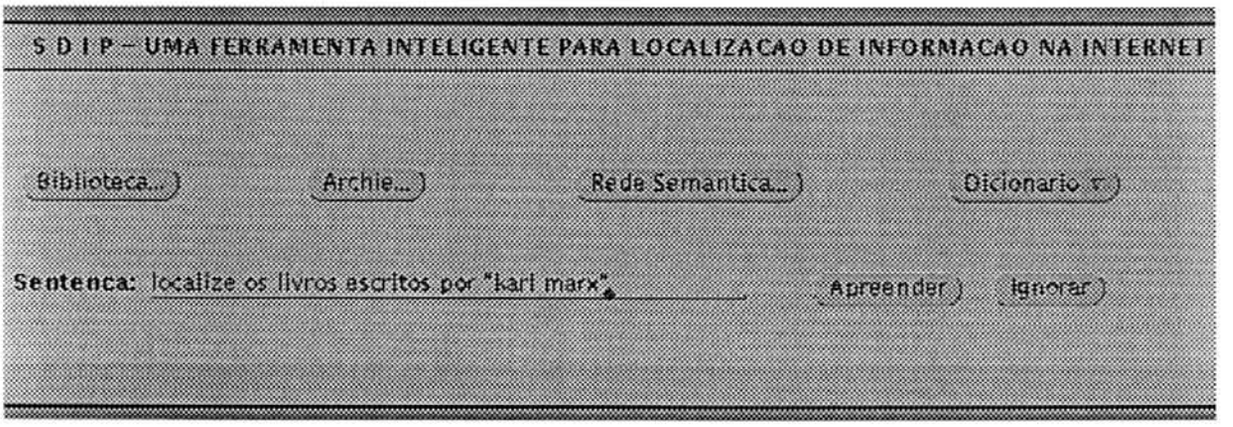

Figura 12.7 *Janela principal com a presença das estruturas que permitem o aprendizado do significado de novas sentenças.*

Tabela 12.3 Especificação em SDL-PR do módulo de interface gráfica.

STATE	ST_MAIN;	
INPUT	S_ENV_IG(CMD);	/(COMANDOS DOS USUÁRIOS*/
DECISION	CMD;	
('ARCHIE'):		
NEXTSTATE	ST_ARC;	/*USA O ARCHIE*/
('FTP'):		
NEXTSTATE	ST_FTP;	/*SISTEMA FTP*/
('BIBLIOTECA'):		/*SISTEMA DE CONSULTAS BIBLIOGRÁFICAS*/
NEXTSTATE	ST_SABI;	
('DICIONÁRIO'):		/*DICIONÁRIO DO SISTEMA*/
NEXTSTATE	ST_DIC;	
('VISUALIZA BASE'):		/*VISUALIZAÇÃO DA BASE DE CONHECIMENTO*/
NEXTSTATE	ST_BASE;	
('MODIFICA BASE'):		/*MODIFICA A BASE DE CONHECIMENTO DO SDIP*/
NEXTSTATE	ST_MODIFY;	
('SENTENÇA'):		/*INTERFACE EM LíNGUA NATURAL*/
NEXTSTATE	ST_SENT;	
('PARAR'):		
NEXTSTATE	ST_STOP;	
ENDDECISION;		
ENDSTATE	ST_MAIN;	

Tabela 12.4 Especificação em SDL-PR do módulo de interface gráfica.

STATE	ST_ARC;	/*USA O ARCHIE*/
OUTPUT	S_IG_ARC;	/*ENVIA CONSULTA*/
INPUT	S_ARC_IG;	/*RECEBE RESPOSTAS*/
OUTPUT	S_IG_ENV;	/*RESPONDE AOS USUÁRIOS*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_ARC;	
STATE	ST_FTP;	/*USA O PFTP*/
OUTPUT	S_IG_FTP;	/*ENVIA COMANDO*/
INPUT	S_FTP_IG;	/*RECEBE RESPOSTAS*/
OUTPUT	S_IG_ENV;	/*RESPONDE AOS USUÁRIOS*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_FTP;	
STATE	ST_SABI;	/*SISTEMA DE CONSULTAS BIBLIOGRÁFICAS*/
OUTPUT	S_IG_FS;	/*ENVIA CONSULTA*/
INPUT	S_FS_IG;	/*RECEBE RESPOSTAS*/
OUTPUT	S_IG_ENV;	/*RESPONDE AO USUÁRIO*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_SABI;	

Tabela 12.5 Especificação em SDL-PR do módulo de interface gráfica.

STATE	ST_BASE;	/*VISUALIZA BASE DE CONHECIMENTO*/
OUTPUT	S_IG_PIS_1;	/*ENVIA COMANDO*/
INPUT	S_PIS_IG_1;	/*RECEBE RESPOSTAS*/
OUTPUT	S_IG_ENV;	/*MOSTRA A BASE DE CONHECIMENTO*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_BASE;	
STATE	ST_MODIFY;	/*ALTERA A BASE DE CONHECIMENTO*/
INPUT	S_ENV_IG;	/*LÊ NOVOS DADOS*/
OUTPUT	S_IG_PIS_1;	/*ENVIA DADOS PARA O PIS*/
INPUT	S_PIS_IG_1;	/*RECEBE RESULTADO*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_MODIFY;	

Tabela 12.6 Especificação em SDL-PR do módulo de interface gráfica.

STATE	ST_SENT;	/*USO DE SENTENÇAS*/
OUTPUT	S_IG_PIS_1;	/*ENVIA SENTENÇA*/
INPUT	S_PIS_IG_1(RESP);	/*RECEBE RESULTADO*/
DECISION (TRUE):	RESP;	/*INTERPRETOU CORRETAMENTE*/
...		/*EXECUTA OPERAÇÕES EXPRESSAS NA SENTENÇA*/
OUTPUT	S_IG_ENV;	/*MOSTRA RESULTADOS*/
NEXTSTATE (FALSE):	ST_MAIN;	/*ERRO DE INTERPRETAÇÃO*/
NEXTSTATE	ST_LEARN;	/*VAI APRENDER O SIGNIFICADO DA SENTENÇA*/
ENDDECISION;		
ENDSTATE	ST_SENT;	

Tabela 12.7 Especificação em SDL-PR do módulo de interface gráfica.

STATE	ST_LEARN;	/*APRENDE O SIGNIFICADO DE UMA SENTENÇA*/
INPUT	S_PIS_IG_2;	/*MUDA PARA O ESTADO DE APRENDIZADO*/
INPUT	S_ENV_IG;	/*APRENDE COM O USUÁRIO*/
OUTPUT	S_IG_PIS_2;	/*ENSINA O PIS*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_LEARN;	

Tabela 12.8 Especificação em SDL-PR do módulo de interface gráfica.

STATE	ST_DIC;	/*ALTERA DICIONÁRIO*/
INPUT	S_ENV_IG;	/*LÊ INFORMAÇÕES DA NOVA PALAVRA*/
OUTPUT	S_IG_PIS_1;	/*ENVIA OS DADOS PARA O PIS*/
INPUT	S_PIS_IG_1;	/*RECEBE RESULTADOS*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_DIC;	
STATE	ST_STOP;	/*MATA OS DEMAIS PROCESSOS*/
OUTPUT	S_IG_ARC(KILL);	/*MATA O PA*/
INPUT	S_ARC_IG;	
OUTPUT	S_IG_FTP(KILL);	/*MATA O PFTP*/
INPUT	S_FTP_IG;	
OUTPUT	S_IG_FS(KILL);	/*MATA O FS*/
INPUT	S_FS_IG;	
OUTPUT	S_IG_PIS_1(KILL);	/*MATA O PIS*/
INPUT	S_PIS_IG_1;	
STOP;		
ENDSTATE	ST_STOP;	
ENDPROCESS	P_IG;	
ENDBLOCK	B_IG;	

13 PROCESSO INTERPRETADOR DE SENTENÇAS

13.1 Introdução

Neste capítulo, discutiremos detalhadamente o último e, sem dúvida, o mais complexo dos componentes do ambiente **SDIP**: o **Processo Interpretador de Sentenças**.

Quando iniciamos o planejamento e o desenvolvimento efetivo do **SDIP**, tínhamos como objetivo principal a construção de um sistema no qual o diálogo com os usuários pudesse ser realizado através de sentenças em língua natural, ao lado de uma interface gráfica convencional. De fato, o objetivo era construir-se um modelo de interface que conjugasse as duas alternativas adotadas de uma maneira complementar e eficiente.

Este objetivo, aliado ao desejo de se implementar um sistema modular, levou-nos a desenvolver um processo que ficasse responsável, única e exclusivamente, pelo trabalho inteligente da interface com os usuários, ou seja, todas as funções relacionadas com a interpretação e aprendizado do significado de sentenças, englobando, também, o gerenciamento de uma base de conhecimento auxiliar e um dicionário de palavras e expressões da língua portuguesa.

De maneira análoga ao que observamos e destacamos em relação aos demais componentes do **SDIP** estudados até o momento, a opção por um projeto modular, com a conseqüente implementação de um processo encarregado apenas de um conjunto de tarefas logicamente relacionadas, permite-nos ampliar rápida e eficientemente as potencialidades do **Processo Interpretador de Sentenças**, sem afetar de nenhuma forma os demais componentes do sistema.

Assim, caso quiséssemos expandir o espectro das classes de sentenças interpretadas pelo **SDIP**, incorporando, por exemplo, o reconhecimento de orações re-

lativas, teríamos de modificar exclusivamente o **Processo Interpretador de Sentenças**, tornando esta expansão transparente para os demais componentes do SDIP.

Como tivemos a oportunidade de destacar nos capítulos 7 e 12, a Interface Gráfica e o **Interpretador de Sentenças** são os únicos componentes do **SDIP** que apresentam um comportamento diferenciado no que concerne a sua funcionalidade, uma vez que, dependendo da situação, ambos podem operar como usuários ou prestadores de serviços no diálogo por eles mantido.

Particularmente, no que concerne ao processo ora em estudo, ele funcionará como um servidor nas seguintes situações:

- Inicialização do sistema;
- Procedimento de término de execução, iniciado a partir da Interface Gráfica;
- Solicitação para interpretar o significado de uma sentença. Este serviço é utilizado pela Interface Gráfica e pelo **Processo Servidor de Correio Eletrônico**;
- Inclusão de novas palavras no dicionário do sistema;
- Modificações e visualização da base de conhecimento do sistema.

Por outro lado, o **Processo Interpretador de Sentenças** comportar-se-á como um cliente, ou seja, usará os serviços providos pela Interface Gráfica, nas seguintes ocasiões:

- Execução das operações expressas em uma sentença, interpretada previamente pelo processo;
- Aparecimento de uma palavra desconhecida durante a análise léxica e sintática de uma sentença;

- Desconhecimento, por parte do **Interpretador**, do significado de uma sentença formulada por um usuário.

Para tornar disponível à Interface Gráfica e ao **Processo Servidor de Correio Eletrônico** os serviços providos pelo **Interpretador de Sentenças**, adotou-se uma solução similar àquela usada nos demais componentes do **SDIP**, ou seja, construiu-se uma API, na qual estão implementadas estas facilidades.

Por outro lado, quando o **Interpretador de Sentenças** opera como um cliente da Interface Gráfica, ele se serve também de uma API especialmente desenvolvida para a efetivação do diálogo com este processo.

13.2 O Sistema de Representação de Diálogos

Como pode se depreender a partir da breve descrição realizada na parte introdutória deste capítulo, o **Processo Interpretador de Sentenças** constitui-se no componente de maior complexidade dentro do ambiente **SDIP**, principalmente se considerarmos que todo o desenvolvimento deste módulo é feito exclusivamente pelo analista, codificando-o diretamente em uma linguagem de programação, sem o auxílio de nenhum "software" adicional.

Em virtude da complexidade envolvida no integral desenvolvimento do **Processo Interpretador de Sentenças**, aliado à existência de alguns protótipos que implementam parte do que seria necessário a este componente, resolvemos usar um desses sistemas como a base sobre a qual construiríamos o processo enfocado neste capítulo. Mais precisamente, utilizamos um protótipo, denominado **Sistema de Representação de Diálogos (SRD)**, desenvolvido como parte do trabalho de mestrado do doutorando Sérgio Antônio Andrade de Freitas, descrito em [FRE 93].

O **SRD** é um sistema que funciona como um monitorador do diálogo mantido por dois agentes, representando-o através de DRSs. Em linhas gerais, dois

agentes, o ouvinte e o falante dialogam por intermédio do **SRD**, enquanto o sistema representa o discurso recebido usando as DRSs.

Para uma melhor compreensão por parte do leitor a respeito do funcionamento do **SRD**, podemos particioná-lo em cinco componentes logicamente interrelacionados [FRE 93], a saber:

1. Analisador léxico e sintático, o qual, a partir de uma sentença formulada em língua portuguesa, produz uma árvore de derivação sintática que a representa;

2. Dicionário, onde estão armazenadas as palavras necessárias à análise léxica e sintática de uma sentença, com suas respectivas informações gramaticais, tais como: gênero, número, etc;

3. Gerador de DRSs. Componente que, a partir da árvore de derivação gerada pelo analisador sintático, produz as DRSs a ela associadas;

4. Focalizador. Componente que resolve os pronomes anafóricos presentes em um discurso, utilizando como metodologia ou formalismo a teoria do foco;

5. Base de conhecimento. Usada pelo focalizador como fonte de informações, as quais são algumas vezes necessárias à execução do algoritmo de focalização, além de manter as relações e os objetos de interesse do sistema. Note-se que a base de conhecimento do **SRD** é uma estrutura estática, fazendo parte do próprio código do sistema. Este fato impede que, em tempo de execução, esta estrutura seja modificada dinamicamente, fazendo com que o **SRD** não seja capaz de adaptar-se a uma nova realidade que a ele se apresente.

Desta forma, o **SRD** ao receber um discurso o analisa léxica e sintaticamente, produzindo uma árvore de derivação que, por sua vez, é passada ao gerador de DRSs. Neste componente, é gerada a DRS que representa o significado semântico

do discurso inicialmente recebido. Nesta mesma fase, existe uma integração entre o gerador de DRSs, o focalizador e a base de conhecimento, permitindo que se resolvam os pronomes anafóricos presentes no discurso formulado por cada um dos participantes.

No que concerne ao **SDIP**, usamos do **SRD** os componentes que realizam a análise léxica e sintática do discurso, bem como o dicionário de palavras. Todavia, promovemos algumas modificações em ambos os componentes, objetivando adequá-los a nossa realidade diferenciada. Quanto aos demais constituintes do sistema, eles foram completamente rescritos, exceção feita ao focalizador, o qual não está presente no **Interpretador de Sentenças**, fazendo com que, pelo menos na atual versão do **SDIP**, não sejam resolvidos pronomes anafóricos que por ventura apareçam em um discurso.

Nas próximas seções do presente capítulo, discutiremos detalhadamente a estrutura e o funcionamento do **Processo Interpretador de Sentenças** e cada um dos componentes que o constituem atualmente.

13.3 Estrutura e Funcionamento do Processo Interpretador de Sentenças

Nesta seção, discutiremos o **Processo Interpretador de Sentenças** em seus aspectos mais gerais, enfocando sua estrutura e funcionamento básicos. Nas próximas subseções, abordaremos mais detalhadamente cada um dos principais aspectos aqui introduzidos, servindo-nos de exemplos para ilustrar o funcionamento interno do processo.

Na codificação do **Interpretador de Sentenças** foram utilizadas duas linguagens de programação: C [KER 78, SUN 88, SUN 89a] e Prolog [BRAT 86, PER 87]. A primeira destas linguagens foi usada na construção das APIs que possibilitam a comunicação do **Processo Interpretador de Sentenças** com a Interface Gráfica e com o Processo Servidor de Correio Eletrônico. A Linguagem Prolog,

por outro lado, foi usada para a codificação dos componentes internos do processo, exceção feita ao analisador sintático que foi escrito em DCG ("Descritive Clauses Grammar").

Observando a figura 13.1, onde mostramos a estrutura lógica de blocos do **Interpretador de Sentenças** e as figuras 13.2 e 13.3, podemos facilmente compreender os caminhos percorridos por uma sentença ou comando, desde o momento em que ingressam no processo, até a sua conclusão.

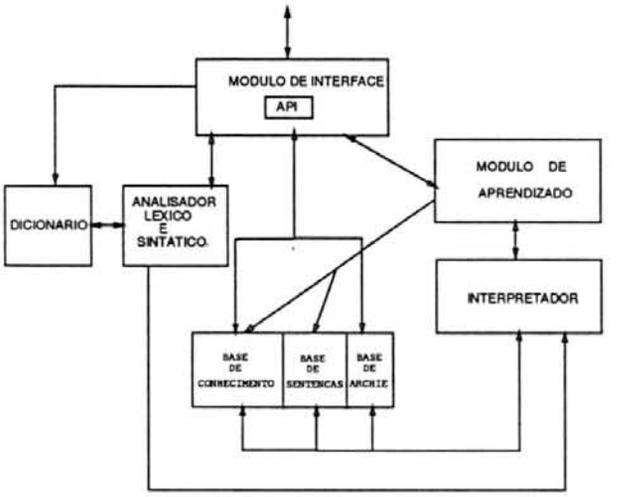

Figura 13.1 *Estrutura lógica de blocos do Processo Interpretador de Sentenças.*

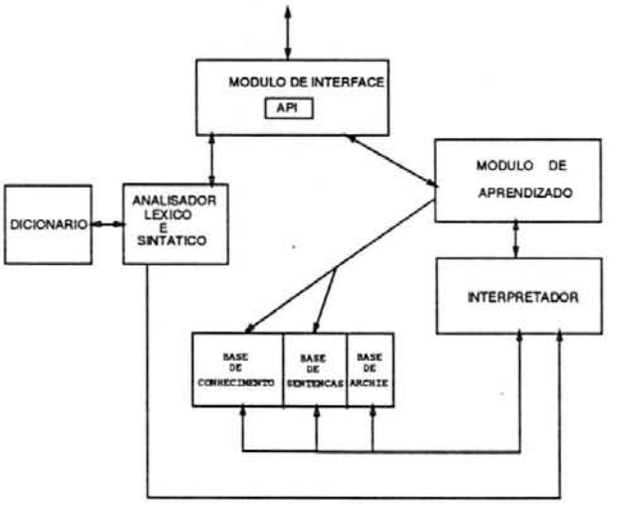

Figura 13.2 *Caminho percorrido por uma sentença dentro do Processo Interpretador.*

Inicialmente, consideremos o processamento ao qual é submetida uma sentença em língua portuguesa, ilustrado pela figura 13.2. Após ter ingressado no

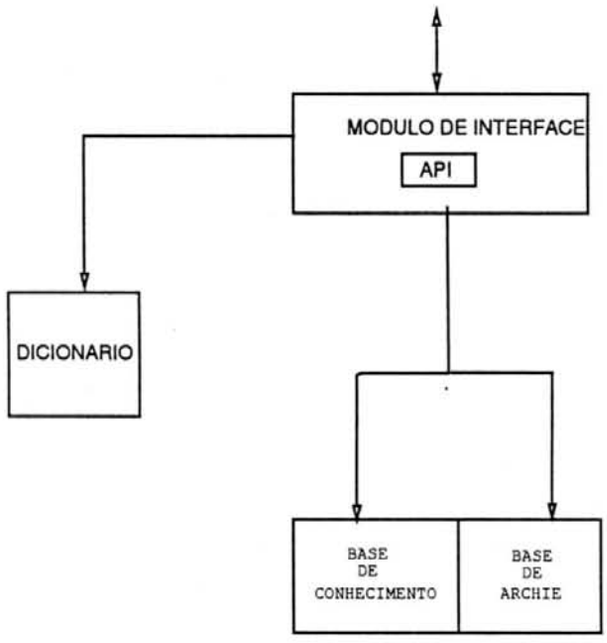

Figura 13.3 *Caminho percorrido por um comando dentro do Interpretador.*

processo através do Módulo de Interface Externa, a sentença é passada ao analisador léxico e sintático do sistema, onde, com o uso do dicionário de palavras, é gerada uma árvore sintática representativa da sentença recebida.

Na eventualidade de uma determinada palavra ser desconhecida para o sistema, situação caracterizada pela inexistência da mesma no dicionário, o analisador léxico faz uso do módulo de interface para contactar o **Ambiente Gráfico** e solicitar ao usuário o fornecimento das informações gramaticais que descrevem e caracterizam este termo desconhecido.

Posteriormente, a árvore de derivação, produzida no analisador sintático, é processada no módulo de interpretação. Neste ponto, caso a sentença seja reconhecida pelo sistema, os comandos que nela estão implicitamente expressos são enviados, novamente através do módulo de interface externa, ao **Ambiente Gráfico** para neste processo serem efetivamente executados. Caso contrário, é iniciada uma fase de aprendizagem, na qual será determinado o significado da sentença desconhecida e os comandos que estão a ela associados, fazendo-se uso das facilidades especialmente implementadas no **Ambiente Gráfico**. Com o término da fase de

aprendizagem, o **Processo Interpretador** estará apto a interpretar e executar os comandos expressos pela sentença, outrora desconhecida.

Finalmente, cabe ainda destacar que as bases de conhecimento, informações Archie e de sentenças genéricas desempenham um papel fundamental no processo de interpretação das sentenças, posto que esta fase constitui-se, basicamente, de operações realizadas sobre este conjunto de informações. Por outro lado, observamos que a fase de aprendizado do significado de uma sentença utiliza tão somente informações contidas na base de sentenças genéricas.

Observando o caminho percorrido por um comando oriundo da Interface Gráfica dentro do **Interpretador de Sentenças**, mostrado na figura 13.3, notamos que este difere substancialmente daquele seguido por uma sentença em língua portuguesa. Como observamos na figura 13.3, após terem ingressado no processo através do módulo de interface externa, os comandos atuam diretamente sobre as bases de informações do **Interpretador de Sentenças**, mais precisamente dicionário, base de conhecimento e base de informações Archie, posto que, como vimos no capítulo anterior, o **Ambiente Gráfico** implementa tais facilidades, havendo a necessidade da existência de comandos a elas associados.

A partir deste momento, daremos início à descrição pormenorizada a respeito de cada uma das estruturas que aparecem na figura 13.1, culminando, no fechamento do capítulo, com uma especificação em SDL-PR do **Interpretador de Sentenças**.

13.3.1 Módulo de Interface Externa

Este componente constitui-se na porta por onde fluem as informações e comandos que entram e saem do **Processo Interpretador de Sentenças**. Na realidade, o módulo de interface externa é uma associação composta por duas APIs ou bibliotecas de funções, escritas em linguagem C, além das cláusulas Prolog que inte-

gram estas bibliotecas com os módulos principais, todos codificados completamente na linguagem Prologe DCG.

De um lado, temos a biblioteca de funções encarregada de receber e tratar os comandos provenientes do **Ambiente Gráfico** ou do **Processo Servidor de Correio Eletrônico**, sendo usada no momento em que o **Interpretador de Sentenças** funciona como um processo servidor.

De outra parte, existe a biblioteca onde estão implementadas as funções que permitem ao processo descrito neste capítulo utilizar-se dos serviços providos pelo **Ambiente Gráfico**.

13.3.2 Análise Léxica e Sintática das Sentenças

O discurso formulado por um usuário, ao ingressar no **Processo Interpretador de Sentenças** através do módulo de interface externa, deve ser avaliado com respeito a sua constituição e estrutura, objetivando verificar se de fato ele obedece às regras definidas pela língua portuguesa. Mais precisamente, devemos averiguar a correção de um discurso, por um lado, em relação à consistência do conjunto de termos individuais que o compõe e, por outro lado, com respeito à estrutura sintática definida pelo agrupamento ordenado destes termos.

Os analisadores léxico e sintático do Processo Interpretador de Sentenças são os componentes responsáveis pela realização efetiva desta tarefa inicial de avaliação de um determinado discurso. Eles recebem do módulo de interface externa uma sentença em língua portuguesa e verificam a sua correção morfológica e estrutural, produzindo como resultado de seu trabalho uma ou mais árvores de derivação sintática, representativas do discurso inicial.

Não obstante, exista uma interdependência entre os procedimentos envolvidos na análise léxica e sintática de uma sentença, abordaremos separadamente cada uma destas fases por considerarmos que, desse modo, fica facilitada, tanto a explanação, quanto a compreensão do tema por parte do leitor.

Inicialmente, consideremos a fase de análise léxica de uma sentença na qual são identificados e classificados cada um dos termos que a constituem.

O analisador léxico do **Processo Interpretador de Sentenças** é constituído por um conjunto de cláusulas Prolog e um dicionário de termos, onde ficam armazenadas permanentemente o conjunto de palavras reconhecidas pelo sistema.

No que concerne a seu funcionamento, o analisador léxico é ativado pelo sintático, sempre que este necessita identificar um componente de uma sentença, possibilitando-lhe, desta forma, construir a árvore de derivação sintática que a representa. Caso seja impossível para o analisador identificar o item léxico, é iniciado um procedimento semi-inteligente para, com a ajuda do usuário do sistema, classificar o termo desconhecido. Ao concluir-se a fase de aprendizado, o termo é inserido no dicionário de palavras do sistema com o intuito de, nas próximas oportunidades, possibilitar ao analisador léxico a sua identificação inequívoca.

Como dissemos anteriormente, o analisador ora em estudo tem a função de identificar gramaticalmente um termo componente de uma sentença. Portanto, para que ele seja capaz de executar efetivamente a sua tarefa principal, deve existir uma base de dados no sistema, onde este componente possa pesquisar e armazenar informações inerentes a sua função. Esta base de dados, denominada *dicionário de palavras do sistema*, é composta por um conjunto de arquivos Prolog, onde estão armazenadas todas as palavras identificáveis pelo analisador léxico, bem como suas respectivas informações gramaticais.

No momento da inicialização do **Processo Interpretador de Sentenças**, esta base de dados é carregada para a memória do interpretador Prolog, tornando a identificação de uma palavra um processo mais rápido e eficiente, uma vez que a pesquisa não é realizada em um meio de armazenamento secundário.

Por outro lado, a inclusão de uma nova palavra no dicionário é realizada em duas etapas. Em um primeiro momento, o termo Prolog é incluído diretamente na memória do interpretador Prolog para, logo depois, esta informação ser gravada

no arquivo correspondente, o qual é mantido em disco, tornando a alteração do dicionário um fato permanente, perceptível nas próximas vezes em que um usuário utilizar o **SDIP**.

Nas tabelas 13.1, e 13.2, mostramos as classes gramaticais de cada uma das palavras reconhecidas pelo sistema, o arquivo no qual elas estão armazenadas e, finalmente, o termo Prolog que as representa.

Logo depois, fazemos alguns comentários a respeito das informações mostradas nas tabelas 13.1 e 13.2, destacando aquelas de que não encontramos explicações em gramáticas da língua portuguesa, uma vez que foram criadas com o intuito de satisfazer as necessidades impostas pela implementação do **Processo Interpretador de Sentenças**.

Tabela 13.1 Classes gramaticais das palavras reconhecidas pelo SDIP.

Classe Gramatical	Arquivo (s)	representação
Adjetivo	a.pro	a(Pal,Num,Gen)
Advérbio	adv.pro	adv(Pal)
Artigo	d.pro	d(pal,Num,Gen)
Conjunção	c.pro	c(Pal,Tc)
Preposição	pr1.pro, pr2.pro	pr(Pal,Prep-Pal), pr(Pal)
Pronome	p.pro	p(Pal,Num,Gen,Pes,Tp)

Onde:

Todas as classes gramaticais de palavras mostradas nas tabelas 13.1 e 13.2, exceto as classes Objeto e Computador/diretório/Arquivo/Endereço, apresen-

Tabela 13.2 Complementação das informações mostradas na tabela anterior.

Classe Gramatical	Arquivo	representação
Substantivo		
Próprio	np.pro	np(Pal,Num,Gen)
Comum	n.pro	n(Pal,Num,Gen)
Verbo	v.pro	v(Pal,Inf,Tmp,Num,Gen,Pes)
	vdes.pro	vdes(Desin,Conj,Tmp,Num,Gen,Pes)
	vrad.pro	vrad(Rad,Inf,Conj)
	vtra.pro	vtra(Transit,Inf,Sem)
Numeral	num.pro	num(Pal,Tn,Num,Gen)
Objeto	complex.pro	complex(List,Num,Gen,To,Termos)
Computador/ Diretório/Arquivo/ Endereço	hda.pro	hda(Pal,Num,Gen,Thda)

Pal	= Palavra
Gen	= Gênero (masculino ou feminino)
Num	= Número (singular ou plural)
Tc	= Tipo da conjunção (conjuntiva ou disjuntiva)
Prep-Pal	= Preposição+palavra primitiva
Pes	= Pessoa (primeira, segunda ou terceira)
Tp	= Tipo do pronome (pessoal, demonstrativo etc)
Inf	= Verbo no infinitivo
Tmp	= Tempo verbal (presente, pretérito perfeito etc)
Desin	= Desinência verbal
Conj	= Conjugação (primeira, segunda ou terceira)
Rad	= Radical do verbo
transi	= Transitividade verbal (direta, indireta etc)
Sem	= Informação semântica
Tn	= Tipo do numeral (cardinal ou ordinal)
List	= Lista de termos formadores da expressão
To	= Define o comportamento da expressão (substantivo, nome próprio etc)
Termos	= Descrição individualizada dos termos que formam a expressão
Thda	= Define o tipo da entrada (Arquivo, diretório, etc)

tam comportamento e funções de acordo com o que está definido nas gramáticas da língua portuguesa.

Os argumentos associados a cada uma das palavras de uma dada classe gramatical servem para caracterizá-las, por exemplo, quanto a seu número, gênero, etc. É importante notar que a caracterização dependerá da classe gramatical da palavra. Assim, por exemplo, uma conjunção não possui o qualificador gênero, uma vez que não faria sentido determinar se este termo é masculino ou feminino.

Consideremos a classe gramatical preposição. Na tabela 13.1, observamos que as palavras desta classe podem ser representadas de duas maneiras distintas:

- A forma *pr(Pal)* é usada para representar preposições simples, como, por exemplo, a, de, com etc;

- Por outro lado, o fato *pr(Pal, Prep-Pal)* é usado para representar termos que contenham preposições, contudo associadas a uma palavra de outra classe gramatical, como no caso do termo *nesta*, formado pela preposição *em*, e pelo pronome demonstrativo *esta*.

Agora, voltemos nossa atenção para as palavras da classe gramatical verbo. Na tabela 13.2, observamos que existem quatro fatos Prolog envolvidos na representação deste tipo de palavra.

Na verdade, os verbos são divididos em duas subclasses: os regulares e os irregulares. Assim, o fato Prolog *v/6* é usado para representar única e exclusivamente os verbos irregulares. Por sua vez, os fatos *vdes/6* e *vrad/3*, combinados, representam os verbos classificados como regulares. Finalmente, o fato *vtra/3* determina a transitividade de um dado verbo, seja ele regular, ou irregular.

É importante notar que um dos argumentos que descreve a palavra da classe gramatical verbo é a sua informação semântica, estando presente no fat Prolog *vtra/3*. Este campo é usado para indicar o tipo de ação que está associada ao verbo considerado. Os valores, por ele assumidos, limitam-se tão somente ao conjunto de ações que o sistema pode de fato realizar, excluindo-se, desta maneira, verbos que não sejam significativos para o **SDIP**.

A seguir, relacionamos os valores que podem ser assumidos pelo campo ora em estudo.

Na tabela 13.3, mostramos exemplos de sentenças que ilustram o uso das informações semânticas listadas abaixo, com seus respectivos verbos associados.

- **pesquisar** -> verbos usados para iniciar uma procura por informações;
- **mostrar** -> verbos que são utilizados para solicitar ao sistema que exiba uma dada informação, seja ela um arquivo, diretório, resultados de consulta etc;

- **recuperar** -> indica ao sistema que o usuário deseja trazer, para a máquina local, um determinado arquivo;

- **sair** -> verbos que sinalizam para o **SDIP** abandonar um determinado estado ou posição, como, por exemplo, passar do estado em que o sistema está em funcionamento para o estado de inatividade, encerrando sua execução;

- **usar** -> verbos utilizados para a seleção de um editor para a visualização de um arquivo, o servidor destino de futuras consultas etc;

- **mover** -> indica ao sistema para se deslocar em uma estrutura como, por exemplo, um diretório;

- **enviar** -> indica que uma determinada informação deve ser encaminhada a um dado destino;

- **pôr** -> indica ao sistema que uma informação deve ser colocada em um dado local, um diretório ou arquivo, por exemplo.

Tabela 13.3 Exemplos de sentenças associadas a cada um dos tipos de informação semântica definidas para os verbos.

Informação Semântica	Sentença
Pesquisar	Localize informações sobre a arquitetura de rede SNA
Mostrar	Mostre o diretório sna
Recuperar	Recupere o arquivo sna.ps
Sair	Abandone o sistema de consultas bibliográficas
Usar	Utilize o servidor archie archie.ans.net para as novas consultas
Mover	Entre no diretório sna
Enviar	Encaminhe os resultados da consulta ao usuário islu@cesup.ufrgs.br
Pôr	Salve os resultados da consulta no arquivo result.txt

Para concluirmos o estudo das informações mostradas nas tabelas 13.1 e 13.2, consideremos as expressões das classes gramaticais *Objeto* e *Computador/Diretório/Arquivo/Endereço.* Termos pertencentes a estas classes não correspondem a nenhum tipo gramatical, definido na língua portuguesa.

A classe Objeto foi criada para possibilitar ao sistema o reconhecimento de sentenças mais complexas, nas quais se tornaria difícil para um analisador sintático simples, sem a interferência do usuário, identificar corretamente os seus componentes, determinando os limites existentes entre cada um deles.

Considere a sentença 13.1:

Frase 13.1 Ana deu o livro de poesias de Pedro a Carina.

Como se pode observar no exemplo acima 13.1, seria difícil para um analisador sintático simples identificar quais as partes da sentença que correspondem aos complementos nominais, objeto direto e objeto indireto. Desta forma, os usuários devem auxiliar o analisador sintático na realização desta tarefa, identificando para ele que termos da sentença formam um conjunto consistente. Na sentença 13.2, mostramos uma maneira alternativa de como construir o exemplo anterior 13.1.

Frase 13.2 Ana deu o livro_de_poesias_de_Pedro a Carina.

.

Uma situação em que este tipo de construção é imprescindível acontece quando, em uma sentença, aparece o título de um livro. Não obstante este termo forme um conjunto, em geral a sua presença impossibilita uma análise sintática correta da sentença, posto que este componente possui, por assim dizer, uma vida sintática própria. Neste caso, o título do livro deve aparecer, ou da forma mostrada acima na sentença 13.2, onde seus termos estariam separados por um sinal de sublinha ('_'), ou, alternativamente, aparecer envolvido por aspas (' " " ').

Somando-se as classes de palavras mostradas nas tabelas 13.1 e 13.2, definimos ainda um outro termo reconhecido pelo analisador léxico do **Interpretador de Sentenças**. Esta nova categoria gramatical foi criada com o intuito exclusivo de tornar possível a representação de palavras não pertencentes ao léxico da língua portuguesa, particularmente nome de arquivos, diretórios e servidores, bem como endereços de correio eletrônico.

As informações que descrevem os termos pertencentes a essa categoria estão codificadas no fato Prolog *hda/4*, no qual os argumentos contêm respectivamente o item léxico, seu gênero e número, além de seu tipo (arquivo, diretório etc).

Finalmente, destaque-se que, ao contrário dos anteriores, termos pertencentes a esta classe, em algumas situações bem particulares, possuem um tempo de vida limitado por sua presença dentro do contexto do sistema. assim, por exemplo, somente faria sentido manter no dicionário de palavras os nomes dos arquivos pertencentes a um diretório, durante o intervalo de tempo em que estes fossem visíveis, tanto para o sistema, quanto para os seus usuários.

Prosseguindo o nosso estudo a respeito do processamento ou análise, a qual é submetida um discurso no momento que este ingressa no **Processo Interpretador de Sentenças**, veremos agora o componente responsável pela análise sintática de uma sentença, ou seja, o conjunto de regras que a verifica sob o aspecto estrutural.

As regras que formam o analisador sintático do **Processo Interpretador**, as quais definem as classes de sentenças da língua portuguesa reconhecidas pelo **SDIP**, foram codificadas em DCG. Todavia, não aprofundaremos neste texto a discussão a respeito de tais regras, bem como acerca das classes de sentenças por elas reconhecidas, uma vez que, como dissemos anteriormente, estas são similares àquelas usadas no SRD. O leitor interessado em conhecer este conjunto de regras, deve consultar [FRE 93], onde o autor aborda detalhadamente este tópico.

No restante desta subseção, destacaremos apenas algumas características adicionais por nós definidas e agregadas às regras de reconhecimento sintático discutidas em [FRE 93]. São elas:

- Possibilidade do reconhecimento de sentenças na forma imperativa, ou seja, aquelas iniciadas por um verbo conjugado segundo as regras definidas na língua portuguesa para este tipo de construção, cujo sentido é de exortar o ouvinte a cumprir uma ordem por ela expressa;

- Os nomes próprios, substantivos e outros termos de uma sentença, podem ser compostos, isto é, formados por mais de um termo individual. Um exemplo deste tipo de construção seria o nome próprio *João da Silva*, formado por dois nomes próprios simples e uma preposição;

- Graças à atuação conjunta dos analisadores léxico e sintático, foi possível incrementar a complexibilidade das sentenças reconhecidas pelo **SDIP**, principalmente aquelas onde existe a presença simultânea do complemento nominal e do objeto indireto, o que, nas regras originais, ocasionava a geração de árvores sintáticas incorretas. Um exemplo ilustrativo desta classe de sentenças é mostrada em 13.3.

Frase 13.3 Marcos deu a Cláudia um livro de poesias.

Antes de concluirmos esta subseção, vejamos um exemplo de como trabalham conjuntamente os analisadores léxico e sintático, validando uma sentença da língua portuguesa, transformando-a em uma árvore de derivação sintática que a representa.

Considere a frase 13.4:

Frase 13.4 Encontre os livros de "Jean Paul Sartre[1]".

[1]Jean Paul Sartre (1905:1980), romancista, dramaturgo e filósofo francês. Introdutor do pensamento existencialista na França.

Após passar pela fase de análise léxica e sintática, a sentença 13.4 é transformada na lista Prolog mostrada na figura 13.4, a qual representa a árvore de derivação sintática a ela associada.

Na figura 13.4, observamos que, a partir da frase inicial 13.4, foi construída uma estrutura, onde estão representadas informações sintáticas ou estruturais, como, por exemplo, sujeito, predicado e objeto direto, bem como informações léxicas, exemplificadas por nomes próprios, substantivos, verbos etc.

```
[[or,[],[],[sv,[ev,[],at
,[v(encontre,encontrar,spr,s,_,_,pesquisar)]]
,[[sn1,[],[sn,objeto,d(os,p,m,o)
,[en,[],n(livros,p,m),[]]],[orel]]]
,[[sn1,pr(de),[sn,objeto,[],[en,[],
complex([jean,paul,sartre],s,m,np
,[np(jean,s,m),
np(paul,s,m),
np(sartre,s,m)]),[]]],[orel]]]]]]
```

Figura 13.4 Estrutura produzida pela análise léxica e sintática de uma sentença.

13.3.3 Interpretação do Significado das Sentenças

Como vimos na subseção anterior 13.3.2, o processamento de uma sentença na fase de análise léxica e sintática, produz como resultado uma árvore de derivação sintática que a representa estruturalmente.

Nesta subseção, veremos como essa estrutura intermediária é processada, possibilitando que o sistema interprete o seu significado e, conseqüentemente, seja

capaz de executar os comandos que estão implicitamente expressos na sentença original.

Primeiramente, é produzida, a partir da ADS que ingressou no **Interpretador**, a DRS que representa o significado semântico da sentença original.

Na figura 13.5, mostramos a DRS gerada a partir da ADS que aparece na figura 13.4. Destaque-se que, em ambos os casos, as estruturas são representadas através de construções Prolog, não obedecendo, em muitas oportunidades, aos formalismos definidos na literatura da área.

```
ref(1,n(sistema,s,m) )

ref(2,n(livros,p,m) )

ref(3,complex([jean,paul,sartre],s,m,np
,[np(jean,s,m),
np(paul,s,m),
np(sartre,s,m)]) )

drs(0,[3,2,1],[cond(encontrar,[1,2,3])] )
```

Figura 13.5 DRS gerada a partir da árvore sintática mostrada na figura anterior.

Na figura 13.5, observamos que a representação de uma DRS no **SDIP** obedece às regras que seguem:

- Os referentes da DRS são representados pelo fato Prolog *ref/2*, onde o primeiro argumento é um número que individualiza o referente, enquanto o segundo refere-se ao próprio objeto ao qual o referente está associado;
- A estrutura que representa a DRS real é o fato Prolog *drs/3*, no qual o primeiro argumento representa a posição hierárquica da DRS considerada, possibilitando, desta maneira, que se representem subDRSs. O segundo argumento é formado por uma lista Prolog, composta pelos índices que

permitem o acesso aos referentes presentes no universo da DRS considerada.

Note-se que, ao contrário da representação adotada por Hans Kamp em [KAM 90], onde eram usadas letras para representar os referentes, por razões de implementação, utilizamos números com o mesmo intuito.

Finalmente, o terceiro argumento do fato em drs/3 representa as condições, onde aparecem os verbos da sentença original e seus referentes associados.

Após a fase de construção da DRS, esta mesma estrutura sofre uma transformação adicional com o intuito de produzir um fato Prolog único, que possa ser mais facilmente trabalhado do que a DRS original, composta por diversos fatos.

Na figura 13.6, mostramos a estrutura resultante da transformação sofrida pela DRS da figura 13.5.

```
frase([ref(1,n(sistema,s,m)),

ref(2,n(livros,p,m)),

ref(3,complex([jean,paul,sartre],s,m,np
,[np(jean,s,m)
,np(paul,s,m)
,np(sartre,s,m)]))]

,[drs(0,[3,2,1],[cond(localizar,[1,2,3])])])
```

Figura 13.6 Fato Prolog gerado com o processamento da DRS mostrada na figura anterior.

A partir deste momento, estamos prontos para determinar os comandos que estão implicitamente expressos na sentença 13.3 para, posteriormente, executá-los. Para tanto, é realizada uma pesquisa na base de sentenças genéricas do sistema, objetivando localizar uma entrada onde esteja representada a sentença considerada.

Na eventualidade de não se encontrar esta informação, o **Processo Interpretador de Sentenças** inicia o procedimento descrito na subseção 13.3.4, que lhe permitirá construir e incluir esta nova entrada na base de sentenças genéricas.

Contudo, por agora, suponhamos que o **Interpretador** localize uma entrada na base de sentenças genéricas, onde esteja representada a estrutura mostrada na figura 13.6.

Na figura 13.7, são mostradas duas informações importantes. Na parte superior, aparece um esboço da entrada da base de sentenças genéricas que satisfez a pesquisa realizada pelo **Interpretador**. Ainda observando a parte superior da figura 13.7, constatamos que uma entrada desta base de dados é representada pelo fato Prolog *sentence/2*, onde o primeiro argumento é a própria estrutura que está sendo procurada, enquanto o segundo é composto pelos comandos que estão implicitamente expressos pela sentença original. Também, observamos que em ambos os argumentos, aparecem letras maiúsculas, que serão instanciadas com os valores apropriados no momento em que houver o casamento do primeiro argumento do fato *sentence/2*, com a estrutura que serve como chave de busca.

Finalmente, na parte inferior da figura 13.7, mostramos a aparência da entrada da base de sentenças genéricas, quando ocorre o casamento com a estrutura usada como chave de busca. Aqui, podemos observar que as variáveis visíveis na parte superior da figura 13.7, agora estão instanciadas com os valores adequados.

Com o término desta fase, o sistema já está capacitado a executar os comandos que atenderão o pedido do usuário, expresso na sentença original. Para isso, o **Interpretador** ativa a regra Prolog, cuja cabeça é o segundo argumento da entrada localizada dentro da base de sentenças genéricas, com seus argumentos devidamente instanciados.

Nesta regra, serão feitas algumas tentativas de refinar a consulta ou ordem formulada pelo usuário, utilizando-se, conforme o caso, a base de conhecimento, base de informações Archie ou ambas.

```
sentence(frase([ref(1,n(sistema,s,m)),

ref(2,n(livros,p,m)),

ref(3,complex(Autor,Num,Gen,np,List

,[drs(0,[3,2,1],[cond(localizar,[1,2,3])])])

,search_library(Autor,Title,Subject,Publisher,Servers))
```

```
sentence(frase([ref(1,n(sistema,s,m)),

ref(2,n(livros,p,m)),

ref(3,complex([jean,paul,sartre],s,m,n
,[np(jean,s,m)
,np(paul,s,m)
,np(sartre,s,m)]))]

,[drs(0,[3,2,1],[cond(localizar,[1,2,3])])])

,[search_library([jean,paul,sartre],_,_,_,[server('asterix.ufrgs.br','sabibib')])])
```

Figura 13.7 Entrada da base de sentenças genéricas, considerada em dois momentos.

Considerando o exemplo particular mostrado na figura 13.7, a regra *search_library/5* realizará adicionalmente a geração da expressão na linguagem CDS-ISIS correspondente à consulta formulada pelo usuário, baseando-se nas informações presentes na sentença original (nome do autor), somando-se àquelas obtidas com o refinamento da mesma.

Finalmente, através de comandos enviados à Interface Gráfica, as operações que estavam expressas na sentença original são executadas, concluindo-se, desta forma, um ciclo de operação do sistema, iniciado no momento da submissão de uma sentença em língua portuguesa por parte do usuário.

13.3.4 Aprendendo o Significado de Sentenças

Nesta subseção, discutiremos a situação ocorrida quando o **Interpretador**, ao pesquisar na base de sentenças genéricas, não consegue encontrar uma entrada, cujo primeiro argumento seja similar à estrutura gerada a partir da sentença original, usada como chave de busca. Sendo assim, o **Processo Interpretador de Sentenças** deve construir uma estrutura que contenha esta informação e inseri-la na base de sentenças genéricas para que seja usada em futuras oportunidades.

O aprendizado do significado de uma dada sentença e, conseqüentemente, a determinação dos comandos por ela implicitamente expressos é feito por intermédio do ambiente gráfico, com o auxílio indispensável do usuário que formulou a consulta inicial.

Mais precisamente, no momento em que o **Interpretador** não consegue identificar o significado de uma sentença, uma mensagem é enviada ao ambiente gráfico, fazendo-o passar para o estado de aprendizagem. Neste momento, o usuário, utilizando-se das janelas, menus e demais estruturas presentes no ambiente gráfico, deve realizar todas as operações que estão expressas em sua sentença original, ou seja, aquelas ações que deveriam ser executadas automaticamente pelo **SDIP** com o, intuito de satisfazer plenamente o desejo do usuário, manifesto originalmente em língua natural. Ao concluir-se esta fase, o controle é retomado pelo **Processo Interpretador de Sentenças**, mais especificamente pelo componente de aprendizado, onde, finalmente, é construída a entrada que é inserida na base de sentenças genéricas.

Um exemplo de como se utilizar a Interface Gráfica para ensinar ao sistema o significado de uma sentença em língua portuguesa é mostrado no próximo capítulo deste trabalho 14, na seção 14.3.9.

13.3.5 Base de Conhecimento

O objetivo principal do **SDIP** é, como dissemos anteriormente, auxiliar os usuários na tarefa de localizar informações dentro da rede Internet.

Todavia, a capacidade efetiva do sistema em apoiar os usuários nesta atividade dependerá fundamentalmente da quantidade, qualidade, estrutura e complexibilidade dos conhecimentos ou informações, relevantes para este fim, colocadas à sua disposição. Sendo assim, quanto maior for o tempo e esforço dedicados à elaboração, organização e construção da base de conhecimento do sistema, melhores e mais completas serão as respostas por ele oferecidas aos usuários finais.

Considerando os fatos acima relatados, aliado ao objetivo de aumentar-se a flexibilidade e a potencialidade do **SDIP**, construiu-se, junto ao **Processo Interpretador de Sentenças**, uma base de conhecimento, cuja a finalidade é fornecer os subsídios necessários à correta interpretação das sentenças, principalmente no que tange ao aperfeiçoamento da consulta por ela expressa e, também, ao tipo de informação que o usuário deseja de fato recuperar.

Com o intuito de possibilitar aos usuários a facilidade de construir uma base de conhecimento adequada a sua realidade e necessidades, o sistema, por intermédio da Interface Gráfica, provê o conjunto de funções que lhes permitem visualizar, incluir e excluir entradas da base de conhecimento.

Uma descrição detalhada de como utilizar estas facilidades será realizada no próximo capítulo 14, na seção 14.3.8.

O conhecimento mantido e utilizado pelo **Processo Interpretador de Sentenças** está representado por intermédio de uma rede semântica [WIN 84]. Em cada uma das entradas da base de conhecimento está definido um arco da rede , o qual expressa uma relação existente entre dois grupos compostos por um ou mais objetos.

A seguir, definimos o conjunto das relações que podem ser usadas na construção das entradas da base de conhecimento do SDIP.

Na tabela 13.4, mostramos exemplos do uso destas relações.

Tabela 13.4 Exemplos de fragmentos de conhecimento construídos a partir das relações definidas no SDIP.

Relação	Exemplo
é_parte_de	{Nível de transporte} é_parte_de {Modelo de Referência OSI}
é_um	{TCP} é_um {protocolo de transporte}
são	{{SDLC}, {HDLC}, {BSC}} são {protocolos de ligação de dados}
é_autor_de	{Graciliano Ramos} é_autor_de {{Memórias do Cárcere}, {Vidas Secas}}
são_autores_de	{{Brain Kernighan}, {Dennis Ritchie}} são_autores_de {The C Programming Language}
é_editor_de	{Prentice-Hall} é_editor_de {Computer Networks}
é_sinônimo_de	{FDDI} é_sinônimo_de {Fiber Distributed Data Interface}
escreve_sobre	{Georg Wilhelm Friedrich Hegel} escreve_sobre {filosofia}

- **é_um** -> estabelece uma relação de generalidade entre dois objetos;
- **são** -> define uma relação de generalidade, existente entre vários objetos com um único objeto;
- **é_autor_de** -> estabelece uma relação de autoria de um ou mais trabalhos;

- **são_autores_de** -> estabelece uma relação entre um conjunto de autores com suas respectivas obras;
- **é_editor_de** -> define a relação existente entre um editor e as obras ou autores por ele publicados;
- **é_sinônimo_de** -> estabelece uma associação entre objetos sinônimos;
- **escreve_sobre** -> define uma relação entre autores e assuntos sobre os quais eles escrevem;
- **é_parte_de** -> define uma relação em que um objeto é parte integrante de outro.

Particularmente, no tocante à implementação, cada uma das entradas da base de conhecimento é representada pelo fato Prolog *net/3*, onde o primeiro e o terceiro argumentos definem o universo ou o conjunto de objetos sobre os quais atua a relação que, por sua vez, está definida no segundo argumento.

13.3.6 Base de Informações Archie

Como vimos anteriormente, o **SDIP** é capaz de, alternativamente, encaminhar consultas ao sistema Archie, auxiliando os usuários a utilizar adequada e eficientemente os recursos proporcionados por este servidor de informações, residente na rede Internet.

Além de oferecer aos usuários a possibilidade de formular consultas ao sistema Archie por intermédio de expressões ou sentenças em língua natural, o **SDIP** também é capaz de determinar qual o servidor que enviará as respostas que melhor atenderão às necessidades do usuário, levando-se em consideração os seguintes critérios de avaliação:

1. Tempo de resposta observado para cada um dos servidores Archie considerados;

2. Número de respostas distintas recebidas de cada servidor consultado. Portanto, caso existam dois ou mais arquivos idênticos, eles serão contabilizados somente uma única vez;

3. Quantidade de ítens do tipo diretório que fazem parte da resposta enviada pelos servidores Archie;

4. Número de ocorrências que diferenciam as respostas enviadas pelos diversos servidores consultados, ou seja, considera-se tão somente aqueles arquivos e/ou diretórios conhecidos por um único servidor;

5. Quantidade de respostas positivas recebidas, independentemente de sua categoria ou redundância.

Observe que a seleção ou ordenamento dos servidores que serão consultados baseia-se unicamente em critérios quantitativos. Contudo, acreditamos que tais critérios são suficientes para que, via de regra, as consultas dos usuários sejam encaminhadas ao servidor que lhe fornecerá o melhor conjunto de respostas, no menor espaço de tempo.

Adicionalmente, é importante destacar dois detalhes a respeito da seleção do servidor Archie ao qual será encaminhada uma determinada consulta. São eles:

- A ordem na qual são consultados os servidores é verificada periodicamente, permitindo, desta forma, que alterações quantitativas verificadas nos conjuntos de respostas recebidas, bem como no tempo de acesso, reflitam-se no ordenamento dos sistemas Archie utilizados;

- O tempo de resposta medido em uma rede similar à Internet sofre uma grande influência do horário em que se está usando um determinado serviço, em virtude da variação do tráfego de dados na rede. Desta forma, dividimos o dia de vinte e quatro horas em seis intervalos iguais de quatro horas e, associamos a cada um deles, diferentes listas ordenadas de servidores Archie. Por conseguinte, consultas que sejam realizadas em

um determinado horário serão encaminhadas conforme determina a informação mantida na lista de ordenamento a ele associada. Adotando esta metodologia, esperamos que os tempos de resposta observados em uma consulta sejam sempre aceitáveis, visto que em cada uma das seis listas de ordenação está em primeiro lugar o servidor de menor tempo de acesso no intervalo considerado.

O **Processo Interpretador de Sentenças** mantém um conjunto de dados denominado base de informações Archie, que lhe permite, a partir de uma sentença em língua natural, determinar a chave de busca e os parâmetros de operação que serão usados no momento da consulta, além das listas de servidores, ordenadas segundo os critérios anteriormente descritos, que lhe informam qual o sistema Archie que deve ser contactado inicialmente.

Adicionalmente, em cada uma das entradas da base de informações Archie está presente uma palavra e/ou expressão em língua portuguesa, a qual é usada como campo de indexação pelo **Interpretador de Sentenças**. No momento em que este componente retira da consulta, já transformada em uma DRS, com a ajuda da base de conhecimento, a informação que de fato constitui-se no núcleo da consulta do usuário, ele a utiliza para identificar a entrada e, conseqüentemente, o servidor Archie que deverá receber a requisição, bem como seus parâmetros de funcionamento. A operação de acesso ao servidor será efetivada por intermédio da Interface Gráfica, assemelhando-se ao que é feito quando de uma consulta ao sistema SABI.

De maneira análoga à base de conhecimento, o conjunto de dados descritos nesta subseção também pode ser modificado pelos usuários, tornando-o mais adequado as suas necessidades individuais. No próximo capítulo 14, na seção 14.3.4, descrevemos detalhadamente como os usuários devem proceder para construir uma base de informações Archie personalizada.

Tabela 13.5 Especificação em SDL-pr do Processo Interpretador de Sentenças.

BLOCK	B_PIS;	/*DEF. DO PIS*/
SIGNALROUTE R1		
FROM P_CREATE	TO ENV	WITH S_PIS_IG_1
FROM ENV	TO P_CREATE	WITH S_CR;
SIGNALROUTE R2		
FROM PIS	TO ENV	WITH S_PIS_IG_1, S_PIS_IG_2
FROM ENV	TO PIS	WITH S_IG_PIS_1, S_IG_PIS_2;
SIGNALROUTE R3		
FROM P_CREATE	TO ENV	WITH S_PIS_SCE
FROM ENV	TO P_CREATE	WITH S_CR;
SIGNALROUTE R4		
FROM PIS	TO ENV	WITH S_PIS_SCE
FROM ENV	TO PIS	WITH S_SCE_PIS;
SIGNALROUTE R5		
FROM PIS	TO ENV	WITH S_PIS_ENV
FROM ENV	TO PIS	WITH S_ENV_PIS;
CONNECT C_IG_PIS	AND	R1,R2;
CONNECT C_SCE_PIS	AND	R3,R4;
CONNECT C_PIS_ENV	AND	R5;

13.3.7 Especificação em SDL-PR do Interpretador de Sentenças

Nesta seção, mostraremos uma especificação em SDL-PR, que descreve o comportamento do **Processo Interpretador de Sentenças** (tabelas 13.5, 13.6, 13.7, 13.8, 13.9 e 13.10). A partir desta representação em SDL-PR [INT 88, BEL 91], o leitor compreenderá claramente o funcionamento básico e o papel desempenhado por este processo dentro do ambiente **sdip**.

Tabela 13.6 Especificação em SDL-pr do Processo Interpretador de Sentenças.

PROCESS	P_CREATE;	/*INICIA O PIS*/
START;		
STATE	ST_CREATION;	
INPUT	SYSTEM SDIP/	/*SIGNAL USADA
	BLOCK B_PIS/	PARA CRIAR
	R1 S_CR,	UM PROC. */
	SYSTEM SDIP/	
	BLOCK B_PIS /	
	R3 S_CR;	
CREATE	PIS;	/*ATIVA O PIS*/
DECISION	SENDER;	
(P_IG):	OUTPUT S_PIS_IG_1;	
(P_SCE):	OUTPUT S_PIS_SCE;	
ENDDECISION;		
STOP;		
ENDSTATE	ST_CREATION;	
ENDPROCESS	P_CREATE;	

Tabela 13.7 Especificação em SDL-pr do Processo Interpretador de Sentenças.

PROCESS	PIS;	/*PIS*/
DCL	FRASE	BOOLEAN;
STATE	ST_MAIN;	
INPUT	S_IG_PIS_1(CMD),	/*SIGNAL TRÁS
	S_SCE_PIS(CMD);	OS COMANDOS
		PARA O PROC.*/
DECISION	CMD;	
('CONFIGURAÇÃO'):		/* COMANDOS QUE
		ALTERAM AS BASES
		DE INFORMAÇÕES*/
NEXTSTATE	ST_CONFIGURE;	
('INFORMAÇÕES'):		/*LÊ INFORMAÇÕES
		DAS BASES DE
		DADOS */
('INTERPRETAR'):		/*INTERPRETA
		SENTENÇAS*/
DECISION	SENDER;	
(P_IG):		
NEXTSTATE	ST_INT_IG;	
(P_SCE):		
NEXTSTATE	ST_INT_SCE;	
ENDDECISION:		
('PARAR'):		
DECISION	SENDER;	
(P_IG):	OUTPUT S_PIS_IG_1;	
(P_SCE):	OUTPUT S_PIS_SCE;	
ENDDECISION;		
STOP;		
ENDDECISION;		
ENDSTATE	ST_MAIN;	

Tabela 13.8 Especificação em SDL-pr do Processo Interpretador de Sentenças.

STATE	ST_CONFIGURE;	/*MODIFICA AS BASES DE DADOS*/
OUTPUT	S_PIS_ENV;	/*GRAVA OS NOVOS DADOS*/
OUTPUT	S_PIS_IG_1;	/*INFORMA RESULTADOS DA OPERAÇÃO*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_CONFIGURE;	
STATE	ST_INFO;	
INPUT	S_ENV_PIS;	/*LÊ AS BASES DE DADOS*/
OUTPUT	S_PIS_IG_1;	/*ENVIA OS DADOS PARA O MIG*/
NEXTSTATE	ST_MAIN;	
ENDSTATE	ST_INFO;	

Tabela 13.9 Especificação em SDL-pr do Processo Interpretador de Sentenças.

STATE	ST_INT_IG;	/*INTERPRETA FRASES VINDAS DO MIG*/
...		/*PROCESSO DE INTERPRETAÇÃO*/
DECISION (TRUE):	FRASE;	/*CONSEGUIU INTERPRETAR*/
OUTPUT NEXTSTATE (FALSE):	S_PIS_IG_1; ST_MAIN;	/*NÃO CONSEGUIU INTERPRETAR*/
OUTPUT	S_PIS_IG_1, S_PIS_IG_2;	/*INFORMA A IG*/ /*PASSA PARA O EESTADO DE APRENDIZADO*/
NEXTSTATE	ST_WAIT;	/*REALIZA APRENDIZADO*/
ENDDECISION; ENDSTATE	ST_INT_IG;	

Tabela 13.10 Especificação em SDL-pr do Processo Interpretador de Sentenças.

STATE INPUT	ST_WAIT; S_IG_PIS_2;	/* APRENDE*/ /*APRENDE COM A AJUDA DO MIG*/
...		/*PROCESSO DE APRENDIZADO*/
NEXTSTATE ENDSTATE	ST_MAIN; ST_WAIT;	
STATE	ST_INT_SCE;	/*INTERPRETA AS FRASES VINDAS DO SCE*/
...		/*PROCESSO DE INTERPRETAÇÃO*/
OUTPUT ENDSTATE ENDPROCESS ENDBLOCK	S_PIS_SCE; ST_INT_SCE; PIS; B_PIS;	/*RESPONDE AO SCE*/

14 USANDO O SDIP

14.1 Introdução

O aproveitamento de todas as potencialidades e facilidades oferecidas por uma aplicação, depende fundamentalmente do nível de conhecimento que o usuário possui a respeito do sistema, bem como da experiência prática por ele adquirida com a contínua e freqüente utilização do sistema.

Uma vez que o público alvo do **SDIP** é constituído, em grande parte, por pessoas não especializadas na área de informática, devemos discutir o assunto interface usuário e sistema de uma maneira clara e objetiva, sem deter-nos excessivamente em particularidades. Assim, neste capítulo, faremos uma descrição detalhada das facilidades, potencialidades e maneiras de usar adequadamente o ambiente **SDIP**.

Além de satisfazer aos usuários iniciantes, que buscam neste texto uma orientação precisa no que concerne ao uso do sistema, também escrevemos este capítulo de forma que o leitor especializado possa analisar com clareza a filosofia e o desenho de interface aqui adotados.

Inicialmente, estudaremos brevemente o uso do **SDIP** através do envio de mensagens de correio eletrônico ao sistema.

Posteriormente, descreveremos a interface gráfica do sistema, apresentando cada uma de suas facilidades e singularidades características.

14.2 Usando o Sistema Através do Correio Eletrônico

Os usuários que não têm acesso a uma estação de trabalho, que lhes permita a utilização do **SDIP** através de sua interface gráfica, não estão impossi-

bilitados de se beneficiar das facilidades implementadas pelo sistema. Para tanto, basta os usuários encaminharem uma mensagem de correio eletrônico ao sistema, contendo em seu corpo um discurso em língua portuguesa, que expresse inequivocamente a sua necessidade ou desejo. Ao receber a mensagem, o sistema interpretará o discurso redigido pelo usuário, realizará as operações necessárias com o intuito de obter os resultados esperados e, finalmente, os encaminhará novamente ao usuário, colocando-os em uma mensagem de correio eletrônico.

Como dissemos no capítulo 11, somente o serviço de recuperação de dados bibliográficos, com o uso de sentenças em língua portuguesa, está a disposição dos usuários através desta interface. Portanto, para acesso aos serviços providos pelo sistema Archie, combinada a transferência de arquivos, deve-se obrigatoriamente utilizar a interface gráfica.

Antes de mostrarmos alguns exemplos de sentenças reconhecidas pelo sistema, devemos estabelecer algumas restrições, no que concerne à estrutura sintática e morfológica das sentenças, bem como em relação à constituição dos termos léxicos componentes. É importante lembrar que tais restrições aplicam-se, de maneira idêntica, ao uso de sentenças em língua portuguesa dentro da interface gráfica do sistema. São elas:

1. Somente serão reconhecidas pelo sistema as sentenças, cuja estrutura seja representável pelas regras sintáticas definidas em [FRE 93] e extendidas neste trabalho, conforme descrito no capítulo 13, seção 13.3.2;

2. Não é permitido o uso de pronomes de qualquer tipo e, por extensão, é vedada a presença de anáforas dentro das sentenças;

3. Caso exista mais de uma frase em um mesmo discurso, elas serão interpretadas não individualmente, mas como se formassem um conjunto único e inseparável, no qual existe uma relação entre os componentes;

4. Cada linha de texto é processada isoladamente, ou seja, o contexto do sistema renova-se à medida em que cada uma delas é interpretada;

5. Termos de uma sentença, que precisam ser processados como uma única unidade léxica, devem aparecer envolvidos por aspas (" "), como, por exemplo, títulos de livros formados por mais de uma palavra, nos quais a sua estrutura complexa impediria a interpretação correta da sentença;

6. Nome de arquivos, diretórios e servidores, quando presentes em uma sentença, deverão estar envolvidos por apóstrofos (' '), em virtude da não existência de regras que definam claramente a sua forma de construção, permitindo a ocorrência das mais variadas e estranhas combinações de caracteres;

7. Em algumas situações, faz-se necessário correlacionar ítens de uma sentença, através de um sublinhado ('_'), para que o sistema não seja induzido a interpretá-la erradamente. Um exemplo deste fato é verificado quando, em uma sentença, existir um complemento nominal e o sistema o interpretar como um objeto indireto. Neste caso, deve-se unir o nome e seu complemento, utilizando-se um sublinhado.

A seguir, mostramos alguns exemplos de sentenças reconhecidas pelo sistema **SDIP**:

Frase 14.1 Localize os livros escritos por Hegel[1].

Frase 14.2 Encontre os livros de autoria de "Pablo Neruda"[2].

Frase 14.3 Encontre os livros_de_filosofia escritos por Nietzsche[3].

Frase 14.4 Localize livros de redes_de_computadores.

[1]Georg Wilhelm Friedrich Hegel (1770:1831), filósofo alemão que viveu na segunda metade do século dezoito e início do século dezenove. Autor de importantes trabalhos no campo da Filosofia e do Direito.

[2]Poeta e romancista chileno. ganhador do Prêmio Nobel de Literatura em 1971.

[3]Friedrich Wilhelm Nietzsche (1844:1900), importante filósofo alemão do final do século dezenove.

Nas sentenças 14.1 e 14.2, o sistema procurará por todos os livros catalogados no SABI, cujos autores sejam respectivamente Hegel e Pablo Neruda. Por outro lado, na sentença 14.3, o usuário expressa o seu desejo de localizar livros escritos por Nietzsche, na área de filosofia. Finalmente, na sentença 14.4, o **SDIP** solicitará ao sistema SABI que localize todos os livros, cujo assunto central seja redes de computadores.

É importante destacar que, em nenhum momento, o usuário ficará sem nenhum tipo de resposta do sistema. Caso ocorra um problema de má formação estrutural da sentença, o **SDIP** enviará uma mensagem ao usuário, relatando-lhe o acontecido.

Adicionalmente, se não for possível para o sistema interpretar o significado de uma sentença, ele solicitará a intervenção do administrador, que o ajudará na realização desta tarefa. No momento em que for concluída esta fase de aprendizagem, o sistema estará apto a reconhecer e tratar sentenças similares a que ocasionou o início da fase de aprendizagem, além de ser capaz de responder imediatamente à consulta formulada pelo usuário.

14.3 O Ambiente Gráfico

Como vimos anteriormente, o **SDIP**, além de implementar uma interface baseada na troca de mensagens de correio eletrônico entre os usuários e o sistema, também possui um módulo gráfico que permite aos usuários um diálogo interativo com o **SDIP**.

Em virtude de o módulo gráfico possibilitar um diálogo cooperativo entre os usuários e o sistema, foi possível acrescentar algumas facilidades adicionais, não presentes no modelo de interface anterior. A seguir, relacionamos as principais diferenças e facilidades adicionais existentes no ambiente gráfico, e não implementadas pela interface descrita na seção anterior 14.2:

- Além de permitir a formulação de uma consulta ao sistema SABI com o uso de sentenças em língua natural, o módulo gráfico também possibilita o uso convencional daquele sistema, porém através de um ambiente composto por janelas, menus e campos;

- As consultas não são dirigidas unicamente ao SABI, sendo possível recuperar informações bibliográficas mantidas por outros sistemas similares;

- O módulo gráfico, além do SABI, também utiliza os serviços providos pelo sistema Archie com o intuito de localizar informações. As consultas dos usuários podem ser expressas em sentenças da língua portuguesa, palavras ou expressões chave ou, ainda, padrões de busca característicos do sistema Archie;

- A interface gráfica permite a seus usuários recuperar arquivos por intermédio do protocolo FTP;

- O usuário pode adicionar ítens léxicos desconhecidos ao dicionário do sistema;

- O usuário pode ensinar ao sistema o significado das sentenças que por ventura ele desconheça;

- A base de conhecimento do **SDIP** pode, a qualquer momento, ser alterada pelos usuários.

É importante destacarmos que o uso de sentenças da língua portuguesa somente aplica-se a consultas formuladas a um dos servidores, Archie ou SABI, bem como à manipulação dos resultados obtidos nas mesmas. Por conseguinte, operações, tais como, atualização da base de conhecimento ou do dicionário do sistema, somente poderão ser efetivadas através dos menus, janelas e demais estruturas postas à disposição dos usuários pelo ambiente gráfico do **SDIP**.

14.3.1 Instalação

Os procedimentos necessários à instalação do sistema **SDIP** são bastante simples, possibilitando que o próprio usuário os execute.

Inicialmente, deve-se expandir o arquivo *"sdip.tar.Z"*, utilizando para este fim o programa *uncompress*.

Posteriormente, através do programa *tar*, deve-se extrair do arquivo *"sdip.tar"* todos os arquivos e diretórios que formam o sistema. Neste momento, será criado um conjunto de diretórios, dentro dos quais se localizam os códigos fonte de cada um dos módulos do sistema.

Finalmente, o usuário deve, dentro do diretório *"dir_graphic"*, executar o comando *make*, iniciando a geração dos programas executáveis apropriados para a sua plataforma computacional. Nesta mesma fase, também são criados dois arquivos de configuração no ("home directory") do usuário, os quais serão detalhados na próxima seção 14.3.2.

Destaque-se que, seguindo estes procedimentos, o usuário criará um ambiente que executará em uma única estação de trabalho. Todavia, para instalar o sistema distribuído em diversas estações, o usuário deve proceder da seguinte forma:

- Manter na estação central, obrigatoriamente, os dois arquivos de configuração e o diretório associado a **Interface Gráfica** ("dir_graphic");
- Determinar as estações onde residirão cada um dos demais módulos do **SDIP**, criando em cada uma o respectivo diretório com seus arquivos;
- Modificar o arquivo de configuração *.sdiphostsrc*, fazendo-o refletir a nova disposição do sistema.

14.3.2 Inicialização do Sistema

Antes de ingressarmos de fato na descrição das estruturas que formam a interface gráfica, mostrando exemplos de como utilizá-las adequadamente, devemos tecer alguns comentários a respeito da configuração inicial do sistema, notoriamente no que concerne à disposição dos processos e arquivos de configuração.

Como tivemos a oportunidade de enfatizar, os processos componentes do **SDIP** não precisam residir em uma única estação de trabalho, podendo, por exemplo, estarem distribuídos em um conjunto de computadores, todos conectados a uma rede local. No entanto, é importante que o usuário tenha em mente as seguintes observações:

- O componente do sistema que implementa o ambiente gráfico necessita de dois arquivos de configuração inicial. Ele os procurará no ("home directory") do usuário, na estação em que o processo estiver executando;
- O processo que implementa o protocolo de transferência de arquivos FTP, sempre terá como base de referência o ("home directory") do usuário, na estação em que ele estiver executando;
- Os processos envolvidos no interfaceamento com o sistema SABI deverão residir em uma estação de trabalho, na qual eles executem como o usuário ("root") ou com previlégios idênticos. Esta restrição advém do fato de que a função *rcmd()*, usada por estes processos, somente será bem sucedida caso tenha sido chamada por um processo que execute como ("root") ou equivalente;
- Resultados relativos a consultas encaminhadas ao sistema SABI devem ser considerados como estando vinculados ao módulo gráfico. Desta forma, quando, por exemplo, pedirmos ao sistema para salvar em arquivo os resultados de uma consulta bibliográfica, ele o fará, tendo como

referência básica a estação em que estiver executando o módulo do ambiente gráfico.

No momento da inicialização, o ambiente gráfico procura no ("home directory") do usuário dois arquivos de configuração do sistema. Caso estes não sejam localizados, ele os cria a partir de informações padronizadas, especificadas no momento da compilação do módulo gráfico.

O primeiro arquivo procurado, denominado *.sdiphostsrc*, armazena informações referentes aos componentes do sistema, mais especificamente, seus tipos ou classes, estações de trabalho onde residem e os nomes dos programas que devem ser executados. Neste arquivo, deve aparecer obrigatoriamente uma entrada associada a cada um dos componentes do **SDIP**.

A figura 14.1 mostra as informações armazenadas no arquivo *.sdiphostsrc*, para uma configuração, onde o módulo gráfico e o processo FTP residem na estação *linus.cesup.ufrgs.br*, enquanto os demais componentes devem ser executados na estação de trabalho *penta.ufrgs.br*.

penta.ufrgs.br	archie	/home/penta/islu/dir_archie/archie
linus.cesup.ufrgs.br	ftp	/home/darwin/islu/dir_ftp/ftp
penta.ufrgs.br	library_sabi	/home/penta/islu/dir_sabi/sabi
penta.ufrgs.br	smart	/home/penta/islu/dir_frase/frase
linus.cesup.ufrgs.br	graphic	/home/darwin/islu/dir_graphic/graphic

Figura 14.1 Arquivo .sdiphostsrc.

É importante notar que o arquivo *sdiphostsrc* é formado por uma seqüência de linhas de texto, onde cada uma delas subdivide-se em três campos, separados por um caractere de espaço. No primeiro campo está especificado o nome da estação, onde deve ser ativado o processo servidor. O segundo campo, informa-

nos o tipo do servidor através de um mnemônico significativo. Finalmente, o terceiro campo determina o nome do arquivo correspondente ao código executável do servidor a ser ativado. Além disso, este campo pode, quando necessário, conter a seqüência de diretórios que conduzem ao arquivo executável correto.

O segundo arquivo de configuração, denominado *.libraryrc*, determina o local onde reside, e o tipo do sistema de recuperação de dados bibliográficos ali encontrado, bem como uma mensagem que permita aos usuários identificá-lo, e a ("username") que deve ser usada para obter-se o direito de acesso ao sistema. Observe que na versão atual do **SDIP**, este arquivo pode ser formado por mais de uma entrada, porém todas com o mesmo tipo de sistema de recuperação de dados bibliográficos, uma vez que o SABI constitui-se no único servidor deste tipo que pode ser contactado presentemente.

```
asterix.ufrgs.br:sabi:"Sistema instalado na UFRGS":"sabibib"
```

Figura 14.2 Arquivo .libraryrc.

A presença deste arquivo (*.libraryrc*) é fundamental para que o **SDIP** seja capaz de determinar, qual o componente especializado e a ("username") que deverão ser utilizadas para acessar os serviços bibliográficos de um dado sistema. Este arquivo será fundamental em futuras versões do **SDIP**, onde esperamos desenvolver processos adicionais que lhe possibilite interagir com outros sistemas de recuperação de dados bibliográficos, além do SABI anteriormente implementado.

A figura 14.2 mostra as informações características, armazenadas no arquivo . *libraryrc*. Observamos que, no presente caso, o separador de campos é o caractere ':', ao contrário do caso anterior, onde esta função era exercida pelo caractere de espaço.

Na hipótese de que não ocorram erros no momento da leitura dos arquivos de configuração ou na fase de inicialização dos processos componentes do

SDIP, o usuário visualizará a janela principal do sistema, podendo, a partir deste momento, utilizar as facilidades implementadas pelo ambiente. Caso contrário, o sistema mostrará uma mensagem, relatando ao usuário o erro ocorrido, e terminará sua execução.

14.3.3 Janela Principal

Após o sistema passar sem a detecção de erros pela fase de inicialização, o usuário observará a abertura da janela principal do **SDIP**, mostrada na figura 14.3.

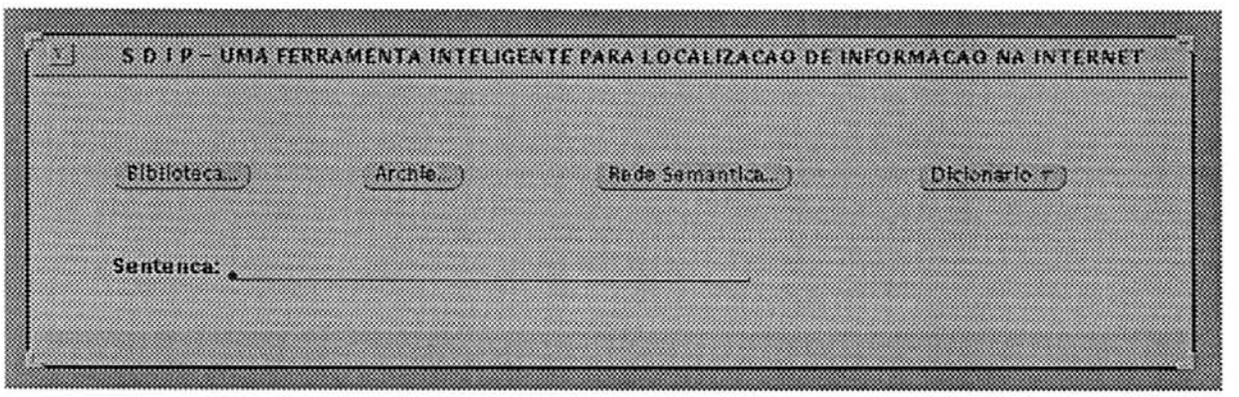

Figura 14.3 *Janela principal do SDIP.*

A partir da janela principal, o usuário pode dirigir-se a um dos cinco submódulos do **SDIP**, a saber:

1. Sistema Archie, no qual o usuário poderá utilizar os serviços providos por esse sistema, em combinação com funções de transferência e visualização de arquivos;

2. Sistema de bibliotecas, permitindo o uso de serviços de pesquisa de dados bibliográficos;

3. Base de conhecimento, possibilitando a realização de operações, tais como, inclusão e exclusão de informações da base de conhecimento do sistema;

4. Dicionário, onde o usuário pode incluir novos termos léxicos, desconhecidos pelo sistema;

5. Sistema de interpretação de sistemas, no qual estão implementadas todas as funções inerentes à interface através de sentenças da língua portuguesa.

Nas próximas subseções, veremos em detalhe cada um destes submódulos.

14.3.4 Janela do Sistema Archie

Utilizando esta janela, mostrada na figura 14.4, o usuário pode interagir com o sistema Archie, realizando uma consulta simples ou inteligente, neste último caso, com a supervisão do **Processo Interpretador de Sentenças**, bem como recuperar os arquivos que ele julgar interessantes com o uso do protocolo FTP e, finalmente, visualizá-los, utilizando-se das ferramentas apropriadas para este fim.

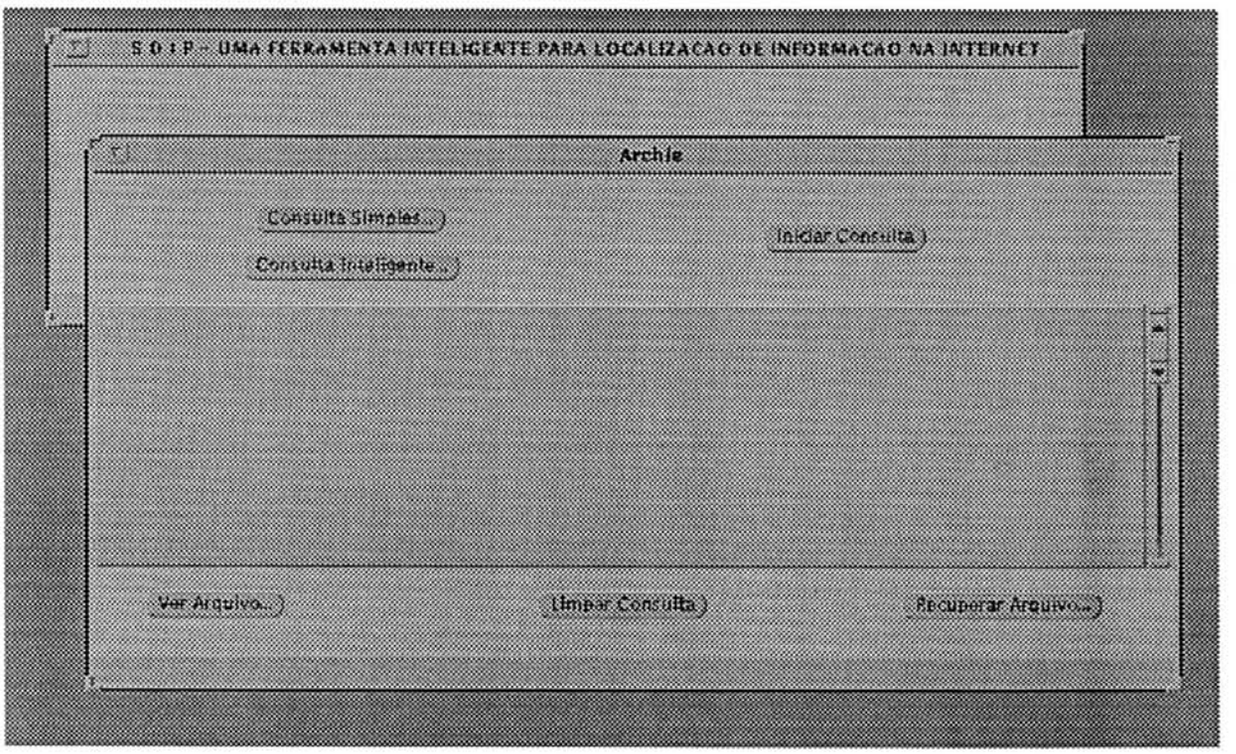

Figura 14.4 *Janela que permite o acesso ao sistema Archie.*

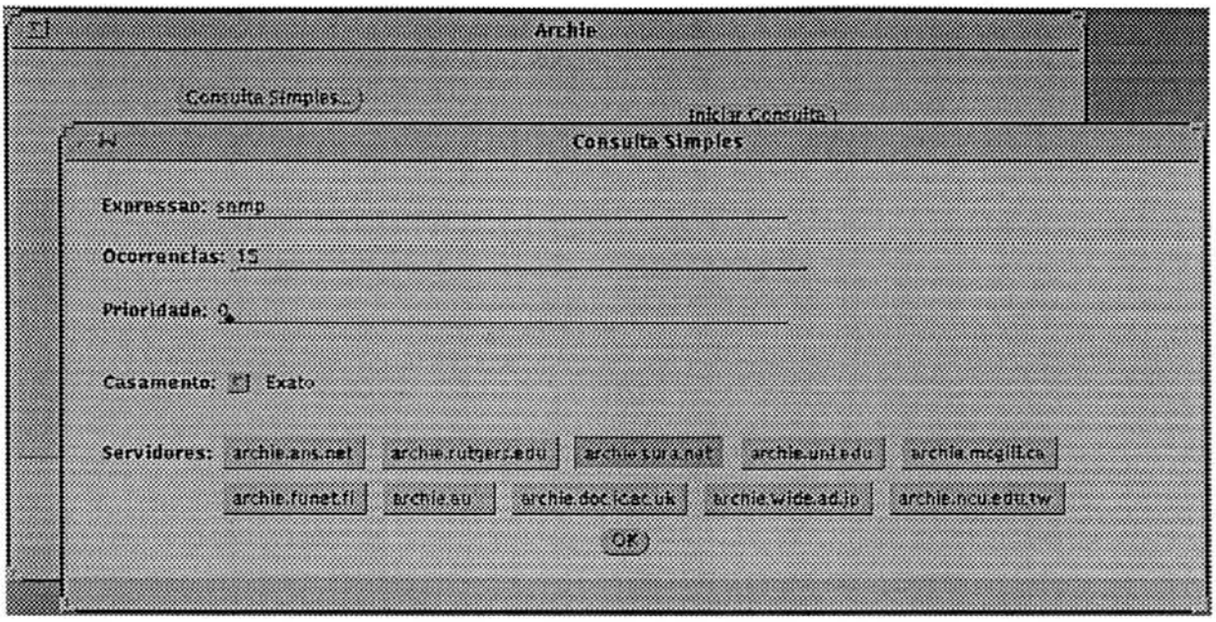

Figura 14.5 *Exemplo de uma consulta simples.*

A figura 14.5 mostra como é realizada uma consulta simples. Neste tipo de operação, os usuários devem ajustar os parâmetros de funcionamento, selecionar os servidores Archie que serão consultados, bem como especificar o padrão de busca que será usado na pesquisa.

Na figura 14.6, mostramos os resultados obtidos a partir da consulta simples, ilustrada na figura 14.5.

Neste momento, os usuários podem verificar as respostas obtidas, navegando no conjunto de ítens mostrados como uma lista de opções.

Nas figuras 14.7, 14.8 e 14.9, mostramos o caminho percorrido pelo usuário até a recuperação de um arquivo por ele considerado relevante.

Por outro lado, em uma consulta inteligente, o usuário informa uma expressão ao sistema e este, com o auxílio do **Processo Interpretador de Sentenças**, determina todos os parâmetros de operação, servidores Archie a serem consultados e os padrões de busca que devem ser usados, com o intuito de obter o conjunto de respostas que satisfaçam as necessidades do usuário.

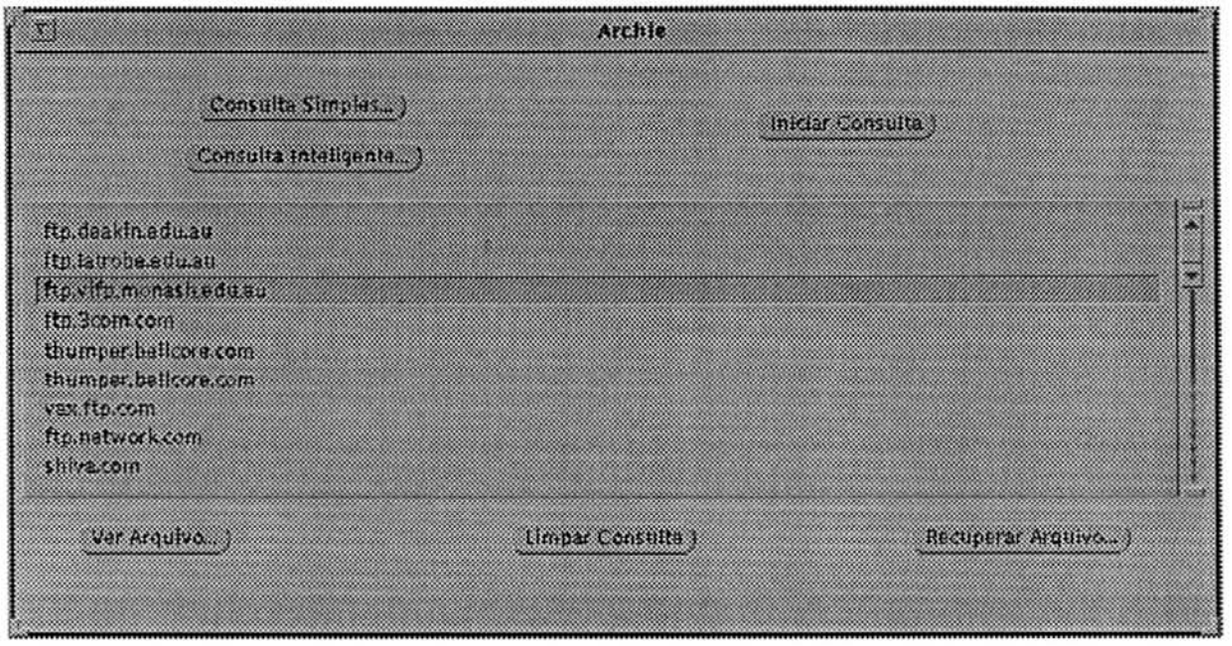

Figura 14.6 *Respostas enviadas pelo servidor Archie.*

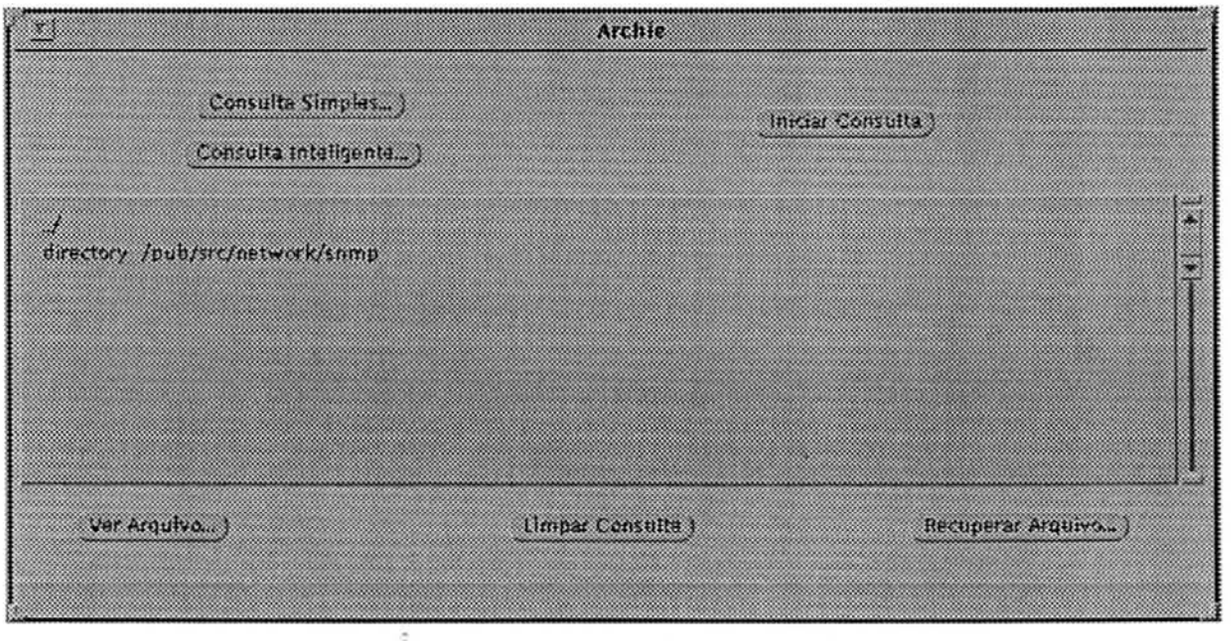

Figura 14.7 *O usuário visualiza as respostas associadas a um servidor FTP específico*

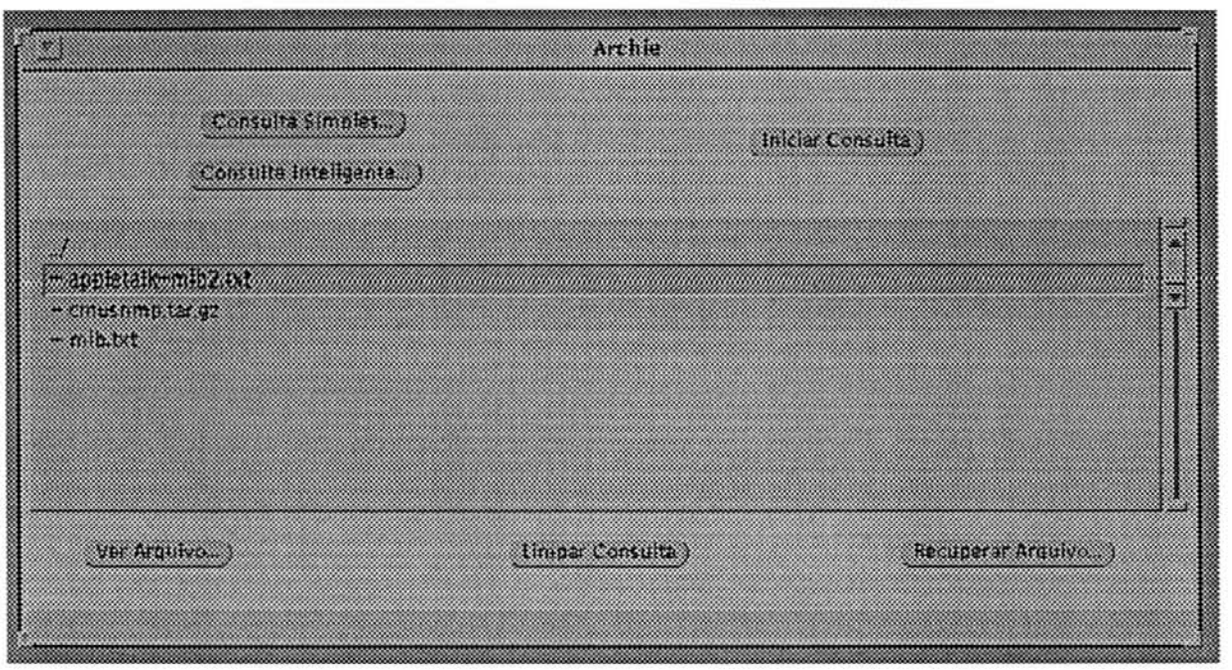

Figura 14.8 *O usuário verifica os arquivos contidos em um diretório.*

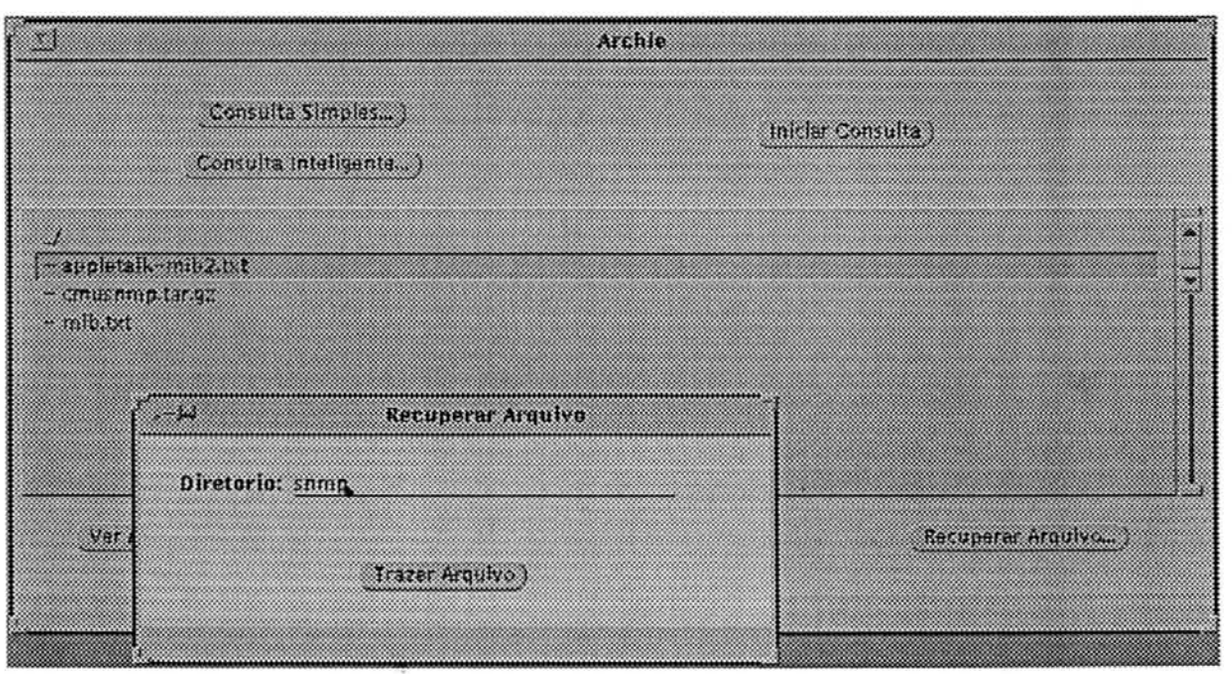

Figura 14.9 *Usuário recupera o arquivo selecionado.*

Nas figuras 14.10 e 14.11, mostramos respectivamente a formulação de uma consulta inteligente ao sistema e os resultados finais obtidos.

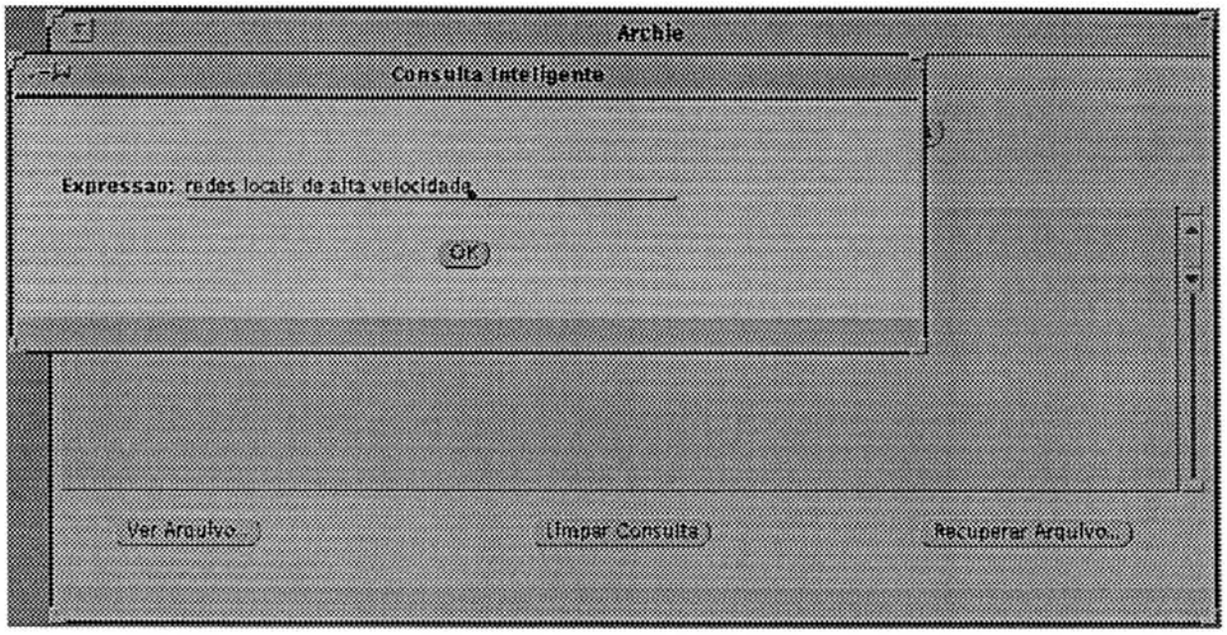

Figura 14.10 *Uso do sistema Archie através de uma consulta inteligente.*

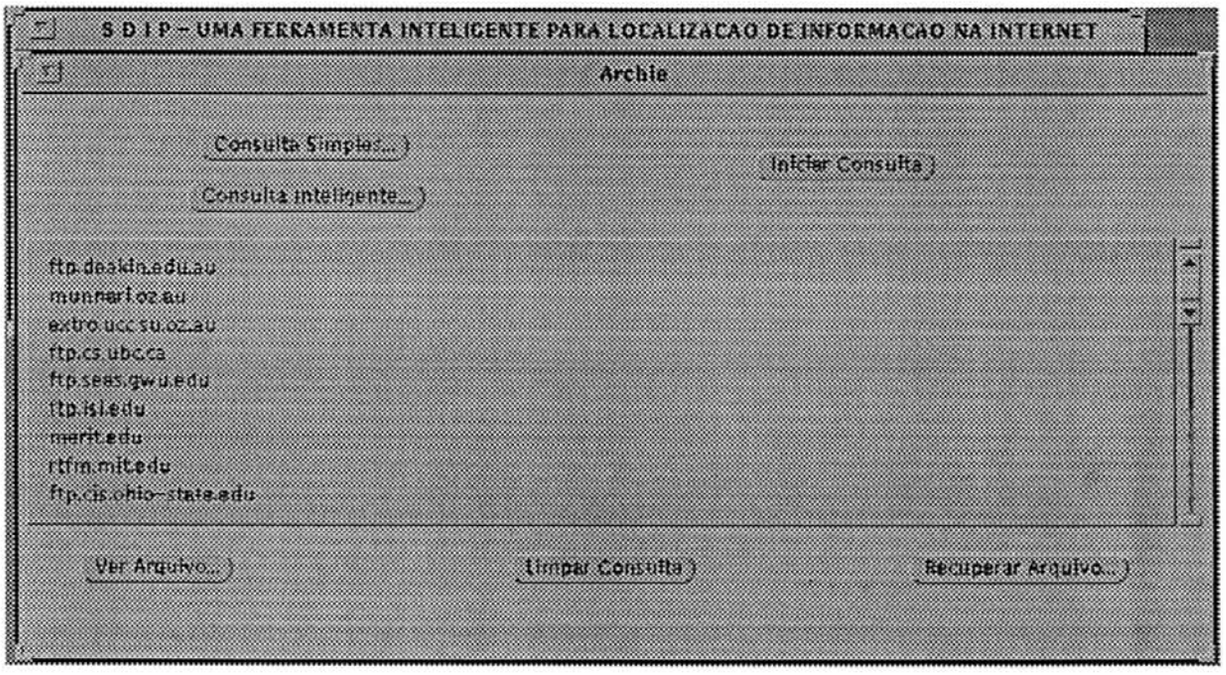

Figura 14.11 *Resultados obtidos com a formulação de uma consulta inteligente.*

É importante notar que há uma relação evidente entre a expressão especificada na consulta (*redes locais de alta velocidade*) e as respostas obtidas, como,

por exemplo, arquivos e diretórios cujo nome raiz é *FDDI*, os quais provavelmente devem conter informações relacionadas com as redes locais de alto desempenho de mesmo nome.

14.3.5 Janela de Acesso a Sistemas para Consultas Bibliográficas

Continuando o nosso estudo a respeito das facilidades gráficas providas pelo ambiente **SDIP**, vamos discutir nesta subseção o uso do subsistema de acesso a gerenciadores de informações bibliográficas.

Ao selecionar o botão *biblioteca*, o usuário visualizará a janela mostrada na figura 14.12. Por intermédio das estruturas contidas nesta janela secundária, os usuários podem realizar um conjunto de operações, desde a consulta a um sistema de gerência de dados bibliográficos, até o salvamento dos resultados obtidos em um arquivo local.

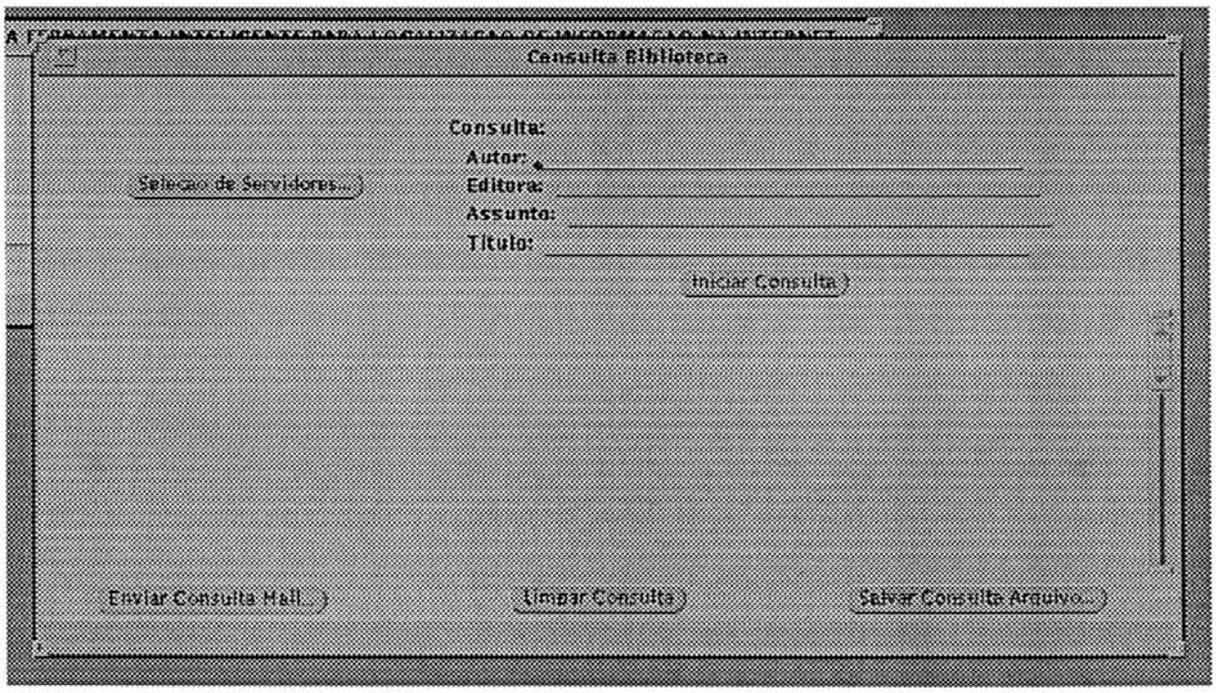

Figura 14.12 *Janela principal do módulo de acesso a sistemas de pesquisas bibliográficas.*

Primeiramente, vejamos como o usuário deve proceder para realizar uma pesquisa bibliográfica.

Inicialmente, ele deve fornecer ao sistema o máximo de informações possíveis a respeito das obras desejadas, objetivando diminuir a quantidade de dados irrelevantes recuperados. Para tanto, o usuário deve preencher os campos denominados: autor, título, assunto e editora, sempre que tais informações forem de seu conhecimento. Note-se que não é necessário o preenchimento de todos os campos, contudo quanto mais precisos e abundantes forem os dados, mais concisos serão os resultados obtidos.

Após ter preenchido pelo menos um dos campos acima referidos, o usuário deve selecionar os servidores bibliográficos para os quais ele deseja que seja encaminhada a sua consulta. Selecionando o botão *Seleção de Servidores*, o usuário terá acesso a uma lista contendo todos os servidores bibliográficos conhecidos pelo sistema, possibilitando-lhe optar por aqueles que melhor atenderem as suas necessidades.

Neste momento, o usuário pode informar ao sistema, selecionando o botão *Iniciar Consulta*, que a pesquisa pode ser iniciada.

As figuras 14.13, 14.14 e 14.15 mostram respectivamente o momento em que uma consulta é iniciada pelo usuário, e os resultados finais por ela produzidos.

Adicionalmente, a janela de acesso a sistemas de pesquisas bibliográficas também permite a seus usuários, quando for do interesse dos mesmos, salvar em arquivos ou enviar através do correio eletrônico, os resultados finais obtidos em uma consulta.

As figuras 14.16 e 14.17 ilustram como o usuário deve proceder para realizar essas operações, na ordem em que elas foram anteriormente referidas.

Finalmente, selecionando o botão *Limpar Consulta*, o usuário faz com que este subsistema retorne a seu estado inicial, ou seja, os campos com seus conteúdos indeterminados e o descarte dos resultados obtidos em uma consulta anterior.

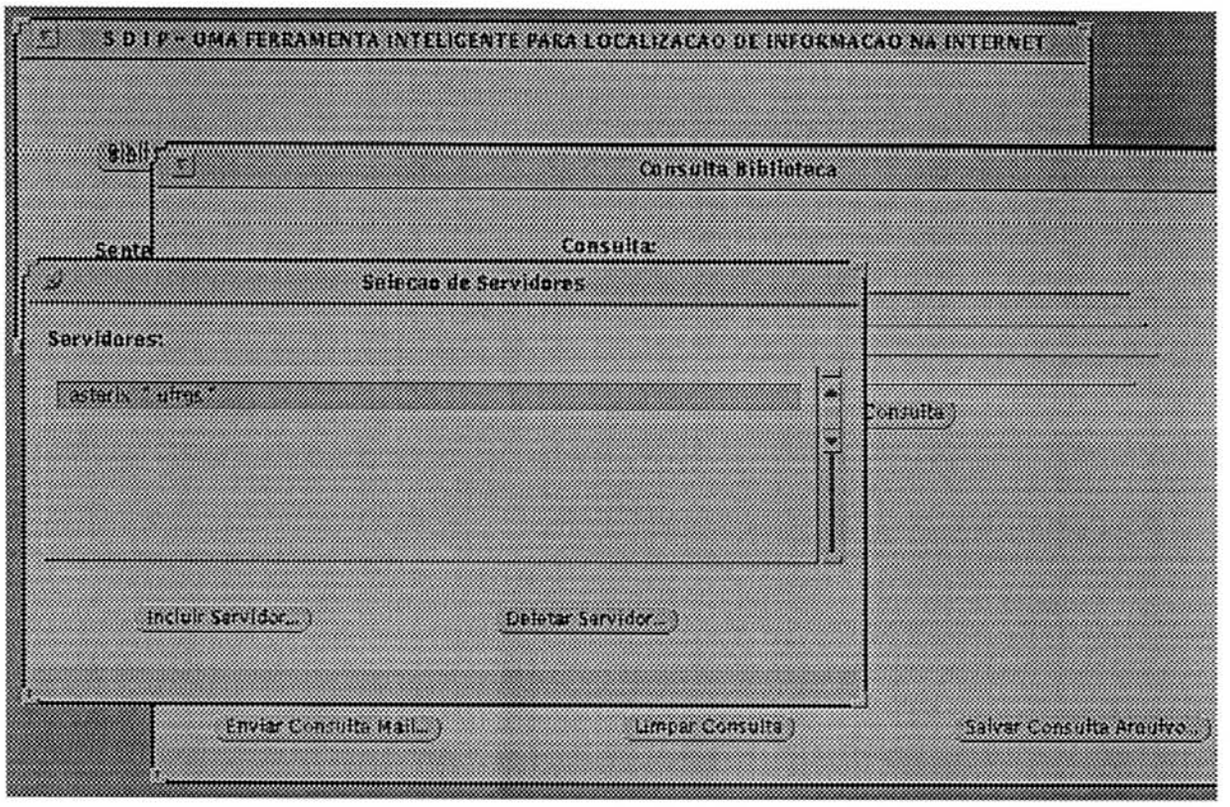

Figura 14.13 *Seleção dos servidores que serão consultados.*

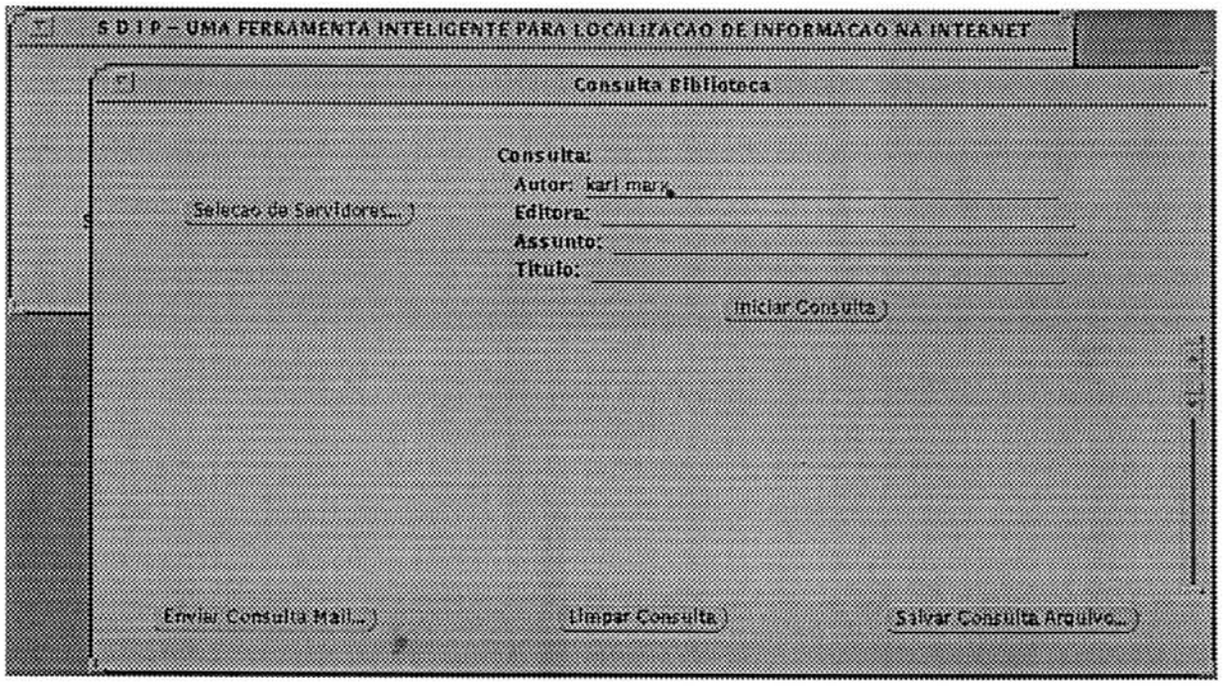

Figura 14.14 *Início da consulta bibliográfica, tendo como base o nome de um autor*

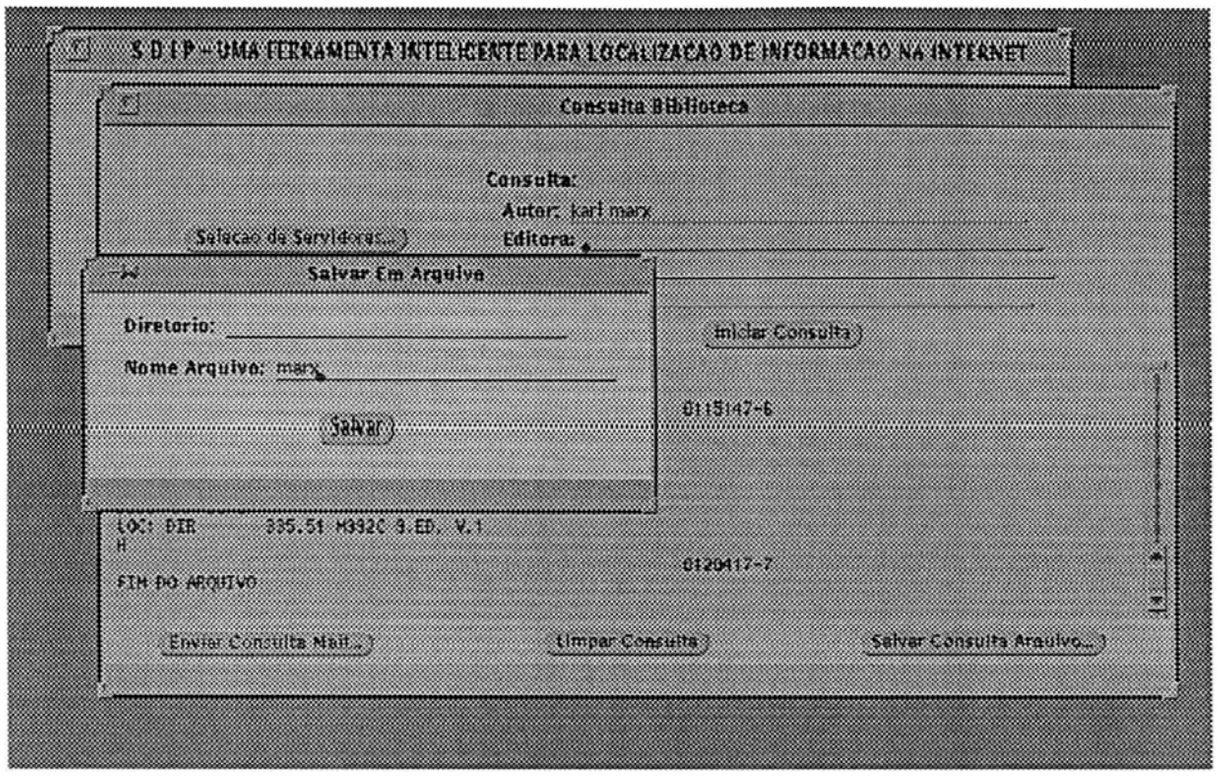

Figura 14.15 *Resultados obtidos a partir da consulta bibliográfica.*

A janela que nos permite interagir com sistemas de gerenciamento de dados bibliográficos, possui ainda duas estruturas importantes para a configuração do **SDIP**.

A primeira delas, a janela *Inclui Servidor*, deve ser utilizada pelos usuários sempre que estes desejarem ampliar a quantidade de sistemas deste tipo, conhecidos pelo **SDIP**. Neste caso, os usuários devem especificar o endereço Internet do computador onde reside o servidor, seu tipo, uma mensagem que o identifique para os demais usuários do sistema e, finalmente, a ("username") que permite o acesso público aos serviços de recuperação de dados bibliográficos.

A figura 14.18 ilustra o uso da facilidade descrita acima.

Por outro lado, o botão *Deletar Servidor* realiza a operação inversa, ou seja, modifica a configuração do sistema, excluindo os servidores de informações bibliográficas selecionados pelo usuário.

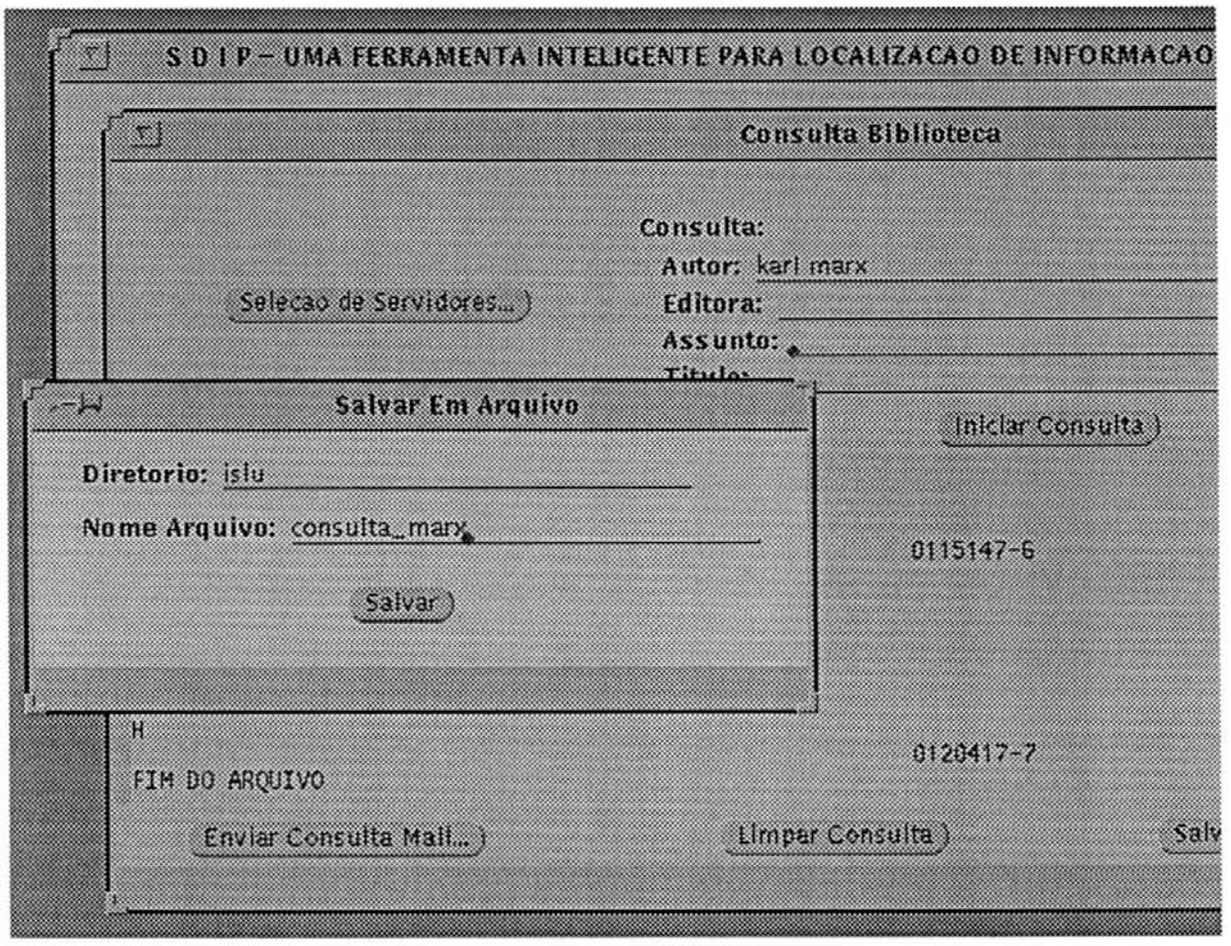

Figura 14.16 *Salvamento dos resultados de uma consulta em arquivo local.*

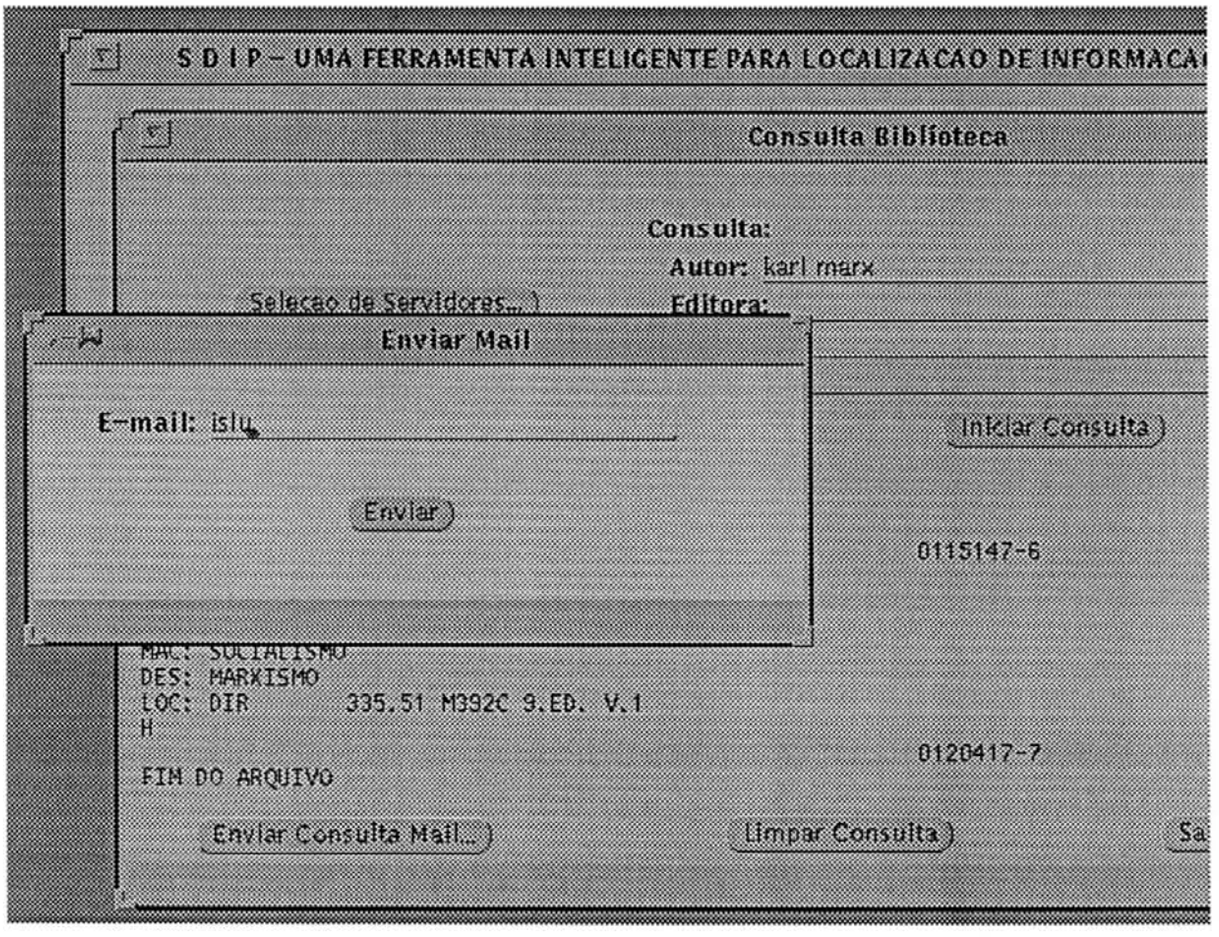

Figura 14.17 *Envio dos resultados obtidos em uma consulta, através do correio eletrônico.*

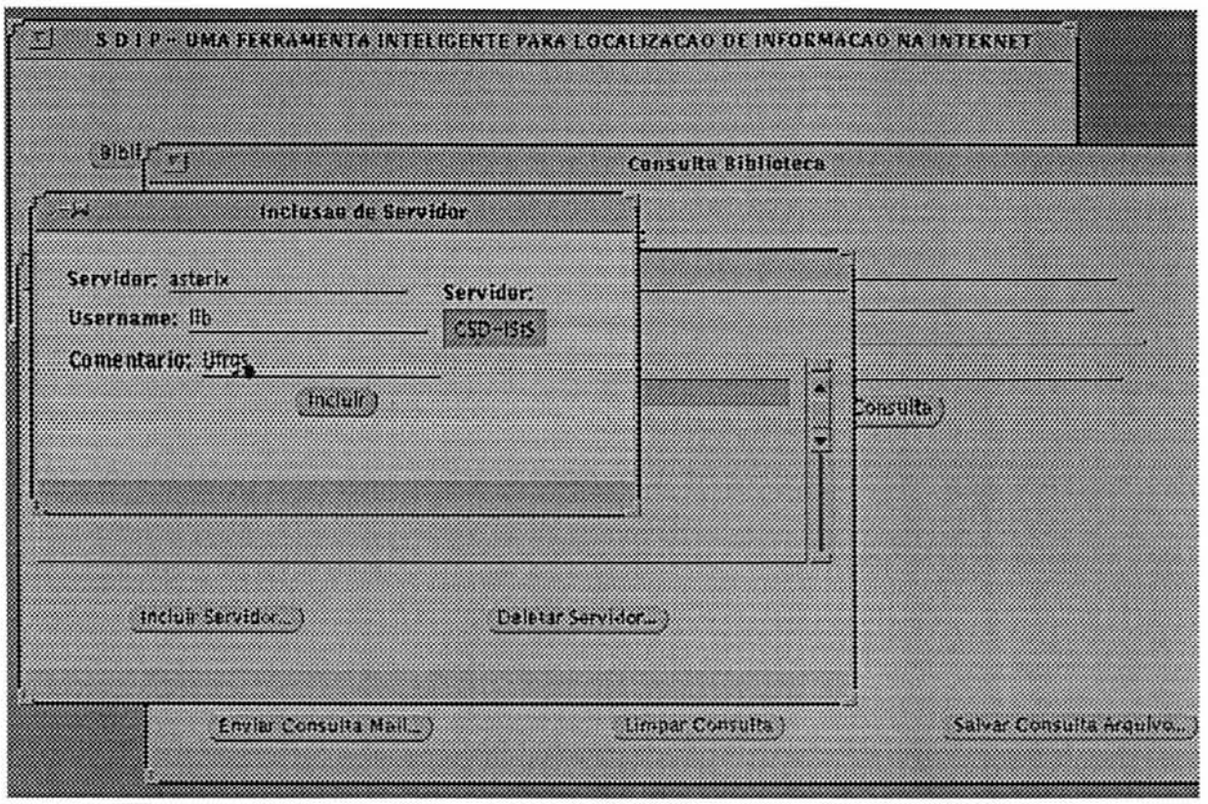

Figura 14.18 *Exemplo de inclusão de um sistema para consultas bibliográficas.*

14.3.6 Menu do Dicionário de Palavras

Outra estrutura importante que aparece na janela principal do ambiente gráfico do **SDIP** é o menu do dicionário de palavras do sistema. Abrindo esta estrutura, os usuários do ambiente **SDIP** verão um conjunto de opções, mostradas na figura 14.19. Em particular, destaque-se que a cada uma das classes de palavras reconhecidas pelo **SDIP** está associada uma entrada deste menu de opções.

Por intermédio deste subsistema, é possível incluir-se novas palavras e expressões no dicionário do **Processo Interpretador de Sentenças**, quando estas forem desconhecidas pelo mesmo.

Todavia, esta não se constitui na única maneira através da qual o **SDIP** pode expandir o seu vernáculo. Assim, uma outra alternativa será usada sempre que, ao interpretar uma sentença, o sistema encontrar uma palavra por ele desconhecida. Neste momento, inicia-se um processo de aprendizado semi-inteligente do

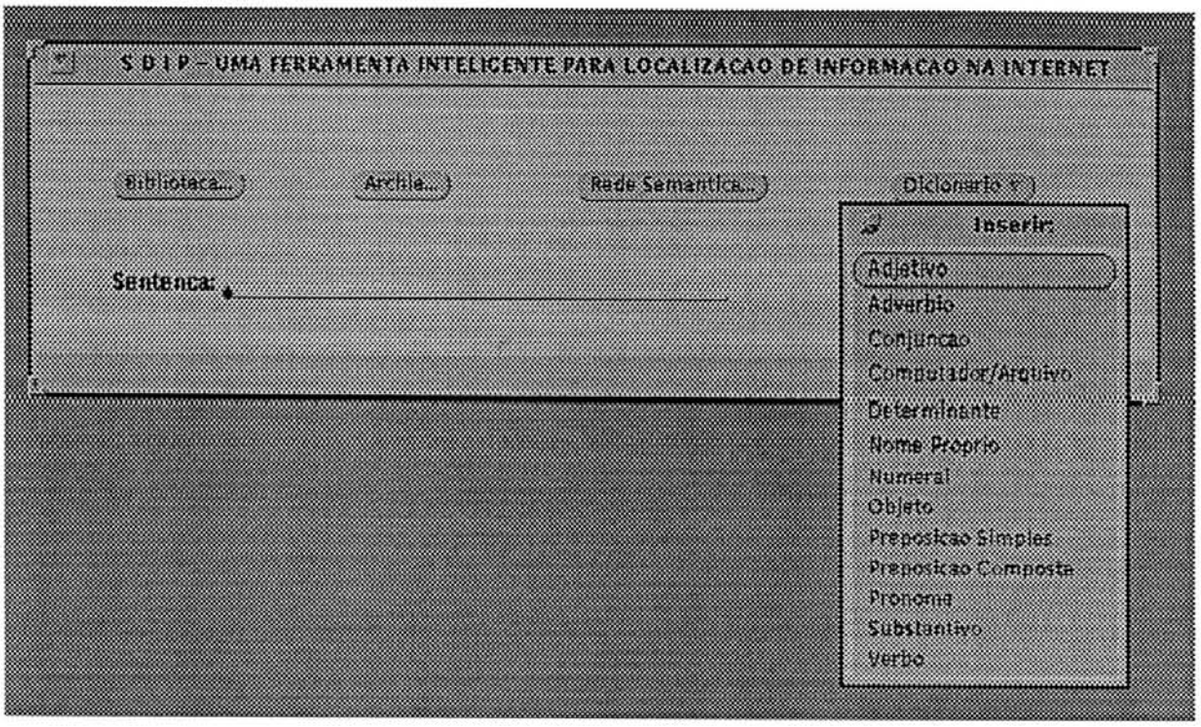

Figura 14.19 *Menu do dicionário de palavras do* **SDIP**.

novo termo, no qual o **SDIP** questiona o usuário, solicitando informações que lhe permite classificar e caracterizar corretamente o item léxico considerado.

Em razão de que existem muitas opções no menu do dicionário, ilustraremos a sua utilização, incluindo uma palavra de uma única classe gramatical. Posteriormente, o exemplo pode ser facilmente estendido para palavras de outras classes, variando apenas as informações sintáticas e semânticas associadas a cada uma delas, as quais devem ser fornecidas quando da inclusão. Assim, mostraremos como o usuário deve proceder para incluir um novo verbo no dicionário do sistema.

Em um primeiro momento, deve-se selecionar a opção do menu, mostrado na figura 14.19, associada à palavra da classe gramatical verbo. Neste momento, o sistema abrirá uma outra janela (figura 14.20), formada por um conjunto de campos, os quais serão usados para a caracterização da palavra que será inserida no dicionário. Finalmente, como ilustra a figura 14.21, o usuário deve fornecer as informações gramaticais associadas ao termo, e indicar ao sistema, através do botão *Inserir*, quando a operação estiver concluída.

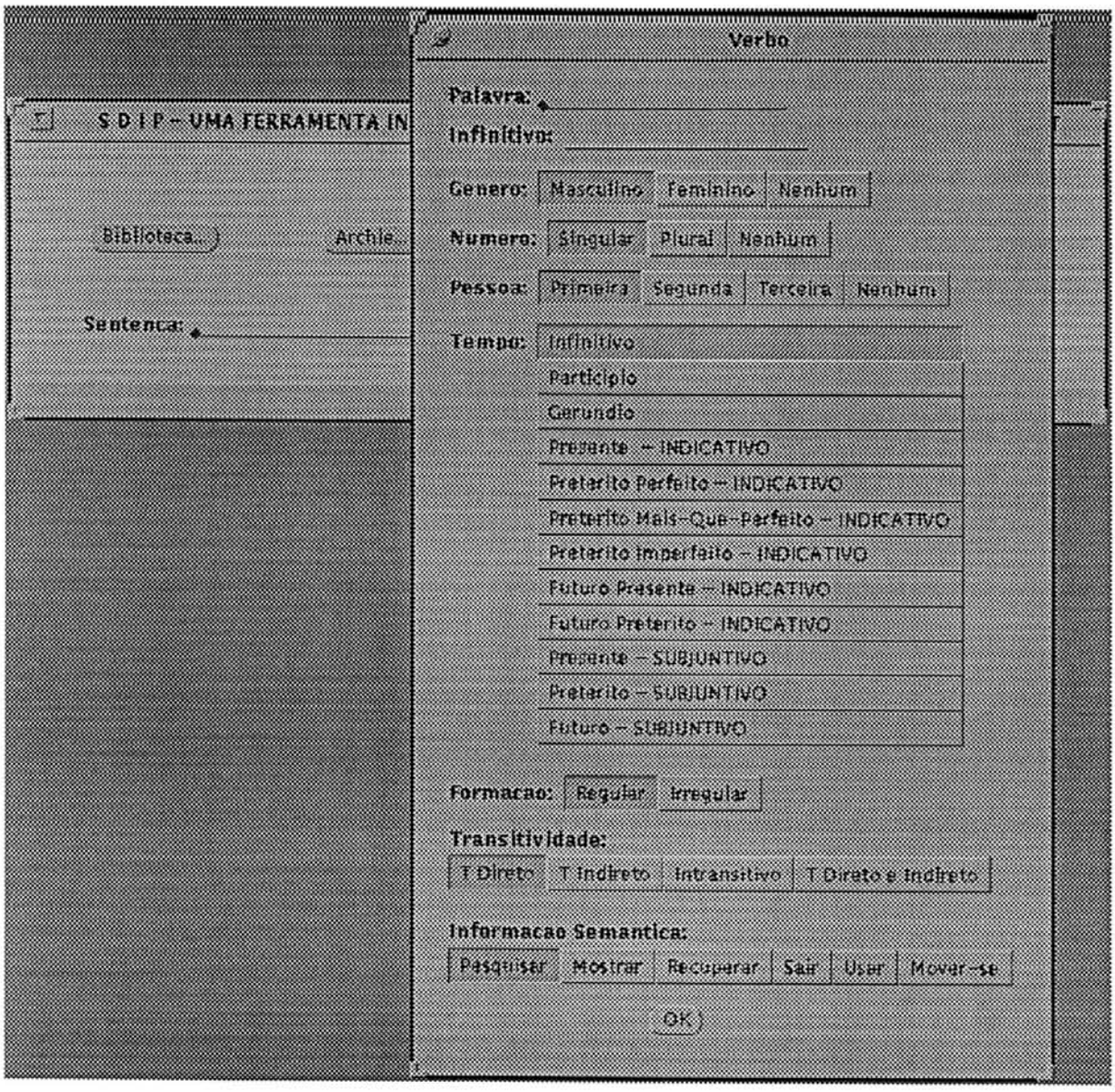

Figura 14.20 *Conjunto de campos que caracterizam uma palavra da classe gramatical verbo.*

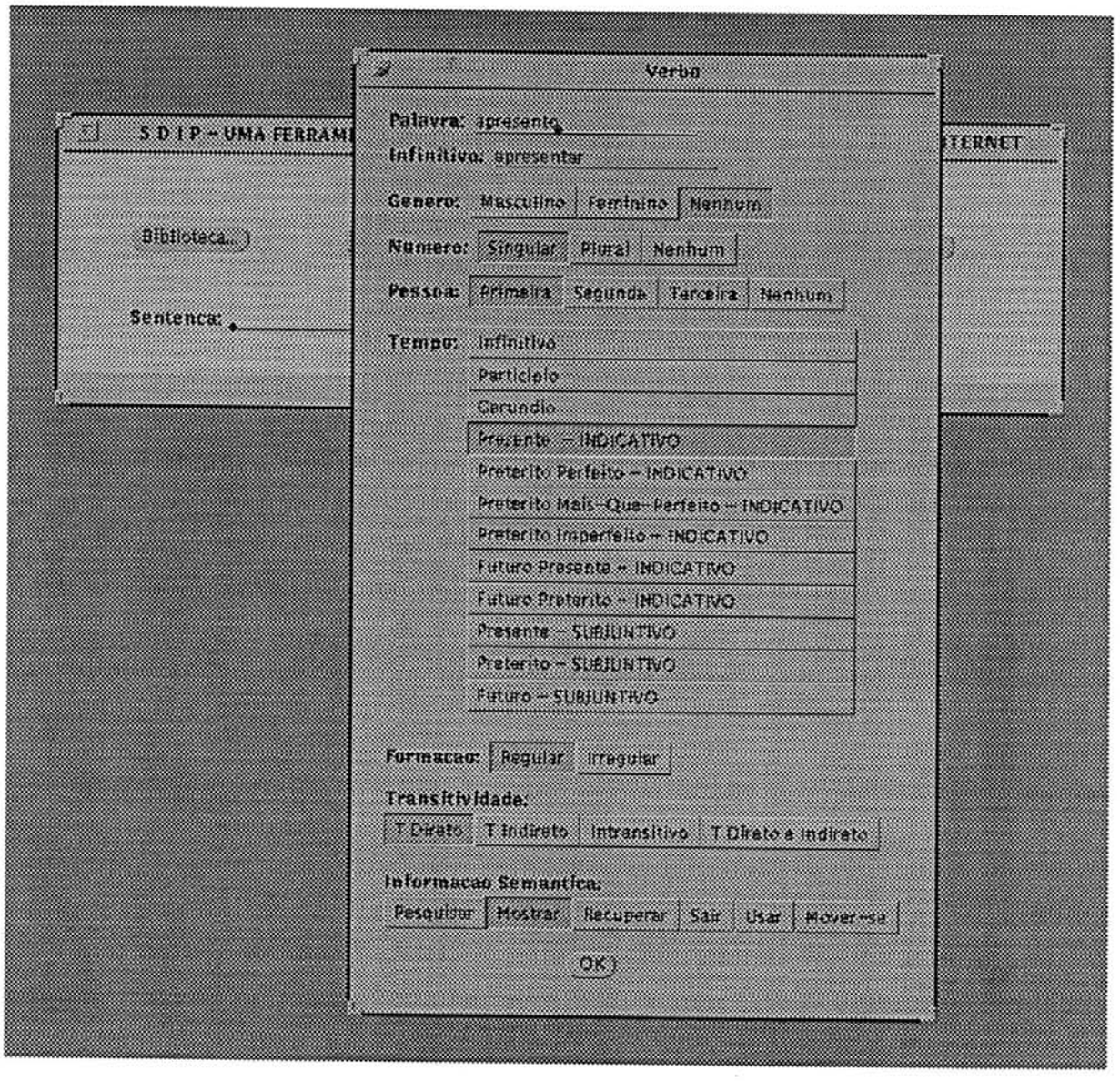

Figura 14.21 *Especificação dos conteúdos dos campos que descrevem um verbo.*

14.3.7 A Interface em Língua Natural

A grande inovação que distingue o ambiente **SDIP** dos demais sistemas hoje existentes, é a presença de uma interface em língua natural, a qual permite a seus usuários utilizarem-se, quase que integralmente, das facilidades implementadas pelo sistema, única e exclusivamente através de sentenças da língua portuguesa. Objetivamente, somente as facilidades relativas à manipulação da base de conhecimento e dicionário do sistema não podem ser acessadas, usando-se sentenças da língua portuguesa.

Para utilizar este tipo de interface, o usuário deve escrever a sua sentença no campo *Sentença* e posteriormente pressionar a tecla enter, no momento em que a formulação estiver concluída, como ilustra a figura 14.22.

Seria impossível mostrarmos neste trabalho exemplos de todas as formas de sentenças que os usuários poderiam utilizar-se para interagir com o **SDIP**. No entanto, a título de ilustração, mostraremos dois exemplos completos, cada um deles associado a um sistema de pesquisa de informações implementado no **SDIP**.

Na figura 14.22, mostramos um exemplo de uma sentença que induziria o sistema a efetuar uma consulta ao servidor SABI.

Na figura 14.23, mostramos os resultados obtidos a partir da consulta formulada.

Consideramos importante observar que a figura 14.23 é bastante semelhante àquela em que é realizada a consulta ao sistema SABI através da janela *biblioteca*, exceto pela presença da sentença em língua natural no campo *Sentença*. Dessa maneira, caso desejássemos, poderíamos, a partir deste momento, interagir diretamente com o sistema, utilizando-nos da própria janela *biblioteca*.

Na figura 14.24, solicitamos ao sistema que envie os resultados obtidos na consulta ao usuário **islu@inf.ufrgs.br**.

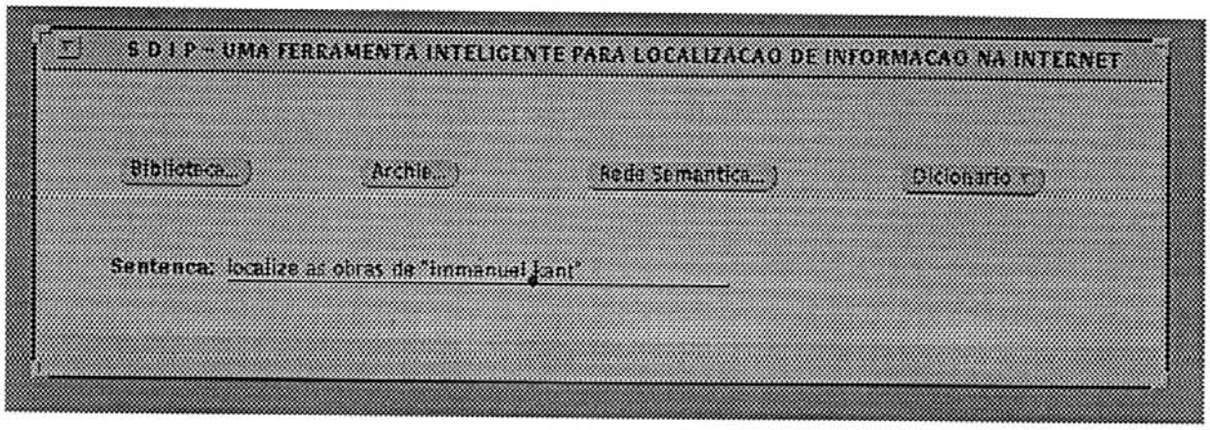

Figura 14.22 *Consulta ao sistema SABI, usando sentenças em língua natural.*

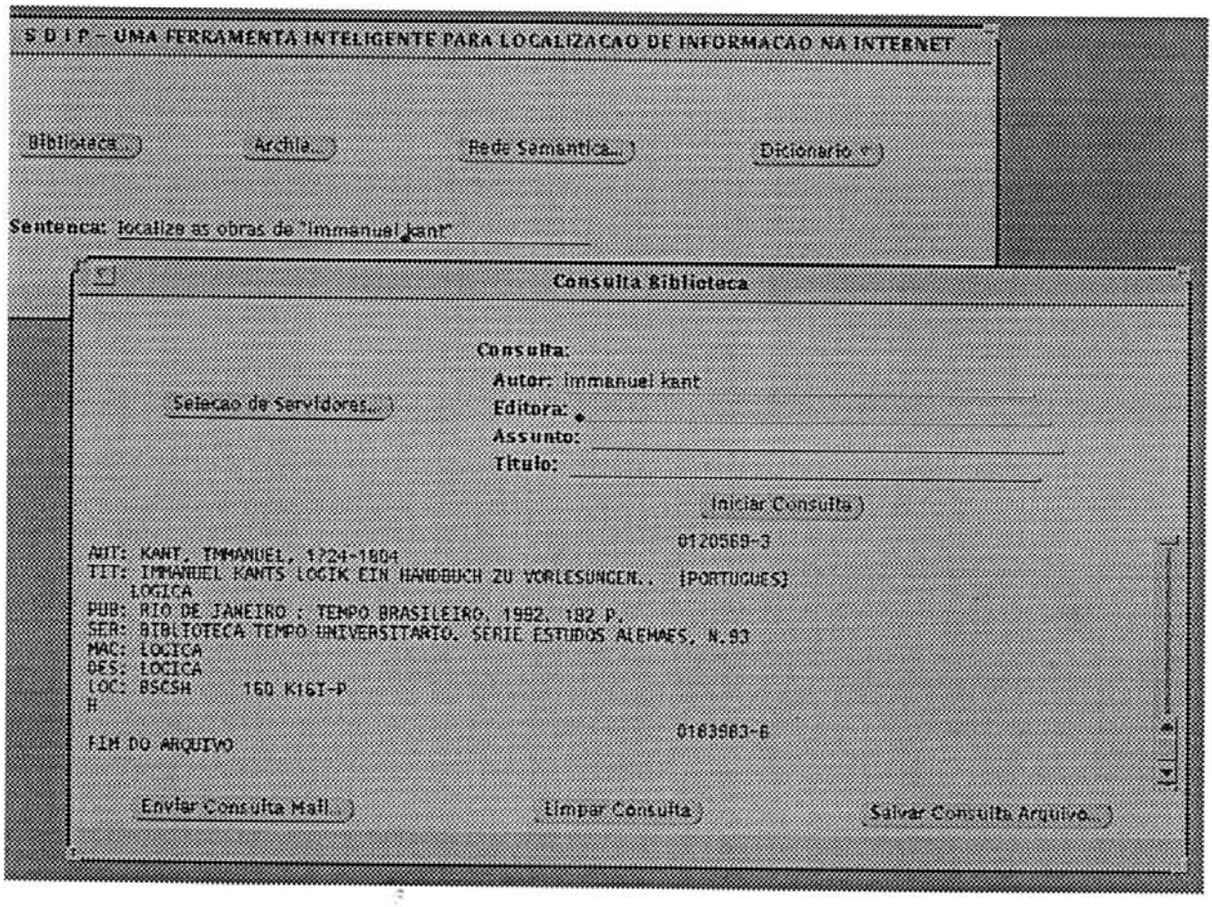

Figura 14.23 *Resultados obtidos pela consulta mostrada na figura anterior.*

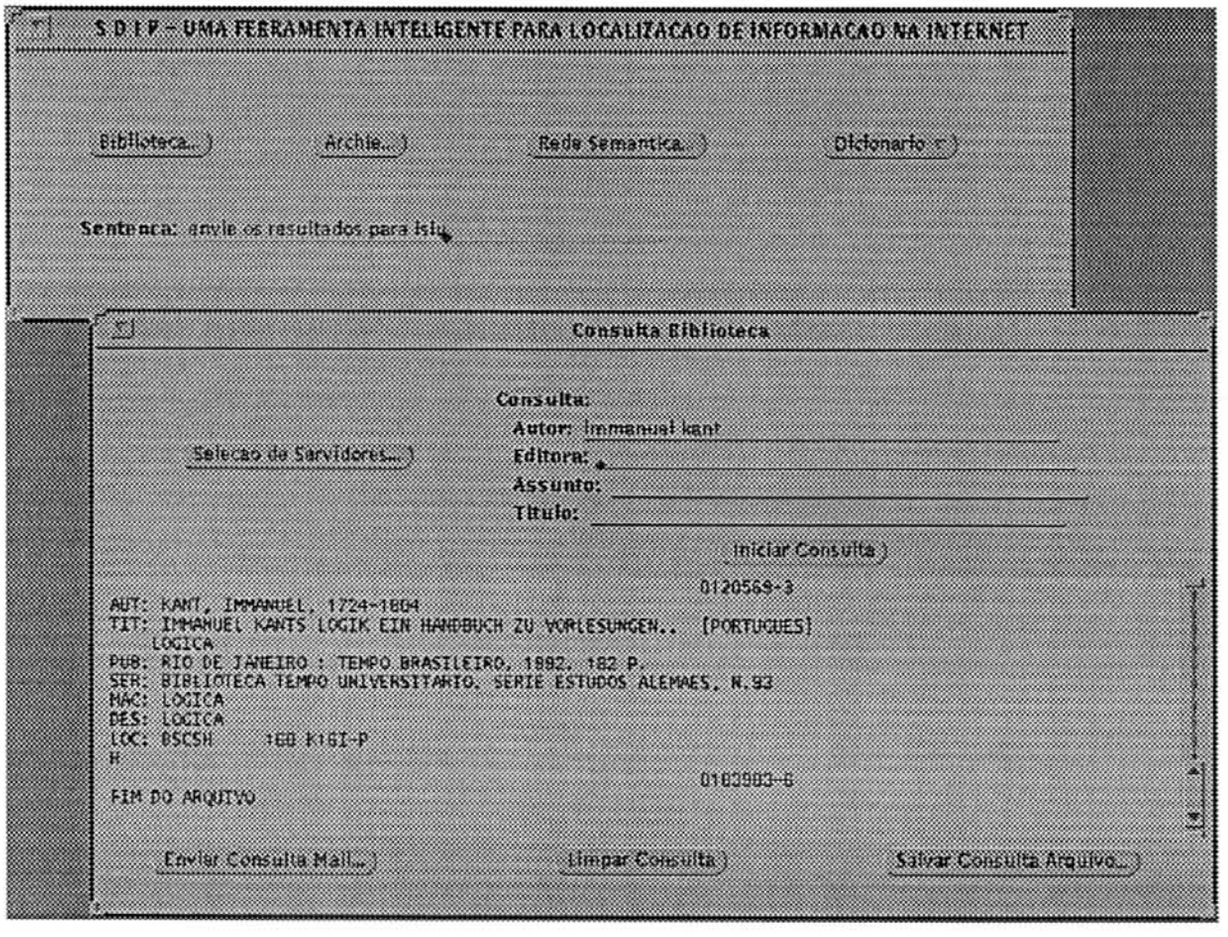

Figura 14.24 *Uso de língua natural para solicitar ao sistema que envie os resultados obtidos através do correio eletrônico.*

Finalmente, na figura 14.25, temos um exemplo de uso de sentença em língua natural na utilização do sistema Archie.

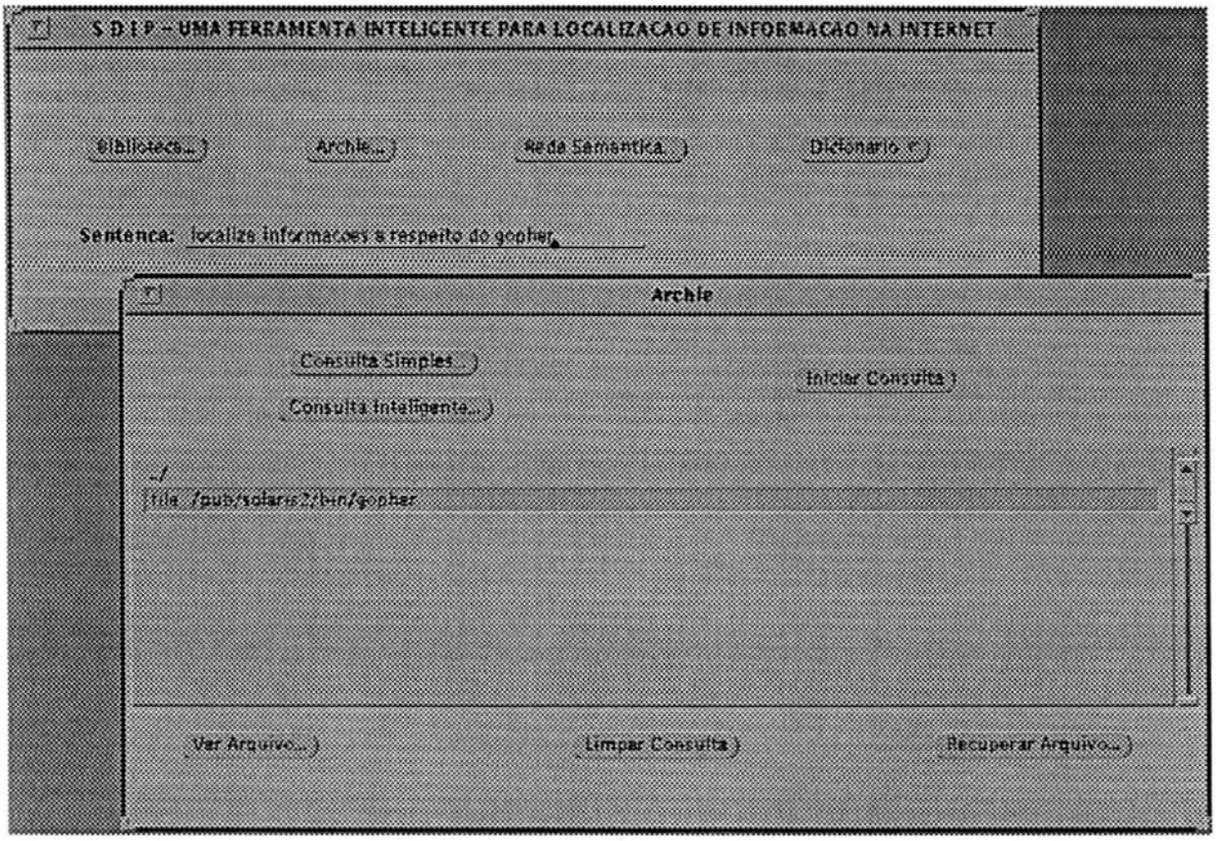

Figura 14.25 *Consulta ao sistema Archie, usando-se língua natural.*

Na figura 14.26, mostramos um exemplo de sentença que nos permite recuperar o arquivo selecionado, além de ilustrar a configuração da interface gráfica visualizada pelo usuário naquele momento.

14.3.8 Modificando a Base de Conhecimento do Sistema

Como vimos no capítulo anterior 13, o **SDIP**, mais especificamente o **Processo Interpretador de Sentenças**, mantém uma base de conhecimento que serve de suporte à interpretação das consultas formuladas pelos usuários, permitindo ao sistema obter informações mais adequadas às suas necessidades. Neste particular, uma importante característica inerente ao ambiente **SDIP** refere-se ao fato de que o conhecimento mantido pelo sistema não é estático ou permanente, mas variável,

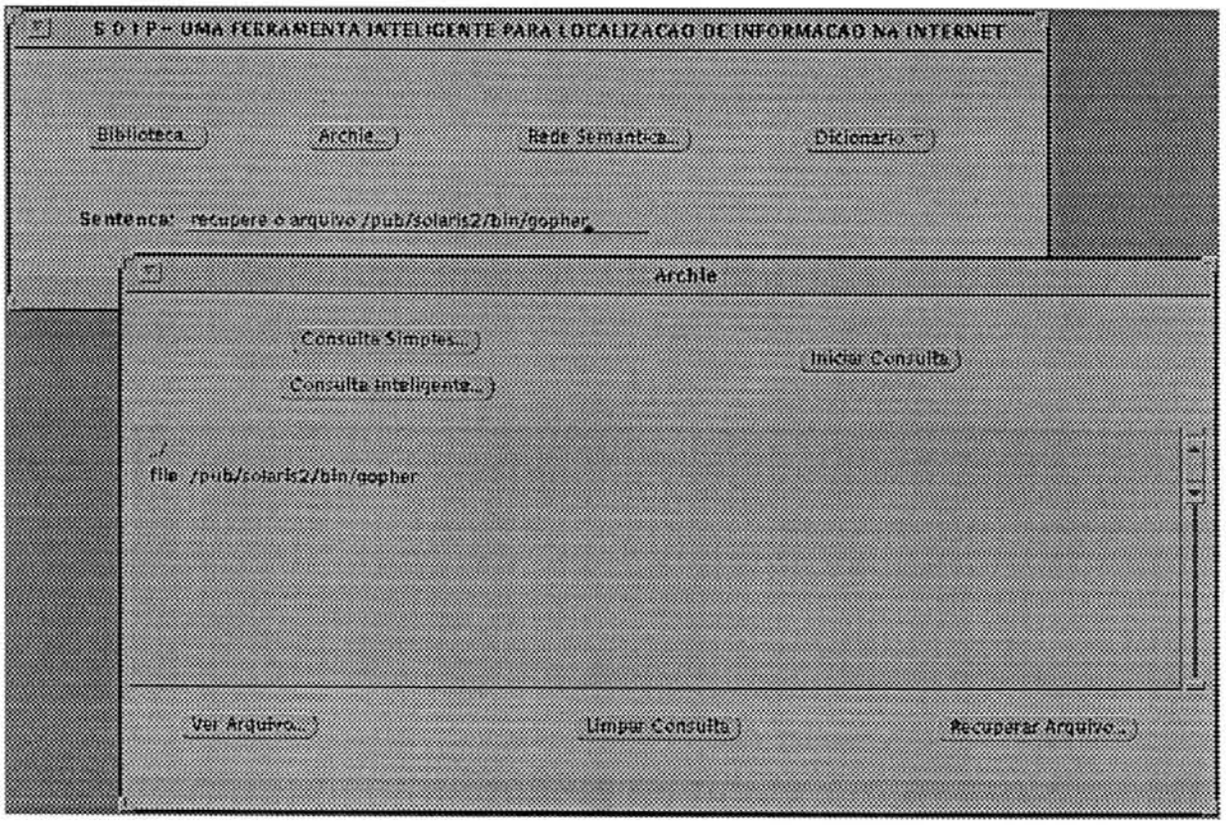

Figura 14.26 *Exemplo de uma sentença usada para recuperar um arquivo remoto.*

ou seja, ele pode ser modificado a qualquer momento, havendo a possibilidade de expandi-lo ou restringi-lo, conforme seja necessário. Desta forma, por exemplo, os usuários podem paulatinamente incrementar a base de conhecimento do sistema, tornando-o cada vez mais ajustado às suas necessidades.

Selecionando o botão *Rede Semântica*, o usuário terá acesso ao módulo que permite a realização de operações sobre a base de conhecimento do sistema.

Na figura 14.27, mostramos a tela principal visualizada pelo usuário ao selecionar o botão *Rede Semântica*.

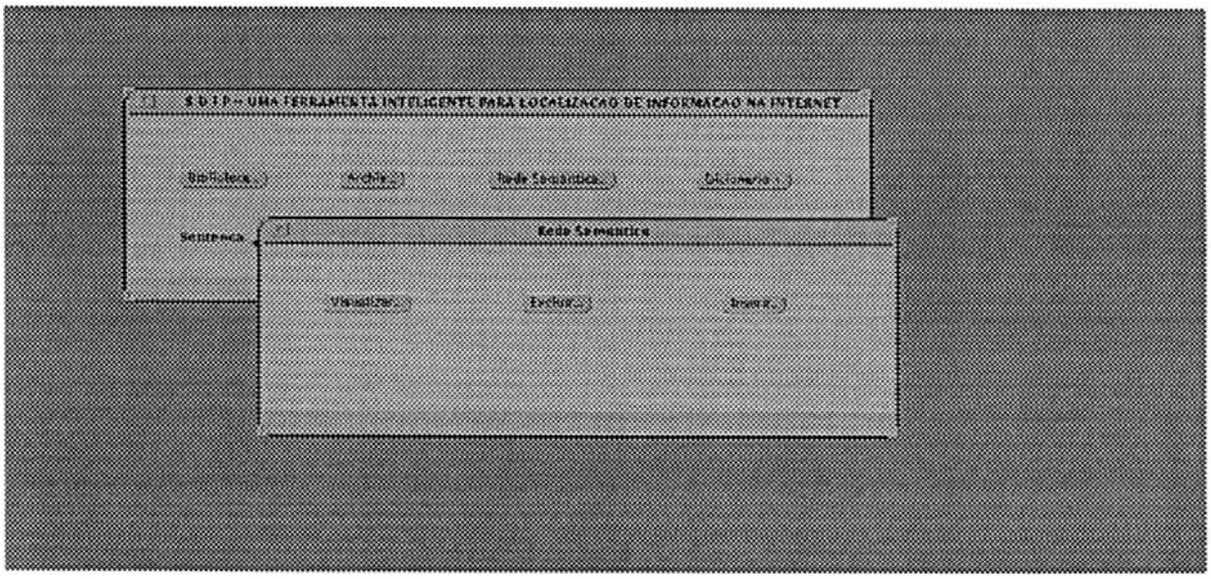

Figura 14.27 *Tela principal associada ao botão* Rede Semântica.

Nesta tela inicial, existe uma série de três operações possíveis de serem realizadas, a saber: visualização, exclusão e inserção de entradas na base de conhecimento do sistema.

O botão *Visualizar*, mostrará ao usuário o conteúdo atual da base de conhecimento mantida pelo sistema.

A figura 14.28 ilustra o uso desta opção.

O botão *Excluir*, permite aos usuários excluir entradas presentes na base de conhecimento. Ao selecionar este botão, o sistema mostrará ao usuário uma tela

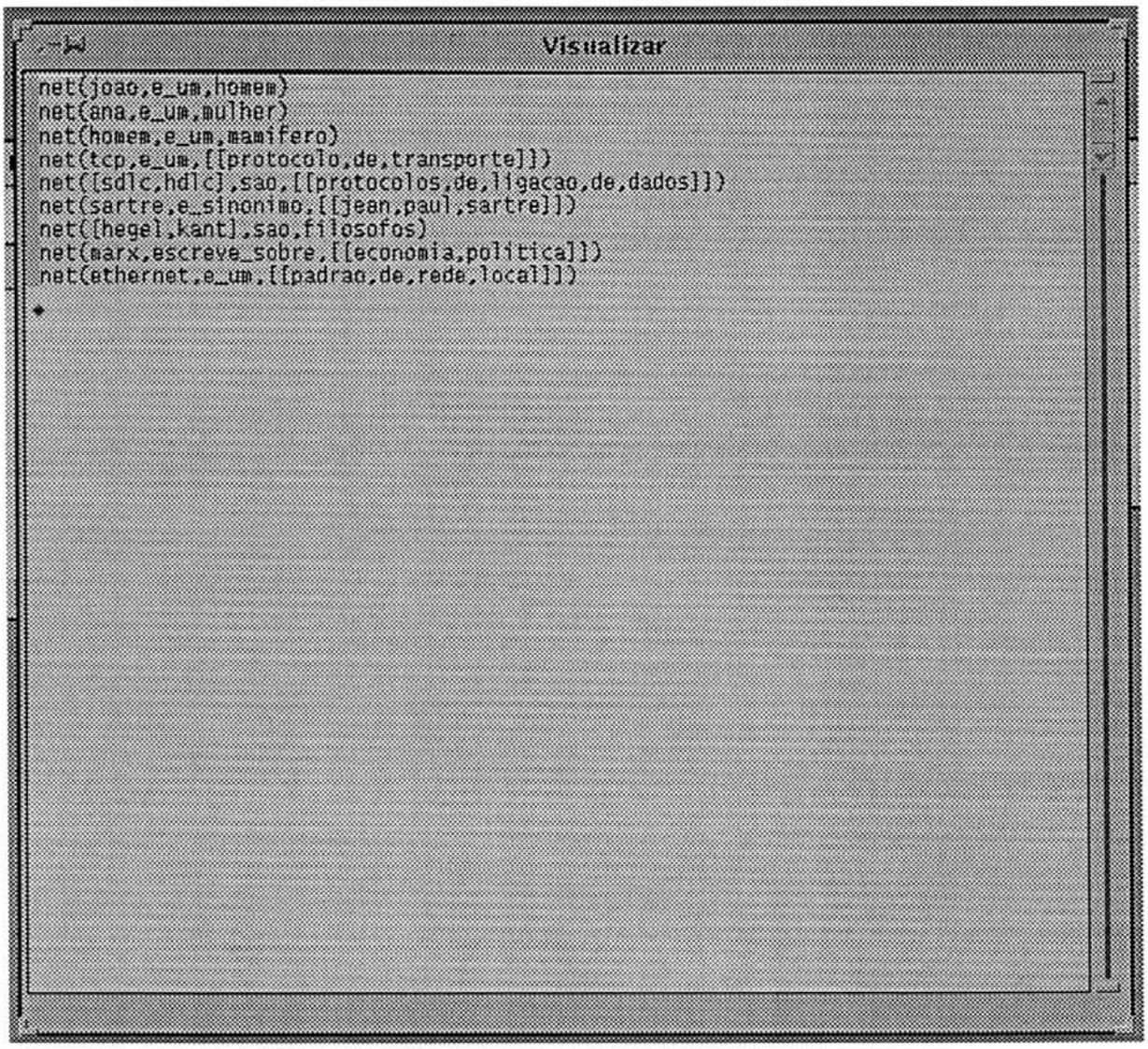

Figura 14.28 *Janela que permite a visualização da base de conhecimento do sistema.*

similar à ilustrada pela figura 14.29, contendo uma lista das entradas da base de conhecimento. Após ter selecionado alguns ítens desta lista, o usuário pode excluí-los definitivamente, usando para esse fim o botão *OK*.

A figura 14.29 ilustra a realização desta operação.

Na figura 14.30, mostramos a base de conhecimento resultante, após o término normal da operação de exclusão.

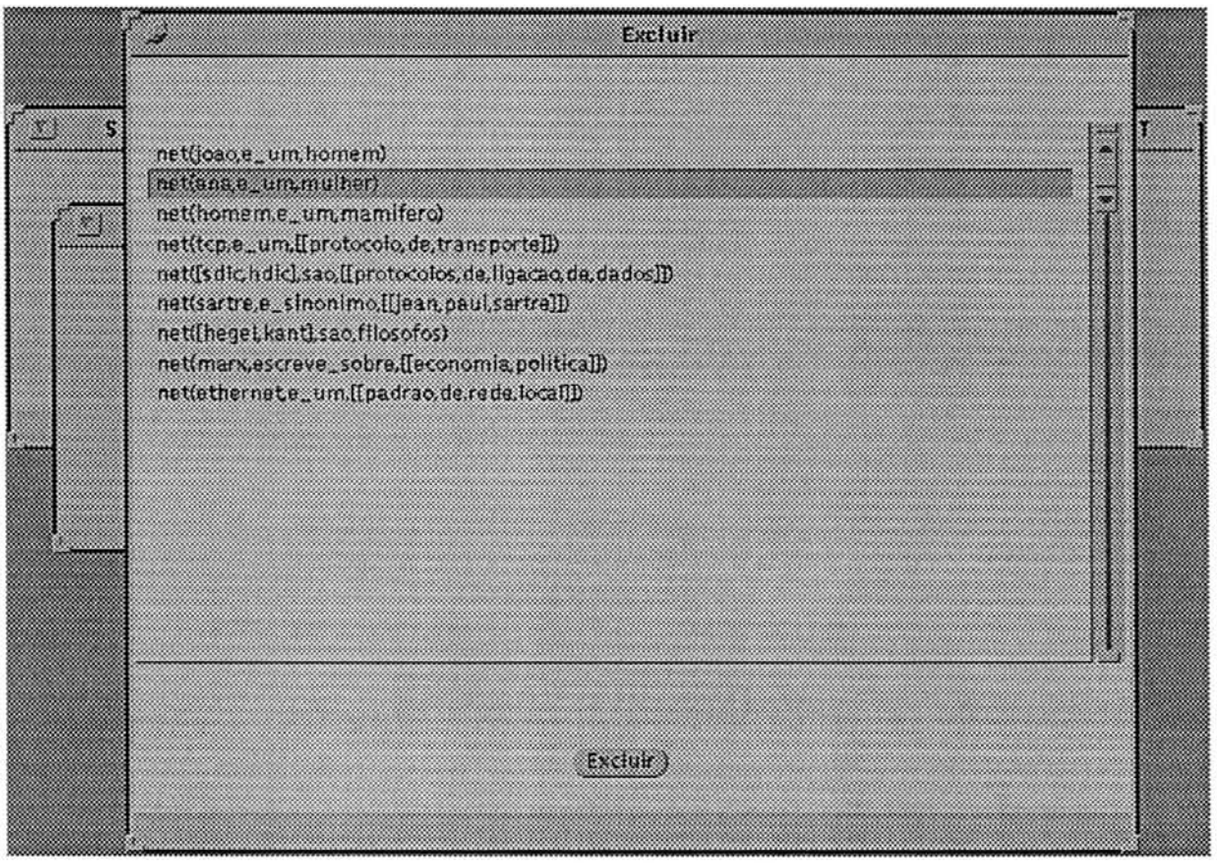

Figura 14.29 *Exclusão de uma entrada da base de conhecimento do sistema.*

Finalmente, vejamos como os usuários devem proceder para incluir uma ou mais entradas na base de conhecimento do sistema, após terem selecionado o botão *Inserir*. Para tanto, suponhamos que se deseja incluir uma entrada que informe ao sistema o seguinte fato:

FDDI é uma rede local de alta velocidade

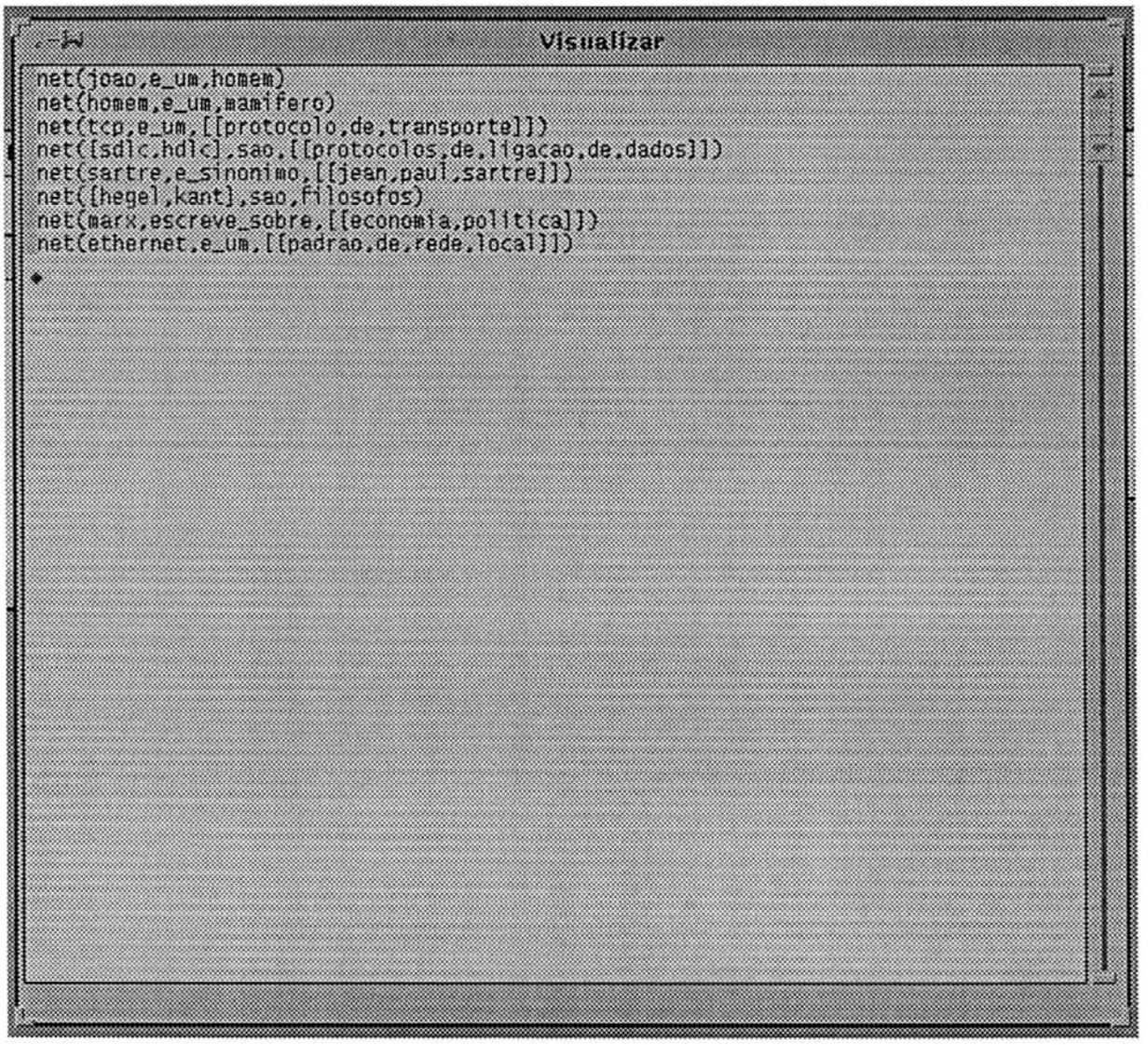

Figura 14.30 *Base de conhecimento resultante, após a exclusão.*

Suponhamos, adicionalmente, que a palavra **FDDI** e a expressão **rede local de alta velocidade**, já estão presentes no dicionário de palavras do sistema.

Na primeira fase da inserção de uma nova entrada na base de conhecimento, o usuário deve indicar ao sistema que palavras e/ou expressões do dicionário ele utilizará para a construção da entrada. Através do menu *seleção de palavras*, o usuário poderá realizar esta operação, sendo necessário apenas, escolher a opção *nome próprio* e selecionar a palavra *FDDI*. Posteriormente, ainda no menu *seleção de palavras*, o usuário deve escolher a opção *objeto* e, após, selecionar a expressão *rede local de alta velocidade*.

As figuras 14.31, 14.32, 14.33 e 14.34 ilustram as operações acima descritas.

Retornando à janela de *inclusão* (figura 14.35), observamos que a palavra e expressão anteriormente referidas, agora aparecem em uma lista de ítens que podem ser selecionados pelo usuário.

Prosseguindo com o nosso exemplo, agora ingressamos na última fase necessária à inclusão de uma nova entrada na base de conhecimento do sistema, qual seja, a construção efetiva da mesma. Para tanto, devemos selecionar as palavras ou expressões e a relação que, unidas, constituirão uma entrada da base de conhecimento do **SDIP**. Por conseguinte, o usuário deve efetuar as seguintes operações:

- Selecionar respectivamente a palavra *FDDI* e o botão *objeto_1*;
- Selecionar a opção *é_um*, que aparece no conjunto de ítens rotulados como *relações*;
- Selecionar a expressão *rede local de alta velocidade* e o botão *objeto _2*;
- Selecionar o botão *INCLUI*, indicando ao sistema que foi concluída a construção da entrada;

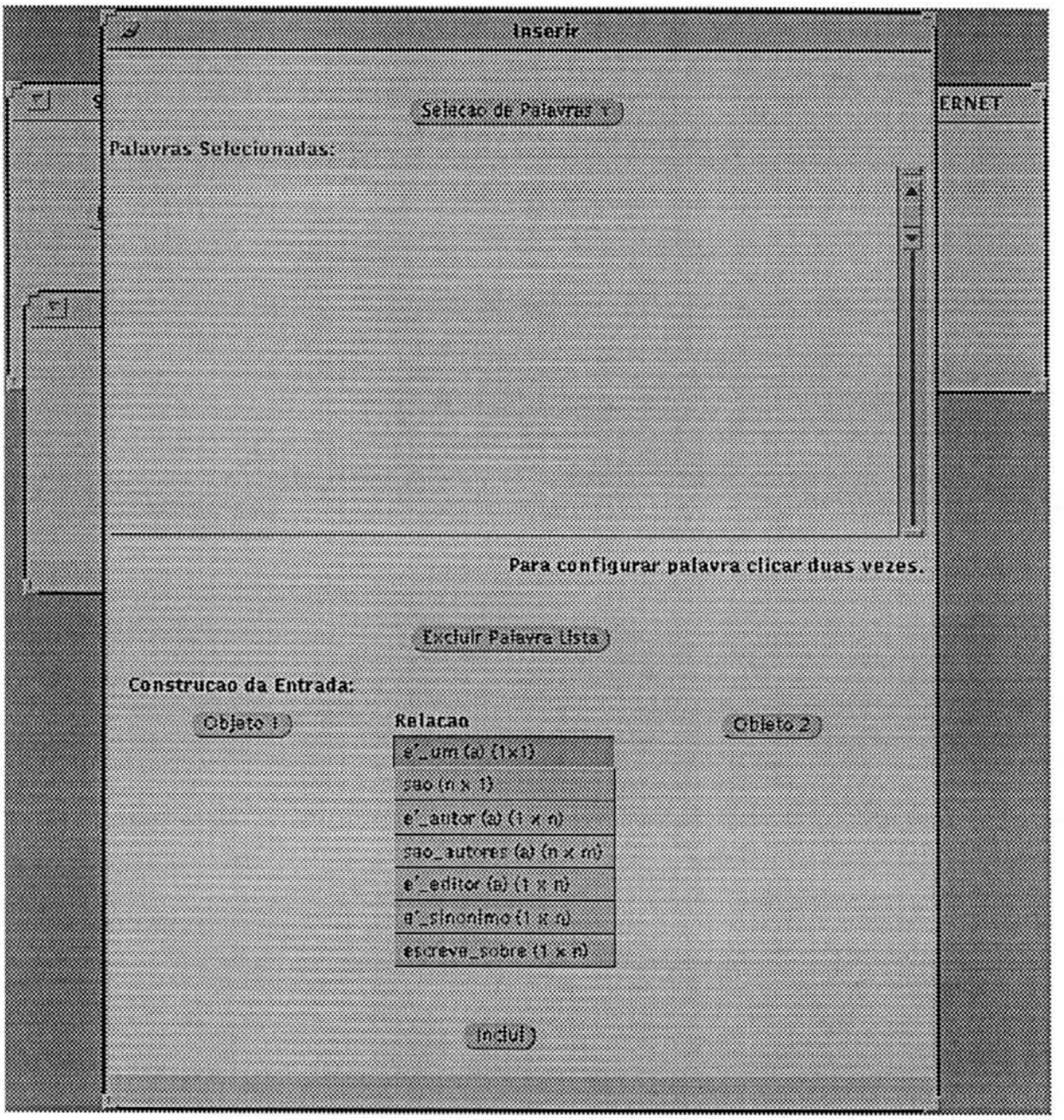

Figura 14.31 *Janela que permite a inclusão de novas entradas na base de conhecimento do sistema.*

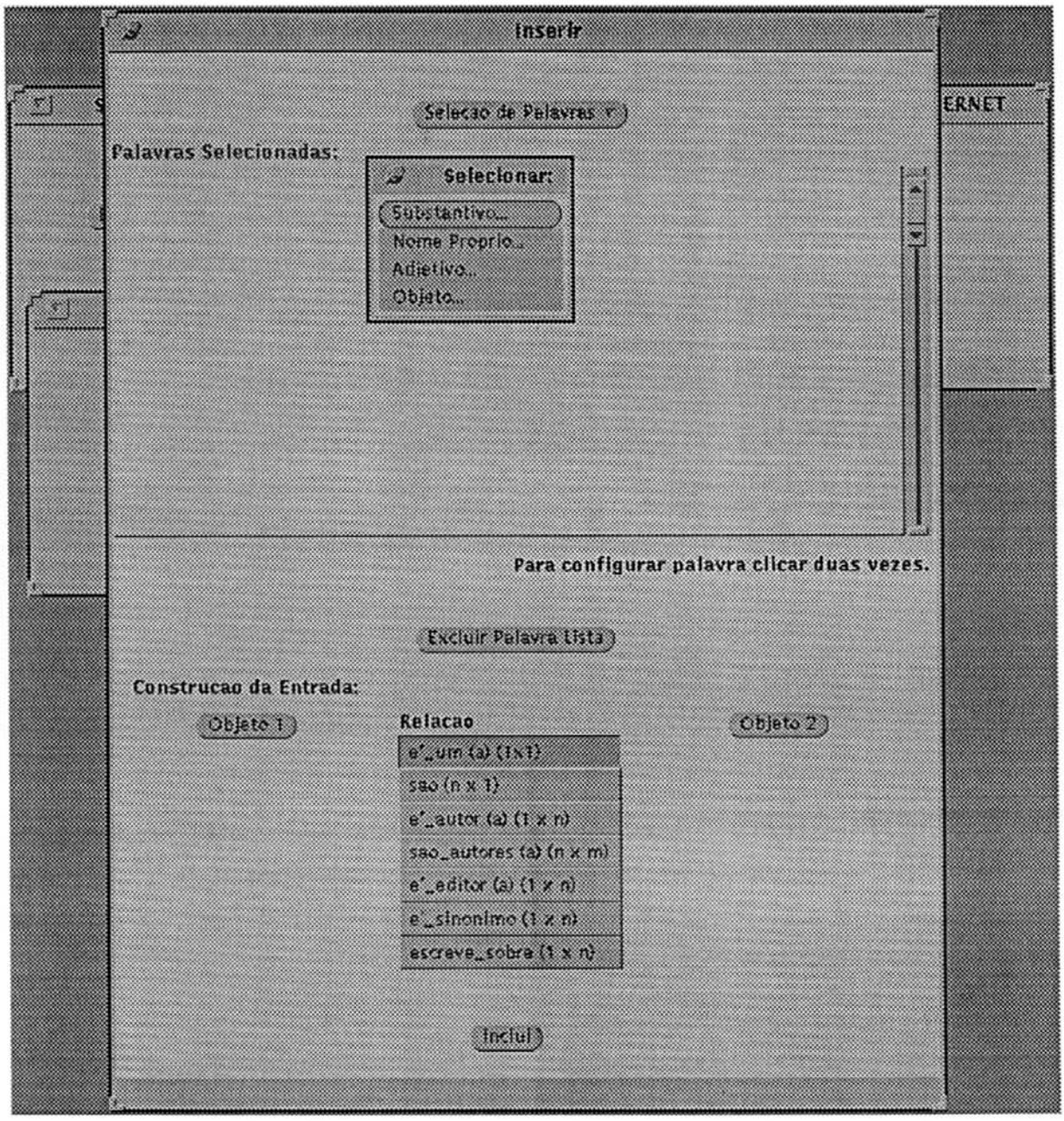

Figura 14.32 *Menu que permite a seleção de palavras que formarão a entrada da base de conhecimento.*

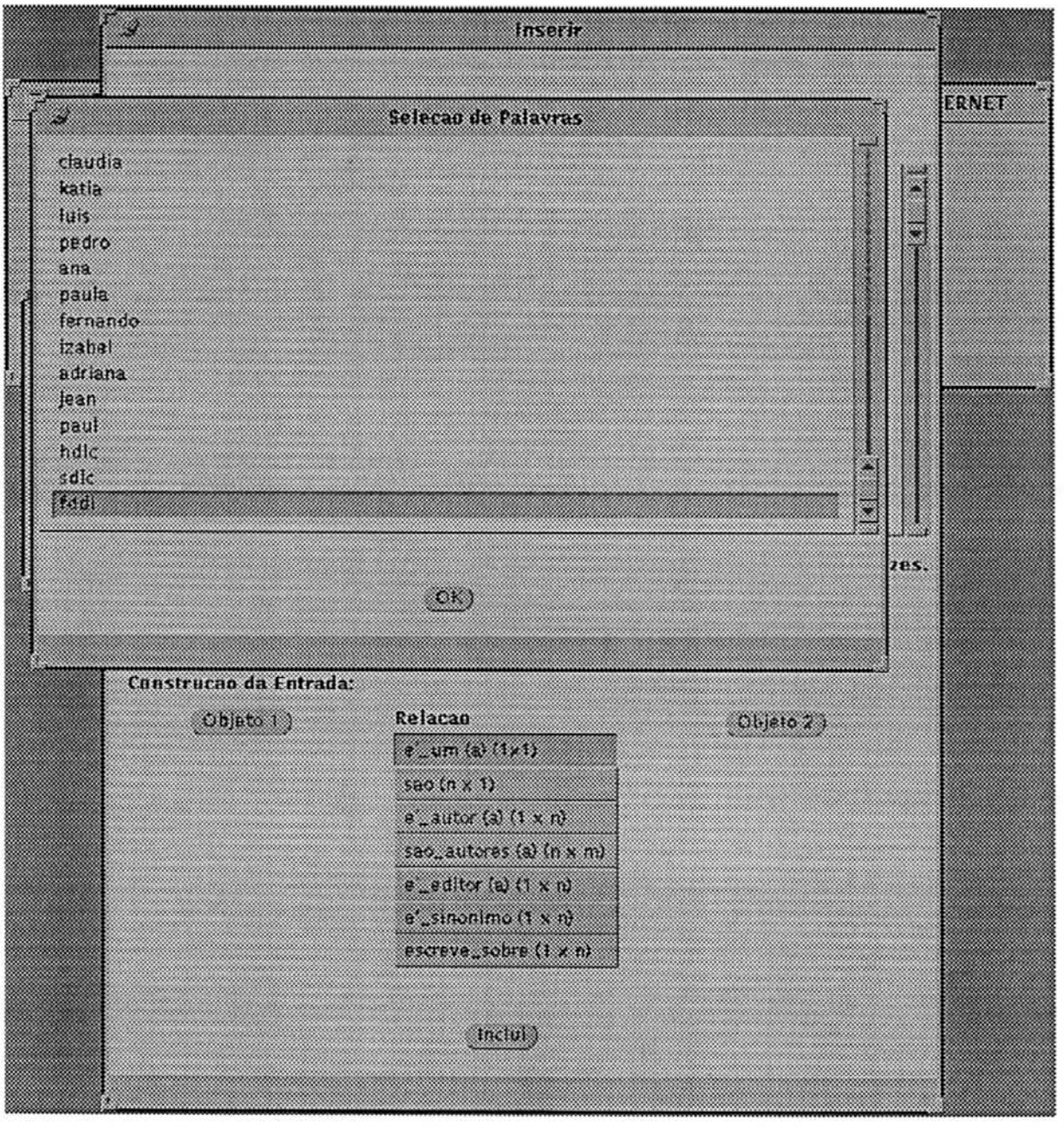

Figura 14.33 *Seleção da palavra FDDI.*

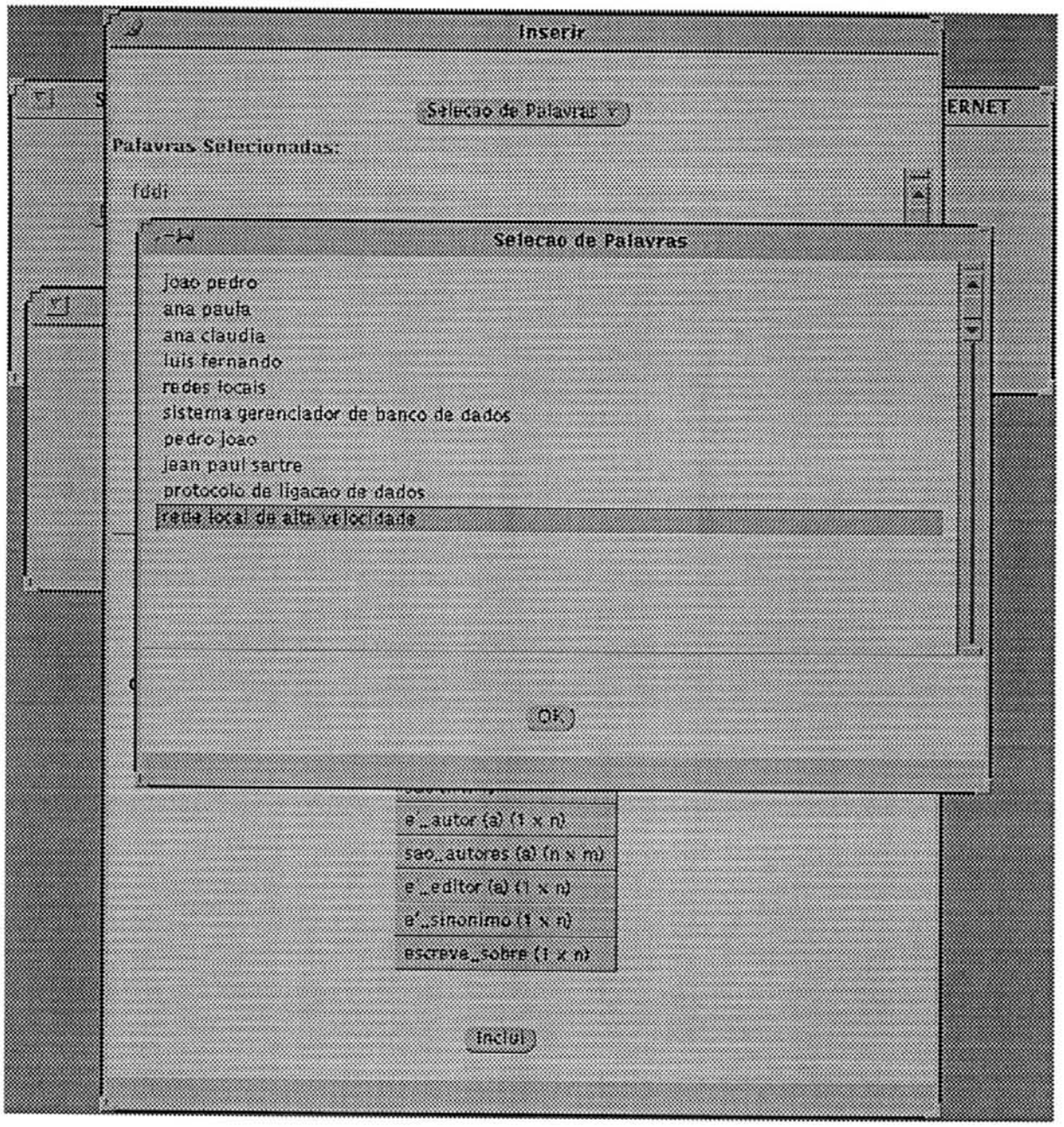

Figura 14.34 *Seleção da expressão* rede local de alta velocidade.

- Abandonar a janela de inclusão, selecionando o ícone apropriado, informando ao sistema que foi concluída a fase de inserção de novas entradas na base de conhecimento.

Neste momento, esta nova informação foi incorporada à base de conhecimento do **SDIP**, podendo ser utilizada pelo sistema no momento em que ele tiver de interpretar uma sentença em língua natural.

A figura 14.35 ilustra os últimos passos necessários à realização da operação descrita nesta subseção.

A figura 14.36 mostra-nos que de fato foi criada uma nova entrada na base de conhecimento no ambiente **SDIP**.

Ainda na janela de inclusão, podemos definir, para cada um dos objetos que formam as entradas inseridas na base de conhecimento, os servidores Archie com seus parâmetros de operação e as chaves de busca que serão utilizadas no momento em que estes termos forem o núcleo de uma consulta que deve ser encaminhada a esses servidores de informação da Internet. Por conseguinte, no momento em que uma consulta referir-se a algum destes termos, o **SDIP** utilizará as informações a ele associadas, objetivando fornecer as respostas esperadas pelos usuários.

Para definir este conjunto de informações, o usuário deve pressionar duas vezes consecutivas o botão *Select*, com o cursor do mouse sobre o termo considerado, por exemplo, *FDDI*.

A figura 14.37 mostra a tela visualizada pelos usuários, no momento em que se conclui a ação anteriormente referida.

Finalmente, a figura 14.38 ilustra a seleção dos servidores Archie que serão consultados, bem como a definição da chave de pesquisa e os valores dos parâmetros de funcionamento a eles associados.

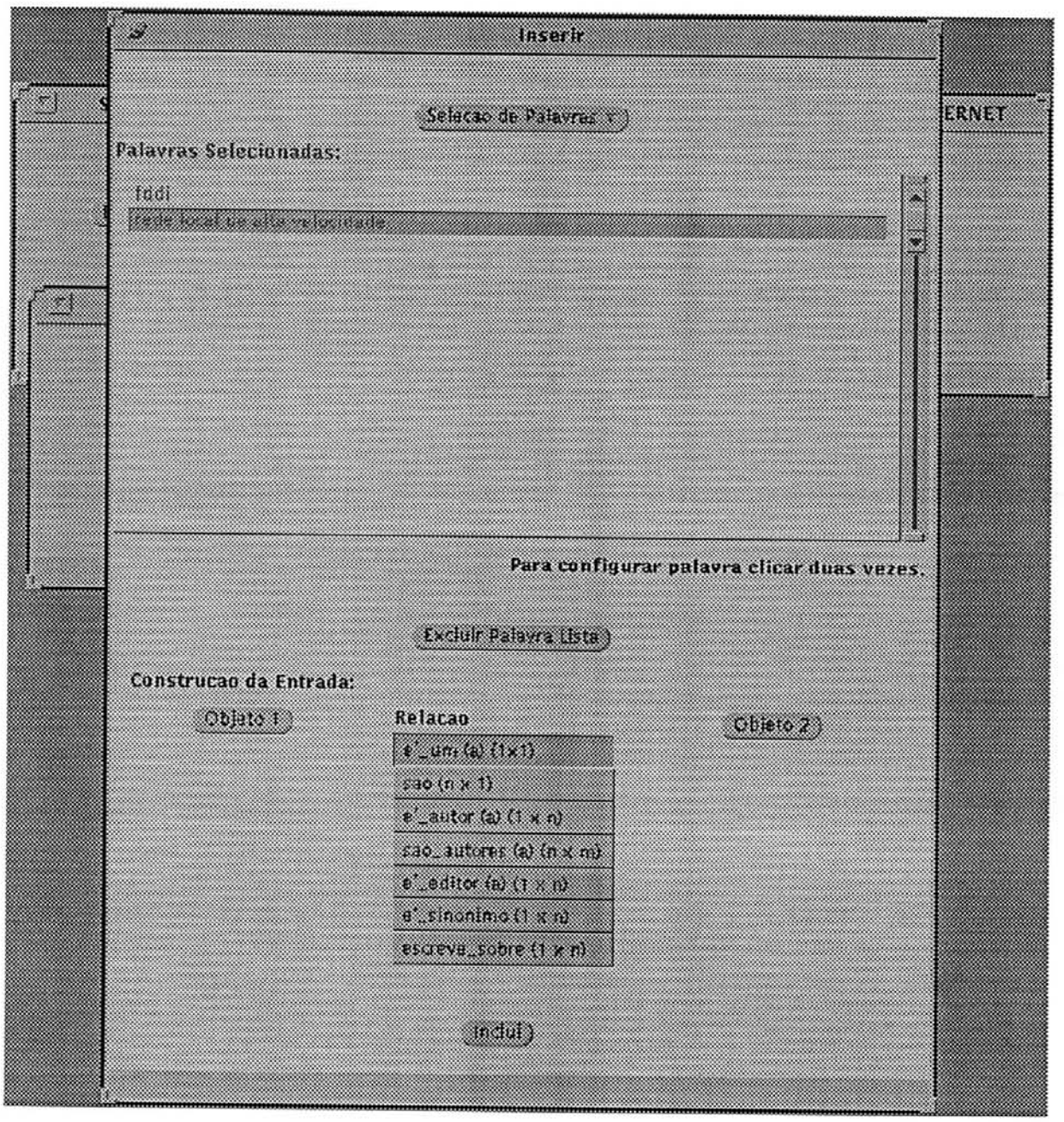

Figura 14.35 *Inclusão efetiva da entrada na base de conhecimento do sistema.*

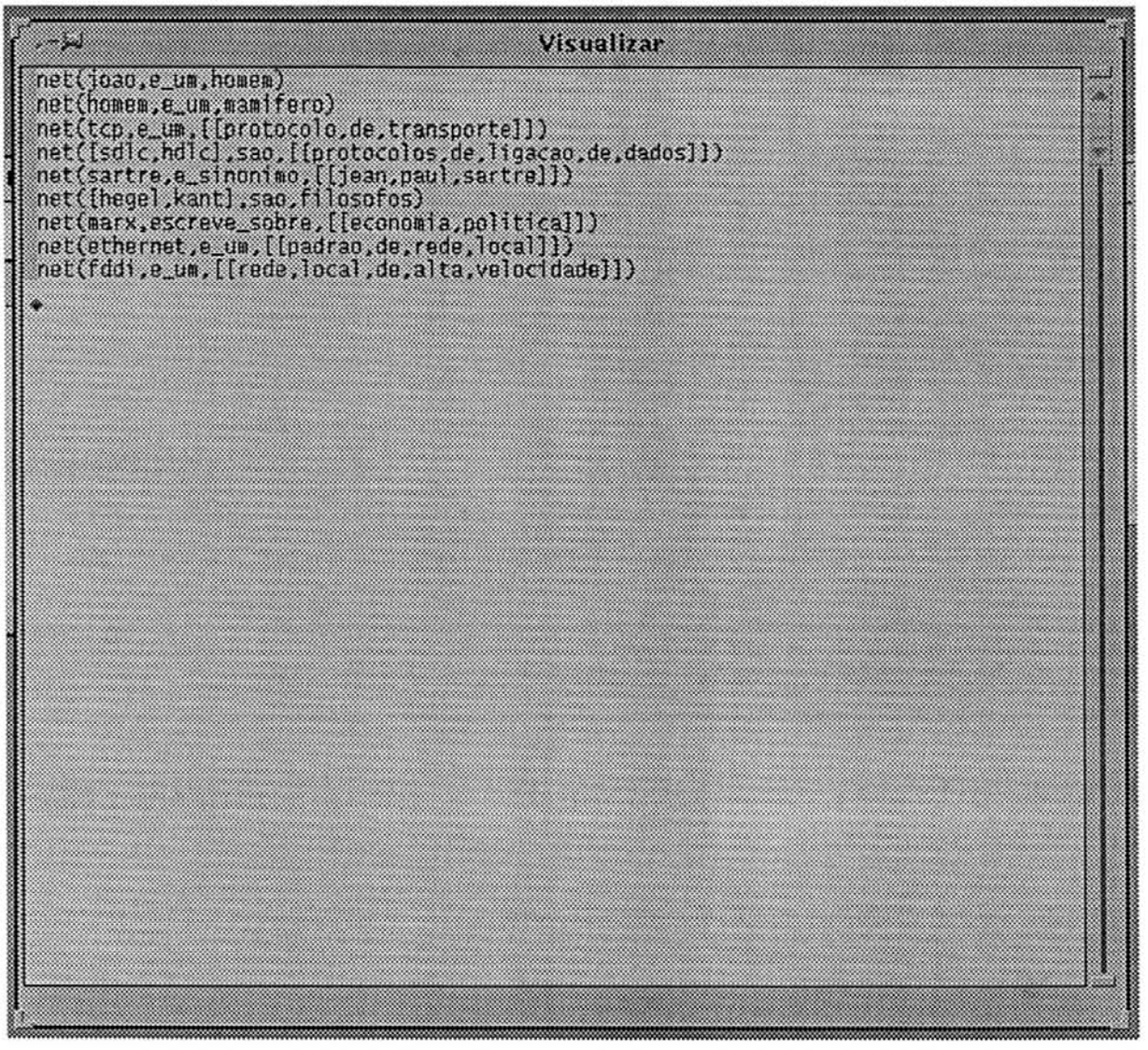

Figura 14.36 *Visualização da base de conhecimento resultante após a inclusão de uma nova entrada.*

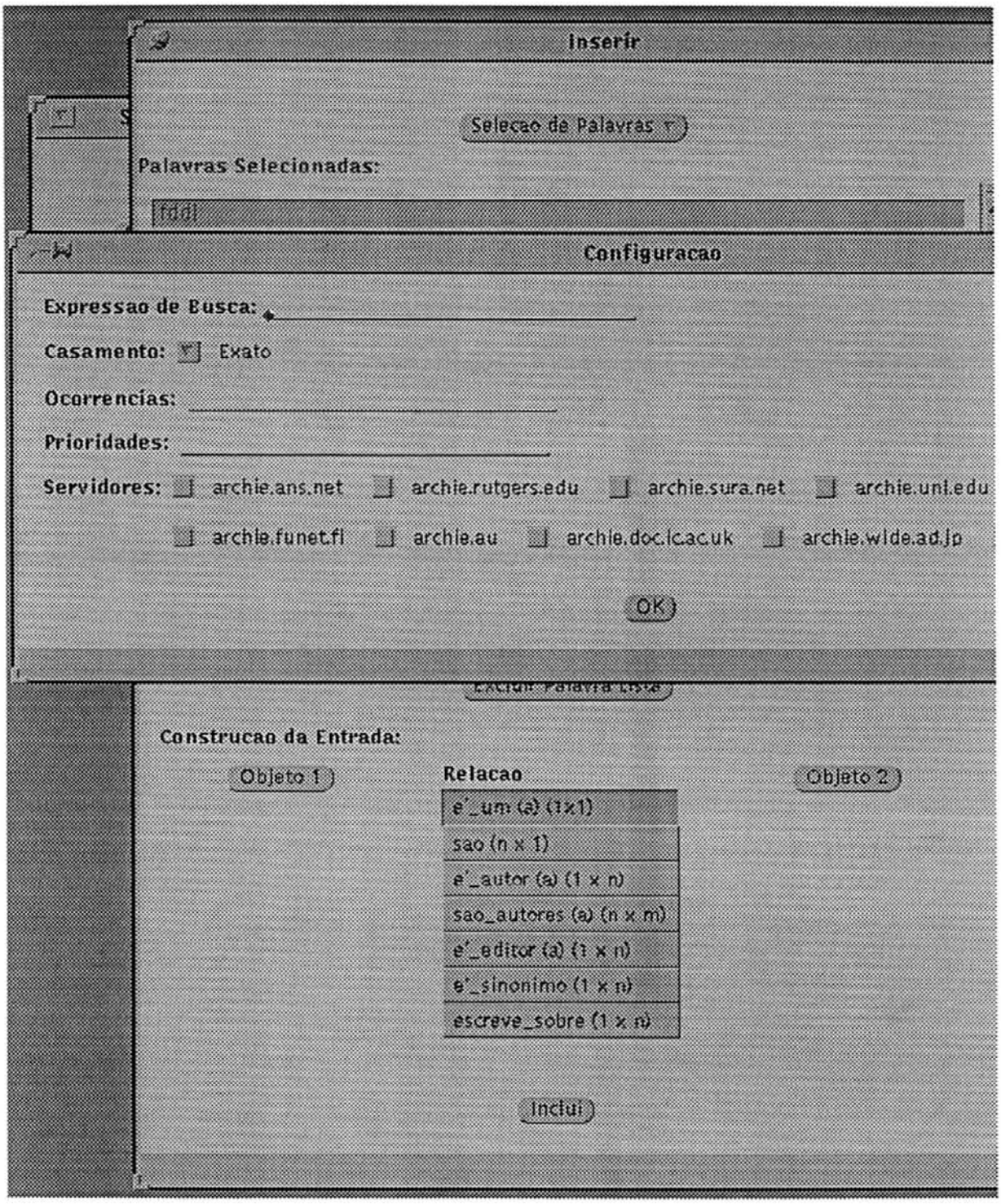

Figura 14.37 *Tela usada para a definição de parâmetros de funcionamento de servidores Archie para uma determinada consulta.*

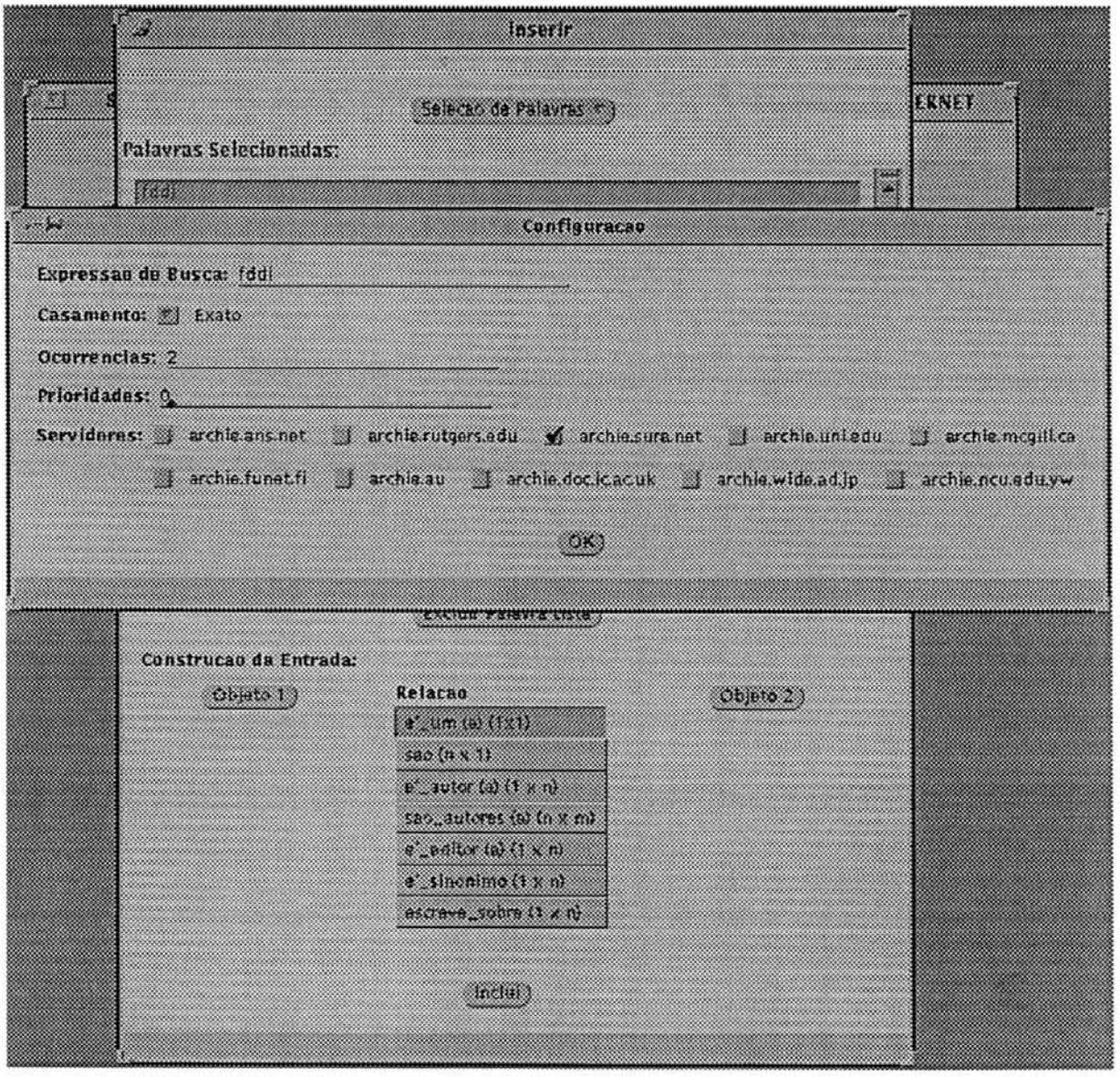

Figura 14.38 *Parâmetros usados em uma consulta, cujo núcleo é a palavra* FDDI.

É importante destacar que a ordem segundo a qual são encaminhadas as consultas aos servidores Archie, poderá ou não ser alterada, dependendo da observância dos critérios de ordenação, descritos no capítulo 13, na seção 13.3.6.

Infelizmente, na presente versão do sistema, o usuário para expandir a base de conhecimento necessita obrigatoriamente saber como representá-lo através de uma rede semântica. No entanto, parece-nos que, embora exista esta restrição, esta flexibilidade é fundamental para o êxito do sistema. No capítulo 15, destacamos as perspectivas deste e de futuros trabalhos que venham a abordar mais profundamente a questão do aprendizado de conhecimentos, propondo formas mais adequadas e amigáveis para obtê-lo e representá-lo.

14.3.9 Aprendendo o Significado de Novas Sentenças

Nesta subseção, discutiremos a última característica importante do **SDIP** posta à disposição dos usuários através da interface gráfica, qual seja, a possibilidade do sistema aprender o significado de sentenças, que para ele sejam desconhecidas.

Para explicar esta propriedade do **SDIP**, consideremos que o sistema desconhece o significado da frase 14.5. Caso o usuário submeta esta frase ao **SDIP**, o sistema entrará no modo de aprendizado do significado de sentenças.

A figura 14.39 mostra a tela principal do **SDIP**, quando o sistema se encontra neste modo de operação, indicado pela presença de dois novos botões: *aprende* e *ignora*.

Caso o usuário selecione o botão *ignora*, o sistema sairá do modo de aprendizado, descartando a sentença formulada anteriormente.

Frase 14.5 Localize os livros escritos por Karl Marx[4].

[4]Karl Marx (1818:1883), fundador do socialismo científico. Nascido na Alemanha, dedicou-se principalmente à filosofia, teoria e economia política. Autor de obras importantes, como o Manifesto Comunista, em parceria com seu amigo Engels, e O Capital, entre muitas outras.

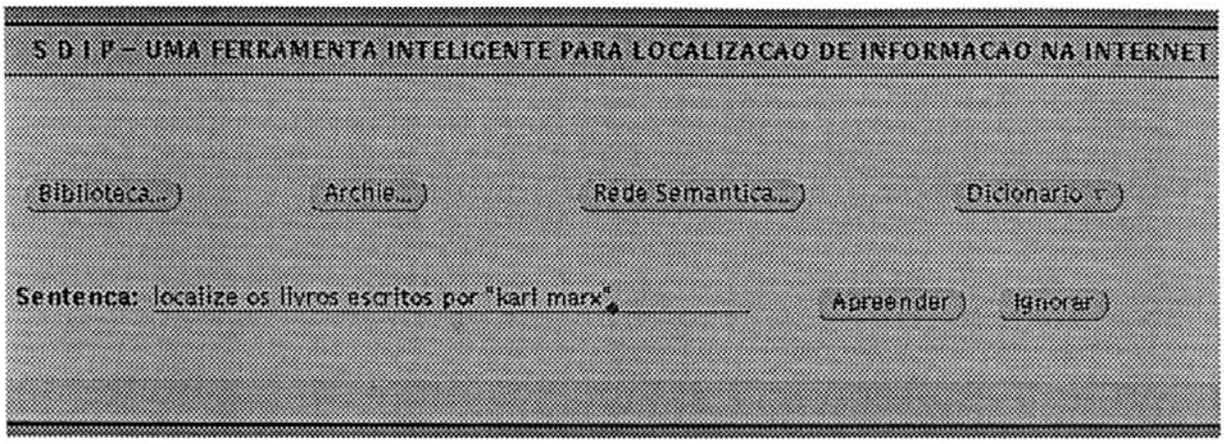

Figura 14.39 *Janela principal do* **SDIP** *quando o sistema se encontra no modo de aprendizado.*

Por outro lado, se o usuário desejar explicar ao **SDIP** o significado da sentença 14.5, ele deverá proceder como segue:

- Utilizando a interface gráfica, o usuário deve realizar todas as operações expressas implicitamente na sentença, ou seja, ele deverá usar a janela do sistema SABI para localizar informações bibliográficas a respeito dos livros de autoria de Karl Marx;
- No momento em que a operação descrita no item anterior estiver concluída, o usuário deverá selecionar o botão *Aprender*, na janela principal do **SDIP**.

Ao concluírem-se os passos acima referidos, o **SDIP** saberá como proceder quando lhe forem submetidas sentenças similares a do exemplo 14.5. É importante destacar que o **SDIP** compreenderá a sentença em seu sentido amplo ou genérico, podendo ser aplicada a outros autores, como, por exemplo, Immanuel Kant[5].

As figuras 14.40, 14.41 e 14.42 mostram o procedimento descrito acima, seguido por um usuário, com o intuito de explicar ao **SDIP** o significado da sentença 14.5.

[5]Immanuel Kant (1724:1804),filósofo alemão do século dezoito. Pertencia à corrente idealista da filosofia, similarmente a Hegel.

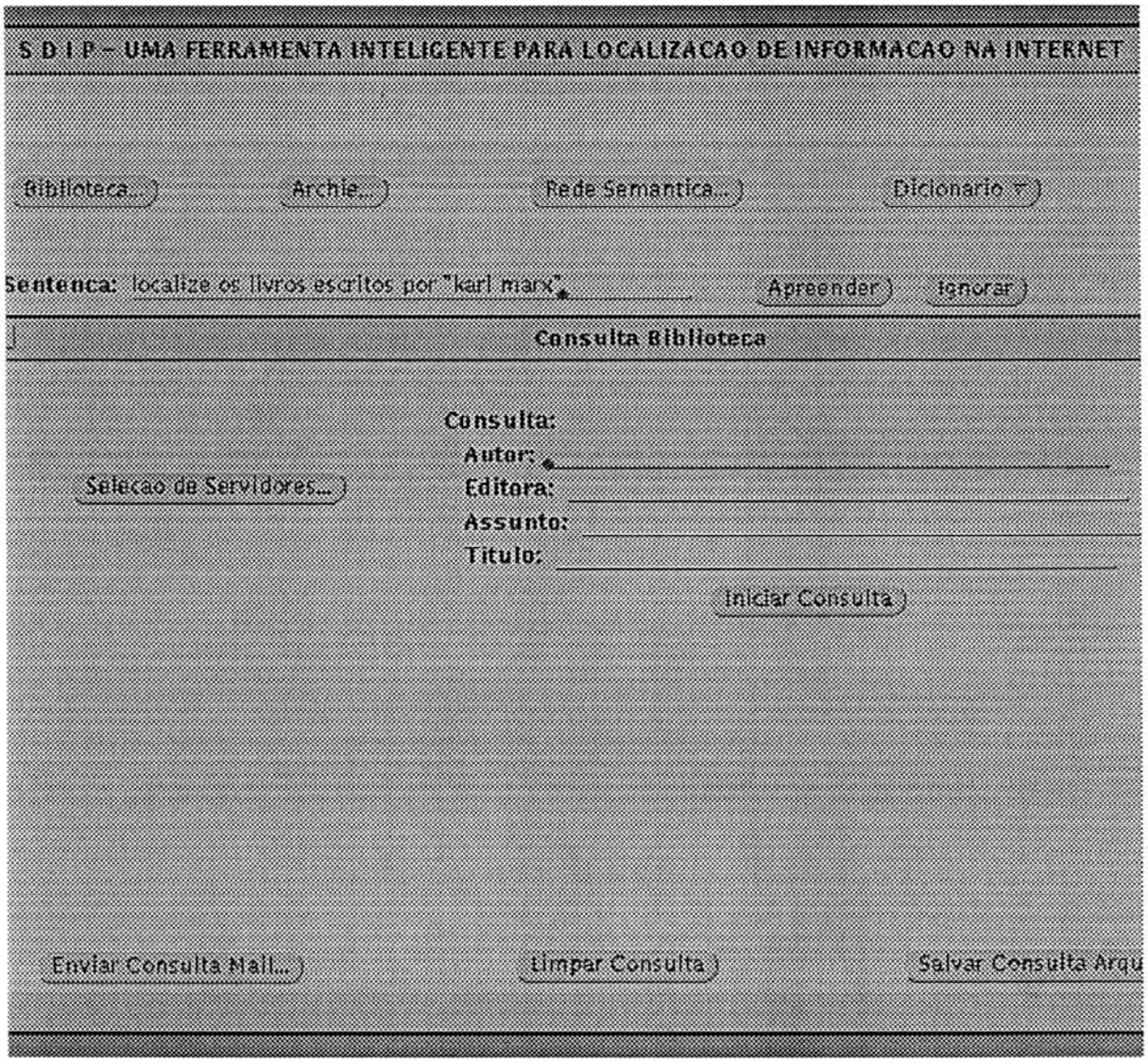

Figura 14.40 *O usuário abre a janela associada ao botão* biblioteca.

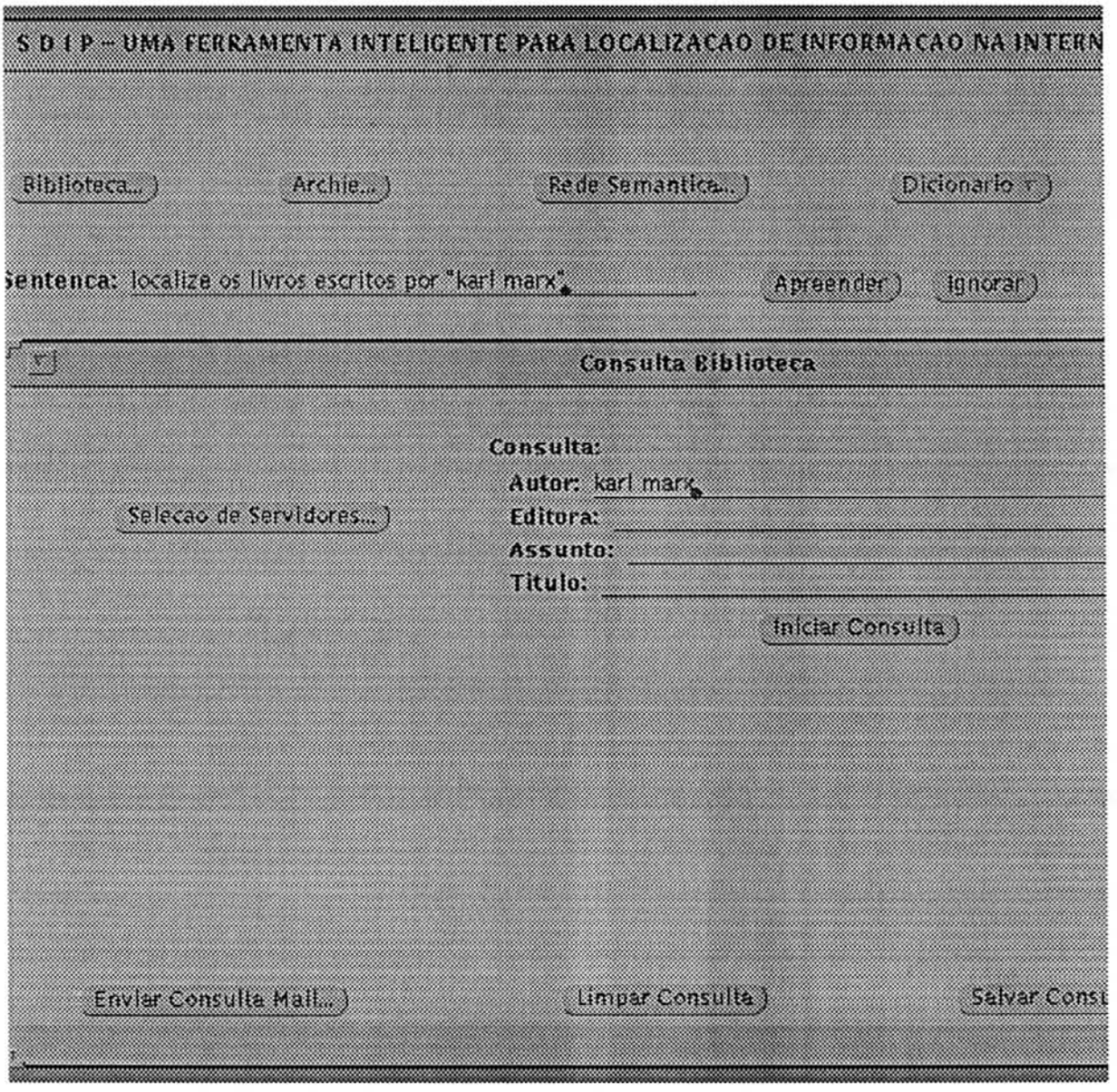

Figura 14.41 *Usuário define o autor e inicia a consulta.*

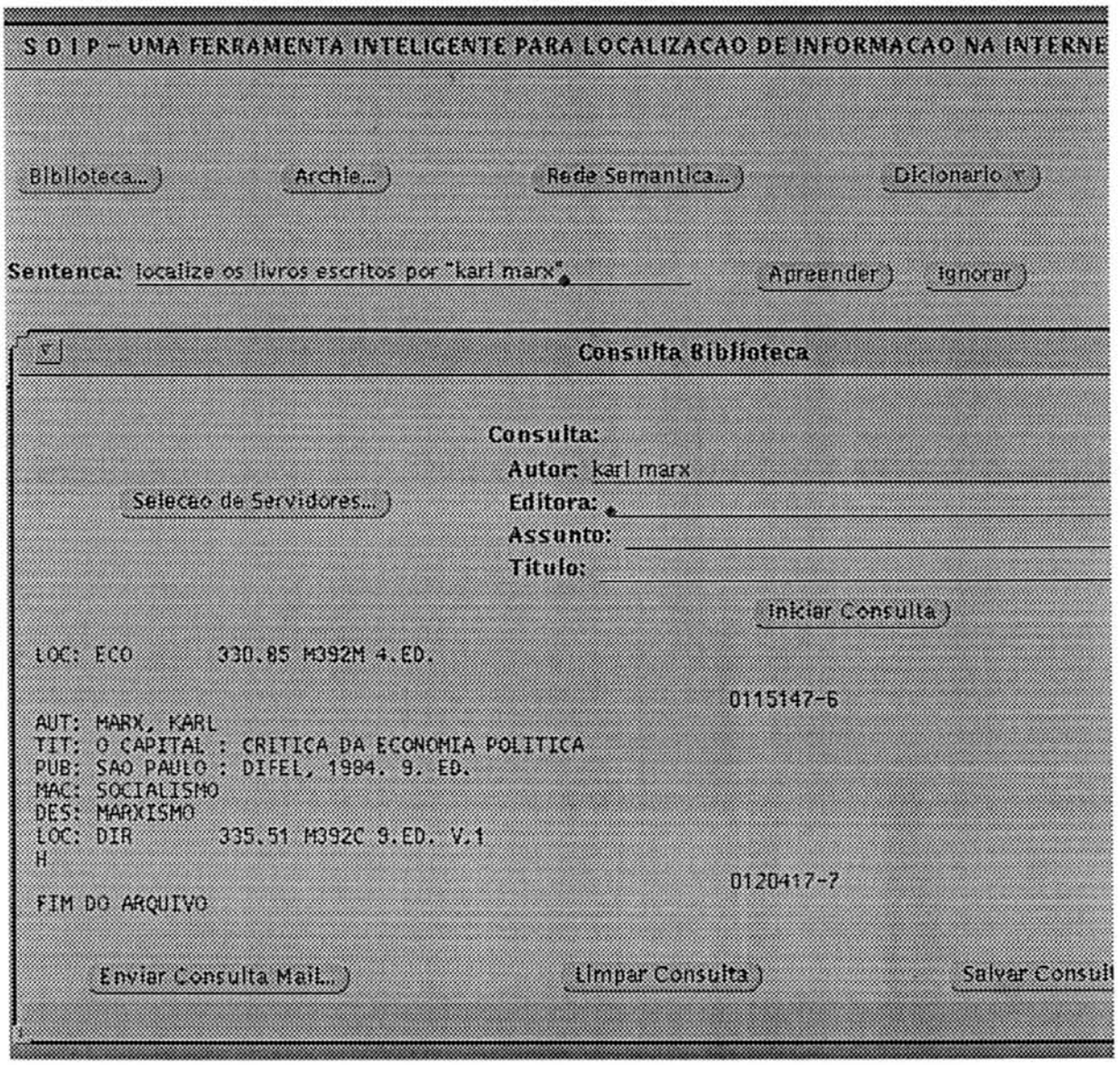

Figura 14.42 *Usuário visualiza os resultados da consulta e seleciona o botão* Aprende.

A figura 14.43 comprova o efetivo aprendizado por parte do **SDIP** do significado da sentença 14.5. Nesta figura, observamos que o sistema obteve as respostas esperadas com a submissão da frase original.

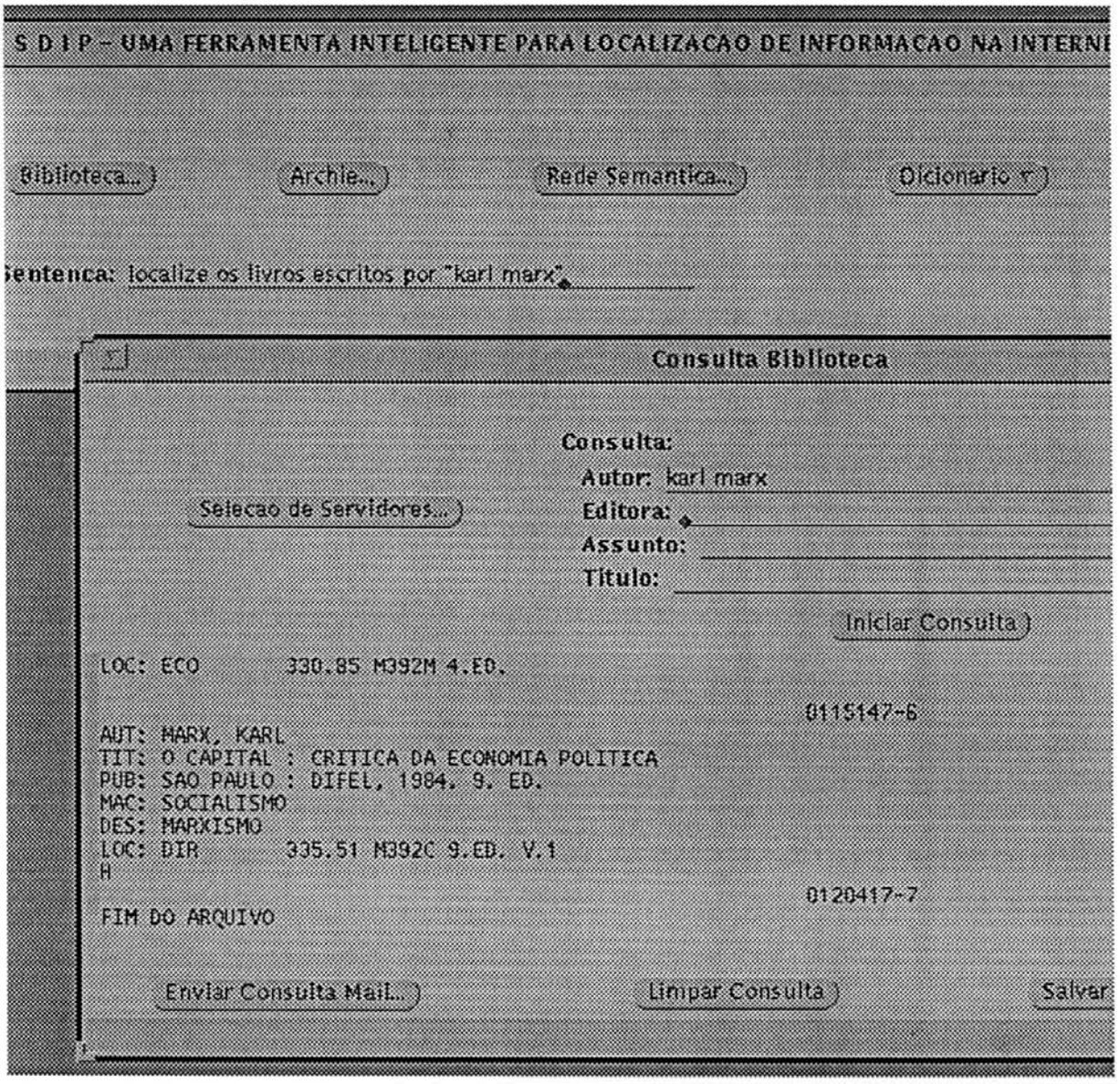

Figura 14.43 *Comprovação do efetivo aprendizado do significado da sentença.*

Antes de concluirmos o presente capítulo, no qual enfocamos a utilização do **SDIP**, tendo como ambiente a interface gráfica do sistema, gostaríamos de destacar que a cada botão, menú, item ou opção mostrada nas diversas janelas, estão associados textos explicativos que descrevem a sua função. Para ler estas informações descritivas, o usuário deve posicionar o cursor, movendo o mouse sobre o item a

respeito do qual ele deseja algum tipo de esclarecimento. Concomitantemente, ele deve pressionar a tecla *Help*, existente no teclado da estação de trabalho. Neste momento, será mostrado ao usuário um breve texto descrevendo a função do item selecionado.

Na figura 14.44, mostramos o texto que descreve o botão *Limpar Consulta*, presente na janela *Biblioteca* da interface gráfica.

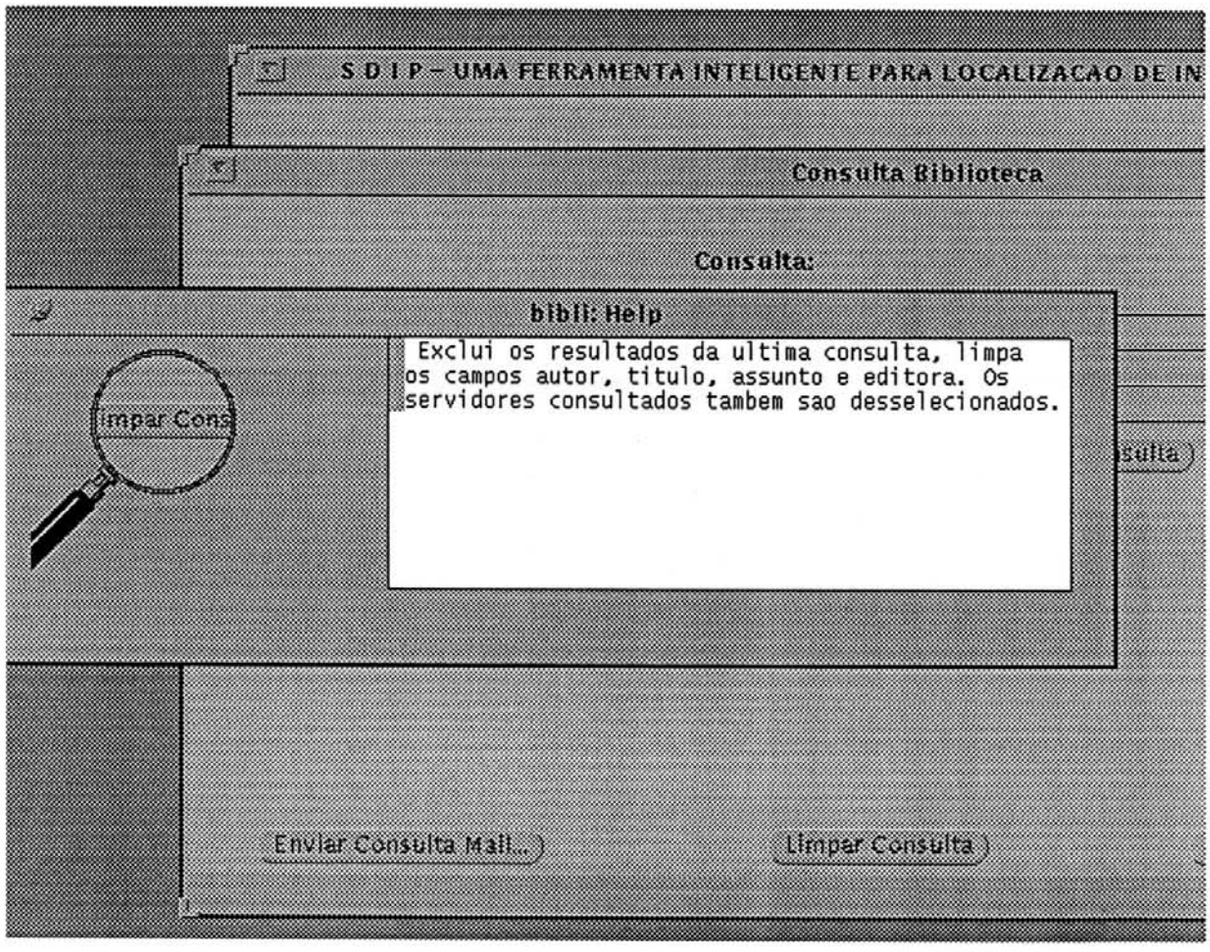

Figura 14.44 *Texto descritivo do botão* Limpar Consulta, *mostrado na janela para consultas bibliográficas do* **SDIP**.

Finalmente, parece-nos importante destacar que a melhor maneira de se conhecer as facilidades e potencialidades de um sistema e sua interface, seria usá-lo freqüentemente, sem preocupar-se com eventuais erros cometidos durante a utilização do mesmo.

15 CONCLUSÃO

15.1 Considerações Gerais

Considerando a proposta inicial do presente trabalho – implementar um sistema inteligente, capaz de auxiliar os usuários a localizar e recuperar informações dentro da rede Internet – e as restrições anteriormente discutidas, principalmente nos capítulos 1, 7 e 13, obtivemos como produto final um sistema que, de fato, constitui-se em um instrumento bastante útil para aquelas pessoas imbuídas do objetivo de explorar plenamente o potencial da Internet como um meio de difusão de informações. Adicionalmente, acreditamos que os resultados deste e de outros trabalhos que venham a ser realizados nesta mesma área, virão a beneficiar principalmente aquele conjunto de pessoas que sentem dificuldade em usar os sistemas atuais, e que, por esta razão, são privadas de uma vasta quantidade de conhecimentos que hoje residem na Internet.

Particularmente no que concerne às contribuições deste trabalho, acreditamos que a principal delas refere-se à constatação de que o **SDIP** é um sistema que não constitui-se, por assim dizer, em uma linha de chegada, mas sim um ponto a partir do qual muitos outros trabalhos e estudos poderão ser desenvolvidos. Neste aspecto, o que nos parece o mais importante a ser destacado diz respeito ao fato de que este conjunto de projetos de pesquisa potenciais não estão restritos a uma única área da Ciência da Computação, envolvendo, por exemplo, pesquisadores ligados à área de inteligência artificial, sistemas de informações, redes de computadores etc. Na seção 15.2, mostramos alguns exemplos ilustrativos de trabalhos que podem ser desenvolvidos, tendo como base o que foi realizado e exposto nesta dissertação.

Finalmente, fazendo um breve apanhado a respeito do que foi desenvolvido e estudado no decorrer desta dissertação, gostaríamos de tecer alguns comentários finais acerca dos servidores de informações encontrados na Internet, da DRT e da implementação do sistema.

No que concerne ao primeiro tópico, os estudos realizados reafirmaram a nossa convicção inicial de que estes sistemas, hoje e cada vez mais no futuro, terão um papel decisivo e destacado dentro da Internet, principalmente na área de difusão de informações e serviços. Neste sentido, estamos certos de que o *WWW* terá uma participação destacada em relação aos demais sistemas, conseqüência advinda principalmente da maior flexibilidade e potencialidade do protocolo de comunicação por ele adotado.

Outro aspecto interessante observado no decorrer deste estudo é a inexistência de um sistema similar ao **SDIP**, no tocante à aplicação de técnicas de inteligência artificial. Neste particular, o único sistema onde é mencionada esta possibilidade é o **WAIS**, todavia a utilização de tais recursos não passam, pelo menos até o presente momento, de meras conjecturas.

A adoção da DRT como formalismo para representação do significado semântico das sentenças foi uma escolha que satisfez plenamente as necessidades e restrições impostas pela implementação. Ainda a este respeito, parece-nos importante destacar que, pelo menos na presente versão do sistema, utilizamos uma fração mínima de todo o potencial da DRT [KAM 90], o que, sem sombra de dúvida, abre ótimas perspectivas para o possível melhoramento da interface em língua natural atualmente implementada.

Finalmente, no tocante à implementação do sistema, muitos aspectos favoráveis ou limitantes já foram discutidos em outros capítulos desta dissertação. Contudo, gostaríamos de enfatizar ainda as vantagens advindas do fato de termos particionado o **SDIP** em vários processos individuais, principalmente no que concerne a programação e teste de cada um dos módulos.

Um outro aspecto relevante a ser relatado refere-se à pouca eficiência do analisador sintático, comparativamente aos demais componentes do sistema, aliado à dificuldade por nós encontrada em promover as necessárias alterações em suas regras de interpretação, visando torná-lo adequado à nossa realidade.

Uma observação importante, anteriormente discutida no capítulo 12, mas que aqui deve ser relembrada, diz respeito ao módulo de interface que permite aos usuários modificar a base de conhecimento do sistema. Neste particular, para que esta facilidade seja efetivamente utilizada por todos os usuários, deve-se trabalhar na construção de uma interface onde não seja necessário representar ou expressar uma realidade diretamente através de uma rede semântica, visto que esta técnica é do conhecimento de um grupo bastante restrito de pessoas.

15.2 Trabalhos Futuros

Um dos mais importantes parâmetros que pode ser utilizado para a avaliação qualitativa de um trabalho de pesquisa, seja ele realizado em nível de graduação, mestrado, doutorado, ou até mesmo na vida profissional, tendo como referência a razão central que motivou o seu desenvolvimento, são as perspectivas, caminhos e idéias por ele inspiradas, tanto em quem o realizou efetivamente, quanto naqueles que porventura tomaram contato com o produto final (trabalho escrito e/ou software).

Neste sentido, tendo em vista os objetivos necessariamente limitados que restringiram a abrangência do presente trabalho, a necessidade e possibilidade de aprofundar os estudos realizados, melhorando, desta forma, os resultados até agora obtidos e, finalmente, a eventual extensão e/ou deslocamento do foco de atenção deste trabalho para outras áreas correlatas, comportam-se como fatores capazes de estimular a criatividade deste e de outros pesquisadores, ensejando-lhes a concepção de projetos que podem ser propostos como trabalhos a serem desenvolvidos futuramente, tanto em nível de graduação, quanto de pós-graduação.

Neste particular, merecem destaque as seguintes propostas de trabalhos de pesquisa, os quais visam complementar ou aproveitar os resultados obtidos e os conhecimentos adquiridos durante o período de preparação desta dissertação:

1. Desenvolvimento de novos processos de interface, capacitando o **SDIP** a interagir com outros servidores de informações existentes na rede Internet. Adicionalmente, devem ser realizados estudos visando definir a estrutura e complexidade do conhecimento que deve ser agregado ao sistema, capacitando-o a interagir coerentemente com estes servidores;

2. Uma das principais limitações verificadas na versão atual do protótipo desenvolvido, refere-se ao fato de que as modificações efetuadas por um usuário, em qualquer base de dados ou de conhecimento do sistema, é visualizada por outras pessoas que utilizam o **SDIP**, mesmo que elas, por exemplo, tenham uma visão diferenciada a respeito da realidade que deve ser representada. Por esta razão, poderiam-se desenvolver trabalhos, cujo objetivo seja promover as necessárias modificações no **SDIP**, capacitando-o a manter informações contextuais de cada um de seus usuários, solucionando-se, desta maneira, o problema acima referido;

3. Nesta dissertação foram estudadas apenas sentenças simples, para as quais a implementação gera uma representação DRS. Consideramos importante para a continuidade deste trabalho, aproximando-o ainda mais da realidade dos usuários, os quais freqüentemente utilizam construções sentenciais mais complexas, expandir a capacidade do analisador sintático e do gerador de DRSs, habilitando-os a reconhecer e representar sentenças de outras classes, tais como: orações relativas, subordinativas etc;

4. A implementação não é capaz de resolver os anáforas que surgirem no corpo de uma dada sentença ou discurso. Neste sentido, seria interessante a realização de estudos comparativos entre a solução proposta pela DRT [KAM 90] e outros algoritmos existentes, incorporando-se posteriormente algum deles à implementação;

5. Incorporação ao protótipo de um analisador capaz de aprender novas regras sintáticas [VIC 85]. Adicionalmente, seria desejável que no mo-

mento em que uma nova regra sintática fosse aprendida ou alterada, esta informação encontre eco nas regras de geração das DRSs;

6. Como tivemos a oportunidade de destacar anteriormente, o analisador sintático constituiu-se em um dos pontos problemáticos do sistema, particularmente em relação a sua performance e facilidade de promoverem-se modificações em suas regras de análise sintática. Por este motivo, consideramos importante a adoção de uma metodologia alternativa para a construção deste analisador, objetivando torná-lo mais eficiente e flexível;

7. Estudo das teorias, metodologias e técnicas orientadas à aquisição, estruturação e representação do conhecimento [BRA 91, BRA 92], tornando possível a construção de uma interface mais adequada para a realização desta tarefa, e que seja capaz de atender às necessidades particulares dos usuários do sistema;

8. Construção de uma interface em língua natural destinada a outros ambientes e realidades, como, por exemplo, um sistema de gerência ou planejamento de redes de computadores. Neste último caso, pode-se agregar ao ambiente módulos que implementem algoritmos de construção e reconhecimento de planos [BAR 93, CRO 93, FER 92, GHA 93, MIL 92], além de outras facilidades destinadas a auxiliar o projetista a determinar a configuração de rede mais adequada a uma realidade considerada.

BIBLIOGRAFIA

[ALL 80] ALLEN, James; PERRAULT, Raymond. Analyzing intention in utterances. **Artificial Intelligence**, Amsterdam, v. 15, p. 143–178, July 1980.

[ALL 84] ALLEN, James F. Towards a General Theory of Action and Time. **Artificial Intelligence**, Amsterdam, v. 23, p. 225–255, July 1984.

[AME 92] AMERICAN NATIONAL STANDARDS INSTITUTE; NATIONAL INFORMATION STANDARDS ORGANIZATION. **Z39.50-1992 (version 2) Information Retrieval Service and Protocol**: American National Standard, Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection. [S.l.]: American National Standards Institute, 1992.

[ANK 93] ANKLESARIA, F.; MCCAHILL, M.; LINDNER, P. et al. **The Internet Gopher Protocol (a distributed document search and retrieval protocol)**. Mar. 1993. 16p. (RFC 1436).

[ARA 89] ÁRABE, Ana Maria Ferreira. Comunicando-se com Bases de Dados Estáticas em Linguagem Natural: o Protótipo de uma Interface. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 7., nov. 1989, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1989. p. 261–275.

[ARD 85] ARAGON, Doris Ferraz de; CASTRO, Ilka Dias de. Um Sistema Formal para Representação de Conhecimento. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 2., nov. 1985, São José dos Campos. **Anais...** São José dos Campos: INPE, 1985. p. 123–126.

[ARD 88] ARAGON, Doris F.; MONARD, Maria C. Uma Implementação dos Sistemas de Representação de Conhecimentos SC. In: SIMPÓSIO

BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 5., nov. 1988, Natal. **Anais...** Natal: UFRN, 1988. p. 348–355.

[ASS 86] ASSIS, Fidelis Sigmarinda Gomes de.; PASSOS, Emmanuel Lopes. Tradutor para Estruturas Lógicas e Interpretador Semântico. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 3., nov. 1986, Rio de Janeiro. **Anais...** Rio de Janeiro: IME, 1986. p. 134–167.

[BAR 93] BARRET, Antony; WELD, Daniel S. **Partial-Order Planning**: Evaluating Possible Efficiency Gains. Seattle, WA: University of Washington, Computer Science Department, Feb. 1993. 32p. (Technical Report, 92–05–01).

[BEL 91] BELINA, Ferenc; HORGRACE, Dieter; SARMA, Amardo. **SDL With Applications From Protocol Specification.** Hertfordshire, Ingland: Prentice-Hall, 1991. 249p.

[BER 92] BERNERS-LEE, T.; CAILLIAU, R.; GROFF, J-D. et al. World-Wide Web: An Information Infrastructure for High-Energy Physics. In: WORKSHOP ON SOFTWARE ENGINEERING, ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS FOR HIGH-ENERGY AND NUCLEAR PHYSICS, Jan. 1992, Seattle, WA. **Proceedings ...** Seattle, WA: [s. n.], 1992.

[BER 92a] BERNERS-LEE, T.; CAILLIAU, R.; GROFF, J-D. et al. World-Wide Web: The Information Universe. **Eletronic Networking: Ressearch, Application and Policy**, [S.l.], v. 2, Spring 1992.

[BER 93] BERNERS-LEE, Tim. **Hypertext Transfer Protocol (HTTP)**: A Stateless Search, Retrieval and Manipulation Protocol. Nov. 1993. (Internet Draft). 29p.

[BER 94] BERNERS-LEE, Tim. **Hypertext Transfer Protocol (HTTP)**: A Stateless Search, Retrieval and Manipulation Protocol. May. 1994. (Internet Draft). 30p.

[BER 94a] BERNERS-LEE, T. **Universal Resource Identifiers in WWW**: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. June 1994. 28p. (RFC 1630).

[BRA 91] BRAGA, José Luís; CARVALHO, Roberto Lins de. Episteme: Aquisição e Estruturação Semi-Automática de Conhecimento. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 7., nov. 1991, Natal. **Anais...** Natal: SBC, 1991. p. 383–394.

[BRA 92] BRAGA, José Luís; CÉSAR, Carlos Neves Lenz; SILVA, Elton José da. Aquisição de Conhecimento Básico no Sistema Episteme. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 9., set. 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p. 170–178.

[BRAT 86] BRATKO, Ivan. **Prolog Programming for Artificial Intelligence**. Wokingham : Addison-Wesley, c1986. 423 p.

[CAR 87] CARRAHER, David William; BORBA, Paulo Henrique; SANTOS, André Luís. Acesso a um Banco de Dados Através de Linguagem Natural. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 4., out. 1987, Uberlândia. **Anais...** Uberlândia: UFU, 1987. p. 217–224.

[COH 91] COHEN, Philip R.; LEVESQUE, Hector J. Intention Is Choice with Commitment. **Artificial Intelligence**, Amsterdam, v. 42, p. 213–261, Mar. 1991.

[COM 88] COMER, Douglas E. **Internetworking with TCP/IP** : Principles, Protocols and Architecture. Englewood Cliffs : Prentice-Hall, c1988. 382p.

[COM 91] COMER, Douglas E.; STEVENS, David L. **Internetworking with TCP/IP**. Englewood Cliffs : Prentice-Hall, c1991. 2v.

[CRO 93] CROUCH, R. S; PULMAN, S. G. Time and Modelity in a Natural Language Interface to a Planning System. **Artificial Intelligence**, Amsterdam, v. 63, p. 265, Oct. 1993.

[DAN 91] DANZIG, Peter; NOLL, John; OBRACZKA, Katia. Distributed Indexing: A Scalable Mechanism for Distributed Information Retrieval. In: ANNUAL INTERNATIONAL ACM/SIGER CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 14., Oct. 1991. **Proceedings...** Washington: [s.n.], 1991. p.220–229.

[DEU 94] DEUTSCH, P.; EMTAGE, A.; MARINE, A. **How to Use Anonymous FTP**. Mai. 1994. 13p. (RFC 1635).

[ELL 90] ELLIS, Margaret A.; STROUSTRUP, Bjarne. **Annotated C++ Reference Manual**. Reading : Addison-Wesley, 1990. 447 p.

[EMT 91] EMTAGE, Alan. **The Archie Server**. Montreal, Quebec: McGill University, Computer Science Department, Oct.1991. 18p. (Technical Report).

[EMT 92] EMTAGE, Alan; DEUTSCH, Peter. **Archie**: An Eletronic Directory Service for the Internet. Montreal, Quebec: McGill University, Computer Science Department, Jan. 1992. 30p. (Technical Report).

[EVS 92] EVSUKOFF, Alexandre; ARAGON, Doris. Uma Formalização de Redes Semânticas para um Sistema Híbrido de Representação de Conhecimento. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA

ARTIFICIAL, 9., set. 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p. 26–37.

[FER 92] FERGUSON, George. **Explicit Representation of Events, Actions and Plans for Assumption-Based Plan Reasoning.** Rochester, New York: University of Rochester, Computer Science Department, June 1992. 48p. (Technical Report, 428).

[FER 93] FERNANDEZ, Luís Fernando. **SAS: Smart Answering System.** Porto Alegre: CPGCC da UFRGS, 1993. 68 p. (Trabalho individual, 350).

[FON 92] FONSECA NETO, Raul. Proposta de uma Arquitetura para o Desenvolvimento de Sistemas Inteligentes. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 9., Set. 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p. 358–372.

[FRA 86] FRAGA, Paltônio Daun. Tratamento Computacional de Línguas Naturais. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 3., nov. 1986, Rio de Janeiro. **Anais...** Rio de Janeiro: IME, 1986. p. 168–177.

[FRE 92] FREITAS, Sérgio Antônio Andrade de. **Um Estudo Sobre a Teoria da Representação do Discurso em Português.** Porto Alegre: CPGCC da UFRGS, 1992. 38p. (Trabalho Individual, 273).

[FRE 92a] FREITAS, Sérgio Antônio Andrade de. A Utilização da DRT em um Sistema de Representação do Discurso. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 9., set. 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p. 383–394.

[FRE 93] FREITAS, Sérgio Antônio Andrade de. **Deiticos e Anáforas Pronominais em Diálogos.** Porto Alegre: CPGCC da UFRGS, 1993. 111 p. (Dissertação de Mestrado).

[FRE 93a] FREITAS, Sérgio Antônio Andrade de; LOPES, José Gabriel Pereira. Um Sistema de Representação do Discurso Utilizando a DRT e a Teoria do Foco. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10., out. 1993, Porto Alegre. **Anais...** Porto Alegre: SBC, 1993. p.345–359.

[FUL 94] FULLTON, M. J.; GOLDMAN, K. J.; KUNZE, B. J. et al. **WAIS over Z39.50-1988**. June 1994. 7p. (RFC 1625).

[GHA 93] GHALLAB, Malik. Managing Temporal Knowledge for Interpretation and Planning. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10., out. 1993, Porto Alegre. **Anais...** Porto Alegre: SBC, 1993. p.17–32.

[HEE 92] HEEMAN, Peter A.; HISRT, Graeme. **Collaborating on Referring Expressions**. Rochester, New York: University of Rochester, Computer Science Department, Aug. 1992. 33p. (Technical Report, 435).

[HEL 90] HELLER, Dan. **XView Programming Manual** : for Version 11 of the X Window System. Sebastopol : O'Reilly & Associates, 1990. 642 p.

[HOV 93] HOVY, E. H. Automated Discourse Generation Using Discourse Structure Relations. **Artificial Intelligence**, Amsterdam, v. 63, p. 341, Oct. 1993.

[INT 88] INTERNATIONAL TELEGRAPH AND TELEPHONE CONSILTATIVE COMMITTEE. **SDL User Guidelines**. Geneva: [s.n.], Nov. 1988. 180p.

[KAH 91] KAHLE, Brewster. **An Information System for Corporate Users**: Wide Area Information Servers. **Connexions: The Interoperability Report**, [S.l.], v. 5, Nov. 1991.

[KAM 88] KAMP, Hans. **Discourse representation theory**: what it is and where ought to go. Heidelberg:Spring-Verlag, 1988. p. 84–111. (Lecture Notes in Computer Science, 320).

[KAM 90] KAMP, H.; REYLE, U. **From Discourse to Logic**: an Introduction to Model Theoretic Semantics of Natural Language. Stuttgart: Institute for Computational Linguistics, University of Stuttgart, 1990. 458p.

[KAN 86] KANTOR, B.; LAPSLEY, P. **Network News Transfer Protocol**: A Proposed Standard for the Stream-Based Transmission of News Feb. 1986. 27p. (RFC 977).

[KER 78] KERNIGHAN, Brain W.; RITCHIE, Dennis M. **The C Programming Language**. Englewood Cliffs : Prentice-Hall, c1978. 228p.

[KOP 85] KOPP JUNIOR, Roberto Velasco; GUERREIRO, Ramiro Affonso Tadeu. Estudos em Linguagem Natural. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 2., nov. 1985, São José dos Campos. **Anais...** São José dos Campos: INPE, 1985. p. 147–150.

[KRO 92] KROL, Ed. **The Whole InterNet** : User's Guide & Catalog. Sebastopol : O'Reilly & Associates, c1992. 376 p.

[LEM 91] LEMOS, Rosana Probst. **Um Estudo do Problema da Ambigüidade no Processamento da Língua Natural**. Porto Alegre: CPGCC da UFRGS, 1991. 36p. (Trabalho Individual, 244).

[LEM 92] LEMOS, Rosana Probst. **Aprendizagem Automática da Semântica de Orações Relativas não Restritivas**. Porto Alegre: CPGCC da UFRGS, 1992. 107p. (Dissertação de Mestrado).

[LOP 88] LOPES, Gabriel P. Metodologia para um Desenvolvimento de Sistemas de Informação Distribuidos com Capacidade para Dialogarem em Português. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 5., nov. 1988, Natal. **Anais...** Natal: UFRN, 1988. p. 125–137.

[LOP 92] LOPES, J.G.P. Architecture for Intentional Participation of Natural Language Interfaces in Conversations. In: BROWN, C.; KOCH, G. (Eds.) **Natural Language Understanding and Logic Programming III.** Cambridge, MA: North-Holland, 1992. p. 312–322.

[LUZ 93] LUZ FILHO, Saturnino Francisco. Representação Semântica de Atitudes Proposicionais Através da Teoria de Atos de Fala. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10., out. 1993, Porto Alegre. **Anais...** Porto Alegre: SBC, 1993. p.361–376.

[LYN 91] LYNCH, Clifford A. The Z39.50 Information Retrieval Protocol: An Overview and Status Report. **Computer Communication Review**, New York, v. 21, p. 58–70, Jan. 1991.

[MAS 90] MASON, Tony; BROWN, Doug. **Lex & YACC.** Sebastopol : O'Reilly & Associates, c1990. 216 p.

[MIL 92] MILLER, Bradford W. **The RHET Plan Recognition System Version 1.0.** Rochester, New York: University of Rochester, Computer Science Department, Oct. 1992. 50p. (Technical Report, 298).

[MIR 86] MIRANDA, Cláudio Silva. Sistema SINE. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 3., nov. 1986, Rio de Janeiro. **Anais...** Rio de Janeiro: IME, 1986. p. 204–213.

[MOU 92] MOUTA, Fernando Augusto; OLIVEIRA, Eugênio da Costa. Adaptative Human-Computer Interface for a Community of Coopera-

ting Intelligent System. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 9., set. 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p. 131-143.

[NAT 88] NATIONAL INFORMATION STANDARDS ORGANIZATION (NISO). **American National Standard Z39.50, Information Retrieval Service Definition and Protocol Specifications for Library Applications.** New Brunswick, NJ : Transaction Publishers, 1988.

[NEU 89] NEUMAN, B. Clifford. **The Virtual System Model for Large Distributed Operating Systems.** Seattle, WA: University of Washington, Department of Computer Science, Apr. 1989. (Technical Report, n. 89-01-07).

[NEU 89a] NEUMAN, B. Clifford. The Need for Closure in Large Distributed Systems. **Operating Systems Review**, [S.l.], v. 23, Oct. 1989.

[NEU 92] NEUMAN, B. Clifford. Prospero: A Tool for Organizing Internet Resources. **Electronic Networking: Research, Applications and Policy**, [S.l.], v. 2, Spring 1992.

[NEU 92a] NEUMAN, B. Clifford. The Prospero File System: A Global File System based on the Virtual System Model. **Computing Systems**, [S.l.], v. 5, Fall 1992.

[NEU 92b] NEUMAN, B. Clifford. **The Virtual System Model**: A Scalable Approach to Organizing Large Systems. Seattle, WA: University of Washington, Department of Computer Science and Engineering, 1992. (ph.d. Thessis).

[OLI 88] OLIVEIRA, Carlos A. O Tratamento Automático de Linguagem Natural em Processos de Aquisição de Conhecimento. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 5., nov. 1988, Natal. **Anais...** Natal: UFRN, 1988. p. 84–93.

[OLI 93] OLIVEIRA, Flávio Moreira de; VICCARI, Rosa Maria; COELHO, Helder. Conceptual Distance and Equilibration in an Inteligent Tutoring Situation. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10., out. 1993, Porto Alegre. **Anais...** Porto Alegre: SBC, 1993. p.289–302.

[PAL 93] PALMER, M. S. The Kernel Text Understanding System. **Artificial Intelligence**, Amsterdam, v. 63, p. 17, Oct. 1993.

[PAU 93] PAULO, Alexandre P. G.; FERREIRA, José L. S. SMILE: Aprendizagem de Conceitos Estruturados a Partir de Exemplos. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10., out. 1993, Porto Alegre. **Anais...** Porto Alegre: SBC, 1993. p.211–224.

[PER 87] PEREIRA, Fernando; TWEED, Christopher. **C-Prolog User's Manual**. Edinburgh: University of Edinburgh, Apr. 1987. 72p.

[PER 91] PEREIRA, Fernando C. M.; POLLACK, Martha E. Incremental Interpretation. **Artificial Intelligence**, Amsterdam, v. 50, p. 37–82, June 1991.

[POS 85] POSTEL, J.; REYNOLDS, J. **File Transfer Protocol**. Out. 1985. 69p. (RFC 959).

[QUA 93] QUARESMA, Paulo; LOPES, José Gabriel. Abducção de Planos e de Intenções em Diálogos. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 10., out. 1993, Porto Alegre. **Anais...** Porto Alegre: SBC, 1993. p.377–388.

[RIN 86] RINO, Lúcia Helena Machado. Um Enfoque de uma Interface em Linguagem Natural. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 3., nov. 1986, Rio de Janeiro. **Anais...** Rio de Janeiro: IME, 1986. p. 126–133.

[SCH 86] SCHIEL, Ulrich. Representação e Recuperação de Informação Temporal Incompleta. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 3., nov. 1986, Rio de Janeiro. **Anais...** Rio de Janeiro: IME, 1986. p. 271–283.

[SCH 90] SCHOFFSTALL, Martin L.; YEONG, Wengyik. A Critique of Z39.50 Based on Implementation Experience. **Computer Communication Review**, New York, v. 20, p. 22–29, Apr. 1990.

[SCH 92] SCHWARTZ, Michael F.; EMTAGE, Alan; KAHLE, Brewster et al. A Comparison of Internet Ressource Discovery Approachs. [S.l.]: University of Colorado, Computer Science Department, July 1992. (Technical report CU-CS-601/92).

[SIQ 87] SIQUEIRA, Idméia Semeghini Próspero; VELLOSO, Helmer Martin. Interação Homem-Máquina Diante de Interface em Linguagem Natural com Português Escrito. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 4., out. 1987, Uberlândia. **Anais...** Uberlândia: UFU, 1987. p. 175–186.

[SMI 88] SMITH, S.; STANFILL, C. **An Analysis of the Effects of Data Corruption on Text Retrieval Performance**. [S.l.]: Thinking Machines Corporation, Dec. 1988. (Technical Report TMC-68).

[STR 92] STROGULSDKI, Heitor. **Uma Proposta de Modelagem de Usuários para Interfaces Inteligentes**. Porto Alegre: CPGCC da UFRGS, 1992. 112p. (Dissertação de Mestrado).

[SUN 87] SUN MICROSYSTEMS. **XDR**: External Data Representation standard. June 1987. 20p. (RFC 1014).

[SUN 88] SUN MICROSYSTEMS. **C Programmer's Guide**. Mountain View: Sun Microsystems, 1988. 121p.

[SUN 88a] SUN MICROSYSTEMS. **RPC**: Remote Procedure Call Protocol specification version 2. June 1988. 25p. (RFC 1057).

[SUN 88b] SUN MICROSYSTEMS. **RPC**: Remote Procedure Call Protocol specification. Apr. 1988. 24p. (RFC 1050).

[SUN 89] SUN MICROSYSTEMS. **Sun C++ Programmer's Guide**. Mountain View : Sun Microsystems, 1989. 284 p.

[SUN 89a] SUN MICROSYSTEMS. **C Programmer's Guide**. Mountain View: Sun Microsystems, 1989. 89p.

[SUN 90] SUN MICROSYSTEMS. **Network Programming Guide**. Mountain View: Sun Microsystems, 1990. 353p.

[SUN 90a] SUN MICROSYSTEMS. **Programmer's Language Guides**. Mountain View: Sun Microsystems, 1990. 348p.

[SUN 90b] SUN MICROSYSTEMS. **Sunview Programmer's Guide**. Mountain View: Sun Microsystems, 1990. 524p.

[SUN 90c] SUN MICROSYSTEMS. **XView Version 2 Reference Manual**: Converting SunView Applications. Mountain View : Sun Microsystems, c1990. 157p.

[SUN 90d] SUN MICROSYSTEMS. **Sunview User's Guide**. Mountain View: Sun Microsystems, 1990. 208p.

[TAN 89] TANENBAU, Andrew S. **Computer Networks**. New York: Prentice-Hall International, 1989. 658p.

[TRA 92] TRAUM, David R.; HINKELMAN, Elizabeth A. **Conversation Acts in Task-Oriented Stoken Dialogue**. Rochester, New York: University of Rochester, Computer Science Department, July 1992. 28p. (Technical Report, 425).

[VAS 92] VASCO, João José P. Furtado; MONGIOVI, Giuseppe; CIME FILHO, Walfredo da Costa. A4 - Ambiente de Apoio a Aquisição Automática de Conhecimento. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 9., set. 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: SBC, 1992. p. 186–189.

[VIC 85] VICCARI, Rosa Maria. **Sistema para Instrução Assistido por Computador em Inteligência Artificial.** Porto Alegre: CPGCC da UFRGS, 1985. 146p. (Dissertação de Mestrado).

[VIC 85a] VICCARI, Rosa Maria. Diálogo em Sistemas que Interagem em Língua Natural: um Exemplo Prático. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 2., nov. 1985, São José dos Campos. **Anais...** São José dos Campos: INPE, 1985. p. 167–172.

[WER 91] WERMELINGER, Michel; LOPES, José Gabriel. Uma Ferramenta para a Aquisição e Representação de Conhecimento Baseada em Grafos Conceptuais. In: SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL, 8., nov. 1991, Brasília. **Anais...** Brasília: SBC, 1991. p. 287–294.

[WIN 84] WINSTON, P. H. **Artificial Intelligence.** Reading, MA: Addison-Wesley, 1984. 524p.

Informática UFRGS

SDIP: Um Ambiente Inteligente para a Localização de Informação na Internet

por

Luis Fernando Nunes Fernandez

Dissertação apresentada aos Senhores:

Prof. Dr. José Palazzo Moreira de Oliveira

Profa. Dra. Maria Janilce Bosquiroli Almeida

Prof. Dr. Michael A. Stanton (UFF)

Vista e permitida a impressão.
Porto Alegre, 03/11/95.

Profa. Dra. Liane Margarida Rockenbach Tarouco,
Orientador.

Prof José Palazzo Moreira de Oliveira
Coordenador do Curso de Pós Graduação
em Ciência da Computação - CPGCC
Instituto de Informática - UFRGS