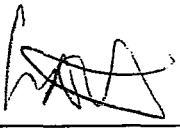
ASSISTENTE INTELIGENTE PARA EXTRAÇÃO DE ELEMENTOS ORIENTADOS A OBJETO DE DISCURSO

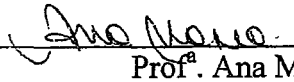
Alberto Tavares da Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

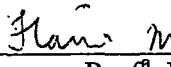
Aprovada por:



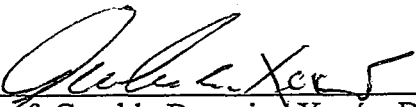
Prof. Luis Alfredo Vidal de Carvalho, D.Sc.



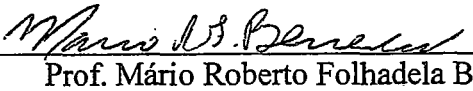
Profª. Ana Maria de Carvalho Moura, Dr.Ing.



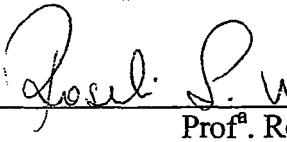
Profª. Flávia Maria Santoro, D.Sc.



Prof. Geraldo Bonorino Xexéo, D.Sc.



Prof. Mário Roberto Folhadela Benevides, Ph.D.



Profª. Roseli Suzi Wedemann, D.Sc.

RIO DE JANEIRO, RJ, BRASIL

FEVEREIRO DE 2008

SILVA, ALBERTO TAVARES DA

Assistente Inteligente para Extração de Elementos Orientados a Objeto de Discurso [Rio de Janeiro] 2008

IX, 213 p., 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2008)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Processamento de Linguagem Natural
2. Lingüística Computacional
3. Lógica Semântica

I. COPPE/UFRJ II. Título (série)

Agradecimentos

A Deus, por mais esse privilégio.

Ao amigo e orientador Luis Alfredo, pela oportunidade de desenvolver esta tese de doutorado, particularmente pelo apoio e confiança em situações diversas durante os últimos cinco anos.

A minha família, Wilma, Eduardo e Marcelo, pela compreensão frente à dedicação exigida pelo doutorado.

Aos meus pais, pela amizade, exemplo e educação.

A minha irmã Olga, pela revisão final do texto.

Ao professor Paúl Ruiz, pelo inestimável auxílio na programação C++.

Aos professores Cícero Roberto Garcez, Maria Claudia Reis Cavalcanti e Ricardo Choren Noya, da Pós-Graduação da Seção de Sistemas e Computação do Instituto Militar de Engenharia, pelo apoio durante a realização do estudo empírico.

Às professoras Ana Maria de Carvalho Moura, Flávia Maria Santoro e Roseli Suzi Wedemann, bem como aos professores Geraldo Bonorino Xexéo e Mário Roberto Folhadela Benevides, por terem participado da minha banca de doutorado.

Aos professores e funcionários da COPPE, minha admiração pelo elevado nível de dedicação e competência.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ASSISTENTE INTELIGENTE PARA EXTRAÇÃO DE ELEMENTOS ORIENTADOS A OBJETO DE DISCURSO

Alberto Tavares da Silva

Fevereiro/2008

Orientador: Luis Alfredo Vidal de Carvalho
Programa: Engenharia de Sistemas e Computação

Este trabalho define um assistente inteligente, i.e., um sistema baseado em conhecimento, que permite a extração de elementos orientados a objeto de textos, ou discursos, em linguagem natural, tendo como principais funcionalidades os analisadores sintático e semântico. O primeiro analisador implementa uma gramática de estrutura sintagmática, processando cada sentença do discurso como entrada e gerando, como saída, os denominados constituintes sintáticos, i.e., estruturas sintáticas e categorias gramaticais, dos vocábulos. O segundo analisador implementa uma técnica lingüística baseada em lógica semântica cujo objetivo é derivar representações de significado a partir de superfícies naturais analisadas sintaticamente com base no reconhecimento de uma conexão entre forma lingüística e significado, permitindo a extração dos elementos orientados a objeto, i.e., classes, atributos, associações e multiplicidades.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

INTELIGENT ASSISTANT FOR EXTRACTION OF OBJECT-ORIENTED ELEMENTS FROM DISCOURSE

Alberto Tavares da Silva

Fevereiro/2008

Advisor: Luis Alfredo Vidal de Carvalho
Department: Computer Science and System Engineering

This thesis defines an intelligent assistant, i.e., a knowledge-based system, which enables eliciting object-oriented elements from a natural language text, or discourse, whose main functionalities are the syntactic and semantic parsers. The first one implements a phrase-structure grammar, taking each sentence of the discourse as input and rendering, as output, the so-called syntactic constituents, i.e., syntactic structures and word classes of the lexicons. The second one implements a linguistic technique based on logical semantics with the purpose of deriving representations of meaning from a syntactically analyzed natural surface based on the recognition of a connection between linguistic form and meaning. The referred technique allows extracting the object-oriented elements, i.e., classes, attributes, associations and multiplicities.

ÍNDICE

Capítulo 1

Introdução

1.1 Contexto

A visualização de um sistema computacional por meio de modelos orientados a objeto é uma melhor prática da Engenharia de Software, pois os modelos representam uma abstração de alguma coisa com o propósito de entendê-la antes de construí-la (RUMBAUGH e BLAHA, 2005). O processo de construção dos referidos modelos é denominado de *modelagem orientada a objeto.* Transversalmente ao conceito de modelagem, existe o processo de desenvolvimento de software, cujas atividades básicas incluem a análise, o projeto, a codificação e o teste de software. A relação de um modelo com as atividades do referido processo não tem, obrigatoriamente, a cardinalidade de um para um, pois um determinado modelo pode ser composto por um ou vários diagramas estáticos e/ou dinâmicos. Na maioria das metodologias orientadas a objeto os modelos são evolutivos, ou seja, um modelo pode iniciar na fase de análise e estar completo na fase de projeto.

A Engenharia de Requisitos, que é um campo de estudo da Engenharia de Software, é o nome dado ao conjunto estruturado de atividades que auxiliam no entendimento dos requisitos requeridos pelos *stakeholders*, ou seja, pessoas que direta ou indiretamente utilizam o sistema, ou a informação por ele gerada, e que documentam as especificações do sistema (SOMMERVILLE, 2005). A Engenharia de Requisitos faz parte do processo de desenvolvimento de software, cujas atividades incluem a elicitação, a modelagem, a validação e a verificação (HOFMAN e LEHNER, 2001). Cabe ressaltar que um determinado modelo gerado na Engenharia de Requisitos pode ser apenas um diagrama ou um modelo na fase de análise do processo de desenvolvimento de software, podendo o mesmo modelo evoluir na fase de projeto. O escopo da Engenharia de Requisitos é a fase

de análise, cuja definição formal pelo corpo de conhecimento da engenharia de software é de Engenharia de Requisitos de Software, que consiste em entender os requisitos por meio da construção de modelos com o objetivo de especificar "o que" é necessário fazer e não "o como" fazer (HILBURN *et al.*, 1999). Uma especificação precisa é um fator chave no desenvolvimento de um software (KOF, 2004).

A análise inicia com as declarações do problema geradas durante a concepção do sistema e, no caso de serem as mesmas incompletas e informais, cabe à análise torná-las mais precisas, bem como expor ambigüidades e inconsistências. É prática comum, também no início da análise, a ocorrência de entrevistas com o usuário ou o fornecimento pelo mesmo de documentos disponíveis dentro do referido domínio, o que inclui processos de negócios, tarefas do usuário, informações sobre sistemas existentes e outros. A identificação de requisitos é uma atividade essencialmente interativa (HICKEY e DAVIS, 2002), pois somente o usuário conhece o problema, de modo que a linguagem natural é a forma mais "natural" que o mesmo tem para se expressar e, principalmente, comunicar-se com o analista. A partir do entendimento do mundo real, o analista deve abstrair os aspectos essenciais em um modelo, sendo este uma representação precisa e concisa do problema que permite responder questionamentos e a construção de uma solução (RUMBAUGH e BLAHA, 2005).

Na última década, a adoção da tecnologia orientada a objeto tem crescido a ponto de tornar-se, nos dias atuais, a tecnologia dominante no desenvolvimento de software (OMG-I, 2004). A despeito das vantagens que a tecnologia orientada a objeto pode prover na comunidade de desenvolvimento de software e seus usuários, os problemas fundamentais associados com as tarefas de identificação de classes, atributos, associações e multiplicidades continuam. As referidas tarefas são usualmente processadas manualmente e dirigidas por heurísticas que o analista adquire por meio de experiência. Ainda nos dias atuais, de acordo com Overmyer *et al.* (2001), as ferramentas básicas para identificação e refinamento de elementos de um modelo conceitual ainda são o lápis e o papel, cujos resultados são posteriormente transferidos para uma ferramenta CASE, o que caracteriza um vazio de automação entre a especificação de requisitos em linguagem natural e a respectiva modelagem conceitual.

A tarefa de construir o denominado modelo conceitual, ou esquema conceitual, é problemática em virtude da forma de abstração e conceitualização da parte relevante do domínio da aplicação a partir dos requisitos. Em muitos casos, analistas são capazes de usar corretamente os conceitos de um modelo conceitual, mas tem dificuldade em abstrair

a realidade de forma a representá-la por meio dos referidos conceitos. Justifica-se essa dificuldade por ser a análise um processo cognitivo, onde a elicitação de requisitos é uma atividade baseada em conhecimento dependente de domínio e a geração de modelos, ou modelagem, uma atividade com conhecimento dependente de modelo. O suporte automatizado ao processo de Engenharia de Requisitos pode melhor refletir o comportamento de resolver problemas de analistas experientes, o que requer identificar, entender e formalizar os mecanismos cognitivos que permitem ao analista abstrair a realidade e representá-la por meio de conceitos e diagnosticá-lo de acordo com o ponto de vista do usuário (ROLLAND e PROIX, 1992).

Segundo Luisa *et al.* (2003), a aplicação de um sistema de processamento de linguagem natural automatizado no suporte aos sistemas de desenvolvimento de software, em geral, e na análise de requisitos, em particular, pode auxiliar o analista na: concentração do problema mais do que na modelagem; interação com outros atores; consideração de vários tipos de requisitos; na definição da rastreabilidade dos requisitos; e no gerenciamento mais eficiente dos problemas de mudança de requisitos.

Considerando que a Engenharia de Requisitos tem, independentemente da técnica de elicitação aplicada, como fonte principal de especificação de requisitos os documentos escritos em linguagem natural e sendo a mesma fortemente dependente da abstração, pode-se concluir que a referida engenharia pode ter um destacado suporte de técnicas de inteligência artificial, tais como, processamento de linguagem natural, representação do conhecimento, raciocínio automatizado, lingüística computacional e programação lógica. Para enfatizar a aplicação da inteligência artificial como ferramenta de apoio à atividade da Engenharia de Requisitos, no caso a geração de modelo conceitual, se destaca a conceituação de Bellman apresentada por Russel e Norvig (2004) sobre inteligência artificial:

> *"(Automatização de) atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado..."*.

1.2 Objetivos

O objetivo geral da presente pesquisa de doutorado é automatizar a geração de modelo conceitual orientado a objeto a partir de especificações de requisitos em linguagem natural. O referido objetivo inclui a construção de um assistente inteligente, i.e., uma ferramenta

automatizada com suporte baseado em conhecimento, denominado de *Assistente Inteligente para Extração de Elementos Orientados a Objeto de Discurso (ASIEEOOD)*. A figura 1.1 apresenta uma visão geral do problema.

Figura 1.1 : Esquema geral do tema de pesquisa proposto.

Os objetivos específicos incluem:

• aplicar métodos da engenharia de software integrados com técnicas da inteligência artificial;

• desenvolver uma técnica lingüística a ser implementada pelo *ASIEEOOD;*

• considerar as atividades de elicitação, modelagem e validação do processo da Engenharia de Requisitos, de acordo com Hoffman e Lehner (2001), não sendo considerada a atividade de verificação;

• gerar o modelo conceitual orientado a objeto de acordo com a notação definida pelo metamodelo diagrama de classe da *Unified Modeling Language* (UML), considerando que os elementos orientados a objeto incluem as classes, os atributos, as associações e as multiplicidades;

• definir uma arquitetura para o sistema de PLN, i.e., o *ASIEEOOD*;

• permitir o processamento de qualquer linguagem natural que possua estruturas sintagmáticas semelhantes ao português, e.g., inglês.

A presente pesquisa, de acordo com os objetivos apresentados, inclui as seguintes áreas de conhecimento: a Inteligência Artificial e a Engenharia de Software. A figura 1.2 detalha os respectivos campos de estudo a serem desenvolvidos na presente proposta.

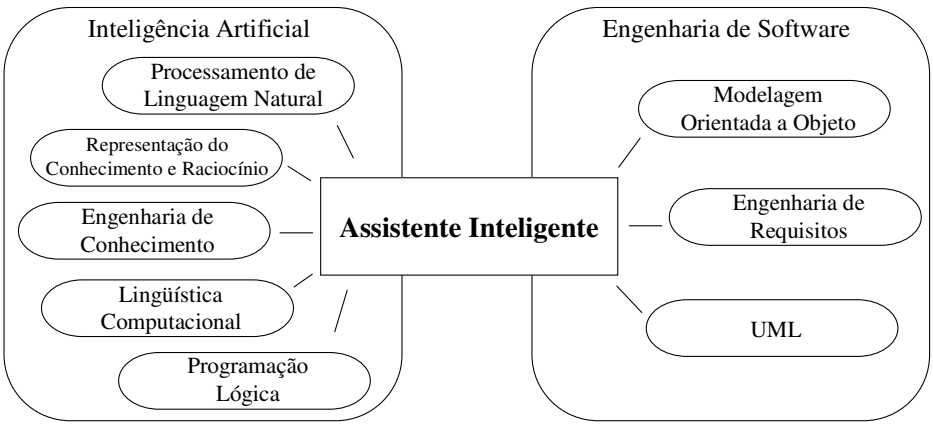


Figura 1.2: Contexto da tese.

1.3 Justificativas

Em Luisa *et al.* (2003), depois de levantados os requisitos de um sistema de processamento de linguagem natural, foi elaborada uma pesquisa eletrônica via internet e enviada a diversas empresas nos Estados Unidos e Europa, onde se pode destacar, para o presente estudo, três perguntas: (i) Qual seria a coisa mais útil que permitiria uma melhora global de eficiência do seu dia-dia? (ii) Quais as duas coisas em seu trabalho que você gostaria de fazer mais eficientemente? (iii) Qual é o nível da terminologia nos documentos de requisitos? Um aspecto importante da presente pesquisa está ligado à relevância do assunto no contexto da Engenharia de Software, ou seja, o papel da linguagem natural como meio de expressar requisitos. Na primeira pergunta, a resposta mais selecionada foi a automação (cerca de 64%). Na segunda, a identificação de requisitos foi o tópico de maior percentual (cerca de 46%), seguido por teste de software (cerca de 34%) e por modelo de requisitos de usuários(cerca de 33%). Na terceira, o item "documentos prévios expressos em linguagem natural comum" teve um alto índice de seleção, 79%. Tais questionamentos ratificam a importância do processo de Engenharia de Requisitos como atividade crucial no desenvolvimento de sistemas e que a aplicação de técnicas lingüísticas em uma ferramenta automatizada poderá gerar um suporte importante a este processo.

A necessidade de aperfeiçoar o processo de software levou a uma aproximação entre as áreas de Engenharia de Software e Inteligência Artificial. Um número crescente de pesquisadores tem usado técnicas de inteligência artificial para apoiar engenheiros de software na realização de suas tarefas, particularmente por meio de assistentes inteligentes que são ferramentas que oferecem suporte baseado em conhecimento para atividades do processo de software (FALBO, 1998).

Um outro aspecto importante a justificar é a adoção da notação do diagrama de classe da UML para o modelo conceitual orientado a objeto a ser gerado. De acordo com o OMG-I (2004), a UML tem emergido como a linguagem de modelagem dominante na indústria do software, pois tem sido aplicada com sucesso em diversos domínios, tais como, saúde, finanças, aeroespaço, comércio eletrônico.

1.4 Metodologia

A hipótese para solução do problema proposto inclui o desenvolvimento de uma técnica lingüística que permita extrair conceitos orientados a objeto de sentenças em

linguagem natural baseado nos constituintes sintáticos dos vocábulos do discurso, i.e., estrutura sintática e categoria gramatical, independentemente do significado da palavra propriamente dito.

No contexto da Engenharia de Requisitos, a elicitação de requisitos ocorre a partir de documentos escritos em linguagem natural, a modelagem, a partir dos elementos orientados a objeto elicitados dos referidos documentos por meio da técnica lingüística citada e a validação, por inspeção visual em um editor gráfico.

Como requisitos lingüísticos básicos destacam-se a definição de um sistema de gramática, de uma linguagem de representação do conhecimento, de uma estrutura de representação do conhecimento e de níveis lingüísticos.

O sistema de gramática proposto é composto pela *gramática de estrutura sintagmática*, ou *PSG-grammar (PSG)*, como o algoritmo formal que permite a geração das seqüências gramaticais (CHOMSKY, 2002) e a *forma de palavra* como o método de reconhecimento de palavra automático, permitindo a implementação de um algoritmo de reconhecimento simples por meio da busca de palavra completa como chave no dicionário analisado (HAUSSER, 2001).

A linguagem de representação do conhecimento adotada é PROLOG, por ser uma linguagem de programação lógica usada, entre outras aplicações, em análise de linguagem natural (RUSSEL e NORVIG, 2004). O nível de significado declarativo da mesma permite informar ao computador o que é verdade e, a partir das referidas verdades, obter conclusões.

A estrutura de representação do conhecimento, i.e., a rede semântica, adotada é a notação do diagrama de classe da *UML.* A referida escolha justifica-se na premissa de que a rede semântica é um sistema especialmente projetado para organizar e raciocinar com categorias, oferecendo auxílio gráfico para visualização de uma base de conhecimento (RUSSEL e NORVIG, 2004).

Os níveis lingüísticos considerados no presente estudo incluem o morfológico, o sintático e o semântico, tal qual ilustra a figura 1.3, onde ocorrem as respectivas análises.

Figura 1.3: Níveis lingüísticos adotados.

A análise morfológica tem como finalidade implementar as funcionalidades relacionadas com a fragmentação do discurso em sentenças, a busca de palavras compostas e a realização da etiquetação morfológica dos vocábulos.

A análise sintática objetiva a verificação da correção da sintaxe das sentenças analisadas do discurso e a geração da estrutura sintática plana da sentença analisada para posterior extração dos constituintes sintáticos, i.e., a estrutura sintática e a categoria gramatical, dos vocábulos que compõem as referidas sentenças.

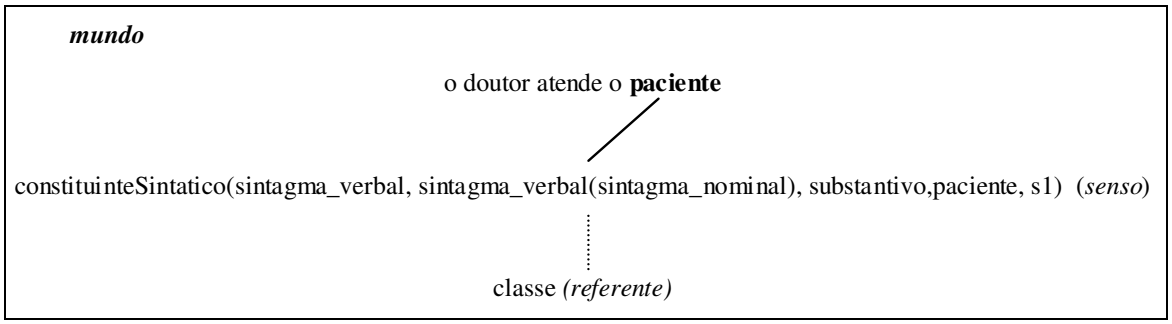


Figura 1.4: Teoria do significado de Frege.

Alinhado com a premissa de Gamut (1991) de que *o núcleo de uma teoria lingüística está em sua teoria semântica*, o presente trabalho tem como núcleo uma teoria semântica fundamentada na teoria estabelecida por Kamp e Reyle (1993), no método indireto de interpretação semântica proposto por Montague para o *The Proper Treatment of Quantification in Ordinary English*, na interpretação semântica baseada na *Teoria do Significado de Frege*, na semântica lógica, ou semântica da teoria do modelo, de acordo com a visão lógica semântica apresentada por Gamut (1991) e Hausser (2001), na não-composicionalidade da gramática e na semântica lógica orientada a discurso. Dos fundamentos apresentados, destaca-se a *Teoria do Significado de Frege* ilustrada na figura 1.4. Em síntese, a superfície da linguagem natural é traduzida para uma linguagem lógica na forma de um predicado cujos argumentos mais significativos estão relacionados com a posição sintática do vocábulo na sentença e a categoria gramatical, sendo o referido predicado a *forma de apresentação* que permite extrair o significado desejado, i.e., o elemento orientado a objeto *classe*, sendo este o *referente*.

A figura 1.5 ilustra o esquema geral do *ASIEEOOD*, sendo o mesmo um sistema baseado em conhecimento com dois atores: o engenheiro de conhecimento e o analista. As principais responsabilidades do engenheiro de conhecimento incluem a criação de regras sintáticas e semânticas que comporão as respectivas bases de conhecimento. No módulo *Editor de Discurso* adota-se o conceito de linguagem controlada, i.e., limitar a estrutura

frasal utilizada pelo analista que realiza a especificação textual do modelo conceitual a fim de permitir uma otimização no processamento computacional do discurso analisado, particularmente na redução da quantidade de estruturas sintáticas possíveis. No *Editor de Conhecimento* são editadas as regras sintáticas e semânticas. As regras semânticas implementam as heurísticas que, por meio de vínculos lógicos com os fatos sintáticos, permitem a extração dos *referentes* do discurso, i.e., os elementos orientados a objeto. O *Editor Gráfico* gera o modelo conceitual a partir dos fatos semânticos, e.g., *classe(médico),* gerados pelo motor de inferência da base de conhecimento.

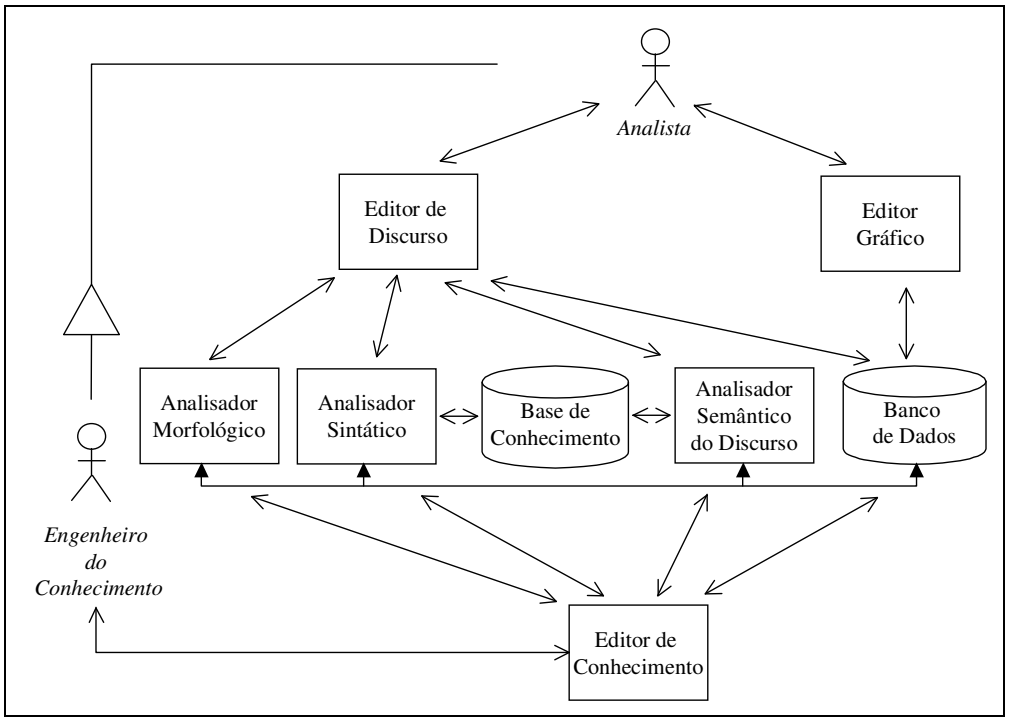


Figura 1.5: Esquema geral do assistente inteligente.

1.5 Estrutura da Tese

O restante desta tese de doutorado está estruturado da forma descrita a seguir.

No segundo capítulo são apresentados os conceitos relacionados com a área de conhecimento da inteligência artificial, particularmente no contexto do processamento de linguagem natural, da lingüística computacional, da representação do conhecimento e raciocínio, da engenharia de conhecimento e da programação lógica.

No terceiro capítulo são apresentados os campos de estudo relacionados com a área de conhecimento da Engenharia de Software, que incluem a modelagem orientada a objeto, a engenharia de requisitos e a UML.

No quarto capítulo são apresentadas técnicas lingüísticas com o objetivo de dissertar sobre trabalhos relacionados com a presente pesquisa.

No quinto capítulo é apresentada a técnica lingüística proposta, a arquitetura do *ASIEEOOD*.

O sexto capítulo apresenta a avaliação da técnica citada por meio de um estudo empírico composto por dois estudos de caso.

No sétimo capítulo são apresentadas as considerações finais do presente trabalho, as contribuições e as propostas de trabalhos futuros.

Capítulo 2

Processamento de Linguagem Natural

Processamento de Linguagem Natural (PLN) é uma área de pesquisa e de aplicação que explora como um computador pode ser utilizado para entender e manipular textos ou fala em linguagem natural a fim de gerar coisas. Pesquisadores de PLN buscam coletar conhecimento sobre como os humanos entendem e usam a linguagem a fim de que ferramentas e técnicas apropriadas possam ser desenvolvidas e que permitam que sistemas computacionais entendam e manipulem a linguagem natural na execução das tarefas desejadas (CHOWDHURY, 2003). O objetivo principal do PLN é ter a habilidade de usar a linguagem natural tal como os humanos.

Um sistema de PLN deve executar as seguintes funções relacionadas: análise ou interpretação dos dados de entrada, mapeamento em alguma linguagem de representação, raciocínio sobre a interpretação para determinar o conteúdo do que deve ser produzido em resposta ao usuário e, finalmente, gerar a resposta (WEISCHEDEL *et al.*, 1999). No contexto da inteligência artificial, o presente estudo adota o *framework* para o PLN ilustrado na figura 2.1, onde estão definidos os campos de estudo necessários para a solução do problema objeto de pesquisa.

Figura 2.1: *Framework* para PLN.

A linguagem natural serve como meio de comunicação em vez de pura representação. Nuseibeh e Eastebrook (2000) destacam a aplicação da lingüística no suporte automatizado à engenharia de requisitos, tendo em vista ter esta engenharia muito de comunicação e que as referidas ferramentas lingüísticas podem ser aplicadas na elicitação de requisitos.

A seguir serão apresentados os principais conceitos dos campos de estudos ilustrados na figura 2.1 com foco no objeto da presente pesquisa.

2.1 Arquitetura para um Sistema de Processamento de Linguagem Natural

Um sistema de PLN no apoio ao processo de desenvolvimento de software em geral, e na análise de requisitos, em particular, pode auxiliar o analista a concentrar-se sobre o problema mais do que na modelagem, interagir com outros atores, levar em consideração vários tipos de requisitos, determinar a rastreabilidade dos requisitos e permitir um melhor gerenciamento das alterações ocorridas nos mesmos (LUISA *et al.*, 2003).

O PLN inclui, entre outras, as seguintes pesquisas (GAL *et al.*, 1991):

• desenvolver e modelar sistemas lingüísticos, onde são adaptadas as teorias gerais da lingüística de acordo com o objetivo definido, com o grau de sofisticação desejado para o processamento e com as restrições associadas com a automatização;

• conceber e implementar modelos e sistemas de PLN, o que inclui a escolha de uma arquitetura, o relacionamento entre linguagem natural e inteligência artificial, em particular as técnicas de representação do conhecimento, a escolha da linguagem de programação e uma fase de avaliação que permita julgar a qualidade do sistema.

A compreensão da linguagem natural exige a aplicação de níveis de análise a fim de que seja possível a extração de significados do texto ou da fala. Os níveis de análise de processamento de uma linguagem natural são os seguintes: o fonético, o fonológico, o morfológico, o léxico, o sintático, o semântico e o pragmático (GAL *et al.*, 1991). O nível fonético diz respeito à composição do som da fala. O nível fonológico diz respeito como os sons se combinam para formar palavras, grupos de palavras e sentenças, tendo como exemplo de aplicação o reconhecimento de voz. O nível morfológico trata da constituição, o que inclui, entre outros aspectos, a derivação e a composição das palavras. O nível léxico define as categorias das palavras, tais como verbos, substantivos, adjetivos e outros. O nível sintático trata de como as palavras se combinam para formar sentenças corretas. O nível semântico trata do significado das palavras, grupos de palavras e sentenças. O nível

pragmático trata de situar uma sentença completamente em seu contexto, particularmente no que toca ao fato de uma determinada palavra ter significados diferentes para contextos diferentes. Existe, também, o nível de discurso que trata com a estrutura de um texto, i.e., um conjunto de sentenças (KAMP e REYLE, 1993) e (CHOWDHURY, 2003). Um sistema de PLN pode envolver todos ou alguns dos referidos níveis.

Uma especificação da arquitetura de um sistema de PLN pode ser usada para obter, com diferentes níveis de performance, três saídas essenciais: análise sintática, semântica e pragmática, textos em linguagem natural ou artificial e diferentes tipos de documentos estruturados. As principais especificações para um sistema de PLN, cuja arquitetura de uso-geral é ilustrada na figura 2.2, incluem (LUISA *et al.*, 2003):

• facilitar, na fase de elicitação de requisitos, a digitalização dos mesmos, a identificação de ambigüidades e contradições nos documentos que descrevem as necessidades dos usuários e a análise automática de respostas a perguntas ilimitadas, interpretando e classificando os respectivos conteúdos;

• modelar requisitos por meio da extração dos conceitos que podem ser incluídos em um modelo conceitual de acordo com o método de desenvolvimento adotado;

• suportar a validação de requisitos por meio do aproveitamento da funcionalidade de geração de um sistema de PLN em produzir descrições em linguagem natural baseada nas estruturas de representação do conhecimento;

• gerar documentação, tais como relatórios diversos;

• permitir a rastreabilidade por meio de vínculos entre os textos usados e os modelos produzidos;

• restringir a escrita de documentos por meio de uma linguagem controlada, ou seja, com restrições na gramática ou no vocabulário.

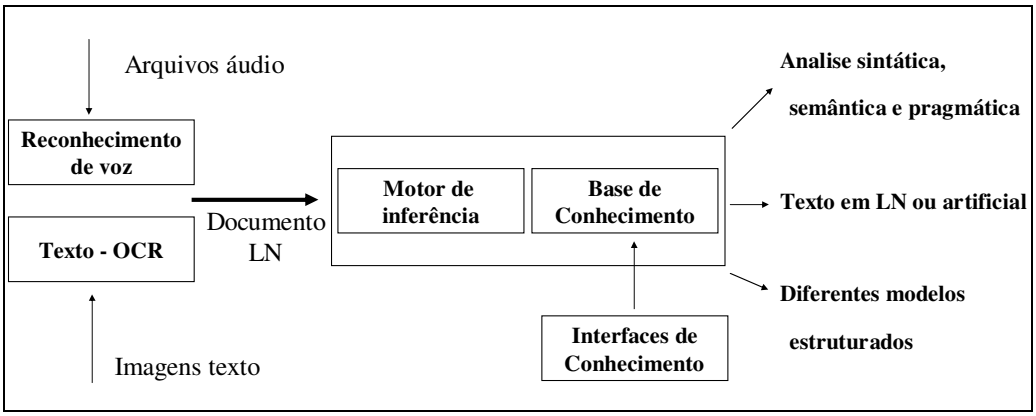


Figura 2.2: Arquitetura de uso-geral para um sistema de PLN.

A figura 2.3 ilustra uma arquitetura, também proposta por Luisa *et al.* (2003), que permite a geração de um modelo conceitual definido pela sintaxe abstrata da UML para o diagrama de classe.

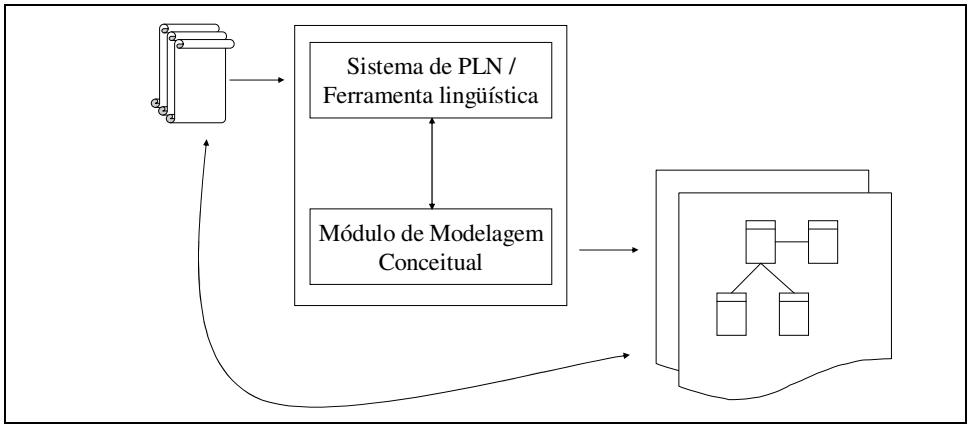


Figura 2.3: Arquitetura de uso-particular para um sistema de PLN.

O sistema LOLITA, *Large-scale, Object-Based, Linguistic Interator, Translator, and Analyser,* é projetado para ser um sistema de propósito geral para PLN que inclui a capacidade de suportar muitas aplicações em múltiplos domínios, tais como, a extração de informação, a tradução, a busca em linguagem natural, o diálogo e um tutorial da língua chinesa. A arquitetura do LOLITA é projetada com um núcleo central e um conjunto de pequenas aplicações. A parte principal do núcleo é a rede semântica, denominada de *Semantic Network* ou *SemNet.* A referida rede semântica é fortemente utilizada durante a fase de análise e os resultados são adicionados à rede. O processo de análise é baseado em regras, não sendo aplicadas técnicas estocásticas, ou outra qualquer, no sistema principal (MORGAN *et al.*, 1993) e (SMITH *et al.*, 1994).

Sowa (2002a) propõe um *framework* para sistemas inteligentes que consiste de uma variedade de componentes especializados junto com linguagens baseadas em lógica que permitem expressar proposições e ações sobre as mesmas. O referido *framework* é modular, flexível e pode ser ajustado a diversas arquiteturas para diferentes tipos de aplicações. O autor apresenta duas técnicas de aquisição de conhecimento que permitem colaborar com a especificação de uma arquitetura de PLN. Na primeira técnica, o sistema é projetado de modo a não permitir nenhuma interação com o usuário, exigindo, portanto, um engenheiro de conhecimento e um especialista de domínio na realização de toda a conversão necessária dos dados de entrada nos respectivos formatos de saída. Na segunda técnica, a indicada pelo autor e ilustrada na figura 2.4, o sistema é projetado como uma ferramenta de extração de conhecimento simples de modo que permita a extração do conhecimento por somente um editor humano.

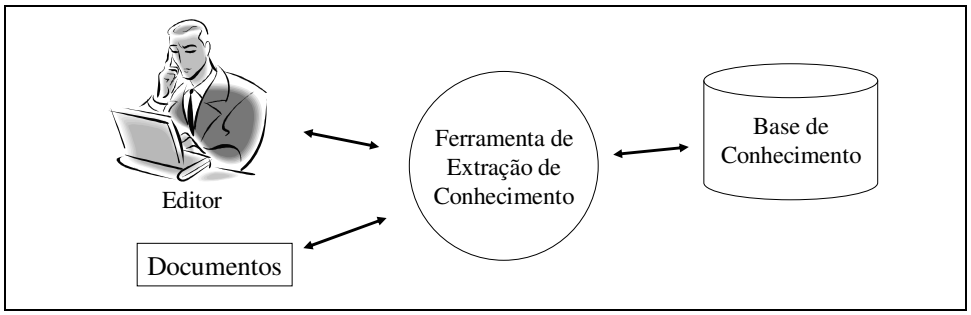


Figura 2.4: Arquitetura básica para uma ferramenta de extração de conhecimento.

2.2 Lingüística Computacional

A linguagem é um conjunto de seqüências de palavras (HAUSSER, 2001). Entretanto, nem toda seqüência de palavras faz parte de uma linguagem, havendo uma distinção entre sentenças gramaticais e as agramaticais, sendo a gramática o dispositivo que permite identificar a referida distinção (PARTEE *et al.*, 1990). De acordo com Carnie (2002), ***L***inguagem é a habilidade do ser humano falar alguma ***l***inguagem, e.g., português e inglês, ou seja, ***l***inguagem é uma instância de ***L***inguagem. A linguagem é um sistema produtivo infinito, i.e., sentenças podem ser produzidas e entendidas sem nunca terem sido ouvidas antes. Outro exemplo da qualidade infinita está relacionado com a denominada *recursão*, o que possibilita que sentenças pertençam a outras sentenças, e.g., *Carlos duvida que[Ana acredita que [João foi para a Europa]]]*.

A lingüística computacional visa o entendimento de processos de linguagens em termos de procedimentos a fim de permitir a sistemas computadorizados a habilidade de gerar e interpretar a linguagem natural, sendo definida como o estudo de sistemas computacionais para o entendimento e geração de linguagem natural (GRISHMAN, 1999). A lingüística computacional combina métodos da gramática tradicional, i.e., que define o conjunto de regras que garantem o bom uso da língua, e da teoria lingüística, i.e., que usa métodos de lógica matemática na descrição de linguagem natural por meio de um sistema de regras formais, com o método de efetivamente verificar hipóteses explícitas de implementar gramáticas formais como programas de computador eficientes e testá-las automaticamente em grandes quantidades de dados reais (HAUSSER, 2001).

2.2.1 Gramática

À gramática cabe definir o conjunto de regras que garantem o bom uso da língua ou o conjunto de regras que permitem organizar as palavras de uma língua em frases

(INFANTE, 2001). A gramática não consiste na enumeração de todas as seqüências possíveis de palavras, que se apresentam em número infinito em todas as linguagens, mas sim na formulação de regras gerais (SOUZA E SILVA e KOCH, 2004).

A gramática tradicional recobre um conjunto de esforços que se destinam a explicar a natureza da linguagem, descrever a estrutura e funcionamento das línguas e regulamentar seu uso consoante padrões, quer lógicos, quer literários de expressão, tendo como partes fundamentais de qualquer construção centrada no verbo o *sujeito* e o *predicado* (AZEREDO, 2003).

A gramática de estrutura sintagmática, ou de constituintes, surgiu em conseqüência do movimento estruturalista impulsionado quase simultaneamente nos EUA e na Europa, cabendo destacado papel ao americano Chomsky. A referida gramática está fundamentada na descrição sintática baseada na identificação dos vários tipos de sintagmas, e.g., sintagma nominal, e na formulação de regras que ordenam as palavras no interior dos sintagmas (AZEREDO, 2003).

Um conceito chave em toda análise gramatical é o da *frase* que é a expressão verbal de um pensamento, ou seja, todo enunciado suficiente por si mesmo para estabelecer comunicação, sendo que toda frase de uma língua consiste em uma organização de elementos lingüísticos agrupados segundo certos princípios, que a caracterizam como uma estrutura; um conjunto de frases, por vezes unitário, forma o texto, que constitui uma unidade semântico-pragmática: semântica, por veicular significados distintos daquele de cada uma das frases tomada isoladamente; e pragmática, por constituir um ato de comunicação, destinado a atuar sobre o ouvinte/leitor de determinado modo (SOUZA E SILVA e KOCH, 2004). Azeredo (2003) define texto como uma entidade pertencente ao domínio do ato verbal de comunicação, isto é, do *discurso*, sendo a frase o menor texto possível; segundo o mesmo autor, as frases que representam analiticamente um conteúdo fazem-no segundo os princípios sintáticos da língua, isto é, o sistema de unidades hierarquizadas - palavras, sintagmas e orações – e relacionadas umas às outras por um conjunto de mecanismos formais.

O objetivo fundamental da análise lingüística de uma linguagem *L* é separar as seqüências gramaticais, que são as sentenças de *L*, das seqüências agramaticais, que não são sentenças de *L*, e estudar a estrutura das seqüências gramaticais. Uma gramática de uma linguagem *L* é essencialmente uma teoria de *L*, ou seja, tal qual uma teoria científica, uma gramática é baseada em um *corpus* de expressões, ou observações, e contém um

conjunto de regras que expressam relações estruturais no meio das sentenças do referido *corpus* (CHOMSKY, 2002).

2.2.1.1 Gramática Gerativa

A base conceitual da gramática gerativa é o monóide livre que, por conter todas as possíveis seqüências de palavras em uma dada linguagem, necessita de critérios formais que permitam distinguir as seqüências bem-formadas das mal-formadas. Como exemplo, a linguagem artificial $a^k b^k$ possui as seguintes expressões bem-formadas: *ab, aabb, aaabbb* etc. Os referidos critérios são denominados, na teoria lingüística, de descrições estruturais e especificados por meio de sistemas de regras recursivas os quais são usados pela lógica e denominados de *gramática gerativa* (HAUSSER, 2001).

A premissa básica da gramática gerativa é que as sentenças são geradas por um subconsciente conjunto de procedimentos que fazem parte de nossas mentes. O objetivo da sintaxe é justamente modelar os referidos procedimentos por meio de regras gramaticais que definem a ordem de colocação de uma palavra em uma determinada sentença, ou seja, as regras permitem "gerar" as sentenças bem-formadas de uma linguagem, logo o nome *gramática gerativa* (CARNIE, 2002).

A metodologia gerativa aplicada à análise de linguagem natural não está limitada a um determinado formalismo de gramática. No contexto da teoria formal da linguagem, destacam-se os seguintes formalismos elementares: gramática de estrutura sintagmática, ou *PSG-grammar*, gramática de categoria, ou *C-grammar*, e gramática associativa à esquerda, ou *LA-grammar* (HAUSSER, 2001).

2.2.1.2 Gramática de Estrutura Sintagmática

A Gramática de Estrutura Sintagmática, ou *PSG (phrase structure grammar),* foi apresentada em 1957 por Chomsky. Posteriormente, o referido autor e outros desenvolveram uma série de formalismos de derivação inicialmente chamados de gramática transformacional (HAUSSER, 2001).

O *sintagma* (*constituent* em inglês) é o principal conceito na teoria sintática baseada na *PSG*. Por definição, sintagma consiste num grupo de elementos que constituem uma unidade significativa dentro da oração e que mantêm entre si relações de dependência e de ordem. Organizam-se em torno de um elemento fundamental denominado núcleo, que pode, por si só, constituir um sintagma, e.g., o núcleo de um sintagma verbal é um verbo. Os sintagmas podem estar encaixados um dentro de outro formando novos sintagmas, i.e.,

uma estrutura hierárquica, tal qual ilustra, em forma de árvore, a figura 2.5 (AZEREDO, 2003).

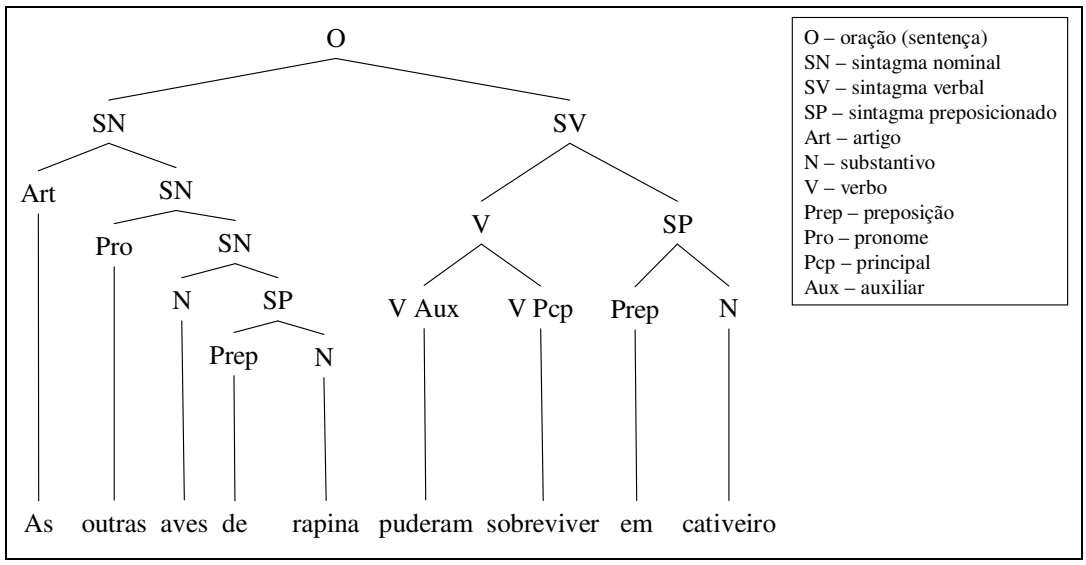


Figura 2.5: Exemplo de estrutura de árvore.

A figura 2.5 ilustra que o sintagma oração *O* é composto de dois sintagmas, o sintagma nominal (SN) e o sintagma verbal (SV). O SN contém um determinante, ou artigo, e um SN. O SV é composto por um verbo e um sintagma preposicionado (SP). O SP contém uma preposição e um substantivo. O segundo SP contém um determinante, ou artigo, e um substantivo. A estrutura é generalizada por meio de *regras de estrutura sintagmática*, de modo que uma árvore de estrutura sintática de uma sentença é gerada a partir das referidas regras. A figura 2.5 possui as seguintes regras de estrutura sintagmática:

O → SN, SV

SN → Art, SN

SN → Pro, SN

SN → N, SP

SV → V, SP

V → VAux, VPcp

SP → Prep, N

A descrição sintática baseada na identificação dos vários tipos de sintagmas e na formulação das regras que ordenam as palavras no interior dos sintagmas caracteriza a *PSG* como uma gramática que permite descrever a estrutura da oração em constituintes, i.e., sintagmas, imediatos, e.g., *O → SN, SV*. A função sintática das referidas unidades passa a ser interpretada unicamente como uma posição estrutural.

A definição algébrica da *PSG* é a seguinte (HAUSSER, 2001) e (GRISHMAN, 1999):

A *PSG* é um quádruplo < V, V_T, S, P> de modo que,

1. V é um conjunto finito de símbolos;

2. V_T é um subconjunto privativo de V, denominado de símbolos terminais;

3. S é um símbolo em V menos V_T, denominado de símbolo inicial;

4. P é um conjunto de regras de reescrita na forma $\alpha \rightarrow \beta$, onde α é um símbolo não-terminal e β é uma seqüência de zero ou mais símbolos terminais e não-terminais.

Os símbolos terminais de V_T incluem as palavras, ou símbolos, da linguagem sendo definida; os símbolos não-terminais, i.e., V menos V_T, são denominados de variáveis. Usualmente, os símbolos terminais são em letras minúsculas e os não-terminais, em maiúsculas. Considerando a figura 2.5, os símbolos não terminais formam o conjunto {O, SN, SV, Art, Pro, N, Prep, V, SP, VAux, VPcp} e os terminais, o conjunto {as, outras, aves, de, rapina, puderam, sobreviver, em, cativeiro}.

De acordo com Souza e Silva e Koch (2004), a gramática de estrutura sintagmática tem como principal limitação atribuir uma única descrição estrutural a cada oração, i.e., não é suficiente para dar conta de certos fatos da língua, tais como: mostrar a relação entre estruturas aparentemente diferentes como (a) e (b); diferenciar estruturas aparentemente idênticas como (c) e (d); desfazer ambigüidades como aquela existente em (e). Azeredo (2003) também apresenta as citadas restrições.

(a) Que eles apresentem propostas é indispensável.

(b) É indispensável que eles apresentem proposta.

(c) O espelho refletiu a sua imagem.

(d) O mestre refletiu um instante.

(e) Marcelo afirmou que Fernando viajou e Sérgio também.

Chomsky (2002) apresenta algumas limitações relativas à *PSG* originalmente apresentada pelo autor, e.g., impossibilidade de formação de novas sentenças por meio do processo de conjunção, descontinuidades com verbos auxiliares, a relação ativo-passiva. Com base nas referidas limitações, o referido autor considera que os fundamentos da estrutura sintagmática são adequados para um determinado núcleo de uma linguagem e que o resto da mesma pode ser derivado por repetidas aplicações de um conjunto apropriado de transformações. A tentativa de estender a referida gramática, a fim de que esta inclua toda a linguagem diretamente, gera a perda da simplicidade de uma *PSG* limitada, bem como o desenvolvimento transformacional.

As referidas limitações levaram Chomsky, em substituição ao modelo sintagmático de análise, propor um modelo com dois níveis de representação estrutural: a *estrutura profunda*, destinada a reunir as informações necessárias à interpretação semântica dos enunciados, e a *estrutura superficial*, que corresponde à forma real do enunciado. Os dois níveis são relacionados por um conjunto de operações gramaticais denominadas de *regras transformacionais* e compõem o nível de representação das estruturas transformacionais (CHOMSKY, 2002) e (AZEREDO, 2003). A Figura 2.6 ilustra uma estrutura profunda da estrutura superficial *o pedestre foi atropelado pela bicicleta*; a estrutura profunda segue a regra de constituição de toda e qualquer frase de uma língua, i.e., $F \rightarrow T + P$, onde F representa a frase, T especifica o tipo de frase (no caso afirmativa e passiva) e P representa a estrutura subjacente, elementar e abstrata dos elementos que compõem a proposição ou oração, i.e., *a bicicleta atropelar o pedestre*. A estrutura profunda permite reduzir o número de estruturas possíveis, representar em ordem direta as relações fundamentais entre os constituintes da oração e ser responsável pela interpretação semântica. Chomsky (2002) afirma que a técnica transformacional tem como objetivo limitar o núcleo de modo que os símbolos terminais existentes nas sentenças básicas sejam derivados por um sistema simples de estrutura sintagmática e possa prover a base por meio da qual todas as sentenças possam ser derivadas por simples transformações.

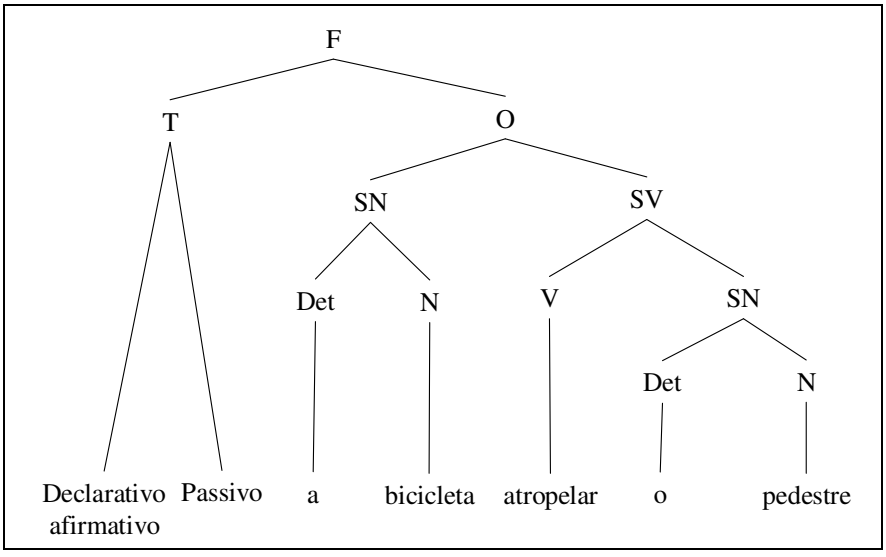


Figura 2.6: Exemplo de estrutura profunda.

A fim de exemplificar a gramática transformacional, a figura 2.6 ilustra a estrutura profunda da frase que representa uma sentença do núcleo da referida gramática. A figura 2.7 ilustra a estrutura obtida após a aplicação de uma transformação passiva, estando inserida também a transformação de afixo. Finalmente, a figura 2.8 ilustra a sentença

resultante em sua forma superficial após a aplicação das regras morfofonêmicas (SOUZA E SILVA e KOCH, 2004).

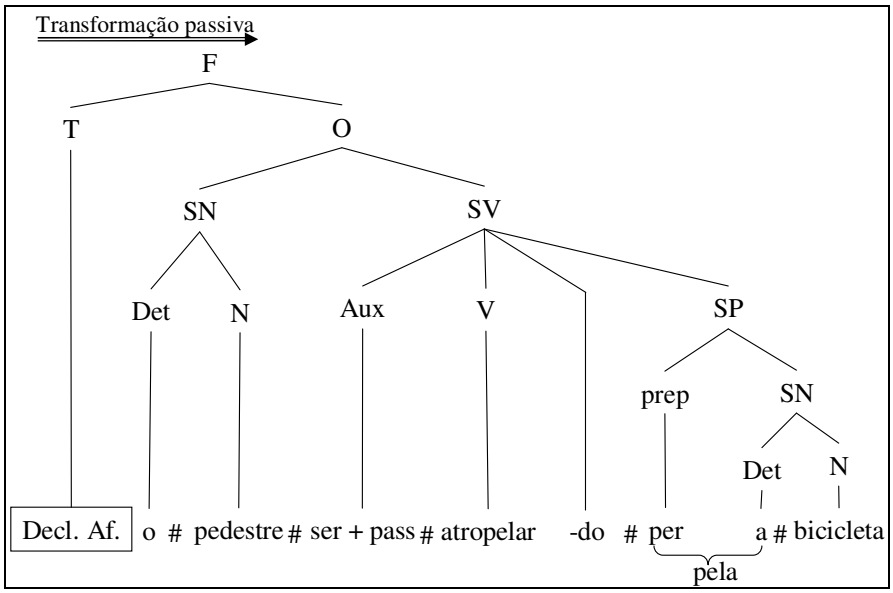


Figura 2.7: Exemplo de transformação passiva.

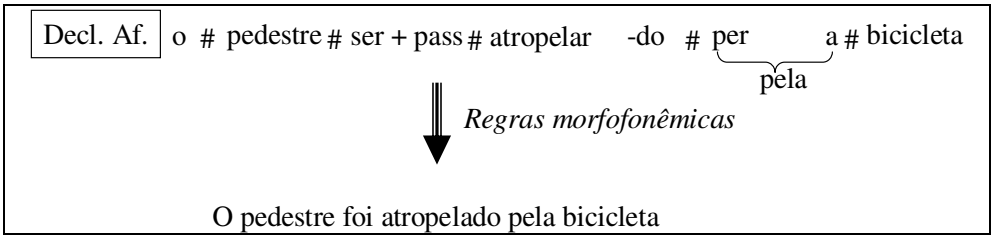


Figura 2.8: Exemplo de aplicação de regras morfofonêmicas.

2.2.1.3 Gramática *LA-grammar*

A denominação da gramática associada à esquerda, i.e., *LA-grammar*, está relacionada com a ordem de derivação conceitual da esquerda para a direita e de baixo para cima, sendo a referida derivação realizada de forma contínua. De acordo com Hausser (2001), a noção formal da estrutura sintagmática, definida com base em estrutura de árvore, não tem aplicabilidade na *LA-grammar*.

Uma *LA-grammar* é especificada por:

i) um léxico *LX*;

ii) um conjunto de estados iniciais ST_s;

iii) uma seqüência de regras r_i, cada uma definida como um par (co_i, rp_i); e

iv) um conjunto final de estados ST_F.

Uma regra associada à esquerda r_i considera uma sentença inicial *ss* e uma próxima palavra *nw* como entrada e tenta uma operação categorial co_i. Se a categoria da entrada

coincide com o padrão de *cat1* e *cat2*, a aplicação da regra r_i ocorre com sucesso e uma saída é produzida. A saída consiste de um par ($ss`$ $rp_i$), onde $ss`$ é a resultante da sentença inicial e rp_i é um pacote de regras contendo todas as regras que podem ser aplicadas após o sucesso da regra r_i. Após uma aplicação positiva de uma regra, o algoritmo obtém a próxima palavra, caso exista, como entrada e aplica o referido pacote de regras. Desta forma, a *LA-grammar* trabalha com os dados de entrada, i.e., as palavras, da esquerda para a direita, encerrando o processamento pela impossibilidade de continuidade gramatical, i.e., entrada agramatical, ou inexistência de uma próxima palavra.

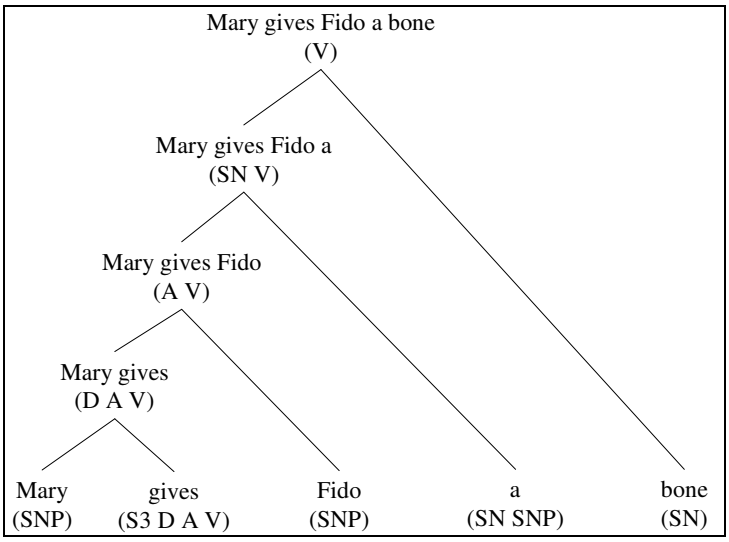


Figura 2.9: Exemplo uma estrutura de derivação *LA-grammar.*

A figura 2.9 ilustra uma aplicação apresentada por Hausser (2001). A categoria *(S3 D A V)*, também denominada de valência, de *gives* indica um verbo (V) que possui um nominativo na terceira pessoa do singular (S3), um dativo (D) e um acusativo (A) como argumentos. A combinação inicial de *Mary* e *gives* é baseada na operação categorial *(SNP) (S3 D A V) => (D A V)*. O segmento de categoria *SNP* (para sintagma nominal singular) de *Mary* cancela o segmento *S3* (para nominativo) da categoria de *gives,* sendo o resultado da primeira combinação uma sentença início da categoria *(D A V)*, indicando uma expressão intermediária que necessita de um dativo (*D)* e de um acusativo (*A)* para tornar-se uma sentença completa. A nova sentença combina com a próxima palavra *Fido* da categoria (*SNP)* que, por sua vez, cancela o primeiro segmento categoria da sentença início (*D A V)*. O resultado da segunda combinação é uma sentença início da categoria (*A V)*, indicando a necessidade de um acusativo (A) de modo a tornar-se completa; a referida sentença combina com a palavra *a* da categoria (*SN SNP)*, sendo a operação categorial *(A V) (SN SNP) => (SN V)*, onde o segmento *SNP* da determinante *a* preenche a posição de valência do acusativo (*A*) , enquanto que o argumento *SN* do determinante é adicionado na categoria

de sentença início, abrindo uma nova posição de valência para um substantivo singular. A referida posição de valência é anulada pela categoria (*SN)* da próxima palavra *bone*. O resultado é uma sentença início da categoria verbo (*V)*, sendo que a falta de posições de valência indica que a expressão está completa.

2.2.1.4 Gramática de Restrições

A Gramática de Restrições (*Constraint Grammar - CG)* foi introduzida por Fred Karlsson em 1990 e formatada pela Escola de Helsinski sendo, conceitualmente, mais orientada à análise de sentença do que à geração de sentenças, ao contrário da *PSG*. De acordo com Bick (2000), a *CG* é reducionista em vez de gerativista, o que lhe torna mais tolerante, ou robusta, em relação ao que a *PSG* considera falha de execução.

As palavras em linguagem natural são, em sua grande maioria e vistas isoladamente, ambíguas em relação à categoria gramatical, inflexão, função sintática e semântica, sendo o contexto da sentença o determinante de como uma palavra deve ser entendida. A *CG* é uma técnica gramatical que visa resolver as referidas ambigüidades por meio do estabelecimento de regras para as possíveis leituras da palavra, descartando, de acordo com o contexto da sentença, as leituras inadequadas. Durante a análise propriamente dita, as regras são compiladas em um programa de computador e as múltiplas ambigüidades representadas por linhas de etiquetas alternativas são reduzidas a apenas uma única linha pelo sistema de regras da *CG*. Uma gramática *CG* madura possui entre 2000 e 3000 regras (BICK, 2000).

A figura 2.10(i) exemplifica as quatro leituras possíveis para a forma de palavra "como", denominadas, de acordo com a terminologia da *CG*, de *cohort.* A figura 2.10 (ii) ilustra uma regra típica da *CG* para eliminação da referida ambigüidade.

i) Entrada de CG (saída do analisador morfológico)

```
"<nunca>"
        "<nunca>" ADV
"<como>"
        "como" <rel> ADV
        "como" <interr> ADV
        "como" KS
        "como" <vt> V PR 1S VFIN
"<peixe>"
        "peixe" N M S
"<$.>"
```

ADV – advérbio
KS – conjunção subordinada
V – verbo
N – substantivo
PR – modo presente
S – singular
M – masculino
1 – 1ª pessoa
VFIN – verbo finito
<rel> - relativo
<interr> - interrogativo
<vt> - mono transitivo

ii) SELECT (VFIN) IF (NOT *-1 VFIN) (NOT *1 VFIN)
[selecione para uma dada forma de palavra a leitura do VFIN se não existe (NOT) – nem à esquerda (*-1) nem à direita (*1) – outra palavra que seja VFIN]

Figura 2.10: Exemplo de aplicação da gramática *CG*.

2.2.2 Níveis lingüísticos

A linguagem é uma propriedade psicológica, ou cognitiva, dos humanos composta pelos seguintes níveis lingüísticos básicos: fonológico, morfológico, sintático, semântico e pragmático. O nível fonológico trata da acústica e da articulação da fala. O nível morfológico, da estrutura de formato da palavra, o que inclui a classificação da palavra de acordo com sua *part of speech (PoS),* i.e., categoria gramatical, e a descrição da estrutura de formato da palavra em termos de inflexão, derivação e composição; em lingüística computacional, morfologia aparece no contexto de reconhecimento automático de formação de palavra por meio da lematização, que relaciona a forma da palavra com sua forma básica, e a categorização, que caracteriza as propriedades morfossintáticas da palavra. O nível sintático trata, basicamente, da composição dos formatos de palavras em sentenças gramaticalmente bem-formadas em termos de regras gramaticais. O nível semântico, do significado dos literais, podendo ser dividido em semântica léxica, que descreve o significado das palavras, e em semântica composicional, que descreve a composição dos significados de acordo com a sintaxe. A pragmática estuda como expressões analisadas gramaticalmente são usadas em relação ao contexto da interpretação (HAUSSER, 2001) e (CARNIE, 2002).

Carnie (2002) denomina os referidos níveis como subsistemas da linguagem e não inclui o nível pragmático. Hausser (2001) define que os componentes fonológico, morfológico, léxico, i.e., que lista as palavras analisadas, sintático e semântico são parte da gramática em virtude dos mesmos tratarem com estruturas de formatos de palavras, expressões complexas e sentenças.

2.2.2.1 Nível Morfológico

Os conceitos básicos no nível morfológico incluem a palavra, o morfema e o alomorfe. A *palavra* é um conceito abstrato que é completamente manifestado nas *formas de palavras* associadas, tal qual ilustra a figura 2.11. A *forma base* é utilizada para identificar uma palavra, tal como: para os substantivos a *forma base* é o nominativo singular, e.g., “livro”; para o verbo, o infinitivo impessoal, e.g., ler; e para o adjetivo, o primitivo, e.g., livre (HAUSSER, 2001). A figura 2.12 ilustra um exemplo de leituras morfológicas para a forma de palavra *revista* de acordo com Bick (2000).

WORD	WORD FORM
wolf = $_{def}$	{[wolf (SN) wolf], [wolf's (GN) wolf], [wolves (PN) wolf], [wolves' (SN) wolf]}

SN – substantivo singular
PN – substantivo plural
GN – substantivo genitivo

Figura 2.11: Exemplo de conceituação de palavra.

FORMA DE PALAVRA	FORMA BASE	
revista		
	revista	*(publicação ou inspeção)*
	revestir	
	revistar	*(presente indicativo)*
	revistar	*(imperativo afirmativo)*
	rever	

Figura 2.12: Exemplo de forma de palavra e forma base.

A morfologia tradicional analisa as formas de palavras por meio da separação da mesma em partes elementares, i.e., morfemas, que compõem as unidades de significação mínimas e indecomponíveis da linguagem, e.g., *escol/ar/i/dade*. A noção de morfema é uma abstração que se manifesta concretamente na forma de alomorfes, que significa "forma alternativa", e.g., o morfema *mong* possui dois alomorfes *mong(e)* e *monj(a).* Alomorfia é a variação de um morfema sem mudança no seu significado, e.g., em *infeliz* e *imutável*, como segundo exemplo, tanto *in-* quanto *i-* indicam negação.

2.2.2.1.1 Reconhecimento automático de palavra

Com base nos conceitos morfológicos descritos, existem três formas principais de reconhecimento automático de palavra: forma de palavra, morfemas e alomorfes (HAUSSER, 2001).

O *método de forma de palavra* analisa o léxico, ou dicionário, com base na forma de palavra, e.g., *[wolves (part of speech: Subst, num: Pl, case: N,D,A base form: wolf]*. Permite a implementação de um algoritmo que simplesmente equipara toda a superfície da forma de palavra desconhecida na correspondente chave na análise do léxico. As críticas ao referido método incluem: a elaboração de um léxico com formas de palavras completas é muito mais custosa do que um léxico com formas bases e regras de inflexão que permitam derivar automaticamente a forma de palavra do léxico; e a impossibilidade de

reconhecer neologismos em tempo de execução, a menos que todas as possíveis formas de palavras sejam providas pela análise léxica.

O *método de morfema*, por ser baseado em morfemas, faz uso de um léxico menor e permite a identificação de neologismos em tempo de execução por meio de uma segmentação e concatenação baseada em regras de formas de palavras complexas em seus elementos (morfemas), tendo como requisito único que os elementos sejam lexicalmente conhecidos e seus modos de composição manipulados por regras. A principal desvantagem do método é o algoritmo de reconhecimento mais complexo, pois a análise de uma superfície desconhecida em tempo de execução requer: a segmentação em alomorfes; redução dos alomorfes ao morfema correspondente; reconhecimento do morfema por meio de análise do léxico; e concatenação baseada em regra do morfema para derivação da forma de palavra analisada. Reproduzindo um exemplo em inglês da referência ao presente método, a superfície *wolves* é segmentada em dois alomorfes *wolv/es*, sendo os mesmos reduzidos aos correspondentes morfemas *wolf+s*.

O *método de alomorfe* combina as vantagens dos métodos de forma de palavra e de morfema por meio da aplicação de um algoritmo de reconhecimento simples com um pequeno léxico para análise. A análise baseada em regras permite o reconhecimento de neologismos em tempo de execução. O referido método usa um léxico elementar e um léxico de alomorfe. O léxico elementar inclui as formas básicas das classes de palavras abertas, os elementos das classes fechadas e os alomorfes de afixos usados em flexões, derivações e composições. O léxico de alomorfes é gerado a partir do léxico elementar por meio da aplicação de regras (*allo-rules*) antes do tempo de execução. Durante o tempo de execução, a superfície desconhecida é concatenada da esquerda para a direita e equiparada com os alomorfes disponíveis, sem a redução aos morfemas; durante o referido processo de concatenação, a análise morfossintática da forma de palavra analisada ocorre por meio de regras de combinação (*combi-rules*). Reproduzindo um exemplo em inglês da referência ao presente método, uma entrada de forma básica em inglês *derive* gera, por meio de regras, dois alomorfes: *derive (e.g., derive/s)* e *deriv (e.g., deriv/ing)*. Como exemplo do referido método, a gramática *LA-grammar* implementa um sistema denominado *LA-Morph* baseado em um léxico elementar, um conjunto de regras *allo-rules* e um conjunto de regras *combi-rules.* Uma *allo-rule* toma como entrada uma forma base de palavra do léxico elementar e deriva da mesma um ou mais alomorfes, sendo a referida análise morfológica da forma de palavra representada por uma tripla composta de superfície, categoria sintática e representação semântica, tal qual ilustra a figura 2.13. Desta forma, o léxico elementar é

convertido, antes do tempo de execução, automaticamente no léxico de alomorfes analisados. A descrição completa de todas as regularidades e irregularidades morfológicas de uma linguagem é um longo processo, sendo, entretanto, a escrita das *allo-rules* um significativo investimento, particularmente por reduzir o tamanho da análise do léxico e permitir a identificação de neologismos. As referidas regras somente são aplicadas de novo caso ocorram modificações nas citadas regras ou no léxico elementar. Os alomorfes gerados por meio da *allo-rules*, antes do tempo de execução, permitem que as *combi-rules* assegurem, em tempo de execução, que:

• os alomorfes encontrados na superfície analisada não estejam combinados em uma forma de palavra não-gramatical;

• as categorias dos alomorfes sejam mapeadas em uma categoria resultante, e.g., *swim (NOM V) + s (SX S3) → swims (S3 V)*;

• o resultado correto seja formado durante a interpretação semântica.

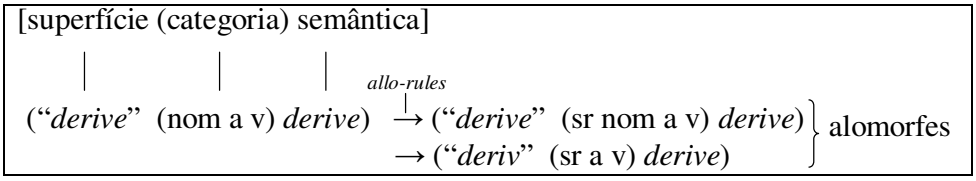


Figura 2.13: Exemplo de derivação de alomorfes a partir de forma base de palavra.

2.2.2.1.2 Etiquetação Morfológica

Uma das principais fontes de problemas para a etapa de análise sintática, ou *parsing* sintático, de uma língua é a pluricategorização gramatical de suas palavras. Como exemplo, o problema de ambigüidade do Português foi quantificado por Bick (2000) em dois *corpus* com um total de 762.000 palavras, sendo a média de duas leituras para cada forma de palavra. A figura 2.14 exemplifica, em notação da gramática *CG*, uma lista de ambigüidades relacionadas com a denominada forma de palavra *galopante*, i.e., na primeira leitura a etiqueta primária determina que a categoria gramatical é "adjetivo" e na segunda, "substantivo". A etiqueta secundária identifica o prefixo *arqui*, que é vinculado à informação *DERP*, do tipo classe de derivação e semântica, sendo *inimigo* a forma base para ambas as leituras e a informação *[SUP]* relativa ao prefixo superlativo.

FORMA DE PALAVRA	FORMA BASE	*TAG* SECUNDÁRIO	*TAG* PRIMÁRIO
arquiinimigos			
	"inimigo"	<DERP arqui– [SUP]>	ADJ M P
	"inimigo"	<DERP arqui– [SUP]>	N M P

Figura 2.14: Exemplo de ambigüidade do Português.

A etiquetação morfológica consiste em se atribuir uma categoria gramatical a cada palavra de um texto, sendo a anotação de um texto com etiquetas morfossintáticas, comumente chamada de *tags*, é útil para várias manipulações subseqüentes do texto, e.g., seleção de palavras de uma determinada categoria ou para desambigüização (ALUISIO e AIRES, 2000). Bick (2000) descreve que o termo *tag* em análise gramatical é usado para designar qualquer cadeia alfanumérica, vinculada a uma palavra, portando meta-informação sobre formas e funções da referida palavra, e.g., *casa N F S*. As principais vantagens do *tag* incluem:

• conter informações de todos os níveis, i.e., morfológico, sintático, semântico, com o mesmo formalismo, o que permite interagir com processos que solucionam ambigüidades;

• facilitar a manipulação em um contexto de dados lingüísticos;

• permitir que informações gramaticais em notação *tag* sejam facilmente filtradas em diferentes esquemas por meio de ferramentas de processamento de texto padrão.

Uma *PoS (part of speech)*, também chamada de categoria gramatical ou classe de palavra, permite identificar qual classe de palavra aparece em diferentes posições em uma sentença, sendo as *PoS* mais importantes os substantivos, verbos, adjetivos, preposições e advérbios. A existência de categorias que podem aparecer em determinadas posições, e outras que não, permite fazer generalizações sobre o comportamento de diferentes tipos de palavras, sendo esta a principal utilidade da *PoS* (CARNIE, 2002). As *PoS* são divididas em duas classes: aberta e fechada. As categorias da classe aberta incluem os substantivos, verbos, adjetivos e advérbios, de modo que os membros são ilimitados, i.e., palavras podem ser criadas a qualquer momento. Por outro lado, as categorias da classe fechada, e.g., preposições, conjunções, determinantes, possuem um número limitado de membros e com menor possibilidade de criação dos mesmos. Hausser (2001) classifica as *PoS* classes abertas de *palavras de conteúdo* e as classes fechadas, de *palavras funções*.

Carnie (2002) afirma que o aprendizado da gramática formal usualmente associa uma *PoS*, tal como um *substantivo* a uma *pessoa, lugar ou coisa*, ou um *adjetivo* a *uma propriedade de um substantivo*, sendo ambas definições baseadas em critérios semânticos. Considerando a frase *Sinceridade é uma qualidade importante*, o referido autor argumenta que, embora *sinceridade* seja uma característica ou atributo, ou seja, uma propriedade associada normalmente com adjetivos, a referida palavra é um *substantivo*. A maior evidência para a não utilização de definição semântica para a *PoS* vem do fato de que a *PoS* de uma palavra pode ser conhecida e o significado da mesma não, sendo esta não-

definição semântica uma característica da *gramática de estrutura sintagmática*, prevalecendo o posicionamento estrutural da palavra na sentença.

A tabela 2.1 ilustra o conjunto de etiquetas, ou *tagset*, adotado pelo Núcleo Interinstitucional de Lingüística Computacional - NILC (ALUÍSIO e AIRES, 2000).

Tabela 2.1: Etiquetas de classes gramaticais adotadas pelo NILC.

Classe gramatical	Tipo	Etiqueta
Adjetivo		ADJ
Advérbio		ADV
Artigo		ART
Conjunção	Coordenativa	CONJCOORD
	Subordinativa	CONJSUB
Interjeição		I
Locução	Adverbial	LADV
	Conjuncional	LCONJ
	Denotativa	LDEN
	Prepositiva	LPREP
	Pronominal	LP
Numeral	Cardinal	NC
	Ordinal	ORD
	Outros Números	NO
Palavra Denotativa		PDEN
Palavras ou Símbolos Residuais		RES
Preposição		PREP
Pronome	Apassivador	PAPASS
	de Realce	PREAL
	Demonstrativo	PD
	Indefinido	PIND
	Interrogativo	PINT
	Oblíquo Átono	PPOA
	Oblíquo Tônico	PPOT
	Pessoal Caso Reto	PPR
	Possessivo	PPS
	Relativo	PR
	Tratamento	PTRA

Classe gramatical	Tipo	Etiqueta
Substantivo	Comum	N
	Próprio	NP
Verbo	Auxiliar	VAUX
	Bitransitivo	VBI
	de Ligação	VLIG
	Intransitivo	VINT
	Transitivo Direto	VTD
	Transitivo Indireto	VTI

Bick (2000) apresenta um analisador léxico para o Português, denominado PALAVRAS, com dois tipos de etiquetas: primária e secundária. A etiqueta primária inclui a *PoS* e características morfológicas, tais como, gênero, número, pessoa do verbo, tempo do verbo, modo do verbo e restrição do verbo (infinitivo, gerúndio, etc). A etiqueta secundária inclui informações léxicas sobre valência e classe semântica. Como exemplo, a palavra *revista* possui o *tag* primário N F S, i.e., *Noun* (substantivo), *Female* (feminino), Singular, e o *tag* secundário <+n> <rr> <CP>, onde <+n> significa uma distribuição denominada pré-nome, e.g., revista Veja, <rr> para objeto legível ou <CP> para Controle e aspecto Perfectivo.

O *parser* PALAVRAS possui dois conjuntos de etiquetas: de classes gramaticais, ilustradas na tabela 2.3, que inclui 14 etiquetas de classes gramaticais, e de flexões, ilustradas na tabela 2.4 (BICK, 2000).

Tabela 2.3: Etiquetas de classes gramaticais adotadas pelo *parser* PALAVRAS.

Classe gramatical	Etiqueta	Observação
Adjetivo	ADJ	
Advérbio	ADV	
Determinantes	DET	e.g., artigos, quantificadores de atributos
Conjunção	KS	Subordinada
	KC	Coordenada
Elemento composto	EC	
Especificador	SPEC	e.g., pronomes indefinidos
Interjeição	IN	
Numeral	N	
Preposição	PRP	
Pronome pessoal	PERS	

Classe gramatical	Etiqueta	Observação
Substantivo	N	
	PROP	Próprio
Verbo	V	

Tabela 2.4: Etiquetas de flexão adotadas pelo *parser* PALAVRAS.

Flexão	Etiqueta	Tipo
Gênero	M	Masculino
	F	Feminino
	M/F	Uniforme ou comum de dois gêneros
Número	S	Singular
	P	Plural
	S/P	Invariável
Caso (desinência)	NOM	Nominativo
	ACC	Acusativo
	DAT	Dativo
	PIV	Prepositivo
	ACC/DAT	
	NOM/PIV	
Pessoa	1, 2, 3	Combinado com número, e.g., 1S, 3P
Tempo	PR	Presente
	IMPF	Imperfeito
	PS	Perfeito simples
	MQP	Mais-que-perfeito
	FUT	Futuro
	COND	Condicional
Modo	IND	Indicativo
	SUBJ	Subjuntivo
	IMP	Imperativo
Restrição	VFIN	Verbo finito
	INF	Infinitivo
	PCP	Particípio
	GER	Gerúndio

No contexto da etiquetação morfológica, destacam-se, também, a intervenção de um pré-processador e a organização do léxico para fins específicos da análise. A intervenção de um pré-processador inclui a necessidade inicial de identificar o que não é uma palavra,

tais como números e pontuações, sendo que as pontuações delimitam sentenças, fazem parte de palavras, e.g., semi-interno e formam não-palavras regulares, e.g., ":"; em segundo, separar as palavras incluindo as que contêm hífens, permitindo a análise morfológica de pronomes pessoais oblíquos átonos nas posições próclise, ênclise e mesóclise, e.g., dar-lhe-ei; e, em terceiro, a identificação de palavras compostas que são tratadas como palavras ordinárias pelo analisador. A organização do léxico do analisador PALAVRAS inclui 10 campos para cada entrada no léxico: raiz da palavra; classe da palavra; regras de combinação, e.g., "i" significa que combina com tempo passado; irregularidades gramaticais; fonética; informação diasistemática, e.g., arcaísmos e palavras de uso regional; sinônimos; classe de palavra sintática, e.g., <vt> significa verbo monotransitivo com objeto acusativo; e número de identificação que auxilia na busca das entradas da raiz. As etiquetas de terminação de inflexão, de sufixo e prefixo permitem determinar a raiz da palavra analisada (BICK, 2000).

2.2.2.1.3 Ambigüidade morfológica

A ambigüidade morfológica, denominada de homonímia, inclui morfemas livres, morfemas amarrados e ambos, tal qual ilustra a figura 2.15. A ambigüidade (1a) e (2) incluem homonímias da mesma categoria gramatical, sendo as ambigüidades (1b), (1c) e (3) homonímia entre categorias gramaticais ou de lexemas (não permite livre flexão de gênero ou de número em uma determinada categoria gramatical, e.g., livro) (BICK, 2000).

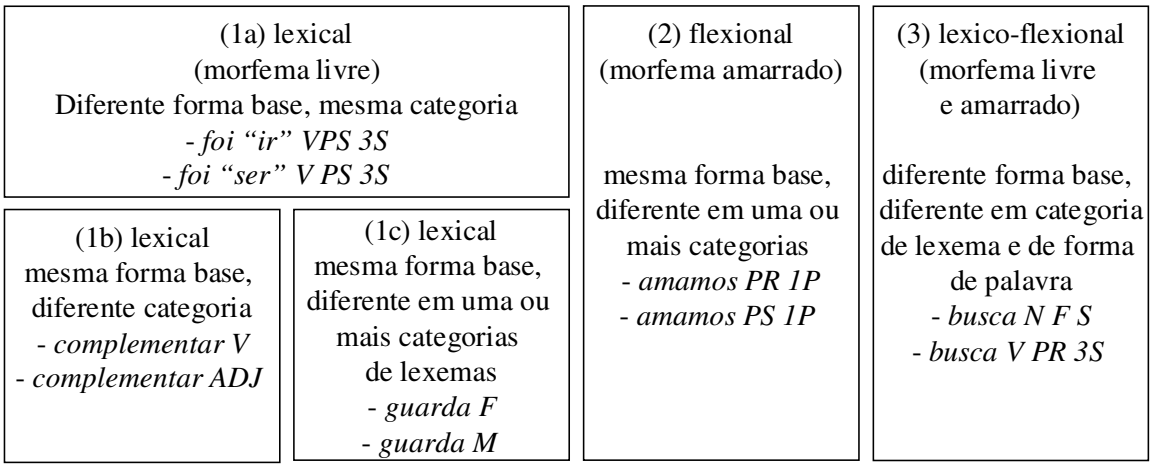


Figura 2.15: Ambigüidade morfológica (homonímia).

A fim de exemplificar, a ambigüidade do Português tem uma média entre 2,0 e 2,1 de leituras para cada forma de palavra. Um tratamento estatístico ilustra a magnitude do problema para as classes de palavras abertas, i.e., N, ADJ, PROP, V, ADV. A tabela 2.5

mostra o risco relativo de uma classe de palavra WC1 ter uma forma de palavra WC1-WC2 ambígua. Como exemplo de interpretação das porcentagens ilustradas, o valor de 15,7% em destaque significa 15,7% de todas as palavras com leitura N são ambíguas com pelo menos um VFIN. Um exemplo característico está na forma da palavra *revista* que tem como forma base os verbos *revestir, revistar (imperativo e indicativo), rever* e dois substantivos com dois significados: *publicação* e *inspeção*.

Tabela 2.5: Freqüências relativas para ambigüidades de classes de palavras.

WC2 → WC1 ↓	N (%)	ADJ (%)	VFIN (%)	INF (%)	GER (%)	PCP (%)	ADV (%)	PROP (%)
N	3,1	13,3	**15,7**	1,1	0,0	3,1	3,0	2,8
ADJ	51,6	1,3	13,2	0,6	0,0	13,0	6,5	5,1
VFIN	35,8	7,7	30,0	12,2	0,1	1,2	3,5	1,8
INF	15,4	2,3	75,4	0,2	0,0	0,0	0,0	0,5
GER	0,7	1,0	2,1	0,0	0,0	0,0	0,0	0,1
PCP	41,2	43,7	7,0	0,0	0,0	1,6	0,3	0,4
ADV	14,8	8,3	7,7	0,0	0,0	0,1	19,2	0,2
PROP	16,6	7,8	4,6	0,2	0,0	0,2	0,3	2,4

2.2.2.2 Nível Sintático

A sintaxe é o estudo dos princípios e processos pelos quais as sentenças são construídas em uma determinada linguagem. A pesquisa sintática de uma determinada linguagem tem como objetivo a construção de uma gramática que possa ser vista como uma ferramenta que permita gerar as sentenças da linguagem sob análise e estudar as propriedades da gramática que fazem isso efetivamente (CHOMSKY, 2002).

Carnie (2002) aplica o método científico na sintaxe. Inicialmente, a observação de dados sobre a linguagem em estudo permite fazer generalizações sobre modelos, e.g., sujeito precede o verbo. A partir destas generalizações, as hipóteses são geradas e testadas sobre mais dados sintáticos e, se necessário, as mesmas são reavaliadas. A partir da referida teoria, o autor denomina as hipóteses de *regras* e o conjunto de hipóteses, que descrevem a sintaxe da linguagem, de *gramática.*

Considerando o escopo da presente pesquisa, serão apresentados conceitos teóricos do nível sintático relacionados com a gramática *PSG* com ênfase no idioma português.

2.2.2.2.1 Estruturas Sintagmáticas do Português

As estruturas sintagmáticas do português incluem o sintagma nominal (SN) e o sintagma verbal (SV), bem como os seguintes sintagmas modificadores: o sintagma adjetivo (SAdj), o sintagma preposicionado (SP) e, para alguns autores, o sintagma adverbial (SAdv) (SOUZA E SILVA e KOCH, 2005) e (AZEREDO, 2003). Na estrutura da sentença, em sua forma base, aparecem como constituintes obrigatórios o nominal e o verbal.

De acordo com Carnie (2002), a *regra de ouro da estrutura em árvore da PSG* é que os modificadores devem estar sempre vinculados dentro da frase que eles modificam. No exemplo ilustrado na figura 2.16, o adjetivo (Adj) *escura* modifica o adjetivo *amarela* sendo, então, *escura* parte de SAdj. A expressão adjetiva *amarela escura* modifica o substantivo *cor*, estando, como conseqüência da referida regra de ouro, a expressão e o substantivo no mesmo SN. Cabe ressaltar que a regra para o SAdj é *SAdj →SAdj, Adj* e não *SAdj → Adj, Adj*, sendo que a regra correta permite a recursividade, i.e., um número indeterminado de modificadores em uma mesma expressão.

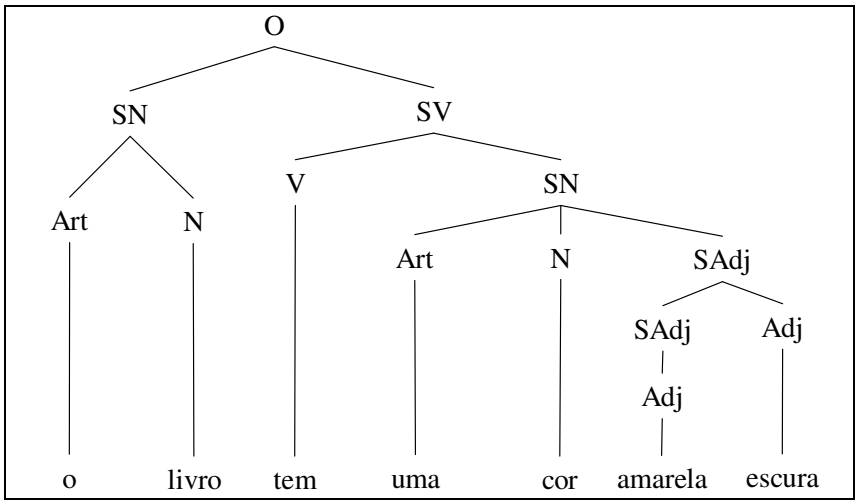


Figura 2.16: Exemplo de aplicação da regra de ouro da estrutura de árvore.

Sintagma Nominal (SN)

O SN comporta necessariamente um núcleo composto por um substantivo comum (N) que poderá vir precedido de determinantes e precedido, ou seguido, por modificadores (SOUZA E SILVA e KOCH, 2004) e (AZEREDO, 2003).

O determinante (Det), quando simples, é representado por um artigo ou pronome, tal qual ilustra a figura 2.17(a). Quando complexo, constitui-se de mais de um elemento, tal qual ilustra a figura 2.17(b), tendo como regra completa *Det → (pré-det) det-base (pós-*

det). Funcionam como elementos-base de um determinante complexo em português o artigo, o pronome demonstrativo ou o pronome possessivo e como pós-determinantes, os numerais e os possessivos, e como pré-determinantes, os quantificadores universais, e.g., todos, nenhum, e os quantificadores partitivos, e.g., alguns, muitos.

A posição de modificador do SN pode ser ocupada por um SAdj, SP ou SAdv, sendo que o SAdj pode anteceder o substantivo, e.g., *a nova residência de Paulo*.

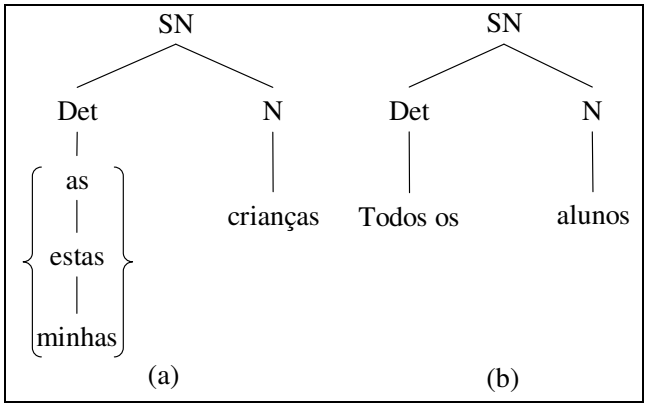


Figura 2.17: Exemplos de determinantes em um SN.

Sintagma Verbal (SV)

O SV é um constituinte caracterizado pela presença obrigatória do verbo, desempenhando privativa e exclusivamente a função de predicado. As diversas estruturas do predicado e, por extensão, da própria oração, dependem de duas classes sintáticas do verbo: predicadores e transpositores.

Os verbos predicadores são núcleos do predicado que compõem classes sintáticas possuidoras de características combinatórias peculiares, tais como:

- verbos que dispensam o SP mas exigem o sujeito, e.g., as folhas *caem* no outono;
- verbos que dispensam o SP mas admitem facultativamente um SN objeto, e.g., este menino não *come* (carne) desde ontem;
- verbos que se constroem com um SN que tanto pode ser o sujeito quanto objeto, e.g., esse tecido *encolhe* após a lavagem ou a lavagem *encolheu* o tecido;
- verbos que obrigatoriamente vêm seguidos de SN, e.g., a polícia *interditou* a ponte;
- SN que servem de objetos seguidos de SP que pode conter um infinitivo, e.g., o porteiro *impediu* o mendigo de entrar na boate.

Os conjuntos de peculiaridades sintáticas características das classes sintáticas apresentadas constituem a predicação de seus verbos, sendo que algumas outras classes podem ser estabelecidas.

A classe sintática dos verbos transpositores introduzem outros constituintes, e.g., SAdj, SN, SP, que podem funcionar como predicadores. Apenas os verbos *ter/haver*, seguidos de particípio, e *ser* são transpositores porque o único papel deles é servir de instrumento para que um constituinte não-verbal possa funcionar como núcleo do predicado, i.e., como predicador. Os verbos compreendidos nesta classe são tradicionalmente rotulados como *auxiliares - ter/haver/ser* – e de *ligação – ser*. Como exemplo, na sentença *o autor dessa história é baiano* o predicador é um SAdj, i.e., *baiano* (AZEREDO, 2003).

Atribui-se a etiqueta *verbo (V)* ao constituinte SV que contém a forma verbal, composta de um só vocábulo, i.e., tempos verbais simples, ou de vários vocábulos, i.e. tempos compostos ou locuções verbais. O SV pode ser representado apenas pelo núcleo, tal qual ilustra a figura 2.18, ou pelo verbo acompanhado de outro sintagma, conforme mostra a figura 2.19 (SOUZA E SILVA e KOCH, 2004).

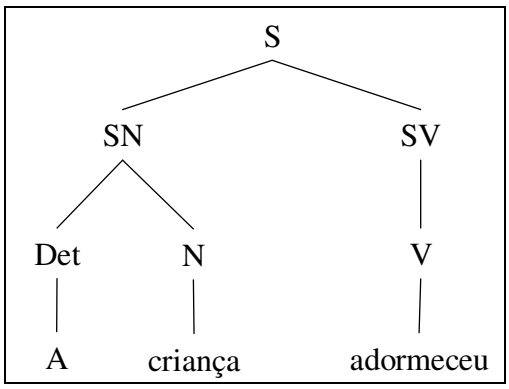


Figura 2.18: Núcleo do SV composto apenas pelo verbo.

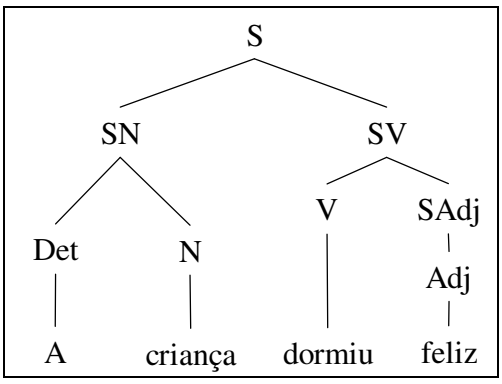


Figura 2.19: Núcleo do SV composto pelo verbo acompanhado de SAdj.

Sintagma Preposicionado (SP)

De uma forma geral, o SP é constituído de uma preposição (*Prep*) seguida de um *SN*, i.e., *SP →prep + SN*.

Os modificadores estruturados em um *SP* são, em sua maioria, expressos por locuções adverbiais, normalmente introduzidas por preposição, sendo, então, possível atribuir-lhes a etiqueta de SP (SOUZA E SILVA e KOCH, 2004). Os referidos autores justificam a referida opção por meio dos seguintes argumentos: muitos advérbios possuem uma locução adverbial correspondente, e.g., rapidamente – com rapidez, aqui – neste lugar; os advérbios constituem um inventário fechado, ao passo que as locuções adverbiais formam, praticamente, um inventário aberto, sendo assim mais econômico englobar a uns e outros sob o rótulo de SP; a descrição torna-se mais coerente, uma vez que toma como base não a estrutura, mas a função desses modificadores que é a mesma, tal como o modificador circunstancial de tempo, e.g, *cedinho* e *de madrugada*. Adotando-se tal posição, a regra de reescrita do SP passa a ser:

$$SP \rightarrow \begin{Bmatrix} prep + SN \\ adv \end{Bmatrix}$$

O SP pode ocorrer de forma independente, i.e., uma terceira divisão da oração, tal qual ilustra a figura 2.20. Ao lado das orações constituídas apenas de SN + SV, como mostra a figura 2.19, tem-se aquelas compostas de SN+ SV + SP.

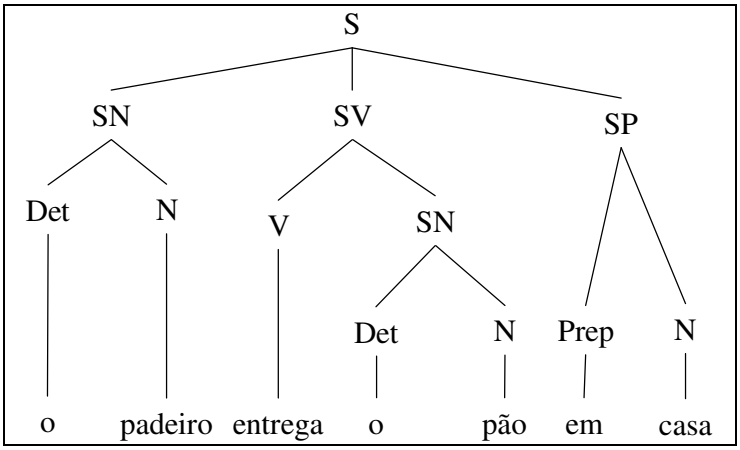


Figura 2.20: Exemplo de independência do SP.

O SP pode, também, apresentar-se dentro de um SN, de um SV ou de um SAdj, funcionando como modificador do núcleo, tal qual ilustra a figura 2.21. As sentenças (b) e (f) exemplificam a função de complementos-nominais dos SP e as sentenças (c) e (d), de complementos-verbais, i.e., acompanham os verbos transitivos, ou seja, de predicação incompleta, que exigem um complemento preposicionado. A sentença (a) exemplifica o

papel de *adjunto* do SP como modificador nominal e a sentença (e), como modificador verbal, i.e., como modificador ou intensificador do processo verbal.

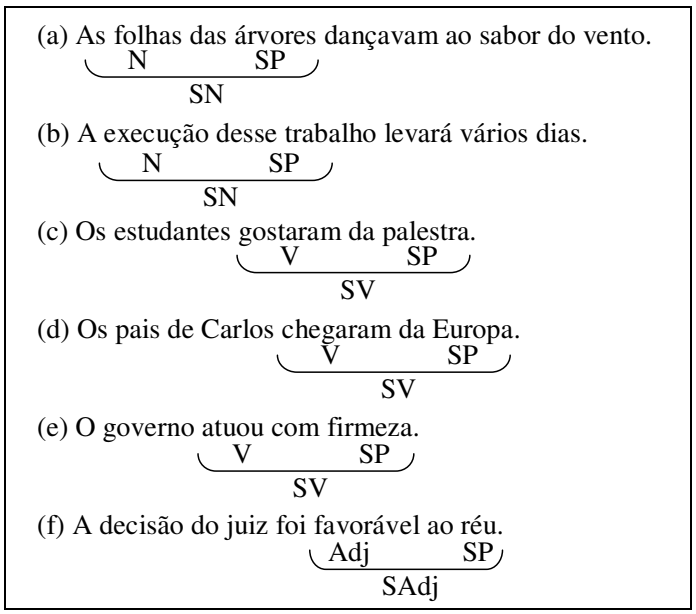


Figura 2.21: Exemplos de SP como modificador dentro de um sintagma.

Sintagma Adjetivo (SAdj)

O SAdj tem como núcleo um adjetivo, tal qual ilustra a figura 2.22, que pode vir isolado ou acompanhado de outros elementos: intensificadores e modificadores adverbiais, antepostos ao núcleo, e de SP, pospostos a ele. O SAdj tem como regra geral: *SAdj → (intensificador) (SP) Adj (SP)* (SOUZA E SILVA e KOCH, 2004).

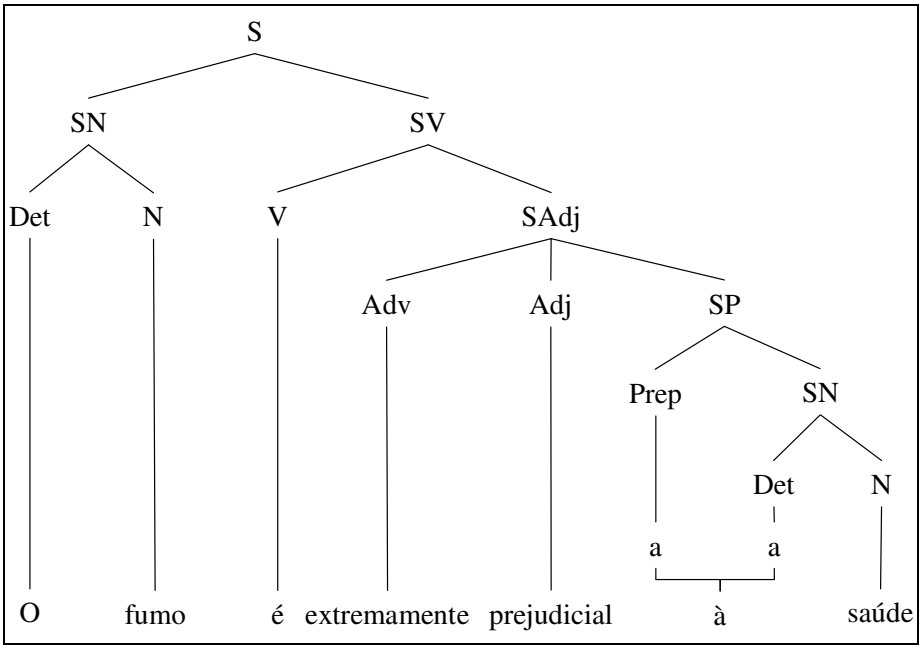


Figura 2.22: Exemplo de sintagma adjetival.

O SAdj desempenha as funções sintáticas de predicador e de modificador, concordando, em ambas as funções, em gênero e número com a base SN com que se relaciona na frase. Como predicador, vem introduzido pelo verbo transpositor *ser*, e.g., os castelos de areia eram grandes e imponentes. Como modificador adjunto, refere-se à base do SN, e.g., ontem tivemos uma noite de prata *lunal*. Como modificador aposto refere-se à totalidade do SN, e.g., *ausentes* os hóspedes e os passageiros, caíamos no ramerrão fastidioso (AZEREDO, 2003).

Sintagma Adverbial (SAdv)

Azeredo (2003) adota de forma independente, como modificador, o sintagma adverbial (SAdv). O SAdv, exemplificado na figura 2.23, tem por núcleo um advérbio (Adv) e pode estruturar-se de forma semelhante ao SAdj, já que alguns advérbios podem vir precedidos de um determinante, e.g., *muito tarde*, precedidos de modificador, e.g., *incrivelmente longe*, ou seguidos de modificador, e.g., *depois do almoço*.

O SAdv pode servir de predicador, geralmente introduzido pelo verbo *ser*, e.g., *a inauguração será amanhã*, ou como modificador junto ao verbo, e.g., *eles conversaram demoradamente,* junto ao adjetivo, e.g., *levemente feridos*, junto ao substantivo, e.g., *um cafezinho agora,* junto a outro advérbio, e.g., *estupendamente bem* (AZEREDO, 2003).

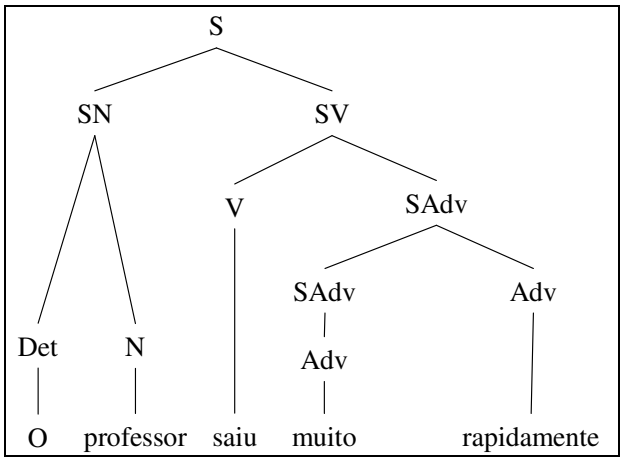


Figura 2.23: Exemplo de sintagma adverbial.

2.2.2.2.2 Testes de Validação de Sintagmas

Carnie (2002) apresenta quatro testes para validação dos agrupamentos de sintagmas: substituição, independência, movimento e coordenação. O autor destaca que, por requererem algum julgamento, nenhum dos testes é absoluto ou perfeitamente seguro.

A *substituição* tem como premissa básica o fato de ser uma simples palavra o menor sintagma, de modo que se é possível substituir um grupo de palavras por uma única palavra

então o referido grupo forma um sintagma. O referido teste é denominado de *comutação* por Souza e Silva e Koch (2004), i.e., a decomposição de uma proposição, ou oração, em unidades menores pode ocorrer por meio do procedimento da comutação, cujas tarefas básicas incluem a segmentação, que determina os subconjuntos em que pode ser decomposta a proposição, e a substituição, que permite verificar quais desses subconjuntos exercem a mesma função. A Figura 2.24 ilustra a aplicação do procedimento de substituição, ou comutação, da árvore representada na Figura 2.5 (AZEREDO, 2003).

as	outras	aves	de	rapina	puderam	sobreviver	em	cativeiro
as	outras	aves	selvagens		sobreviveram		assim	
as	outras	aves			sobreviveram			
as	aves				sobreviveram			
elas					sobreviveram			

Figura 2.24: Exemplo de procedimento de comutação.

O segundo teste é o da *independência* cuja premissa básica é se uma palavra pode operar independentemente em resposta a uma pergunta, então a mesma, provavelmente, constitui um sintagma. Como exemplo, considerar as seguintes sentenças:

a) Paulo *comeu em um restaurante luxuoso.*

b) Paulo *comeu em* um restaurante luxuoso.

Para a pergunta "O que Paulo fez ontem?", a resposta pode ser o grupo de palavras em itálico da sentença (a) e não (b).

O terceiro teste é o do *movimento* cuja premissa básica é se um grupo de palavras pode mover-se dentro de uma sentença, então o referido grupo é um sintagma porque pode mover-se como uma unidade, sendo o mesmo seguro quando mantém o significado da sentença original. Como exemplo, a transformação de uma frase na forma passiva "[A menina] foi aprovada pela [professora de inglês]", para a forma ativa "[A professora de inglês] aprovou [a menina]"

O quarto teste é o da *coordenação*. As estruturas coordenadas são sintagmas ligados por conjunções tais como *e* ou *ou*. O teste funciona se é possível coordenar um grupo de palavras com um grupo similar de palavras, e.g., [João] e [o homem] foram à loja.

2.2.2.2.3 Resolução de Ambigüidade

A regra de ouro permite resolver um dos problemas mais característicos no processamento de linguagem natural que é a ambigüidade, tal qual se observa nas seguintes sentenças:

(a) O homem matou o rei com uma faca.

(b) O homem matou o rei com o cabelo vermelho.

Ambas as sentenças possuem mais de um significado, mas, a fim de exemplificar a solução da ambigüidade por meio da estrutura sintagmática, devem-se considerar as seguintes interpretações (também denominadas de paráfrases):

(c) o homem usou a faca para matar o rei.

(d) o rei com cabelo vermelho foi morto pelo homem.

Na estrutura da sentença *a)*, interpretada pela sentença *c)* e ilustrada na figura 2.25, o verbo "matou" é modificado pelo sintagma preposicionado "com a faca". Na estrutura da sentença *b)*, interpretada pela sentença *d)* e mostrada na figura 2.26, o sintagma preposicionado "com o cabelo vermelho" modifica o substantivo "rei".

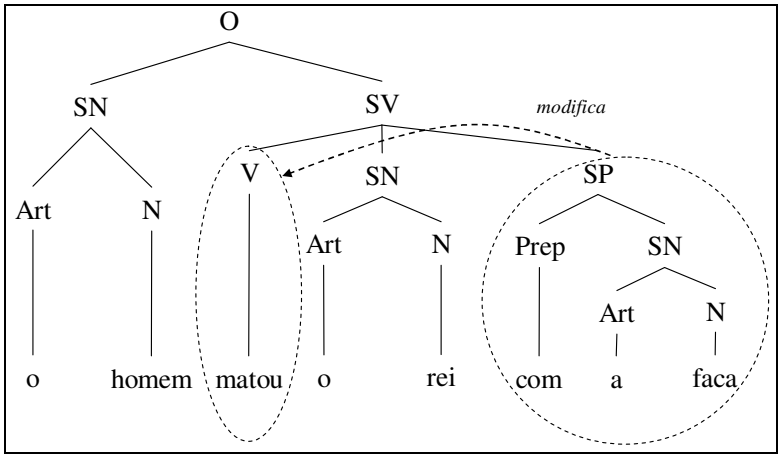


Figura 2.25: Estrutura sintagmática da interpretação c*)*.

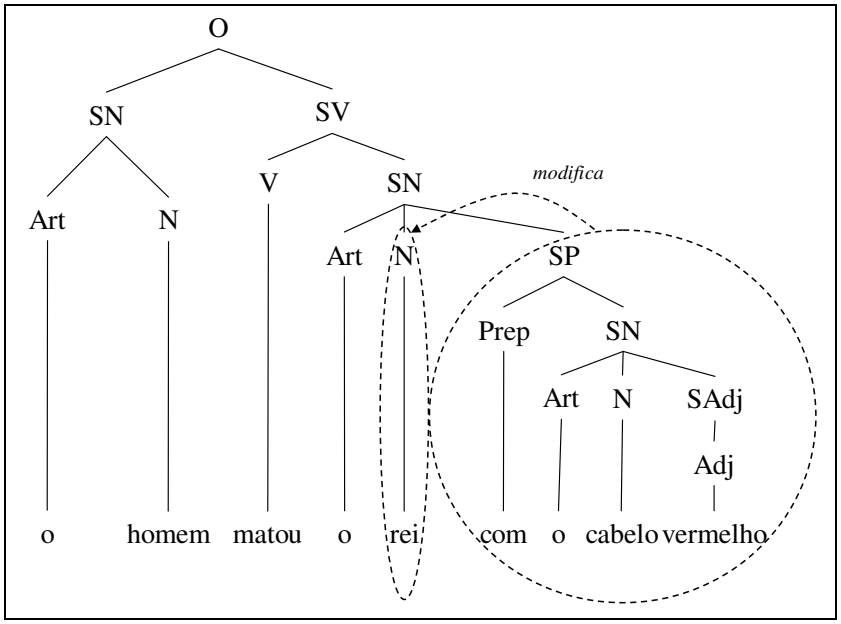


Figura 2.26: Estrutura sintagmática da interpretação *d)*.

Completando a exemplificação da solução da ambigüidade por meio da estrutura sintagmática, deve-se considerar as seguintes interpretações de *a)* e *b)*:

(e) o rei com a faca foi morto pelo homem.

(f) o homem usou o cabelo vermelho para matar o rei.

Na estrutura da sentença a), interpretada pela sentença *e*) e ilustrada na figura 2.27, o substantivo "rei" é modificado pelo sintagma preposicionado "com a faca". Na estrutura da sentença *b)*, interpretada pela sentença *f)* e mostrada na figura 2.28, o sintagma preposicionado "com o cabelo vermelho" modifica o verbo "matou".

Os exemplos apresentados ilustram que as árvores sintáticas permitem capturar as diferenças entre leituras ambíguas de uma mesma superfície de sentença, sendo esta uma importante propriedade das referidas estruturas (CARNIE, 2002).

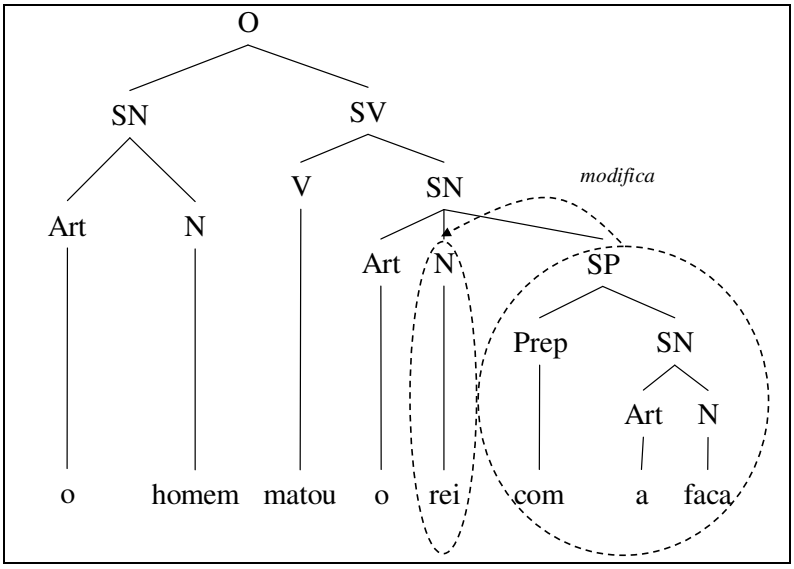


Figura 2.27: Estrutura sintagmática da interpretação *e)*.

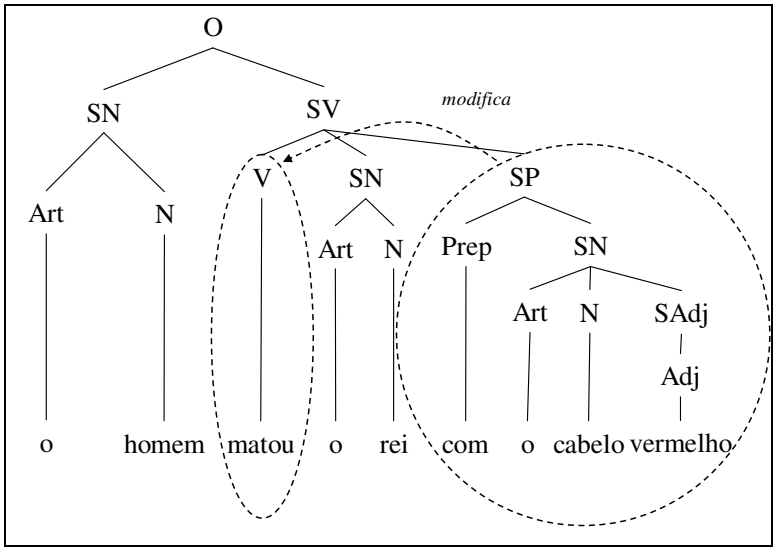


Figura 2.28: Estrutura sintagmática da interpretação *f)*.

Um erro muito comum a ser evitado quando do desenho da estrutura sintagmática de uma determinada sentença é exemplificado por meio da expressão "muito rapidamente" e ilustrado na figura 2.29, i.e., o sintagma adverbial (SAdv) "muito" é modificado pelo advérbio (A) "rapidamente" resultando na fórmula SAdv → SAdv, Adv (CARNIE, 2002). Embora não afirmado pela referência, a citada regra permite a recursão.

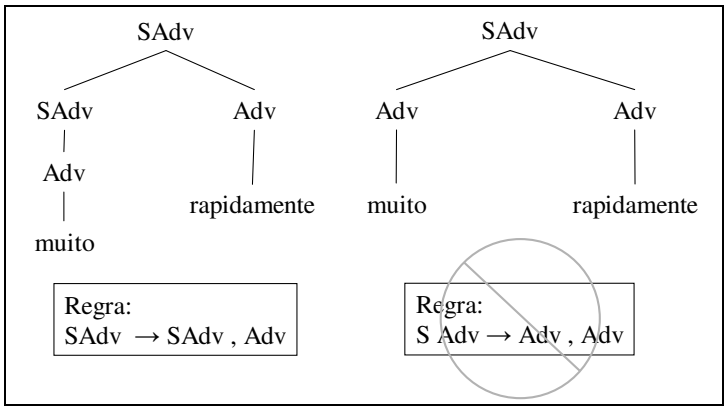


Figura 2.29: Erro comum de estruturação de árvore sintagmática.

2.2.2.2.4 Teoria *X-bar*

A teoria denominada de *X-bar*, ou *X'*, apresenta a necessidade de estruturas sintáticas mais articuladas em substituição a estruturas planas (CARNIE, 2002). A estrutura plana, ilustrada na figura 2.30 e mantida a originalidade do idioma, não apresenta distinção em termos de dominância. Do ponto de vista de estruturação sintática, testes podem conduzir a uma estrutura mais complicada, tal como o teste de constituinte. A figura 2.31 ilustra a aplicação do teste denominado *one-replacement* que enfoca um grupo de nós que não compõem um constituinte na figura 2.30, e.g., na frase *I bought the big [book of poems] with the blue cover not the small [one] with the red cover*, o constituinte *[book of poems]* é representado pelo nó N'_3. O nó *N'* intermediário é necessário (pronunciado "ene-bar") para representar itens vinculados na sentença.

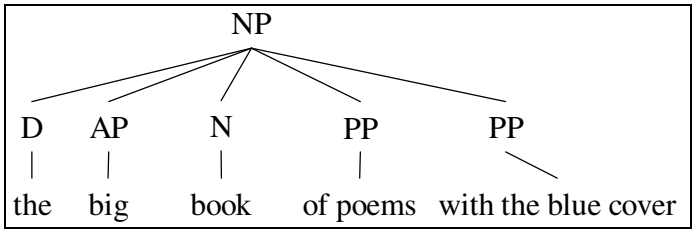


Figura 2.30: Estrutura sintagmática plana.

A estrutura representada pela figura 2.31 gera as seguintes regras *N-bar*:

NP → (D) N';

N' → (AP) N' ou N' (PP);

N' → N (PP).

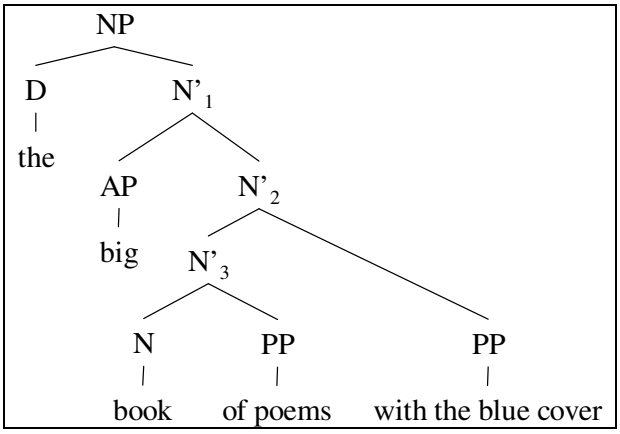


Figura 2.31: Estrutura de acordo com a teoria X-bar.

De modo idêntico, ao sintagma verbal é aplicado o teste de substituição denominado *do-so-replacement*, e.g., *I [eat beans] with a fork but Janet [does so] with a spoon*, tal qual ilustra a figura 2.32. O referido procedimento gera as seguintes regras *V-bar:*

VP → V';

V' → V' (PP);

V' → V (NP).

A regra *VP → V'* permite que toda a sentença a ser substituída seja uma *V' (V-bar)*, e.g., *Kevin [ate spaghetti with a spoon] and Georgi [did so] too.*

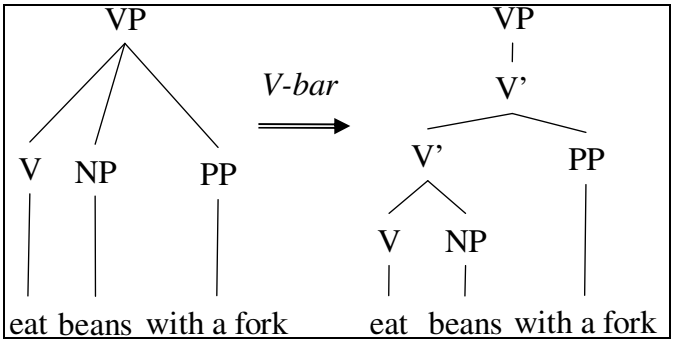


Figura 2.32: Exemplo da aplicação de *V' (V-bar).*

Raciocínio similar é aplicado ao sintagma adjetivo ou adverbial (AP), por meio da *A-bar*, i.e., A', e ao sintagma preposicionado (PP), por meio da *P-bar, i.e., P'*.

Cabe ressaltar que as regras *X-bar* permitem, também, uma generalização relativa à ordem das palavras, de modo que estruturas de frases de diferentes linguagens podem ser comparadas. Como exemplo, em inglês o complemento segue o verbo, tal qual a regra *X' → X (WP)*, mas em turco ocorre o contrário, i.e., *X' → (WP) X*. Parâmetros permitem controlar quais regras são aplicadas para um determinado idioma, sendo um dos

fundamentos da versão da PSG denominada *Princípios e Parâmetros* adotada com destaque na referência (CARNIE, 2002).

A teoria *X-bar* é estendida a sentenças com múltiplas cláusulas. Uma cláusula é um sujeito cuja propriedade é indicada pelo predicado, e.g., *o menino correu, Howard é um estudante de lingüística*. Algumas vezes, cláusulas estão dentro de cláusulas, e.g., *[Peter said [that Danny danced]]*, onde *[Peter said [that Danny danced]]* é denominada de cláusula principal, ou raiz, e a cláusula *[that Danny danced]* é denominada de cláusula embutida ou subordinada. A figura 2.33 ilustra a estrutura sintática da referida sentença, cabendo ressaltar que uma cláusula embutida faz parte de uma cláusula principal.

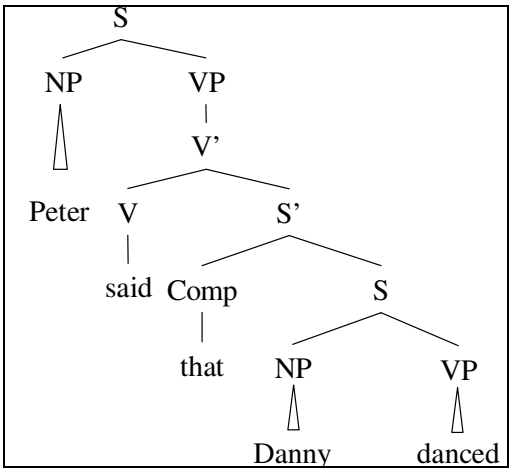


Figura 2.33: Exemplo de sentença com mais de uma cláusula.

2.2.2.3 Nível Semântico

De acordo com Gamut (1991), o núcleo de uma teoria lingüística está em sua teoria semântica, sendo o ponto de conexão entre sintaxe e semântica um dos aspectos mais delicados do estudo lingüístico. Chomsky (2002) apresenta um questionamento se a informação semântica é requerida ou não na seleção de uma gramática e, na mesma linha de raciocínio, questiona: "Como construir uma gramática sem apelo à semântica?". A *PSG*, tal qual concebida pelo referido autor, é uma teoria formal e não-semântica.

O objetivo primário da análise semântica é determinar o significado da sentença (GRISHMAN, 1999). No caso de sentenças declarativas, a semântica busca determinar as regras de inferências existentes nas sentenças da linguagem, o que significa traduzir a linguagem natural em linguagem formal para um sistema computacional. Russel e Norvig (2004) definem que a *interpretação semântica* é o processo de extrair o significado de uma expressão vocal como uma expressão em alguma linguagem de representação. Gamut

(1991) estabelece que, por meio da *teoria referencial do significado*, o significado de um símbolo é aquele ao qual o mesmo se refere.

2.2.2.3.1 Composicionalidade versus Independência da Gramática

O princípio da composicionalidade determina que a interpretação de uma expressão complexa é função da interpretação de suas partes, garantindo que toda regra sintática que permite a construção de certo tipo de expressão tem uma ou mais regras semânticas correspondentes. As considerações semânticas do referido princípio podem influenciar a sintaxe e até mesmo limitar a autonomia desta, i.e., uma teoria semântica composicional não só pressupõe uma teoria sintática mas também impõe restrições sobre a mesma (GAMUT, 1991). Em síntese, a semântica faz parte da gramática.

Hausser (2001), alinhado com o princípio da composicionalidade por meio da *LA-grammar*, determina que, em lingüística, semântica é um componente da gramática que deriva representações de significado a partir de superfícies naturais analisadas sintaticamente. Bick (2000), também alinhado por meio da *CG*, assume que a etiquetação semântica é o próximo passo lógico na análise de nível progressiva antecedida pelas análises morfológica e sintática, i.e., a semântica faz parte da gramática. Gamut (1991) destaca que a composicionalidade do significado e da sintaxe é, entre outras candidatas, uma das formas de computar o significado.

Chomsky (2002) defende a independência da gramática em relação ao significado, i.e., entre a sintaxe e a semântica. O problema da pesquisa sintática está direcionado à construção de uma ferramenta que produza um dado conjunto de sentenças gramaticais e no estudo das propriedades das gramáticas que fazem isso efetivamente, de modo que noções sobre semântica não são consideradas na discussão. O resultado da estrutura gramatical é a definição de um *framework* sintático que possa suportar a análise semântica, sendo o mesmo um ponto de conexão entre sintaxe e semântica.

Alinhado com a referida independência, Carnie (2002) define a gramática como um conjunto de regras que descrevem a sintaxe de uma linguagem. Azeredo (2003) corrobora o conceito de independência da gramática ao afirmar que a gramática compreende tradicionalmente dois níveis estruturais: o da morfologia, cuja maior unidade é o vocábulo, e o da sintaxe, cujo limite máximo é o da oração.

2.2.2.3.2 Teoria Referencial do Significado

Gamut (1991) destaca que o conceito básico da *teoria referencial do significado,* ou *Teoria do Significado de Frege*, é que o significado de um símbolo é aquele ao qual o mesmo se refere. O conceito de senso, ilustrado na figura 2.34, está relacionado com o *modo de apresentação*, ou seja, o critério pelo qual a referência pode ser determinada sob várias circunstâncias.

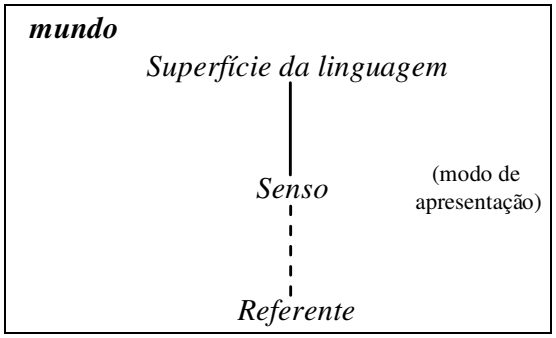


Figura 2.34: Teoria do Significado de Frege.

A figura 2.35 exemplifica um relacionamento entre o conceito *senso* e *referente*; as três linhas *a, b* e *c* têm um ponto de interseção comum *p*, podendo o referido ponto ser caracterizado em diferentes formas: como *a interseção de a e b,* como *a interseção de a e c,* como *a interseção de b com c* e, finalmente, como *a interseção de a,b e c*. Observa-se que o ponto *p* é referência para quatro diferentes nomes, i.e., quatro diferentes descrições têm diferentes *sensos* mas o mesmo *referente.* No contexto do processamento de linguagem natural, Frege conclui que o *senso* de uma sentença é a proposição que a mesma expressa, sendo o *referente* o correspondente valor verdade.

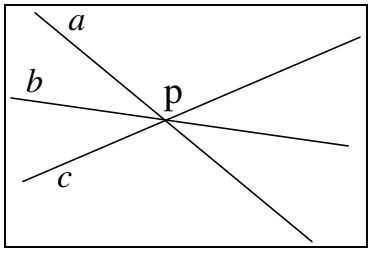


Figura 2.35: Relacionamento *senso versus referente*.

2.2.2.3.3 Interpretação Semântica

Partee *et al.* (1990) afirmam que a distinção entre sintaxe e semântica, na tradição lógica, está proximamente vinculada à distinção entre sistemas formais e suas interpretações.

A *Teoria do Modelo* é o estudo das interpretações dos sistemas formais e enfoca as relações entre teorias e modelos. Um conjunto de axiomas juntos com todos os teoremas deriváveis dos mesmos é chamado de uma teoria, i.e., um conjunto de declarações que possuem vínculos decorrentes de conseqüências lógicas. Um modelo para uma teoria requer: (a) um domínio abstrato e estruturado; e (b) uma interpretação para todas as expressões primitivas da teoria naquele domínio, de modo que, para uma determinada interpretação, todas as declarações na teoria tornam-se verdadeiras para aquele modelo naquela interpretação (PARTEE *et al.,* 1990).

Hausser (2001) apresenta uma estrutura de dois níveis da interpretação semântica, tal qual ilustra a figura 2.36. O primeiro nível consiste de expressões da linguagem analisadas sintaticamente e o segundo, de estruturas semânticas associadas. Os dois níveis são sistematicamente relacionados por meio de um algoritmo de atribuição.

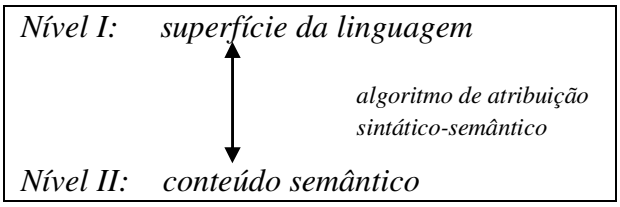


Figura 2.36: Níveis de interpretação semântica.

O referido autor apresenta, também, três tipos de semântica, ilustradas na figura 2.37, que conformam com a estrutura básica da interpretação semântica: linguagens lógicas, linguagens de programação e linguagens naturais. As interpretações semânticas de linguagens artificiais necessitam de dois níveis para determinação das respectivas semânticas e as linguagens naturais, de três níveis, sendo o terceiro o pragmático. No nível II da interpretação das linguagens lógicas, o *modelo teórico de conjunto do mundo* está em conformidade com a teoria do modelo apresentada.

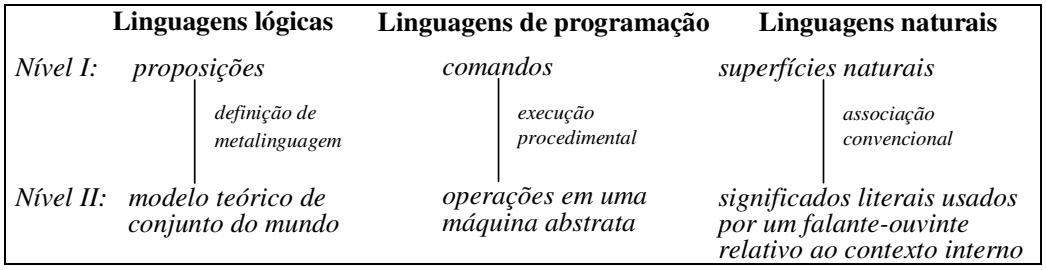


Figura 2.37: Tipos de interpretação semântica.

Gamut (1991) apresenta o método indireto de interpretação semântica proposto por Montague para o *The Proper Treatment of Quantification in Ordinary English* (modelo PTQ). No referido modelo, como ilustra a figura 2.38, as expressões em linguagem natural

são primeiramente traduzidas em expressões de uma linguagem lógica. As mesmas são interpretadas semanticamente e, como conseqüência, ocorre a geração de modelos.

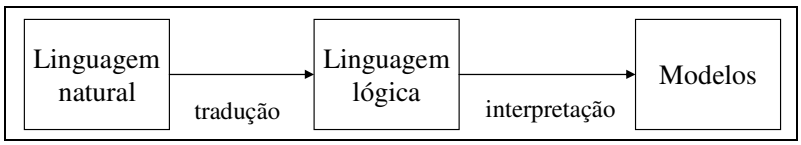


Figura 2.38: Método indireto de interpretação semântica.

No contexto dos níveis apresentados na figura 2.37 para as linguagens lógicas, Hausser (2001) usa a denominação de *semântica lógica* para a referida interpretação. A semântica lógica analisa uma simples proposição, e.g., *Julia dorme,* como uma proposição que tanto pode ser *verdadeira* como *falsa.* Gamut (1991) denomina a *semântica lógica* de *semântica teoria-modelo*, tendo como noções chaves os conceitos de *referente,* tal qual apresentado na *teoria referencial do significado,* e de *verdade*, de modo que o significado de expressões em linguagem natural possa ser obtido por meio dos conceitos *referente* e *verdade.*

A figura 2.39 detalha a interpretação das *linguagens lógicas* ilustrada na figura 2.37. Na definição formal de uma linguagem interpretada, Tarski faz distinção entre a linguagem objeto, i.e., a linguagem a ser interpretada semanticamente, e a metalinguagem, i.e., a linguagem que formula as definições semânticas (HAUSSER,2001). Os termos *linguagem objeto* e *metalinguagem* referem-se a linguagens com diferentes funcionalidades, mas que podem ser implementadas por diferentes linguagens ou por apenas uma, e.g., inglês ou lógica dos predicados (GAMUT, 1991).

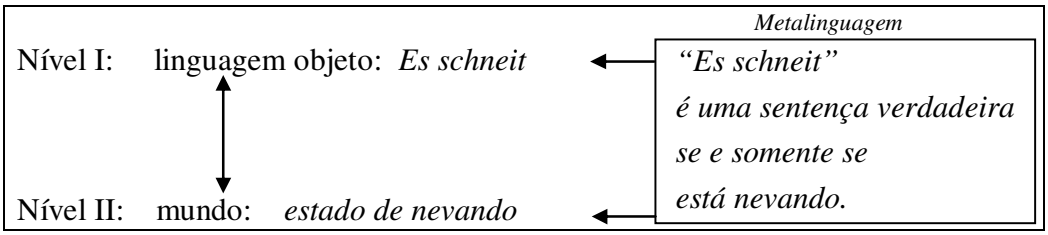


Figura 2.39: Relação entre linguagem objeto e metalinguagem.

2.2.2.3.4 Análise de Discurso

Conceitualmente, a análise semântica da linguagem natural se limita a examinar a informação contida em uma sentença. A análise de discurso é o estudo das conexões entre componentes de diversas sentenças de modo que se tenha um completo entendimento de

um determinado texto ou discurso (GRISHMAN, 1999). O termo *discurso* significa texto com múltiplas sentenças.

A teoria apresentada por Kamp e Reyle (1993), denominada de *Discourse Representation Theory (DRT)*, inclui uma técnica lingüística cujo princípio é interpretar seqüências de sentenças, i.e., um discurso. A teoria é concebida de forma a articular as condições de verdade de uma sentença em linguagem natural e no fenômeno da interpretação da referida linguagem, onde as interpretações das sentenças e discursos são construídas em forma de estruturas abstratas denominadas *Discourse Representation Structures (DRS)*. As referidas construções representam, de uma forma geral, unidades lingüísticas grandes, sendo o processamento realizado sentença por sentença, onde cada sentença processada contribui com a *DRS* já construída pelas anteriores, i.e., uma nova *DRS* é gerada. O segmento do discurso termina com a última sentença processada. A natureza incremental da interpretação está diretamente vinculada com um aspecto fundamental do discurso que é a *natureza coesiva semântica*, ou seja, entender qual informação deve ser adicionada por uma nova sentença de um discurso a uma estrutura de informação obtida de sentenças anteriores. A linguagem que permite formular as *DRS* é uma variante da lógica de predicados, de modo que cada *DRS* pode ser vista como uma fórmula da referida lógica. A gramática adotada por esta técnica é a *generalized phrase structure grammar*, i.e., uma versão da gramática de estrutura sintagmática (KAMP e REYLE, 1993). A figura 2.40 ilustra a tradução de linguagem natural em lógica de primeira ordem.

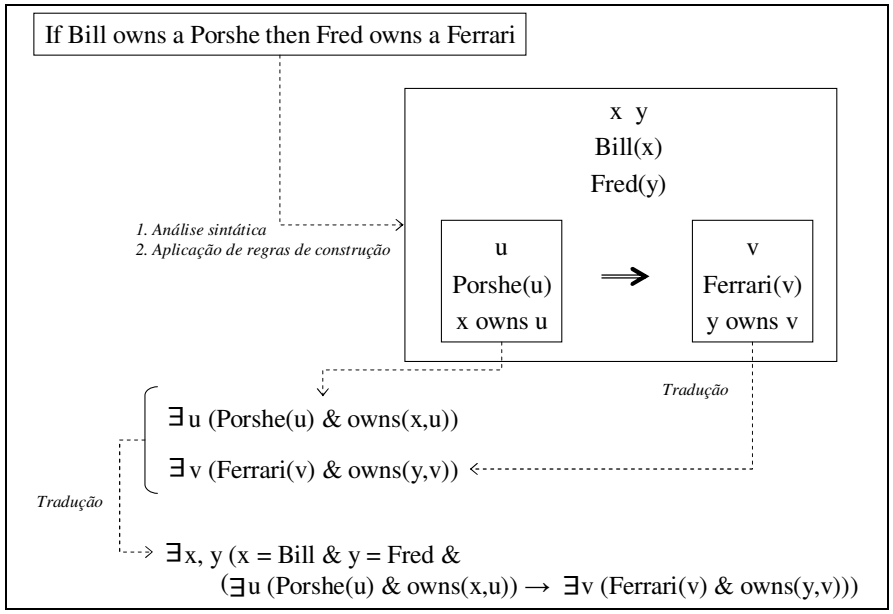


Figura 2.40: Exemplo de *DRS*.

Uma característica própria da DRT é que a interpretação semântica não gera um relacionamento direto entre expressões e um modelo da realidade, sendo criado um nível intermediário de representação semântica onde a informação que convém do discurso é armazenada. A DRT possui, também, um mecanismo composto por um conjunto de regras que convertem estruturas sintáticas em DRS, sendo as referidas regras denominadas de *construction rules* (GAMUT, 1991).

2.3 Representação do Conhecimento e Raciocínio

Representação do conhecimento e raciocínio, de uma forma simplificada, é o estudo do pensamento como um processo computacional. Representação é o relacionamento entre dois domínios, onde o primeiro é destinado a ser representado pelo segundo. O primeiro domínio é, usualmente, mais concreto, imediato ou acessível que o segundo, tal como uma sentença em linguagem natural representada por símbolos da lógica de primeira ordem. O conhecimento, por sua vez, é o relacionamento entre um conhecedor, ou agente, e uma proposição, ou seja, a idéia expressada por uma sentença simples declarativa. O raciocínio é a manipulação formal dos símbolos, que representam uma coleção de proposições aceitas como verdadeiras, para produção de representações de novas proposições, tal como a inferência lógica onde a sentença final representa uma conclusão lógica de proposições representadas por sentenças iniciais (BRACHMAN e LEVESQUE, 2004).

A representação do conhecimento é uma área multidisciplinar que aplica teoria e técnica de três outros campos (SOWA, 2000): a *lógica*, que provê a estrutura formal e as regras de inferência; a *ontologia*, que define qual tipo de coisa que existe no domínio da aplicação; e a *computação*, que suporta as aplicações que diferenciam a representação do conhecimento de pura filosofia. A *lógica* possui uma sintaxe bem definida, mas necessita de um vocabulário para descrever coisas do mundo real. A *ontologia* preenche este vazio, pois é o estudo de todos os tipos de entidades, abstratas ou concretas, do mundo; fornece os predicados do cálculo dos predicados e os identificadores das caixas e círculos do gráfico conceitual. A partir da caracterização dos três campos apresentados, o referido autor define representação do conhecimento como a aplicação da lógica e da ontologia na tarefa de construir modelos computáveis de um determinado domínio. Uma segunda definição determina que a representação do conhecimento é o campo de estudo relacionado com o uso formal de símbolos na representação de uma coleção de proposições aceitas como verdadeiras por um determinado agente (BRACHMAN e LEVESQUE, 2004).

A seguir estão listados os princípios da representação do conhecimento e seu papel na inteligência artificial (SOWA, 2000).

Representação do conhecimento é um substituto (surrogate). Os objetos, eventos e relacionamentos, que não podem ser armazenados diretamente em um computador, são representados por símbolos que servem de *surrogates* de coisas externas. Os símbolos e os respectivos vínculos, por sua vez, formam um modelo do sistema externo. Por meio da manipulação dos *surrogates* internos, o computador pode simular o sistema externo ou raciocinar sobre ele.

Representação do conhecimento é um conjunto de comprometimentos ontológicos. A partir do conceito de ontologia como o estudo da existência, são determinadas, para uma base de conhecimento, as categorias de coisas que existem e podem existir em um domínio de aplicação.

Representação do conhecimento é uma teoria fragmentária do raciocínio inteligente. É implementada por meio da descrição dos comportamentos e interações entre conceitos do domínio.

Representação do conhecimento é um meio eficiente de computação. Além de representar o conhecimento, um sistema de inteligência artificial deve codificar o referido conhecimento de modo que possa ser processado eficientemente.

Representação do conhecimento é um meio de comunicação entre o engenheiro de conhecimento e o especialista no domínio. Permite que o especialista possa entender a linguagem e as notações utilizadas pelo engenheiro, bem como verificar se as definições e as regras representam uma teoria realística do domínio.

Todo Sistema de Representação do Conhecimento e Raciocínio é lógico, devendo ter uma sintaxe bem projetada, uma semântica composicional e procedimentos de inferência (SHAPIRO, 2000). Brachman e Levesque (2004) afirmam que o papel da representação simbólica na construção de sistemas possui duas propriedades importantes: a visão externa de que os sistemas são fundamentados em proposições e que os referidos sistemas são projetados para comportar-se de acordo com as representações simbólicas.

2.3.1 Engenharia do Conhecimento

A *engenharia do conhecimento* é a aplicação da lógica e da ontologia na tarefa de construir modelos computáveis de algum domínio com um determinado propósito (SOWA, 2000). Para Russel e Norvig (2004) e Brachman e Levesque (2004), a *engenharia do conhecimento* é o processo geral de construção de uma base de conhecimento. Um

engenheiro de conhecimento é alguém que investiga um domínio específico, aprende quais conceitos são importantes nesse domínio e cria uma representação formal dos objetos e relações no domínio. A figura 2.41 ilustra a representação de parcela de uma base de conhecimento implementada em Prolog; não há uma cláusula específica representando o que o sistema "aceita como verdadeiro" que a cor da grama é verde, mas por inferência lógica é possível que o sistema deduza que "a cor da grama é verde".

```
cor(X, Y) :- composição(X,Z) ,  cor(Z,Y)
composição(grama, vegetação).
cor(vegetação, verde).
cor(neve, branca).
cor(céu, azul).
```

Figura 2.41: Exemplo de uma base de conhecimento em Prolog.

O processo da *engenharia do conhecimento* proposto por Russel e Norvig (2004) inclui as seguintes atividades:

1. *Identificar a tarefa*. O engenheiro de conhecimento determina a variedade de questões que a base de conhecimento admitirá e os tipos de fatos disponíveis para cada instância específica do problema.

2. *Agregar o conhecimento relevante*. O engenheiro de conhecimento realiza o processo de *aquisição do conhecimento*. Nesta etapa o conhecimento não é representado formalmente, havendo somente um entendimento do escopo da base de conhecimento determinada pela tarefa e o entendimento de como o domínio realmente funciona.

3. *Definir um vocabulário de predicados, funções e constantes*. Inclui a conversão dos conceitos importantes do nível de domínio para o nível da lógica, sendo o resultado conhecido como ontologia do domínio.

4. *Codificar o conhecimento geral do domínio.* O engenheiro de conhecimento escreve os axiomas correspondentes a todos os termos do vocabulário.

5. *Codificar uma descrição da instância específica do problema*. Envolve a escrita de sentenças atômicas simples sobre instâncias de conceitos da ontologia. Uma *base de conhecimento desincorporada* é suprida com sentenças adicionais.

6. *Formular consultas ao procedimento de inferência e obter resposta*.

7. *Depurar a base de conhecimento*. Determina se o procedimento de inferência é consistente.

Newell (1981) descreve que um sistema inteligente pode ser entendido em dois diferentes níveis. O primeiro, denominado de *nível de conhecimento*, permite definir as

questões relacionadas com a adequação da linguagem de representação e as características de suas relações de vínculos lógicos, e o segundo, denominado de *nível de símbolo*, permite definir as questões relacionadas com a arquitetura computacional, as propriedades das estruturas de dados e os procedimentos de raciocínio, incluindo a complexidade de seus algoritmos. O nível de conhecimento permite que um agente racional possa ser descrito e analisado em um nível abstrato definido pelo conhecimento que possui e não pelos programas que executa, ou seja, um sistema inteligente pode ser construído simplesmente informando o que ele precisa conhecer. A construção da base de conhecimento ocorre por meio da inserção das sentenças que representam o conhecimento que o engenheiro de conhecimento tem do ambiente, sendo que, segundo Russel e Norvig (2004), o projeto da linguagem de representação para facilitar a expressão deste conhecimento sob a forma de sentenças é denominado de *abordagem declarativa* para a construção de sistemas, em contraste com a *abordagem procedural* que codifica comportamentos desejados diretamente como código de programa.

2.3.2 Linguagens de Representação do Conhecimento e Raciocínio

Antes que um sistema possa raciocinar, aprender, planejar ou explicar, o mesmo deve ser capaz de formular as idéias envolvidas por meio de uma *linguagem de representação*. Segundo Russel e Norvig (2004), o que falta às linguagens de programação, e.g., C++ e Java, é algum mecanismo geral para derivar fatos a partir de outros fatos, pois cada atualização em uma estrutura de dados é feita por um procedimento específico do domínio. Essa *abordagem procedural* pode ser comparada com a natureza declarativa da lógica, em que o conhecimento e a inferência, inteiramente independente de domínio, estão separados. A denominação de *abordagem declarativa* justifica-se porque sua semântica se baseia em uma relação-verdade entre sentenças e mundos possíveis.

A lógica permite que um subconjunto da linguagem natural seja precisamente formulado para ser expresso em forma computável. Neste contexto destacam-se duas linguagens de representação de conhecimento: a lógica de primeira ordem (SOWA, 2000), (RUSSEL e NORVIG, 2004), (GRISHMAN, 1999), (ALI, 1993) e (BRACHMAN e LEVESQUE, 2004) e a cláusula definida, ou *Horn clause,* sendo esta um subconjunto da LPO (BRACHMAN e LEVESQUE, 2004), (GRISHMAN, 1999), (GAL *et al.*, 1991) e (RUSSEL e NORVIG, 2004).

Uma linguagem lógica de representação do conhecimento deve possuir quatro aspectos essenciais (SOWA, 2000):

• vocabulário, que inclui os símbolos lógicos independentes de domínio, tais como os quantificadores, as constantes dependentes de domínio, as variáveis, cujo escopo da aplicação é governado por quantificadores, e a pontuação, que permite a separação de grupos de símbolos;

• sintaxe, que inclui as regras de gramática ou regras de formação que determinam como os símbolos são combinados para formar sentenças gramaticais;

• semântica, que determina como as constantes e variáveis são associadas com coisas no universo do discurso;

• regras de inferência, que incluem regras que determinam como um modelo pode ser inferido de outro.

A LPO tem como compromisso ontológico, isto é, o que a referida linguagem pressupõe sobre a natureza da realidade, a existência de fatos, objetos e relações (RUSSEL e NORVIG, 2004). Não cabe no presente trabalho descrever a lógica de primeira ordem, cujas referências incluem (RUSSEL e NORVIG, 2004), (SOWA, 2000), (BRACHMAN e LEVESQUE, 2004) e (SOUZA, 2002).

As *cláusulas de Horn* formam um subconjunto da LPO que possibilita um melhor trato computacional, particularmente na implementação do algoritmo de inferência, sendo, do ponto de vista de representação, um subconjunto suficientemente expressivo para alguns propósitos (BRACHMAN e LEVESQUE, 2004). Tal restrição é corroborada por Sowa (2000) ao afirmar que alguns lógicos modernos deliberadamente limitam o expressivo poder da LPO a um subconjunto mais facilmente computável, sendo esta assertiva exemplificada por meio da linguagem Prolog que é baseada em cláusulas definidas de Horn. Russel e Norvig (2004) também afirmam que as bases de conhecimento muitas vezes só contêm *cláusulas de Horn*.

Uma fórmula clausular é um conjunto finito de cláusulas, sendo a cláusula um conjunto finito de literais. O que caracteriza a fórmula clausular é a conjunção de suas cláusulas e a disjunção de seus literais, tal como exemplifica a seguinte fórmula em lógica proposicional: *((p v r) ^ (¬q v r))* (BRACHMAN e LEVESQUE, 2004). A cláusula $(\neg p_1,...,\neg p_n, q)$, que representa uma disjunção com apenas um literal positivo, é uma *cláusula definida de Horn* positiva que pode ser lida como "*se* p_1 *e ... e* p_n*, então q*" e escrita como "$p_1 \wedge ... \wedge p_n => q$". O literal positivo é denominado de cabeça e os literais negativos formam o corpo da cláusula. Uma cláusula definida sem literais negativos, i.e., que simplesmente afirma uma dada proposição, é também denominada de *fato.* De acordo

com Russel e Norvig (2004), a restrição a apenas um literal positivo tem sua importância justificada por três motivos:

- as cláusulas definidas são a base da programação em lógica;
- a inferência pode ser realizada por meio de algoritmos de encadeamento para a frente e para trás, sendo os mesmos fáceis de serem acompanhados por seres humanos;
- a decisão de conseqüência lógica com cláusulas de Horn pode ser feita em tempo linear em relação ao tamanho da base, o que, na prática, significa que a inferência lógica é muito econômica em muitas bases de conhecimento proposicionais.

O raciocínio é a manipulação formal dos símbolos, que representam uma coleção de proposições aceitas como verdadeiras, para produção de representações de novas proposições. O raciocínio lógico, ou *inferência lógica*, envolve o conceito de conseqüência lógica entre sentenças, o que significa que a sentença final representa uma conclusão lógica de proposições representadas por sentenças iniciais, ou seja, dada uma base de conhecimento *BC* e uma sentença qualquer α, determinar se $BC \models \alpha$, isto é, *se α é conseqüência lógica de BC* ou que *BC vincula logicamente α*. Cabe ressaltar que cada sentença é expressa por uma linguagem de representação e representa alguma asserção sobre o mundo. Um processo de raciocínio é logicamente consistente (*logically sound*) sempre que o mesmo produz α; desta forma, é garantido que α é conseqüência lógica, ou seja, *BC* deriva apenas sentenças permitidas. Um processo de raciocínio é logicamente completo (*logically complete*) sempre que o mesmo garante produzir α, sempre que α está vinculado, ou seja, deriva qualquer sentença permitida (BRACHMAN e LEVESQUE, 2004).

A seguir será ilustrado um exemplo apresentado por Brachman e Levesque (2004) que exemplifica parcialmente o formalismo da LPO. A questão é, a partir da figura 2.42, determinar se existe um bloco verde diretamente sobre um bloco não-verde.

Os fatos em *S*, que representam o domínio do problema, são:

$$\{ O(a,b), O(b,c), V(a), \neg V(c)\}, \text{ onde O é o predicado "sobre" e V, "verde".}$$

A sentença α em LPO que representa o objetivo buscado, ou seja, determinar se existe um bloco verde sobre um bloco não verde, corresponde a:

$$\exists x \exists y.\ V(x) \wedge \neg V(y) \wedge O(x,y).$$

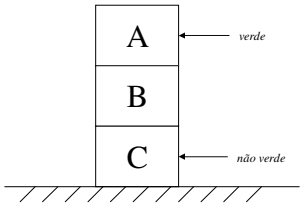


Figura 2.42: Pilha com três blocos.

A sentença α deverá ser verdadeira em *S,* cuja representação é $S \models \alpha$, para tal todas as interpretações que satisfazem *S*, também devem satisfazer α. Existem duas interpretações (*I)* possíveis:

1. Bloco "B" é verde: I $\models$ V(b), tem-se em S, I $\models$ V(b) ^ ¬V(c) ^ O(b,c), que está de acordo com Ǝx Ǝy. V(x) ^ ¬V(y) ^ O(x,y).
2. Bloco "B" não é verde: I $\models$ ¬V(b), tem-se em S, I $\models$ V(a) ^ ¬V(b) ^ O(a,b), que está de acordo com Ǝx Ǝy. V(x) ^ ¬V(y) ^ O(x,y).

Conclui-se que ambas as interpretações, bloco *B* verde e não-verde, são verdadeiras, logo α é uma conseqüência lógica de *S*, ou seja, $S \models \alpha$. Observa-se que determinar o que é implícito em uma coleção de fatos requer formas engenhosas de raciocínio.

A lógica possui, também, uma propriedade denominada composicionalidade, onde o significado de uma sentença é função do significado de suas partes, o que torna mais fácil o sistema de raciocínio. De acordo com Kamp e Reyle (1993), a lógica é a ciência da inferência, sendo baseada em duas hipóteses básicas:

• na inferência correta, as premissas devem ser verdadeiras e a conclusão inferida deve ter uma relação lógica com as premissas de modo a garantir a transferência da verdade contida nestas à conclusão;

• a relação entre premissas e conclusão, que garante a transferência da verdade, é uma relação formal denominada conseqüência lógica ou vínculo lógico, que pode ser analisada como uma relação entre formas lógicas.

As conclusões de um processo de raciocínio oferecem a garantia de serem verdadeiras em qualquer mundo nas quais as premissas são verdadeiras; em particular, se uma base de conhecimento *BC* é verdadeira no mundo real, qualquer sentença α derivada de *BC* por um procedimento de inferência consistente também será verdadeira no mundo real, tal qual ilustra a figura 2.43 (RUSSEL e NORVIG, 2004). Em síntese, raciocínio é o processo de construir novas sentenças a partir de sentenças antigas. O raciocínio lógico deve assegurar que as novas sentenças realmente representem aspectos do mundo real decorrentes dos aspectos que as sentenças já existentes retratam.

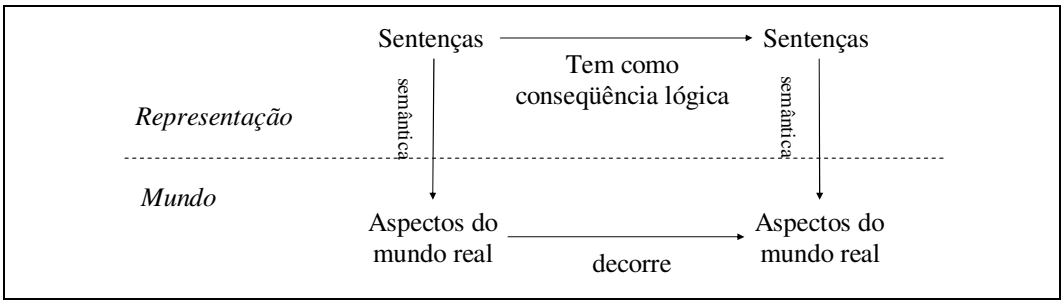


Figura 2.43: Correspondência entre o mundo real e a representação do conhecimento.

O procedimento denominado *Resolução* permite automatizar o raciocínio dedutivo de forma completa e consistente, com o objetivo de determinar se uma sentença ou fórmula α é verdadeira ou não em uma base de conhecimento *BC*, ou seja, $BC \models \alpha$ (BRACHMAN e LEVESQUE, 2004). A resolução é uma regra de inferência (RUSSEL e NORVIG, 2004), (CLOCKSIN e MELLISH, 2003) e (SOWA, 2000). A aplicabilidade do procedimento de *resolução* em uma base de conhecimento implementada em *LPO* apresenta sérias dificuldades computacionais, sendo o referido procedimento mais gerenciável quando aplicado em cláusulas definidas de Horn (BRACHMAN e LEVESQUE, 2004). Russel e Norvig (2004) afirmam, também, que a completeza da resolução faz desta um método de inferência muito importante. Entretanto, em situações práticas, a capacidade total da resolução não é necessária, o que faz com que as bases de conhecimento muitas vezes sejam compostas por cláusulas definidas de Horn.

O procedimento de resolução aplicado às cláusulas definidas de Horn, denominado de *resolução Selected literals, Linear pattern, over Definite clause (SLD)* pode ser implementado por meio de dois algoritmos de busca completos (BRACHMAN e LEVESQUE, 2004): o encadeamento para frente (*forward chaining*) e o encadeamento para trás (*backward chaining*). O algoritmo de *encadeamento para frente* trabalha a partir dos fatos em uma base de conhecimento para os objetivos e o algoritmo de *encadeamento para trás* funciona de modo inverso, ou seja, a partir dos objetivos para os fatos na base de conhecimento, também conhecido como *depth-first*, porque busca resolver novos objetivos somente depois de resolver um objetivo anterior, sendo chamado, ainda, de *left-to-right* porque os objetivos são experimentados da esquerda para a direita. Prolog possui um motor de inferência que implementa o algoritmo de encadeamento para trás (CLOCKSIN e MELLISH, 2003), (BRACHMAN e LEVESQUE, 2004), (RUSSEL e NORVIG, 2004) e (GAL *et al.*, 1991).

Em uma representação clausular, tal como a lógica de Horn apresentada anteriormente, uma base de conhecimento pode ser freqüentemente separada em dois tipos de cláusulas:

fatos e regras. Os fatos são usados para determinar as verdades básicas de um domínio e as regras para estender o vocabulário e expressar novas relações. Um engenheiro de conhecimento pode expressar fatos em LPO sem ter a necessidade de saber como o conhecimento será inferido por um procedimento automatizado em um provador de teorema, tendo em vista que o referido mecanismo tentará todas as aplicações lógicas possíveis sobre todo o conhecimento existente na base de conhecimento na busca de uma resposta ao respectivo questionamento. Quando existe o entendimento da estrutura do domínio ou do problema deve-se evitar usar fatos em qualquer forma ou ordem, sendo sugerido, nestes casos, que o usuário controle o processo de raciocínio por meio de orientações aos mecanismos de raciocínio baseadas no referido domínio, ou seja, é necessário pensar como fazer uso mais efetivo das regras em uma base de conhecimento, sendo o referido processo denominado por Brachman e Levesque (2004) de *Controle Procedimental do Raciocínio*.

Importante a investigar quando se tem uma coleção de proposições é se algo interessante resulta das referidas proposições, i.e., deve-se investigar quais conseqüências as proposições possuem. As proposições consideradas como argumentos verdadeiros são denominados *axiomas ou hipóteses* e as proposições onde se busca a conseqüência lógica a partir dos referidos axiomas denominam-se *teoremas.* Dos referidos conceitos surge a atividade denominada de *prova de teorema* com o objetivo de derivar conseqüências lógicas a partir de proposições dadas, particularmente por meio do princípio da *resolução* que, como regra de inferência, permite automatizar o procedimento de raciocínio dedutivo. De uma forma simplificada, a tarefa de provar teorema é definida como: dada uma fórmula, mostre que aquela fórmula é um teorema, ou seja, a fórmula é sempre verdadeira independente da interpretação dos símbolos que ocorrem na fórmula (BRATKO, 2002).

2.3.3 Estruturas de Representação do Conhecimento

Em 1909, Charles Pierce propôs uma notação gráfica de nós e arcos denominada de grafos existenciais, sendo por ele referida de “a lógica do futuro”. O termo *rede semântica*, i.e., *estrutura de representação do conhecimento*, foi utilizado pela primeira vez em 1961 no sistema de *Margareth Masterman* na Universidade de Cambridge (SOWA, 2002b).

Sowa *et al.*(1991) definem rede semântica como uma estrutura para representação do conhecimento equivalente a um modelo de nós e arcos interconectados, sendo que os nós representam conceitos de entidades, atributos, eventos e estados, e os arcos representam relacionamentos entre os conceitos. Embora as redes semânticas possuam conceitos

comuns, as mesmas divergem em alguns pontos, tais como, em questões filosóficas de significados, métodos de representar quantificadores e operadores da lógica, técnicas de manipulação das redes e execução de inferência, e estilos de diagramação. Alguns sistemas enfatizam a habilidade de destacar proposições e raciocínio e outros enfatizam a definição de novos conceitos na hierarquia de tipo, o que caracteriza a existência de distintas famílias de estruturas de representação do conhecimento. Cabe ressaltar que Stuart Shapiro, quem implementou a primeira rede semântica com suporte integral à lógica de primeira ordem, acredita que uma rede corretamente estruturada pode suportar importantes tipos de raciocínios "subconscientes" que não são diretamente representados em uma forma linear de lógica (SOWA *et al.*, 1991). Shastri afirma que, de uma forma geral, é possível traduzir uma rede semântica em uma linguagem não gráfica e vice-versa (SOWA *et al.*, 1991).

A rede semântica é um sistema lógico cuja sintaxe é definida por sua notação. O respectivo conteúdo semântico é determinado de forma independente por meio do significado dos seus nós e arcos, o que pode ocorrer de modo intencional ou extensional. A técnica intencional inclui o significado básico de um termo em si mesmo e a extensional inclui o significado de um conjunto de coisas referidas pelo mesmo. A técnica extensional usualmente aplicada para a lógica é a denominada *Teoria do Modelo*, que, aplicada à lógica dos predicados, inclui a construção de um modelo como uma estrutura de dados abstrata composto de entidades do domínio do discurso e um conjunto de relações definidas sobre as referidas entidades (SOWA *et al.*, 1991).

De acordo com Analyti *et al.* (1998), por semântica de uma rede entende-se a informação, explícita ou derivada, que uma rede semântica contém, sendo que diversos modelos de redes semânticas são utilizados somente para organizar informações, sem o respectivo formalismo que defina o que seja a semântica de uma rede semântica. As redes semânticas, segundo Shapiro (1978), vêm sendo desenvolvidas como um formalismo de representação do conhecimento e destaca as duas formas de inferência nas referidas redes: a primeira, denominada inferência baseada em caminho, é composta de expressões fundamentadas no cálculo relacional, e a segunda, denominada de inferência baseada em nó, incorpora métodos de representação de regras utilizando a LPO. Russel e Norvig (2004) consideram a rede semântica um sistema especialmente projetado para organizar e raciocinar com categorias, que oferece auxílio gráfico para visualização de uma base de conhecimento, sendo também uma forma de lógica. Para Sowa (2002b), rede semântica é uma representação gráfica declarativa que pode ser utilizada para representar o conhecimento e como suporte automatizado de raciocínio sobre o referido conhecimento.

O sistema de gráfico conceitual, apresentado por Sowa (2000), é uma combinação da lógica de Pierce com a rede semântica utilizada na inteligência artificial, sendo um sistema baseado em pura lógica que representa diretamente as estruturas lógicas, tal qual ilustra a figura 2.44. Para o referido autor, a análise de uma sentença em linguagem natural inclui, a partir da análise sintática, uma interpretação semântica que permite a tradução da sentença em um gráfico conceitual que representa o respectivo significado e, finalmente, o gráfico é traduzido em fórmulas do cálculo de predicados.

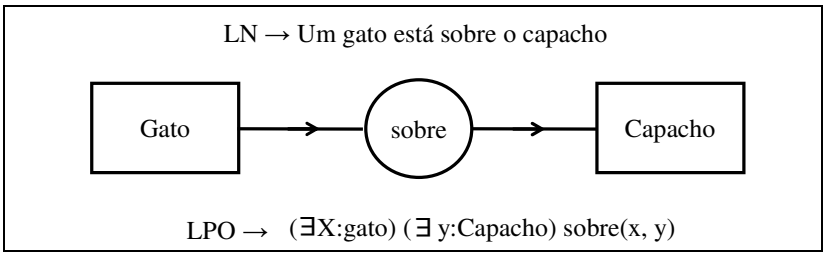


Figura 2.44: Exemplo de Gráfico Conceitual.

Allen e Frisch (1982) propõem uma rede semântica que considera uma variante sintática da LPO e justifica com dois argumentos: primeiro, a rede semântica pode ser mapeada em uma representação em LPO logicamente isomórfica, onde é considerado o conteúdo da representação semântica em vez da representação sintática e, segundo, facilita a recuperação de fatos relevantes por meio de um mecanismo especializado de inferência que permite a recuperação do conhecimento. A figura 2.45 ilustra os referidos argumentos.

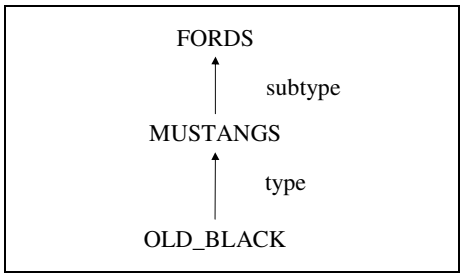


Figura 2.45: Exemplo de rede semântica (ALLEN, 1982).

As seguintes sentenças em LPO são logicamente isomórficas à referida rede:

$$\forall x\ MUSTANGS\ (x) \rightarrow FORDS(x)$$

$$MUSTANGS(OLD\_BLACK)$$

A forma e o conteúdo da rede são capturados por meio dos dois predicados que capturam o significado dos vínculos existentes na referida rede, sendo as sentenças a seguir representativas da semântica existente na mesma:

(a) SUBTYPE(FORDS, MUSTANGS)

(b)TYPE(MUSTANGS, OLD_BLACK)

A partir dos predicados *SUBTYPE* e *TYPE* é questionado como os mesmos podem raciocinar, ou seja, realizar inferência. O axioma a seguir, denominado como *axioma built-in*, que permite definir capacidades de recuperação de conhecimento, é uma solução ao citado questionamento, onde o predicado *SUBTYPE* é transitivo:

$$(c)\ \forall t_1, t_2, t_3\ SUBTYPE(t_1, t_2) \wedge SUBTYPE(t_2, t_3) \rightarrow SUBTYPE(t_1, t_3)$$

O projeto de linguagem de representação de rede semântica de Allen e Frisch (1982) inicialmente identifica as denominadas estruturas primitivas de conhecimento que compõem a base de fatos, tal qual em *(a)* e *(b)* e, a partir destas, são construídos os axiomas *built-in*, tal qual em *(c)*, que permitem implementar a capacidade de raciocínio por meio de inferências.

A *rede semântica estendida*, proposta por Deliyanni e Kowalski (1979), pode ser vista como uma variante da forma clausular da lógica, tendo em vista que a semântica da referida rede é idêntica a da forma clausular da lógica, e cuja correspondência provê a rede com semântica e com regras de inferência. A referida rede é, também, uma estrutura abstrata de dados que permite uma organização dos mesmos de modo que possam ser acessados de forma eficiente na busca por uma solução a uma determinada consulta. Conclusões e condições são representadas por arcos distintos, sendo uma cláusula representada por uma representação de rede de suas conclusões (traço cheio) e condições (traço vazado), tal qual ilustra a figura 2.46, cuja semântica é: *todo animado é um vegetal ou um animal*. A hierarquia *Isa* e *Part-of* , que existem em diversas redes, estão presentes na referida rede semântica estendida.

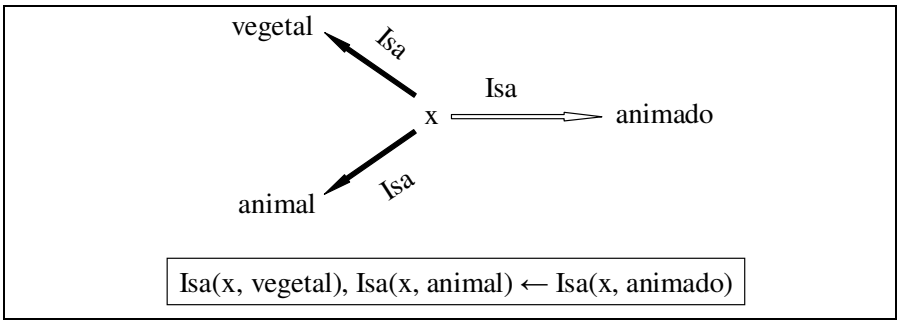


Figura 2.46: Diagramação da rede semântica estendida.

Semantic Network Processing System (SNePS), é um sistema inteligente projetado para representar fatos aceitos por um *agente cognitivo*, tendo como objetivo representar, por meio da lógica, tudo que possa ser expresso por meio de linguagem natural (SHAPIRO e RAPAPORT, 1990) e (SHAPIRO, 2000). SNePS constitui a "linguagem de pensar" do referido agente. SNePS é um sistema de representação do conhecimento e raciocínio

baseado no paradigma de rede semântica, ou seja, possui uma representação gráfica composta por nós e arcos, sendo implementado em *Common Lisp Object System* (CLOS). Os nós representam entidades que, por sua vez, representam objetos individuais, classes de objetos, pessoas, propriedades, abstrações, ações, tempo e proposições, o que a caracteriza como uma rede semântica proposicional tendo em vista estarem as proposições nos citados nós. Os arcos representam relações entre nós.

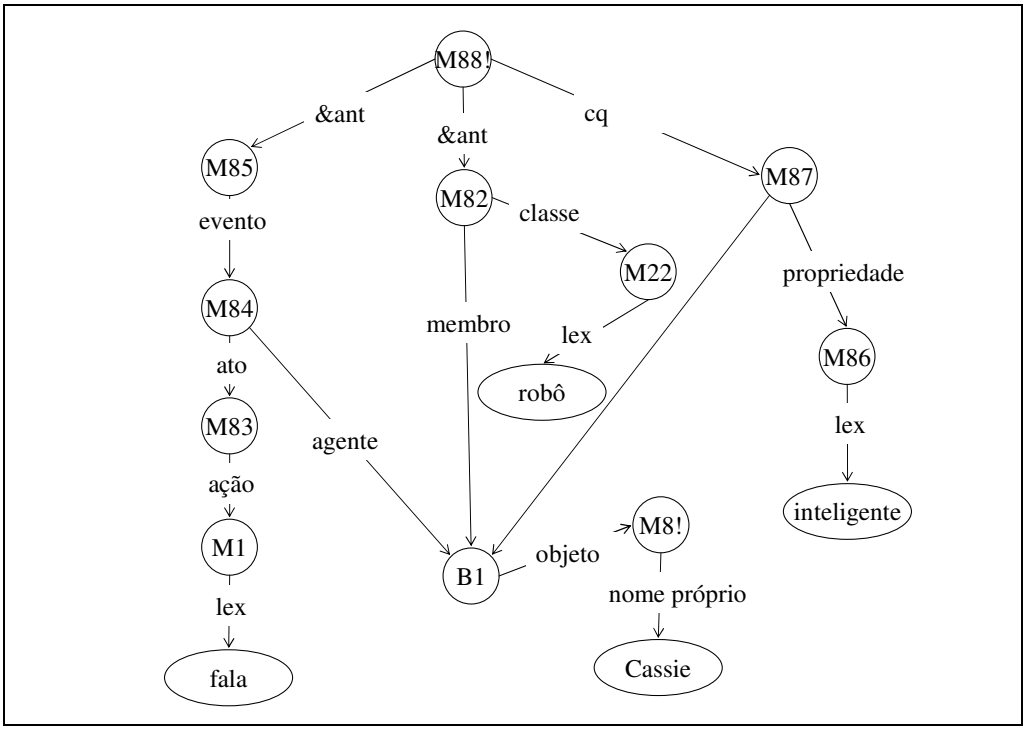


Figura 2.47: Exemplo de Rede Semântica em SNePS.

A figura 2.47 ilustra um exemplo de rede semântica de acordo com o formalismo utilizado em SNePS; o nó *M88!* é declarativo e representa a proposição *Se Cassie é um robô que fala, então Cassie é inteligente*. A sintaxe do SNePS é definida em termos de nós, relações, ligações (par <relação, nó>), condutores (par <relação, um conjunto de nós>) e conjuntos de condutores. A semântica é representada por uma taxonomia de classes semânticas dos nós, que inclui a classe genérica *entidade* que se especializa em *individual, ação* e *proposição*, sendo esta última, especializada em *regra*, *proposição genérica* e *proposição simples*. O nó de proposição de regra permite aplicar a inferência baseada em nó, ou seja, a inferência que utiliza nós, análogos às formulas não-atômicas da LPO, na representação de regras do domínio (*M88!*) de modo a permitir a inferência de um novo nó de proposição (*M87*) declarativa a partir de nós de proposição declarativa previamente definidos (*M82* e *M85*). O nó de proposição genérico pode ser utilizado em inferência baseada em subordinação na resolução de proposições genéricas do tipo *Qualquer robô*

que fala, é inteligente. O nó de proposição simples é todo nó de proposição que não é de regra nem genérica (SHAPIRO *et al.*, 2003). O nó individual é todo nó que não é proposicional nem de ação. Como exemplos de semânticas de domínio planejada dos nós da figura 2.47 , tem-se a semântica do nó *M8!* como sendo *a proposição que o nome de Cassie é "Cassie"*, a semântica do nó *M1, a ação de falar* e a semântica do nó *M88!*, *Se Cassie é um robô que fala, então Cassie é inteligente*.

Embora as notações existentes na UML não sejam comumente chamadas de redes semânticas, Sowa (2002b) as classifica como tal e justifica no fato de que central à UML está uma rede de definição de tipos de objetos, o Diagrama de Objetos, que não é citado nominalmente pelo autor, com vínculos de tipo-subtipo, com atributos que servem para distinguir um tipo de seus subtipos e com mecanismo de herança entre atributos. Outra rede, o Diagrama de Classe, que também não é citada nominalmente pelo autor, é um grafo relacional que permite representar meta informação. A linguagem *Object Constraint Language* (OCL) é considerada uma versão da LPO com uma notação similar a algumas linguagens orientadas a objeto. A discussão sobre a semântica de algumas notações da linguagem UML serão tratadas no Capítulo 3 do presente trabalho.

2.4 Programação Lógica

A linguagem Prolog foi desenvolvida no início da década de 70 pelo pesquisador francês Alain Colmerauer e associados, com a finalidade de desenvolver uma linguagem de programação prática que permitisse a um programador especificar tarefas em lógica, ou seja, informar ao computador o que é verdade e, a partir das referidas verdades, obter conclusões; característica esta que permite gerar programas com uma forte semântica *declarativa* que determina *o que* um programa deve computar, ou seja, dizer ao computador o que é verdade e solicitar que o mesmo teste e infira conclusões (CLOCKSIN e MELLISH, 2003), em contraste com construções de programação convencional que determinam o que a máquina deve fazer e quando.

Prolog é a linguagem de programação mais amplamente utilizada em lógica, sendo usada, entre outras aplicações, em análise de linguagem natural (RUSSEL e NORVIG, 2004). Gal *et al.* (1991) afirmam que existem várias técnicas para tratar linguagem natural, mas a adoção de uma técnica baseada em lógica está em concordância com muitos outros pesquisadores, tendo sido selecionada a linguagem Prolog como a mais apropriada para a referida técnica, particularmente, por causa das vantagens que possui para a programação

lingüística, tais como, modularidade, declaratividade e concisão. Prolog é uma linguagem de programação para computação simbólica, não-numérica, sendo especialmente projetada para resolução de problemas que envolvam objetos e relações entre objetos (BRATKO, 2001).

Prolog possui dois níveis de significado: o declarativo, que diz respeito às relações definidas no programa, e o procedimental, que determina como as relações são avaliadas pelo sistema Prolog, ou seja, define o procedimento para execução de uma lista de objetivos em relação a um determinado programa ou, de forma simplificada, como Prolog responde a uma questão. A técnica declarativa permite uma independência relativa dos detalhes procedimentais, tendo esta uma importância prática pelo fato de serem os aspectos declarativos do programa de mais fácil entendimento do que os aspectos procedimentais. Cabe ressaltar que, em grandes programas, os aspectos procedimentais não podem ser ignorados pelo programador por razões práticas de eficiência computacional. Prolog possui um motor de inferência que implementa o algoritmo de encadeamento para trás que funciona a partir dos objetivos para os fatos na base de conhecimento, sendo também *depth-first*, porque busca resolver novos objetivos somente depois de resolver um objetivo anterior, sendo, ainda, *left-to-right* porque os objetivos são experimentados da esquerda para a direita (CLOCKSIN e MELLISH, 2003), (BRACHMAN e LEVESQUE, 2004), (RUSSEL e NORVIG, 2004) e (GAL *et al.*, 1991).

A sintaxe de um programa Prolog é a da lógica de primeira ordem com fórmulas escritas em *cláusulas definidas de Horn*, que é uma forma normal conjuntiva com um literal positivo (CLOCKSIN e MELLISH, 2003), (BRATKO, 2001), (SOWA, 2000), (BRACHMAN e LEVESQUE, 2004), (GAL *et al.*, 1991) e (RUSSEL e NORVIG, 2004). As cláusulas Prolog incluem fatos, cláusulas sempre verdadeiras, regras, cláusulas verdadeiras se alguma condição é satisfeita, e questões, uma seqüência de um ou mais objetivos a serem satisfeitos. Prolog aceita fatos e regras como um conjunto de axiomas e o questionamento de um usuário como um teorema presumido; Prolog tenta provar o referido teorema, ou seja, demonstrar que o mesmo pode ser logicamente derivado dos axiomas. As variáveis são universalmente quantificadas e a programação recursiva é um dos princípios fundamentais da programação em Prolog, por não ser possível a solução de tarefas de complexidade significante sem o uso da recursão.

Prolog é um provador de teorema e, por ser composto de cláusulas definidas de Horn, é baseado no procedimento de *resolução SLD* (BRACHMAN e LEVESQUE, 2004), (BRATKO, 2001) e (CLOCKSIN, 20033). Russel e Norvig (2004) destacam que Prolog é

uma linguagem incompleta como um provador de teorema para cláusulas definidas porque, para algumas bases de conhecimento, não consegue provar sentenças que são conseqüências lógicas, tendo em vista que um programa Prolog pode ter variações comportamentais com um mesmo significado declarativo; tal afirmativa é corroborada por Bratko (2001), tal qual ilustra a figura 2.48. Na versão 1 o programa é executado com sucesso, sendo que na versão 2 não pode encontrar a resposta. A ordem dos objetivos, dentro de uma mesma cláusula, e das cláusulas é importante, pois, no referido exemplo, ambos os programas têm o mesmo significado declarativo, mas com diferentes significados procedimentais, ou seja, existe programa declarativamente correto que não funciona na prática.

```
predecessor1(X, Z) :- parent(X, Z).
predecessor1(X, Z) :- parent(X, Y), predecessor1(Y, Z).
```

```
predecessor2(X, Z) :- predecessor2(X, Y), parent(Y, Z).
predecessor2(X, Z) :- parent(X, Z).
```

Figura 2.48: Duas versões de um mesmo programa Prolog.

O controle procedimental do raciocínio destaca a necessidade de comunicar ao provador automático de teoremas, cujo método de raciocínio é independente de domínio, algumas orientações baseadas nas propriedades do referido domínio do problema a fim de evitar, por exemplo, os problemas relacionados com a figura 2.48 (BRACHMAN e LEVESQUE, 2004). O referido controle também é examinado por Gal *et al.* (1991), Bratko (2001), Clocksin (2003) e Russel e Norvig (2004), dos quais se destacam, dentro do contexto da linguagem de programação lógica Prolog, as seguintes formas de como o conhecimento pode ser expresso a fim de prover o referido controle:

• regras de formação e estratégias de busca podem ter equivalentes lógicos mas com impactos computacionais distintos em virtude do domínio do problema;

• a especificação da ordem do objetivo pode impactar a computação do algoritmo de encadeamento para trás de acordo, também, com o domínio do problema, em virtude de serem os objetivos processados em concordância com sua ordem na respectiva cláusula, tal como em $G <= G_1 \wedge G_2 \wedge ... \wedge G_n$;

• o mecanismo de "encadeamento para trás" deve ser controlado sempre que for necessário limitar o não determinismo do Prolog, ou seja, sempre que um objetivo tenha sido atingido, o interpretador retrocede e verifica se existe uma outra forma de encontrar o mesmo objetivo;

• o conceito de negação por falha que busca provar que o objetivo *not(G)* é verdadeiro no caso da prova de *G* falhar.

Considerando que a maioria do conhecimento é expressa por fatos elementares e que os mesmos podem ser eliminados ou adicionados, em virtude das mudanças no mundo real, faz-se necessário o controle das referidas atualizações na base de conhecimento particularmente no que toca em relação à integridade no relacionamento entre os fatos.

Prolog é uma linguagem adequada à implementação da *gramática de estrutura sintagmática,* sendo, de fato, um caso especial disponível na estrutura de controle geral provida por este motor de inferência (GRISHMAN, 1999), (GAL *et al.*, 1991), (CLOCKSIN e MELLISH, 2003) e (BRATKO, 2001). A tarefa de analisar uma sentença, em relação a uma determinada gramática, pode ser declarada diretamente em termos de regras em notação Prolog. Muitas implementações Prolog provêem uma extensão notacional denominada de *Definite Clause Grammar* (DCG), ou *Gramática de Cláusula Definida*, que facilita a implementação de gramáticas formais em Prolog. Uma gramática definida em DCG é diretamente executada por Prolog como um analisador sintático, o que permite a decomposição de uma sentença em seus constituintes em um processo denominado de *parsing*. A figura 2.49 ilustra uma regra gramatical em Prolog *Standard* (a) e em DCG (b).

```
(a) sentence (S0, S) :-  noun_phrase(S0, S1), verb_phrase(S1, S)

(b) sentence --> noun_phrase, verb_phrase
```

Figura 2.49: Exemplos de sintaxe padrão e DCG em Prolog.

A DCG foi criada por Pereira e Warren e desenvolvida como ferramenta de modelagem lingüística, sendo incorporada diretamente à linguagem Prolog (CLOCKSIN e MELLISH, 2003), (BRATKO, 2001), (GAL *et al.*, 1991) e (RUSSEL e NORVIG, 2004). Possui um formalismo flexível e permite a representação de qualquer estrutura sentencial que possa ser também representada por uma gramática livre de contexto, sendo, entretanto, uma classe de gramática mais poderosa por permitir parâmetros em categorias não terminais. A denominação DCG advém do fato de que cada regra da gramática pode ser interpretada como uma cláusula definida de Horn. Formalmente, cada regra da DCG tem a forma de uma regra de produção "$X \rightarrow Y$", onde X é um símbolo não terminal e Y é uma forma sentencial. A conjunção é representada por "," e a disjunção, por meio de cláusulas distintas. As regras da forma "$X \rightarrow (Y)$" indicam a correspondência de um símbolo não terminal com um símbolo terminal do vocabulário ou item lexical *Y*. A DCG permite

resolver problemas de análise sintática, indicando se sentenças de certa linguagem natural estão corretamente estruturadas. A DCG permite implementar o fenômeno *dependência de contexto*, i.e, uma frase depende do contexto onde ocorre, tal como a concordância em número que ocorre na mesma. A figura 2.50 ilustra os argumentos *Number* e *Number1* adicionados aos símbolos terminais da gramática e que permitem implementar a citada dependência de contexto.

```
sentence (Number) --> noun_phrase(Number), verb_phrase(Number).
verb_phrase(Number) --> verb(Number), noun_phrase(Number1).
noun_phrase(Number) --> determiner(Number), noum(Number).
noun(Singular) --> [mouse].
noun(Plural) --> [mice].
...
```

Figura 2.50: Exemplo de cláusulas DCG com dependência de contexto.

2.5 Linguagem Controlada

As frases de uma língua não são catalogáveis porque são infinitas tanto em número como, teoricamente, em extensão. As palavras, sim, podem ser listadas em dicionários e mesmo assim muitas se criam ou são modificadas (AZEREDO, 2003).

O uso de uma linguagem natural irrestrita apresenta alguns problemas que corroboram o uso de uma linguagem controlada, tais como, a provável falha do léxico por não conter todas as entradas possíveis, a provável falha da análise sintática e semântica, em virtude da quantidade de construções existentes. O impacto dos problemas citados pode ser minimizado pela aplicação de uma linguagem controlada. Por outro lado, o uso de uma linguagem controlada pode tornar a linguagem natural uma linguagem formal não-natural e podem ocorrer problemas de entendimento por parte do usuário relativos às orientações de como aplicar a linguagem controlada. Sowa (2002a) e Luisa *et al.* (2003) destacam a importância da linguagem controlada para um sistema de PLN.

Sowa (2002a) propõe um *framework* (ver seção 2.1) que inclui um conceito denominado *controlled natural language*, ou seja, o uso de uma linguagem natural controlada que permite a comunicação do editor com a ferramenta. A redução do treinamento de pessoal ocorre em virtude de ser uma linguagem controlada um subconjunto da linguagem natural e as restrições e modelos estarem disponibilizados por meio de menus e com verificação imediata de sintaxe. A redução da complexidade do sistema ocorre, também, durante o processo de extração do conhecimento, pois a

ferramenta pode interagir com o editor e, ao final do processamento, pode traduzir a linguagem controlada para uma linguagem lógica ou outra linguagem de programação. A linguagem controlada permite, ainda, a autodocumentação do sistema e a geração de anotações sobre os documentos originais.

A linguagem controlada denominada *Attempto Controlled Language(ACE)* é um subconjunto do inglês que parece ser informal, mas, de fato, é formal, pois pode ser traduzida em uma linguagem lógica. ACE permite que especialistas expressem especificações com precisão e em termos de aplicação do domínio, reduzindo, desta forma, a distância entre o modelo mental do domínio do problema pelo especialista e da especificação formal e, quase, eliminando a incompreensibilidade do problema. A ACE possui um vocabulário específico do domínio, isto é, palavras com funções predefinidas como determinantes, preposições e conjunções, e palavras com conteúdo definido pelo usuário, tais como substantivos, verbos e adjetivos. Usuários podem estender e modificar o léxico por meio de uma interface simples. A ACE emprega uma gramática restrita por meio de um conjunto de regras de construções e de interpretações. As *regras de construções* definem a forma sintática das sentenças ACE, bem como as restrições de estado para remoção de imprecisões e restrição de ambigüidades. As *regras de interpretação* controlam a análise semântica das sentenças gramaticalmente corretas e resolvem ambigüidades remanescentes. Alguns exemplos de regras de construção da ACE: sentenças declarativas com a estrutura *sujeito + verbo finito (+complemento ou objeto)*; sentenças compostas construídas a partir de sentenças simples; sentenças podem relacionar-se por meio de anáforas, ou seja, pronomes pessoais ou frases substantivadas definidas; verbos podem ser utilizados somente em tempo presente simples, em voz ativa, em modo imperativo e na terceira pessoa; e verbos modais (*can, must etc.*), verbos intencionais (*hope, know etc.*) e advérbios modais (*possibly, probably, etc.*) não são permitidos. Alguns exemplos de regras de interpretação: verbos denotam eventos ou estados e a ordem textual dos verbos determina a ordem temporal dos estados e eventos associados; e frases preposicionais, em posição subordinada, sempre modificam o verbo, enquanto sentenças relativas modificam a frase substantiva antecedente imediata. O analisador sintático traduz textos em ACE em estruturas de representação do discurso, ou seja, em formato padrão da lógica de primeira ordem (LPO) e, opcionalmente, em forma clausular, tal qual ilustra a figura 2.51 (FUCHS *et al.*, 2000).

Em síntese, o objetivo de uma linguagem natural controlada é limitar a estrutura frasal utilizada por um usuário que realiza uma determinada especificação textual a fim de

permitir uma otimização no processamento computacional do texto a ser analisado, particularmente na redução da quantidade de estruturas sintáticas possíveis.

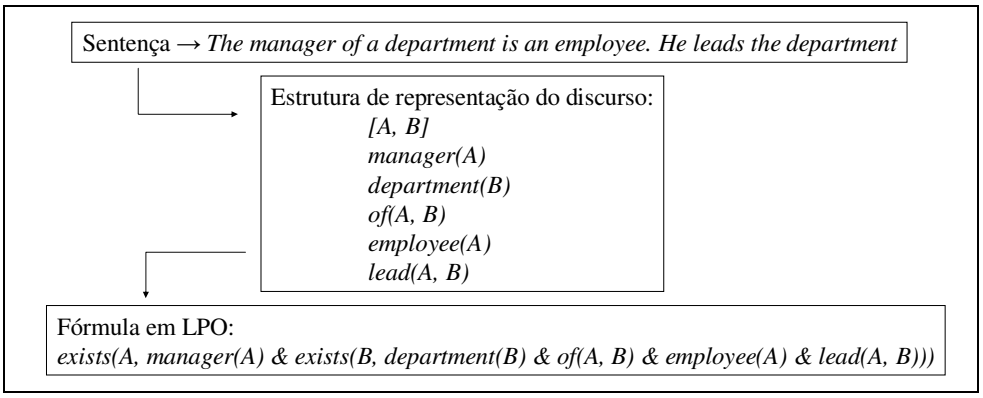


Figura 2.51: Exemplo de aplicação da linguagem ACE.

2.6 Considerações

Um sistema de PLN exige a definição de uma teoria lingüística, sendo esta o fundamento para o projeto de uma arquitetura de software que implemente a solução do problema. Uma teoria lingüística deve considerar a definição de um sistema de gramática, uma linguagem de representação do conhecimento, uma estrutura de representação do conhecimento e de níveis lingüísticos.

No contexto de um sistema de gramática, destaca-se o formalismo das gramáticas gerativas, particularmente a *PSG*. A *forma de palavra* é o método de reconhecimento de palavra automático que permite a implementação de um algoritmo de reconhecimento simples por meio da busca da palavra completa como chave no dicionário analisado (HAUSSER, 2001).

Antes que um sistema possa raciocinar, aprender, planejar ou explicar, o mesmo deve ser capaz de formular as idéias envolvidas por meio de uma linguagem de representação do conhecimento. Cabe destacar que a linguagem de representação do conhecimento Cláusulas Definidas de Horn em notação Prolog é um subconjunto da LPO que possibilita um melhor trato computacional, particularmente na implementação do algoritmo de inferência, sendo, do ponto de vista de representação, um subconjunto suficientemente expressivo para alguns propósitos (BRACHMAN e LEVESQUE, 2004). Prolog é a mais utilizada linguagem de programação lógica, sendo usada, entre outras aplicações, em análise de linguagem natural (RUSSEL e NORVIG, 2004). O nível de significado

declarativo do motor de inferência Prolog permite informar ao computador o que é verdade e, a partir das referidas verdades, obter conclusões, i.e., como um provador de teorema.

A adoção de uma estrutura de representação do conhecimento, i.e., uma rede semântica, permite que um sistema seja especialmente projetado para organizar e raciocinar com categorias e como auxílio gráfico para visualização de uma base de conhecimento, cabendo ressaltar que um projeto do nível de conhecimento por meio de uma base de conhecimento *desincorporada* permite que o engenheiro de conhecimento faça uso intensivo da *abordagem declarativa*.

A definição dos níveis lingüísticos, i.e., fonológico, morfológico, sintático, semântico e pragmático, depende basicamente dos objetivos do sistema de PLN a ser projetado. Segundo Gamut (1991), o núcleo de uma teoria lingüística está em sua teoria semântica.

Uma arquitetura de um sistema de PLN deve incluir o uso de uma linguagem controlada na escrita de especificações em linguagem natural, bem como a edição de regras sintáticas e semânticas que comporão uma determinada base de conhecimento.

Capítulo 3

Modelagem Orientada a Objeto, Engenharia de Requisitos e UML

O escopo do objetivo proposto para o presente trabalho inclui uma forte interface entre as áreas de conhecimento Engenharia de Software e Inteligência Artificial. Em relação à Engenharia de Software, aplicam-se ao referido trabalho conceitos relacionados com a Modelagem Orientada a Objeto, a Engenharia de Requisitos e a UML.

Um modelo é uma abstração de alguma coisa com o propósito de entendê-la antes de construí-la (RUMBAUGH e BLAHA, 2005). Para Maksimchuk e Naiburg (2005), modelo é uma forma visual de representar um negócio, suas regras, a utilização do sistema, as respectivas aplicações, a arquitetura do sistema e interações entre sistemas. Um conceito muito próximo do modelo é o de diagrama que é definido pelo OMG-I (2004) como uma descrição gráfica de uma coleção de elementos de modelo, sendo que cada diagrama é usado com uma finalidade específica de modo a visualizar um aspecto do sistema. Dos conceitos apresentados, um modelo pode ser composto por um ou mais diagramas, o que permitirá captar importantes aspectos do sistema em desenvolvimento. Modelagem, portanto, é o processo de construir modelos que permitirão a implementação do sistema.

A Engenharia de Requisitos, segundo Sommerville (2005), é o nome dado ao conjunto estruturado de atividades que auxiliam no entendimento dos requisitos requeridos pelos *stakeholders*, ou seja, pessoas que direta ou indiretamente utilizam o sistema, ou a informação por ele gerada, e que documentam as especificações do mesmo. A Engenharia de Requisitos faz parte do processo de desenvolvimento de software. As atividades desta engenharia, ilustradas na figura 3.1, incluem a elicitação, a modelagem, a validação e a verificação (HOFMAN e LEHNER, 2001). A elicitação determina as necessidades dos *stakeholders*, a modelagem permite melhorar o entendimento dos requisitos por meio de modelos, a validação permite a legitimação dos requisitos por parte dos *stakeholders* e a

verificação permite checar a consistência interna dos requisitos por meio de uma prova matemática ou técnica de inspeção.

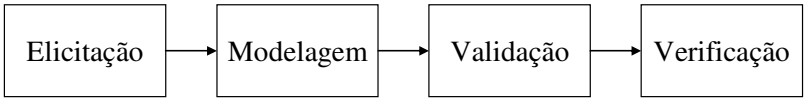


Figura 3.1: Atividades do processo da engenharia de requisitos.

A Linguagem Unificada de Modelagem, ou *Unified Modeling Language(UML),* é uma linguagem visual para especificação, construção e documentação de artefatos de sistemas. A UML é uma linguagem de propósito geral que pode ser usada com os principais métodos orientados a objeto e que pode ser aplicada a todos os domínios de aplicação. Sob a gestão do *Object Management Group (OMG)*, a UML tem emergido como uma linguagem de modelagem dominante na industria de software (OMG-I, 2004), sendo uma notação fortemente baseada em diagramas.

3.1 Modelagem Orientada a Objeto e Engenharia de Requisitos

O processo de desenvolvimento de software provê a base para uma produção de software organizada por meio de uma coleção de técnicas predefinidas e notações. Conjuntamente, a modelagem orientada a objeto permite a construção de sistemas de software complexos, onde o desenvolvedor deve abstrair diferentes vistas do sistema, construir modelos usando notações precisas, verificar se os modelos satisfazem os requisitos do sistema e, gradualmente, adicionar detalhes a fim de que os modelos possam ser implementados. A engenharia de requisitos define o processo de geração dos referidos modelos na fase comumente denominada de *Análise* do ciclo de vida de desenvolvimento de software, mas formalmente denominada de *Engenharia de Requisitos de Software*, de acordo com o corpo de conhecimento da engenharia de software (HILBURN *et al.*, 1999). De acordo com Hickey e Davis (2002), a qualidade dos requisitos é fortemente influenciada por técnicas empregadas durante a atividade de elicitação de requisitos do processo da engenharia de requisitos. As principais técnicas incluem entrevistas, questionários, descrições textuais dos casos de uso, JAD, *viewpoints*, entre outros, cujos produtos finais, particularmente nas técnicas citadas, são especificações de requisitos em linguagem natural. A partir das referidas especificações os modelos são gerados.

Nuseibeh e Eastebrook (2000) destacam a aplicação da lingüística no suporte automatizado à engenharia de requisitos, tendo em vista ter esta engenharia muito de

comunicação e que as referidas ferramentas lingüísticas podem ser aplicadas na elicitação de requisitos.

O relacionamento entre o processo de desenvolvimento de software e a modelagem orientada a objeto ocorre por meio dos modelos a serem criados em cada um dos estágios do referido processo ou pela evolução dos mesmos durante o desenvolvimento do software. O relacionamento entre o processo de desenvolvimento de software, a modelagem orientada a objeto e a engenharia de requisitos se restringe aos modelos gerados na fase de *análise*.

Maksimchuk e Naiburg (2005) descrevem um sistema por meio dos seguintes modelos: *negócio*, aplicado a processos de negócios, fluxo de trabalho e organização; *requisitos*, aplicado para captura de requisitos e comunicação; *arquitetura,* aplicado no entendimento de alto nível do sistema e interações entre diferentes sistemas; *aplicação*, aplicado na definição de arquitetura de baixo nível; e *banco de dados,* aplicado no projeto de banco de dados. Como exemplo de composição de um modelo, o modelo *arquitetura* inclui os diagramas de classe, de componentes e de implantação da UML. A modelagem inicia com o *modelo de caso de usos do negócio* que define o contexto do sistema. Considerando o escopo da presente pesquisa com foco no modelo *arquitetura*, este é decomposto em arquitetura lógica e física. A arquitetura lógica é independente da tecnologia a ser implementada e inclui a elaboração do *diagrama de classe* com respectivos atributos, associações, multiplicidades e papéis entre as classes, ficando fora elementos que irão compor posteriormente o diagrama de classe da arquitetura física, ou seja, o mesmo diagrama com mais detalhes, tais como a navegabilidade entre classes, as operações com respectivas assinaturas e a visibilidade de atributos e operações.

O *Processo Unificado de Desenvolvimento de Software (RUP)* define um ciclo de vida iterativo e incremental com quatro fases: concepção, elaboração, construção e transição. Transversalmente às referidas fases ocorrem os fluxos de trabalho para cada iteração que incluem requisitos, análise, projeto, implementação e teste. Como exemplo de geração de modelos, no fluxo de requisitos é gerado o modelo de caso de uso, composto basicamente pelo diagrama de caso de uso e respectivas descrições, e no fluxo de análise, o modelo de análise composto pelos diagramas de classe e colaboração (JACOBSON *et al.*, 2000).

Larman (2004) adota basicamente o *Processo Unificado* com forte ênfase no processo iterativo e define a vista do sistema por meio de três modelos: *modelo de caso de uso, modelo de domínio e modelo de projeto*. As etapas do processo incluem a *visão geral, concepção e elaboração,* sendo a elaboração composta de iterações onde, para cada

iteração, ocorre a análise orientada a objeto, o projeto orientado a objeto e a tradução de projeto para código. Na análise orientada a objeto são gerados o modelo de caso de uso, que enfatiza a visão dos processos do domínio, e o modelo de domínio, que enfatiza as classes significativas do sistema. O modelo de domínio é composto dos seguintes elementos do diagrama de classe: classes do domínio, associações e atributos.

Rumbaugh e Blaha (2005) propõem três diferentes vistas para a modelagem de um sistema: o *modelo de classe*, que representa os aspectos estáticos, estruturais e de dados do sistema, o *modelo de estado*, que representa os aspectos temporais, comportamentais e de controle do sistema, e o *modelo de interação*, que representa os aspectos de interação do sistema. A fase inicial do processo de desenvolvimento de software, descrito pelos referidos autores, inclui a concepção do sistema e análise, tal qual ilustra a figura 3.2.

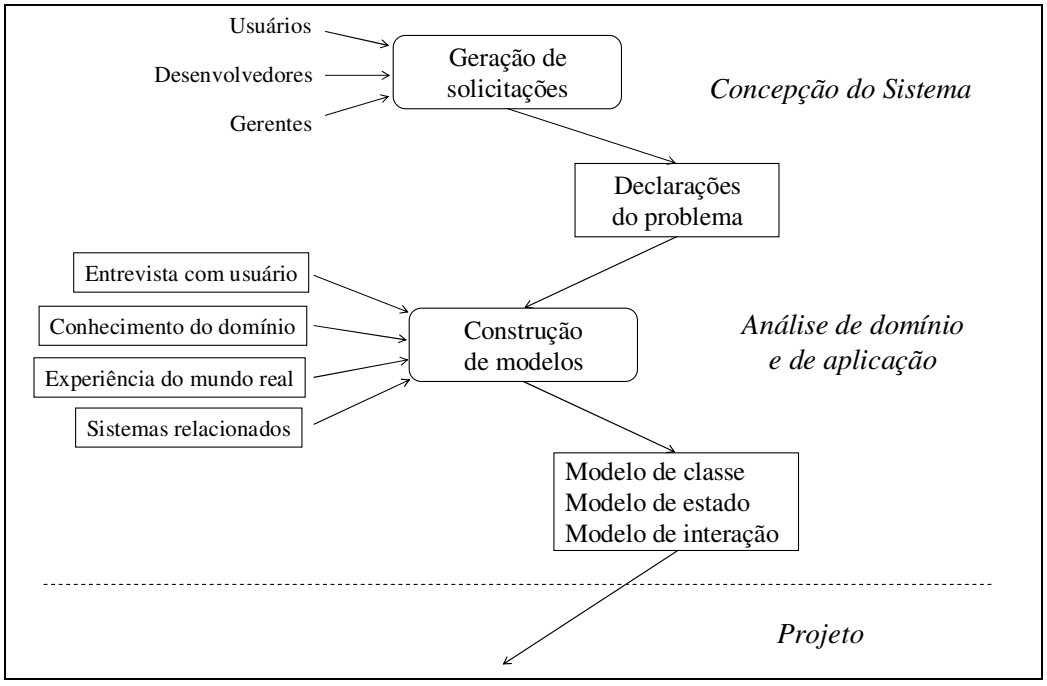


Figura 3.2: Uma visão da análise.

Observa-se que, nos quatro processos de desenvolvimento apresentados, é aplicada de forma implícita o processo da engenharia de requisitos. A técnica principal de elicitação utilizada é a de caso de uso (LARMAN, 2004), (MAKSIMCHUK e NAIBURG, 2005) e (JACOBSON *et al.*, 2000).

3.2 UML

3.2.1 *Infra-estrutura* e *Superestrutura*

As especificações da UML desenvolvidas pelo OMG incluem *UML 2.0: Infrastructure* (OMG-I, 2004) e *UML 2.0: Superstructure* (OMG-S, 2003). A primeira especificação define as construções fundamentais requeridas pela UML 2.0 e a segunda, as construções ao nível de usuário requeridas pela referida linguagem.

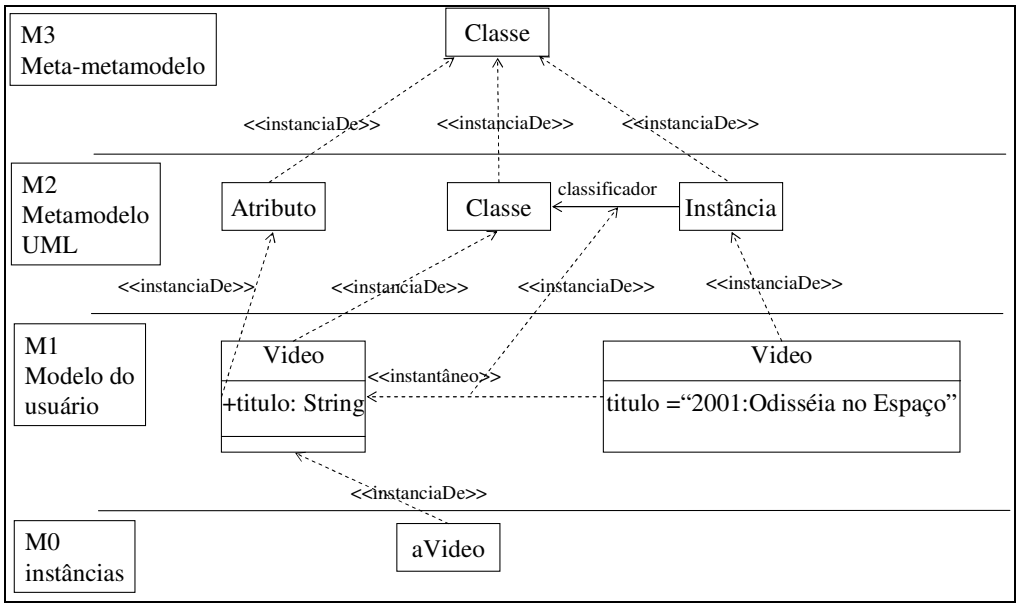


Figura 3.3: Hierarquia de quatro camadas da UML.

A especificação da UML é definida por meio de técnica de metamodelagem, i.e., os modelos são especificados por metamodelos. Embora falte à referida técnica um rigor de uma especificação formal, a mesma oferece a vantagem de ser uma técnica mais intuitiva e pragmática para a maioria dos implementadores. A camada de meta-metamodelagem, referenciada como M3, tal qual ilustra a figura 3.3, define a linguagem de especificação de um metamodelo. A camada de metamodelo, referenciada como M2, define uma linguagem para especificação de um modelo, tal qual a UML. A camada de modelo, referenciada como M1, tem como responsabilidade primária a definição de linguagens que descrevem domínios semânticos, ou seja, permite a geração de modelos de usuários, sendo estes instâncias de metamodelos da UML. A camada mais inferior, referenciada como M0, são as instâncias em tempo de execução de elementos de um modelo. Uma característica específica da metamodelagem é a habilidade de definir linguagens de forma refletiva, ou seja, linguagens que podem ser usadas para definir a si próprias. Tal técnica de especificação, usando metamodelos, não impede que a mesma possa ser especificada, de

forma complementar, por meio de uma linguagem de especificação formal, tal como a *Object Constraint Language* (OCL) (OMG-OCL, 2003) e (WARMER e KLEPPE, 2003) ou a *Object-Z* (DUKE e ROSE, 2000) e (SPIVEY, 1998).

Considerando o escopo do presente trabalho, destaca-se o metamodelo com a sintaxe abstrata do diagrama de classe ilustrada na figura 3.4, onde algumas informações foram omitidas por motivo de legibilidade (OMG-S, 2003).

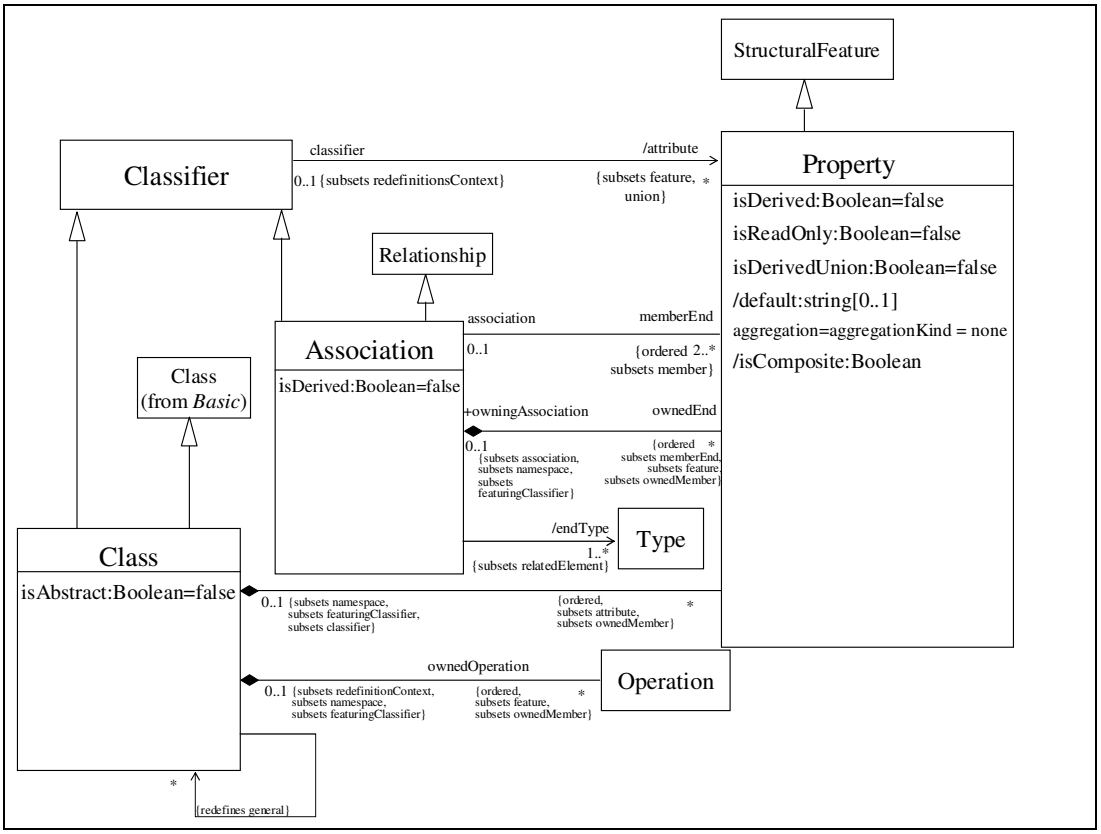


Figura 3.4: Sintaxe abstrata do diagrama de classe.

3.2.2 Semântica da UML

De acordo com Harel e Rumpe (2000), uma linguagem é composta tal qual a descrição ilustrada na figura 3.5.

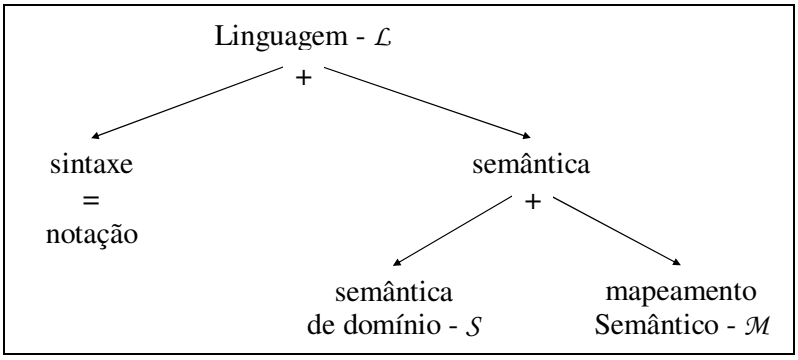


Figura 3.5: Estrutura de uma linguagem.

A semântica de uma linguagem determina o significado de cada expressão da mesma, sendo o referido significado um elemento de algum domínio bem definido. O mapeamento semântico relaciona os conceitos sintáticos aos conceitos do domínio semântico, ou seja, cada elemento sintático é mapeado em algum elemento semântico. O referido mapeamento é representado pela seguinte função, onde a linguagem $\mathcal{L}$ é representada pela respectiva sintaxe:

$$\mathcal{M} : \mathcal{L} \rightarrow \mathcal{S}$$

A semântica da UML se refere à interpretação em tempo de execução dos modelos gerados a partir da UML (SELIC, 2004). No exemplo apresentado por Harel e Rumpe (2000), o referido conceito é aplicado ao *diagrama de classe*, cuja interpretação inclui o conjunto de todas as possibilidades das estruturas de objetos em tempo de execução (*run-time interpretations*).

A semântica, ou significado, dos elementos de modelagem da UML, tal qual descrevem Evans (1998) e France (1999), são apresentados em linguagem natural por meio de uma semântica informal, ou seja, em forma de conceitos chaves básicos que não são definidos formalmente. A UML, cuja estrutura é descrita em (OMG-I, 2004) e (OMG-S, 2003), mantém a referida informalidade na descrição semântica dos conceitos, tal qual exemplifica a definição a seguir:

Elemento	Classe
Definição	Descreve um conjunto de objetos que compartilham a mesma especificação de características, restrições e semânticas.
Semânticas	• O propósito de uma classe é especificar a classificação de objetos e especificar os aspectos que caracterizam a estrutura e o comportamento destes objetos. • Objetos de uma classe devem conter valores para cada atributo membro da classe, de acordo com as características do atributo, por exemplo, seu tipo e multiplicidade. • Quando um objeto é instanciado em uma classe, para todo atributo da classe que tenha uma especificação *default*, se um valor inicial do atributo não é especificado explicitamente para instanciação, então o valor *default* especificado é avaliado para ajustar o valor inicial do atributo do objeto. • Operações de uma classe podem ser invocadas a partir de um objeto, dado um particular conjunto de substituições para os parâmetros da operação.

	Uma invocação de uma operação pode causar mudanças nos valores dos atributos de um determinado objeto. A mesma pode retornar um tipo de retorno como resultado, onde o tipo de resultado para a operação tenha sido definido. A invocação de uma operação pode, também, causar alterações nos valores de outros objetos que podem ser acessados por meio de navegação, direta ou indiretamente, a partir dos objetos nas quais as operações são invocadas. A invocação de uma operação pode causar, ainda, a criação e eliminação de objetos.

Um papel típico do metamodelo é definir as semânticas de como elementos de modelos são instanciados. Como exemplo, considere a figura 3.6, onde as metaclasses *Association* e *Class*, ambas definidas como parte do metamodelo UML, são instanciadas em um modelo de usuário de modo que as classes *Pessoa* e *Carro* são ambas instâncias da metaclasse *Class* e a associação entre ambas as classes é uma instância da metaclasse *Association*. A semântica da UML define o que ocorre quando elementos do modelo definido pelo usuário são instanciados no nível M0 (ver figura 3.3), ou seja, quando é gerada a instância de *Pessoa*, uma instância de *Carro* e uma instância da *Associação* entre as referidas instâncias de classes. As citadas instâncias muitas vezes são denominadas de instâncias em tempo de execução (*run-time instances*) (OMG-I, 2004).

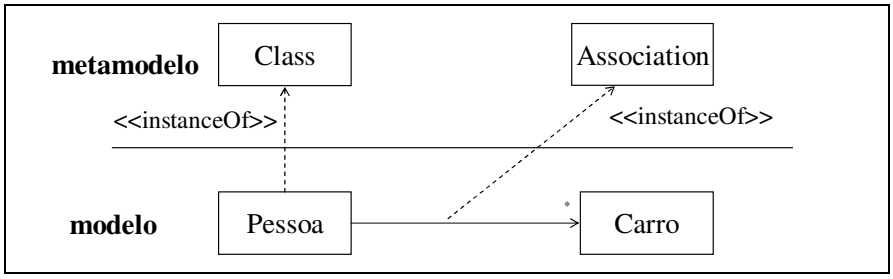


Figura 3.6: Um exemplo de metamodelagem.

3.3 Considerações

O termo *modelagem orientada a objeto* é muito empregado na comunidade de desenvolvimento de software tendo em vista que os principais autores aplicam os conceitos de construção de modelos durante o processo de desenvolvimento de software. De uma forma implícita, as atividades da engenharia de requisitos são aplicadas em metodologias propostas por renomados desenvolvedores das mesmas (LARMAN, 2004),

(MAKSIMCHUK e NAIBURG, 2005), (RUMBAUGH e BLAHA, 2005) e (JACOBSON *et al.*, 2000).

O Processo da Engenharia de Requisitos permite definir com maior rigor as atividades necessárias para elicitar, modelar, validar e verificar os requisitos de um sistema dentro da fase comumente denominada em engenharia de software de *Análise* ou formalmente, de *Engenharia de Requisitos de Software* (HILBURN *et al.*, 1999). A partir de uma definição bem formulada do processo da engenharia de requisitos, pode-se aplicá-lo em qualquer processo de desenvolvimento de software, sendo tal afirmação corroborada por Sommerville (2005), que, ao abordar aspectos atuais da engenharia de requisitos, ressalta a integração do processo da engenharia de requisitos com o processo de desenvolvimento de sistemas. De acordo com Hickey e Davis (2002), a elicitação de requisitos é uma das mais críticas atividades do processo de desenvolvimento de software, sendo intensiva em conhecimento e uma pobre execução da referida atividade garante uma completa falha do projeto. De acordo com os autores, a aplicação de uma técnica de elicitação não é, possivelmente, aplicável em todas as situações. Tal afirmativa é corroborada por Hofman e Lehner (2001), que, ao identificar os fatores de sucesso em projetos de software, detectou a aplicação de técnicas de elicitação diversas em um mesmo projeto, tais como, entrevistas não estruturadas, *brainstorming*, grupos definidos, *workshops*. Os referidos autores definem que um ciclo de engenharia de requisitos é um conjunto de atividades que contém pelo menos uma atividade de elicitação, uma de modelagem, uma de validação e uma de verificação, sendo o produto final, a título de exemplo, um modelo de dados.

Nuseibeh e Eastebrook (2000) destacam a aplicação da lingüística no suporte automatizado à engenharia de requisitos, tendo em vista ter esta engenharia muito de comunicação e que as referidas ferramentas lingüísticas podem ser aplicadas na elicitação de requisitos.

A linguagem de modelagem UML fornece a notação do diagrama de classe para o modelo conceitual objeto desta pesquisa, cujos elementos incluem as classes, os atributos, as associações e as multiplicidades. O entendimento da semântica existente na UML permite um maior rigor no mapeamento semântico entre a sintaxe do modelo, expressa por meio de metamodelo, e o domínio semântico, onde estão as interpretações em tempo de execução dos modelos UML.

Importante ressaltar que um modelo gerado a partir do diagrama de classe da UML pode tornar-se uma *rede semântica* e, como tal, uma forma expressiva de representação do conhecimento.

Capítulo 4

Técnicas Lingüísticas para Geração de Modelos Conceituais a partir de Linguagem Natural – Trabalhos Relacionados

A idéia de que existe uma relação entre o mundo lingüístico e o conceitual é antiga, tal qual descreve o trabalho de Chen (1983). Na área da orientação a objeto, Abbott (1983) propôs designar um substantivo a uma classe, um verbo a um método e um adjetivo a um atributo. Estas orientações têm sido refinadas por diversos autores.

O presente capítulo visa apresentar trabalhos relacionados com o presente tema de pesquisa de doutorado e, como tal, descreve seis técnicas lingüísticas que objetivam gerar um modelo conceitual a partir de documentos diversos contendo especificações de requisitos em linguagem natural.

4.1 Técnica Lingüística Proposta por Block *et al.* (1990)

O objetivo do trabalho desenvolvido por Block *et al.* (1990) é a implementação de um protótipo de sistema que detecta objetos e relacionamentos a partir de linguagem natural. Os autores o classificam como um sistema especialista por usar regras que incorporam conhecimento sobre projeto orientado a objeto para extração de informação de uma especificação de software. O desenvolvimento do trabalho foi composto de duas atividades: a derivação de um conjunto de regras para extração de objetos, e seus relacionamentos no texto, e a implementação de um sistema com as referidas regras. As regras permitem uma identificação inicial de objetos e relacionamentos na especificação de software. O método é composto das regras descritas a seguir.

• *Palavras e frases que identificam classes:* substantivos comuns; substantivos modificados por adjetivos à esquerda e à direita; sinônimos de expressões substantivas com ”kind" seguido por uma expressão preposicional "of" e uma sentença que inclui um verbo no passado, onde o sujeito do verbo está no plural ou um substantivo de grandeza.

• *Palavras e frases que identificam uma instância:* uma sentença contendo uma forma do verbo intransitivo "exists"; uma sentença contendo um verbo no passado e onde o sujeito do verbo não é um substantivo de grandeza nem está no plural; substantivos; sinônimos com "instance"; qualquer expressão substantivada definida ou indefinida; um substantivo próprio; uma expressão substantiva possessiva no singular; uma expressão substantiva possessiva indefinida no plural.

• *Palavras e frases que identificam relacionamentos:* substantivos modificados por expressões adjetivas à esquerda e à direita identificadas por um relacionamento AKO (*a kind of)* ou ISA *(is a)* dependendo do substantivo chave da expressão; advérbios agindo como modificadores relativos, tais como "particularly" e "especially”; uma expressão absoluta; frases onde o verbo principal é um verbo de conjunção; expressões substantivas que são modificadas por uma expressão preposicional que inclui a preposição ‘of’ e onde o substantivo-chave identifica uma classe, enquanto o objeto da preposição identifica uma instância ou um relacionamento de uma instância de classe.

De acordo com a figura 4.1, as regras foram implementadas em um sistema com três componentes primários: um arquivo de interface, um analisador gramatical e uma máquina de inferência. O arquivo de interface busca por sentenças individuais no arquivo texto, sendo cada sentença traduzida para uma lista de símbolos das palavras contidas nas respectivas sentenças, ou seja, ocorre um reconhecimento de palavras. A lista é apresentada ao analisador gramatical que, por sua vez, é composto de um analisador sintático e de um esquema de detalhamento da sentença analisada, cujo produto é um grafo acíclico direto (DAG) que representa a estrutura profunda da sentença analisada. O DAG é processado semanticamente pela máquina de inferência, onde estão implementadas as regras heurísticas, tendo como resultado as instâncias, as classes e os relacionamentos.

O teste das regras propostas foi executado por meio de um pequeno experimento com dezenove alunos da graduação no contexto da disciplina Projeto Orientado a Objeto. Os objetivos do experimento foram: verificar se diferentes projetistas derivariam o mesmo projeto de um mesmo texto e verificar se as regras permitiriam encontrar os mesmos objetos e relacionamentos, tal qual um projetista humano. O resultado identificou que 90% dos alunos identificaram os mesmos objetos.

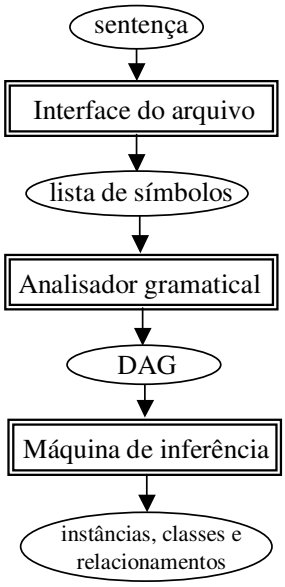


Figura 4.1: Técnica lingüística proposta por Block *et al.* (1990).

4.2 Técnica Lingüística Proposta por Overmyer *et al.* (2001)

Overmyer *et al.* (2001) propõem uma metodologia, suportada por uma ferramenta denominada LIDA (*LInguistic assistant for Domain Analysis*), que visa prover uma assistência lingüística ao processo de desenvolvimento de software. A metodologia proposta considera que o analista inicie seu trabalho com descrições textuais do domínio do problema visando identificar os elementos do modelo e seus relacionamentos, que podem estar em documentos com conceitos operacionais, em casos de uso, em descrições de tarefas e outros. Usualmente, o analista lê os documentos, ilumina ou sublinha os substantivos e verbos, e tenta associá-los a potenciais objetos e métodos. A experiência prática sugere que convenções usando categorias gramaticais das palavras para identificar objetos e métodos, tal como associar classes com substantivos, relacionamentos com verbos e atributos com adjetivos, são naturais e práticas. A maioria dos documentos que descrevem as funções de um sistema contém mais substantivos, verbos, advérbios e adjetivos que os necessários no esforço de identificação dos objetos a serem utilizados no sistema; surge, então, a necessidade de reduzir-se a quantidade de palavras a um subconjunto destas e que sejam diretamente aplicáveis ao desenvolvimento do sistema. Antes de preocupar-se com a identificação de classes, o analista deverá ter preparado um conjunto de casos de uso, ou cenários, que representem os conceitos operacionais para o

sistema proposto, o que lhe permitirá saber, *a priori*, como o sistema funciona e o auxiliará na identificação de quais classes são relevantes. A metodologia é consistente com o Processo de Desenvolvimento Unificado e a ferramenta LIDA utiliza notação UML.

A ferramenta LIDA possui as seguintes características: processamento independente de domínio; interfaces com diferentes vistas, tais como, textos e diagramas; as palavras podem ser vinculadas aos elementos dos modelos de acordo com seu tipo, sendo as mesmas codificadas por cor; as palavras podem ser ordenadas ou organizadas de acordo com seu tipo; uma vista de contexto permite mostrar somente sentenças com determinadas palavras; e exporta e importa modelos de ferramentas *case*. A arquitetura do sistema é composta de duas interfaces principais, tal qual ilustra a figura 4.2: o ambiente de análise de texto e o ambiente de edição de modelo.

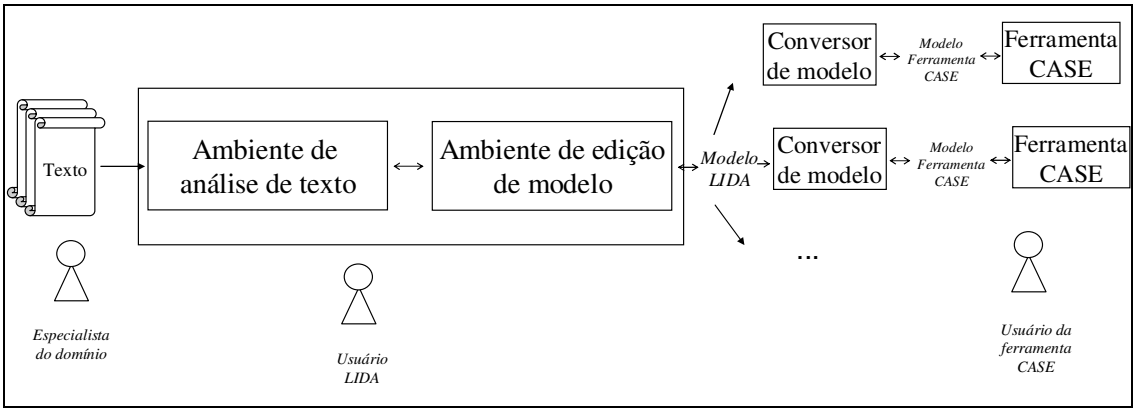


Figura 4.2: Arquitetura do sistema LIDA.

O ambiente de análise de texto do LIDA permite associar um item léxico ao elemento correspondente do modelo candidato, representando o componente que provê a funcionalidade central do método proposto Os principais aspectos incluem a leitura de arquivo texto (tipo RTF e ASCII), a associação de categorias gramaticais por meio do software *MXPOST* e a possibilidade de marcação, pelo usuário, de uma palavra ou frase como um elemento candidato do modelo. O ambiente de edição de modelo é uma ferramenta que oferece a funcionalidade necessária para construir o modelo dos elementos candidatos marcados no analisador de texto do LIDA. O conversor de modelo permite a conversão para uma outra ferramenta *case*.

O fluxo geral do processo de identificação de classes do LIDA está ilustrado no diagrama de atividades da figura 4.3. O analista primeiro importa o documento a ser analisado e o software destaca as categorias gramaticais das palavras. Em seguida, o analista trabalha com a lista de substantivos marcando os candidatos a classes relevantes e,

interativamente, removendo aquelas que, após reflexão, não se qualificam como classes ou os substantivos candidatos a atributos. Após a identificação das classes candidatas, o analista enfoca a lista de adjetivos para seleção dos atributos das classes. Prosseguindo no diagrama de atividades, o analista trabalha na lista de verbos para selecionar candidatos a métodos ou regras. A seguir, o analista usa o Modelador do LIDA que permite, graficamente, associar atributos, métodos e regras com as respectivas classes. A próxima atividade inclui a verificação semântica do modelo resultante usando o Explicador do Modelo do LIDA. Finalmente, o analista interage com o modelo a fim de realizar correções, e.g., nos relacionamentos entre classes.

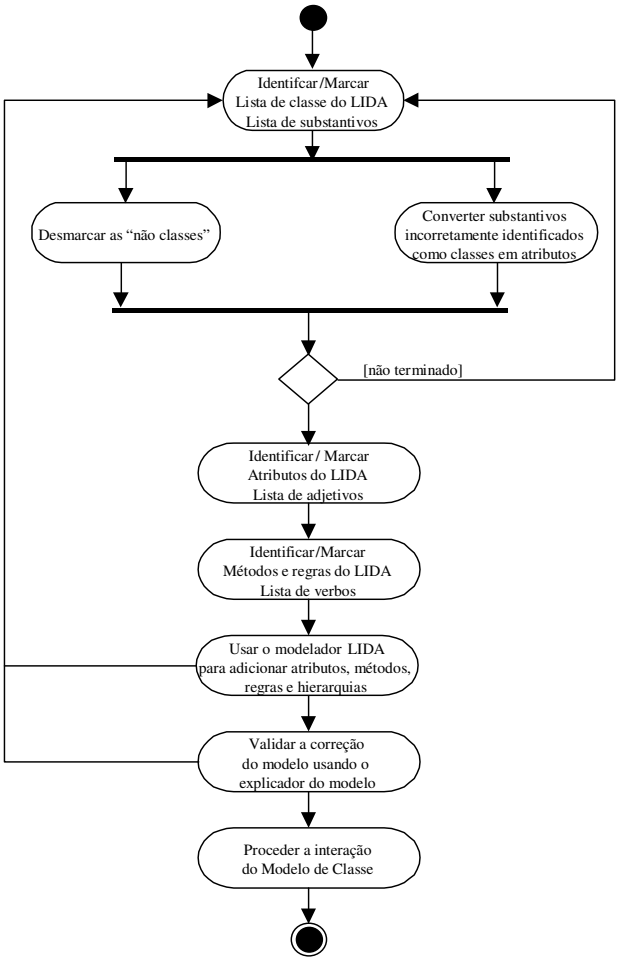


Figura 4.3: Processo de modelagem e identificação de classes do LIDA.

4.3 Técnica Lingüística Proposta por Juristo *et al.* (1999)

A técnica lingüística proposta por Juristo *et al.* (1999) tem os seguintes requisitos de interface:

• a descrição do problema é expressa em inglês que, por ser em linguagem natural, é composta de um conjunto de estruturas lingüísticas que a gramática define, tais como, sujeitos, verbos e complementos, e que, por sua vez, permite a correta construção de sentenças;

• os modelos gerados são expressos por meio de conceitos orientados a objeto, baseados no conjunto de diagramas e estruturas do referido paradigma.

O objetivo da técnica é relacionar, de forma justificada, as estruturas da linguagem natural e os conceitos orientados a objeto. Para tal, o método restringe o escopo da linguagem a um subconjunto, em virtude da extensão e da riqueza da mesma, e usa conceitos matemáticos para formalmente estabelecer o relacionamento entre o referido subconjunto da linguagem natural e os conceitos orientados a objeto, sendo criada para tal uma *linguagem de utilidade.* A fundamentação matemática do método consta em traduzir as estruturas lingüísticas da linguagem de utilidade em representações matemáticas e os modelos orientados a objeto em estruturas matemáticas. Estabelecidas as equivalências matemáticas entre ambas as representações, pode-se garantir que as estruturas orientadas a objeto modelam corretamente as estruturas da linguagem natural, pois as respectivas representações são equivalentes, tal qual ilustra a figura 4.4.

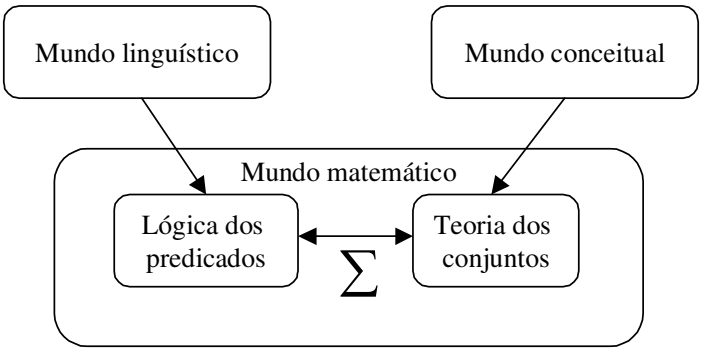


Figura 4.4: Relação entre os mundos lingüístico e conceitual.

A Figura 4.5 ilustra os estágios do método denominado *COrrespondence between LInguistic and COnceptual PAtterns* (COLICOPA).

No estágio 1, o analista extrai a informação requerida pelo processo de análise orientada a objeto, principalmente os requisitos funcionais oriundos da descrição textual.

No estágio 2, o analista busca por sinônimos e homônimos nas referidas funcionalidades. Os dois primeiros estágios incluem um refinamento da descrição do problema a fim de que sejam eliminadas as ambigüidades existentes.

No estágio 3, as informações que representam os dados que o sistema utiliza, denominados requisitos estáticos, são separadas das informações que descrevem o comportamento do sistema, denominados requisitos dinâmicos, tal qual exemplificado na tabela 4.1.

No estágio 4, as estruturas da linguagem natural são representadas em uma linguagem de utilidade e descrevem os requisitos estáticos, sendo gerada um subconjunto da referida linguagem denominada de *linguagem de utilidade estática (SUL).* A *SUL* permite a posterior identificação de classes e relacionamentos a serem expressos no modelo de objetos, sendo composta por um conjunto de oito categorias de estruturas de linguagem natural, formalmente descritas na tabela 4.2.

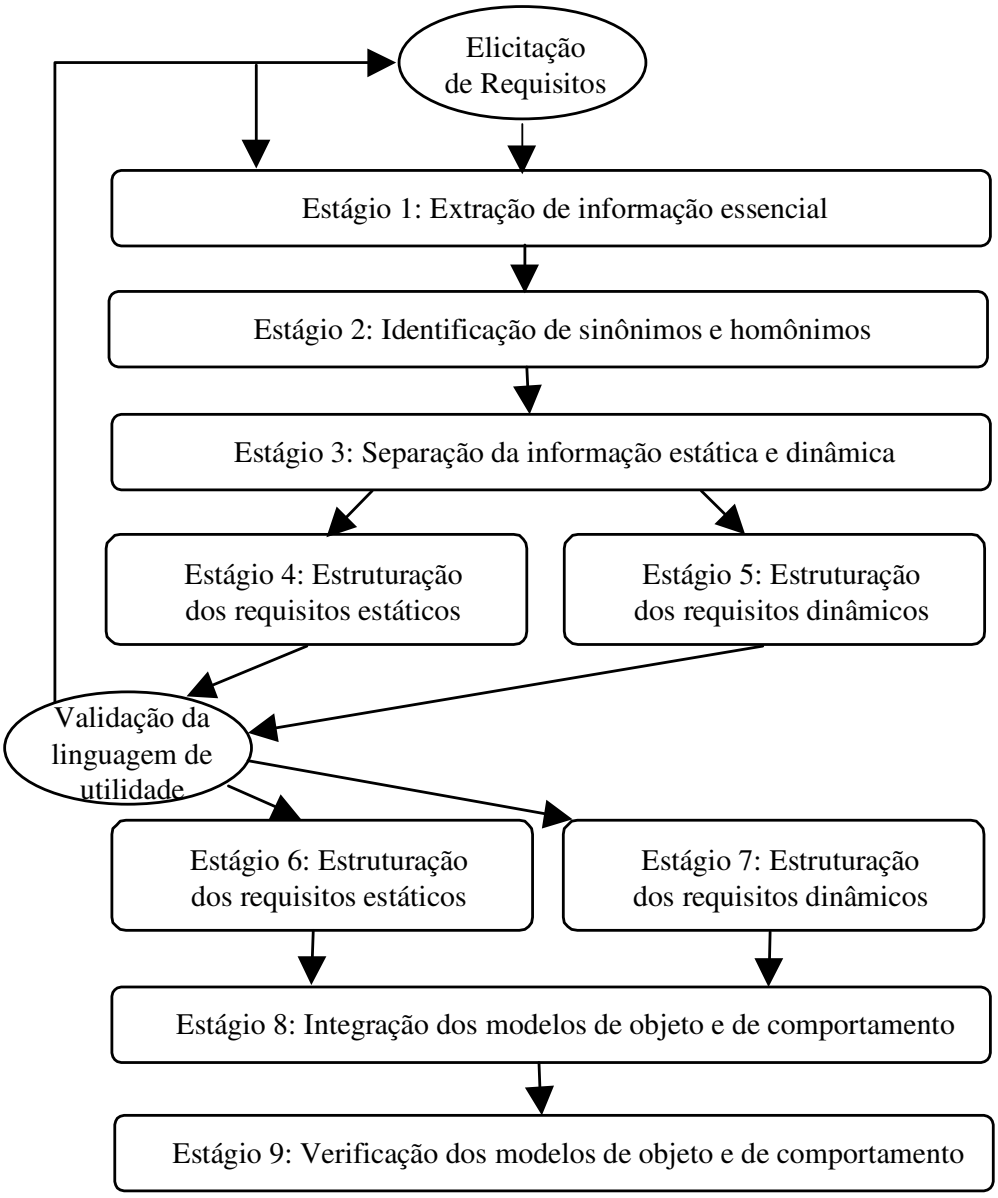


Figura 4.5: Fases do método *COLICOPA*.

Tabela 4.1: Resultados do estágio 3 do *COLICOPA*.

Requisitos estáticos	Requisitos dinâmicos
Vendors may be employees or companies. Employees receive a salary amount and a commission amount, whereas companies only receive a comission amount. Each order corresponds to one vendor only, and each vendor has made at least one order, which is identified by an order number. The same salary may be paid to different sales employees. The same commission amount may be paid to different companies and to different sales employees.	*1. All vendors are paid monthly.*
	4. When a sale is made, the vendor reports the order to the system.
	3.Once the orders are entered into the system, it sends confirmation to the customer and forwards the orders to the suplliers weekly.
	4. When the suppliers deliver the vehicles to the customers, they notify the company, which then pays them.

Tabela 4.2: Estrutura da Linguagem de Utilidade Estática (SUL).

Estrutura	Descrição	Exemplo
Classification (l_1)	Sentenças que expressam diferentes tipos de classificações entre o sujeito e o predicado. Dividem-se em três tipos - $l_{1,1}$, $l_{1,2}$, $l_{1,3}$ - dependendo sobre onde a classificação é expressa e o número de características expressas.	
Bottom up ($l_{1,1}$)	O conjunto de conceitos no qual o sujeito pode ser dividido é expresso no predicado.	*Men and women are types of people.*
Top down ($l_{1,2}$)	O conjunto de conceitos no qual o predicado pode ser dividido é expresso no sujeito.	*People can be men or women.*
Multiple ($l_{1,3}$)	As características de todos os complementos são expressas no sujeito.	*A person is a rational being and an animal.*
Complement enumeration (l_2)	Sentenças que expressam um permanente relacionamento entre o sujeito e o predicado. Estruturas l_3 e l_4 são dois casos particulares deste tipo de sentença.	*An author writes books and articles.*

Estrutura	Descrição	Exemplo
Composition (l_3)	Sentenças que expressam que um conceito é composto por *n* componentes. Dividem-se em dois tipos - $l_{3,1}$ e $l_{3,2}$ - dependendo onde os componentes são expressos.	
Component composition ($l_{3,1}$)	O predicado representa os elementos dos quais o sujeito é composto.	*A computer contains a monitor and a CPU.*
Content composition ($l_{3,2}$)	O sujeito representa os elementos dos quais o predicado é composto.	*The monitor and CPU are part of a computer.*
Identification (l_4)	Sentenças que expressam que o sujeito/predicado inevocadamente representam predicado/sujeito.	*A customer is identified by his code*
Adjacent Complements (l_5)	Estas são extensões da estrutura tipo l_2 onde mais componentes são adicionados.	*A customer can tranfer money to an account on a particular date.*

A transformação de requisitos estáticos na *SUL* ocorre de acordo com as seguintes orientações:

- usar cláusulas afirmativas simples;
- expressar os verbos em terceira pessoa do singular ou do plural em voz ativa ou passiva;
- substituir cláusulas do tipo "There is X in Y" ou "X exists in Y" por "Y has X";
- neste ponto, ignorar sentenças sem um sujeito ou complemento;
- se um modificador se refere a diversos grupos nominais, especificar a relevância em cada um;
- desconsiderar advérbios e adjetivos modais;
- substituir qualquer determinador numérico explícito, tal como 2 ou 100, por um mais representativo, tal como, "a", "some".

A tabela 4.3 mostra a *SUL* para os requisitos estáticos da tabela 4.1, obtidos pela aplicação das referidas orientações.

Tabela 4.3: Linguagem de Utilidade Estática (SUL) para os requisitos estáticos.

Estrutura	Gramática *SUL*	Exemplo
Bottom up ($l_{1,1}$)	*<$l_{1,1}$ Bottom-up classification structure>:= Nominal_Group (<Other_Nominal_Group> <Sentence_Coordinator> Nominal_Group) <bottom-up_classification_verb><complement>*	*1. $l_{1,1}$Vendors can be sales employees or companies.*
Complement enumeration (l_2)	*<l_2 Structures with complements enumeration>:= Nominal_Group (<Other_Nominal_Group> <Sentence_Coordinator> Nominal_Group) <general_verb><Complement>*	*1. l_2 Sales employees receive a basic wage and a comission.* *2. l_2 Companies receive a commission.* *3. l_2 Each order corresponds to one vendor.* *4. l_2 Each vendor has made at least one order.* *7. l_2 The same commission can be paid to several sales employees and several companies.* *8. l_2 On basic wage can be paid to several sales employees.*
Identification (l_4)	*<l_4 Identification structures>:= Nominal_Group <identify> <complement>*	*1. l_4 An order number identifies an order.*

No estágio 6, ocorre a construção do modelo de objetos. A notação escolhida para descrição do modelo orientado a objeto foi a OMT, de Rumbaugh. A identificação de classes e relacionamentos é uma tarefa que relaciona as estruturas lingüísticas da SUL com as estruturas de representação do modelo orientado a objeto, tal qual ilustra a tabela 4.4.

Tabela 4.4: O relacionamento entre a SUL e os requisitos estáticos do modelo de objeto.

Estrutura SUL	Estrutura orientada a objeto
Classificação (l_1)	
Bottom up ($l_{1,1}$)	C_1 Herança simples

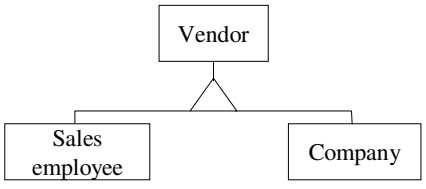
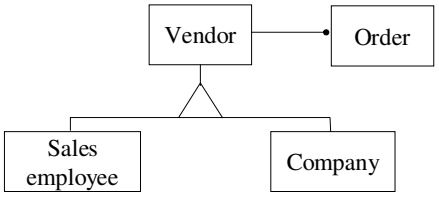
Estrutura SUL	Estrutura orientada a objeto
Top down ($l_{1,2}$)	C_1 Herança simples
Multiple ($l_{1,3}$)	C_2 Herança múltipla
Complement enumeration (l_2)	C_3 Associação binária
Composition (l_3)	
Component composition ($l_{3,1}$)	C_4 Agregação
Content composition ($l_{3,2}$)	C_4 Agregação
Identification (l_4)	C_5 Identificação de associação
Adjacent Complements (l_5)	C_6 N-associação

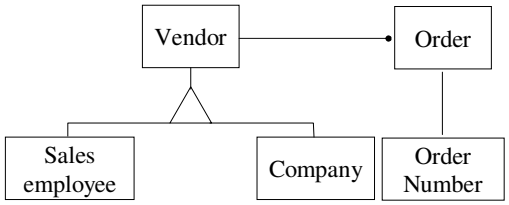
A construção do modelo de objetos segue as atividades descritas a seguir.

1. Identificação de classes: os sujeitos e os complementos são classes.

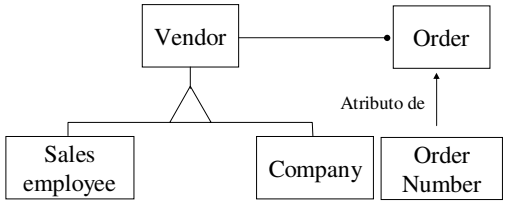
Sentenças	Classes
$l_{1,1}$ *Vendors can be sales employees or companies*	*Vendor* *Sales employee* *Company*
l_2 *Each order corresponds to one vendor* l_2 *Each vendor has made at least one order*	*Order*
l_4 *An order number identifies an order*	*Order number*

2. Identificação de relacionamentos: consta de buscar na tabela 4.4 a estrutura C_x orientada a objeto que corresponda a cada l_x SUL considerada. A posição da classe em cada estrutura orientada a objeto depende da estrutura SUL. Ex.: A estrutura $l_{1,1}$ corresponde a C_1, herança simples, onde o complemento deverá ser a super classe e o grupo nominal no sujeito será a subclasse.

SUL	Estrutura orientada a objeto	Relacionamentos
$l_{1,1}$	C_1 herança simples	
l_2	C_3 associação binária	

SUL	Estrutura orientada a objeto	Relacionamentos
l_4	C_5 identificação de associação	

3. Após a identificação das classes, o analista identifica os atributos das classes examinando as estruturas orientadas a objetos de identificação (C_5), de agregação (C_4) e de associação binária (C_3). Se uma classe do tipo C_5 participa de somente um relacionamento, a mesma é um atributo da outra classe de relacionamento.

l_4	C_5 identificação de associação	

No estágio 7, são estruturados os requisitos dinâmicos. O analista constrói o modelo de comportamento a partir dos requisitos dinâmicos que especificam as interações e eventos que afetam as informações descritas nos requisitos estáticos. Por meio de um procedimento similar ao aplicado aos requisitos estáticos, o analista transformará as descrições textuais dos requisitos dinâmicos em parte da linguagem de utilidade que representa o comportamento do sistema, ou seja, na *linguagem de utilidade dinâmica* (*DUL*). A partir da *DUL* será desenvolvido o modelo de comportamento.

O estágio 8 inclui a integração dos modelos de objetos e de comportamento, tendo como objetivo determinar os métodos das classes. Considerando o escopo do presente trabalho, foram omitidos detalhes relacionados com a parte dinâmica do método formal em estudo; uma descrição completa encontra-se em (JURISTO *et al.*, 1999), (JURISTO e MORENO, 2000) e (MORENO *et al.*, 2000).

4.4 Técnica Lingüística Proposta por Rolland e Proix (1992)

Os referidos autores propõem um suporte automatizado à Engenharia de Requisitos, composta pelas fases de aquisição e validação, por meio de uma técnica lingüística. Na tarefa de aquisição, que inclui a análise e a modelagem, ocorre a geração da especificação conceitual a partir da descrição do problema em linguagem natural. A validação é baseada na geração de textos em linguagem natural a partir das especificações conceituais.

OICSI, acrônimo francês para *ferramenta inteligente para projeto de sistema de informação,* é um sistema especialista com interface gráfica, baseado em regras e que utiliza o paradigma baseado em conhecimento para prover uma ajuda ao analista no processo de engenharia de requisitos. A Figura 4.6 ilustra o processo de análise e de modelagem do *OICSI*.

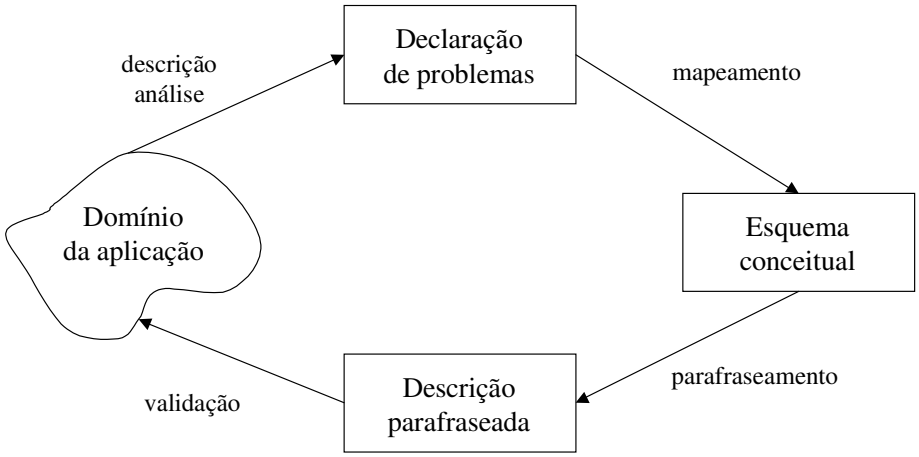


Figura 4.6: Processo de Análise e Modelagem *OICSI*.

A seguir será descrito um exemplo que é, segundo o autor, uma introdução ao comportamento de resolução de problema de um analista por meio de um mecanismo lingüístico:

> "O analista denominado *Ado* ouve a sentença: *Um assinante tem um nome e um endereço*. Provavelmente ele introduz no modelo uma entidade *Assinante* com dois atributos *Nome* e *Endereço*. O que significa que *Ado* utiliza um tipo de conhecimento de senso comum para equiparar a sentença a um esquema conceitual. *Ado* recebe uma segunda sentença: *O colydrena tem um pedistylus e um folicul*. Desta vez *Ado* não está certo do significado da sentença, mas interpreta que no modelo existe uma entidade *colydrena* com dois atributos *pedistylus* e *folicul"*.

Ado, mesmo sem entendimento algum das palavras, encontrou um modelo que descrevesse a situação. O *raciocínio* de *Ado* foi baseado no reconhecimento de um modelo de sentença o qual lhe é coloquial. O conhecimento usado, neste caso, é um conhecimento lingüístico relacionado com manipulação de linguagem que lhe permitiu reconhecer e interpretar o seguinte modelo de sentença: <*grupo sujeito*><*verbo expressando propriedade*><*grupo complemento*>. O conhecimento lingüístico é, certamente, segundo

os autores, o conhecimento mais comum entre os analistas que o aplicam algumas vezes explicitamente, mas o aplicam mais freqüentemente de forma implícita. O objetivo da técnica lingüística implementada na ferramenta OICSI é fazer explícito diferentes tipos de padrões de sentenças de modo a formalizar este tipo de conhecimento lingüístico e suportar o processo de interpretação e modelagem de declarações do problema por meio computadorizado.

As declarações do problema são expressas em francês e automaticamente interpretadas em termos de modelo conceitual OICSI. O processo de geração do esquema conceitual é composto por três etapas: análise, lingüística e mapeamento. A etapa de análise inclui a decomposição de cada sentença em unidades gramaticais, onde, por meio de regras gramaticais contidas em um dicionário, é determinada a natureza gramatical de cada palavra, bem como a classificação dos verbos; tais regras, oriundas da gramática gerativa, permitem verificar se uma sentença pertence à linguagem autorizada bem como a construção de árvores sintáticas. A etapa lingüística, cujo embasamento técnico está fundamentado em "cases" da Teoria de Fillmore, inclui uma equiparação de modelos de modo a associar cada conceito da árvore sintática com um "case" padrão do modelo lingüístico definido. A última etapa, o mapeamento, inclui a construção de uma árvore semântica, constituída de nós e arcos, a partir dos conceitos gerados na etapa anterior. A figura 4.7 ilustra uma rede semântica parcial gerada pelo referido processo, onde cabe ressaltar os quatro conceitos identificados que são objetos, ações, eventos e restrições.

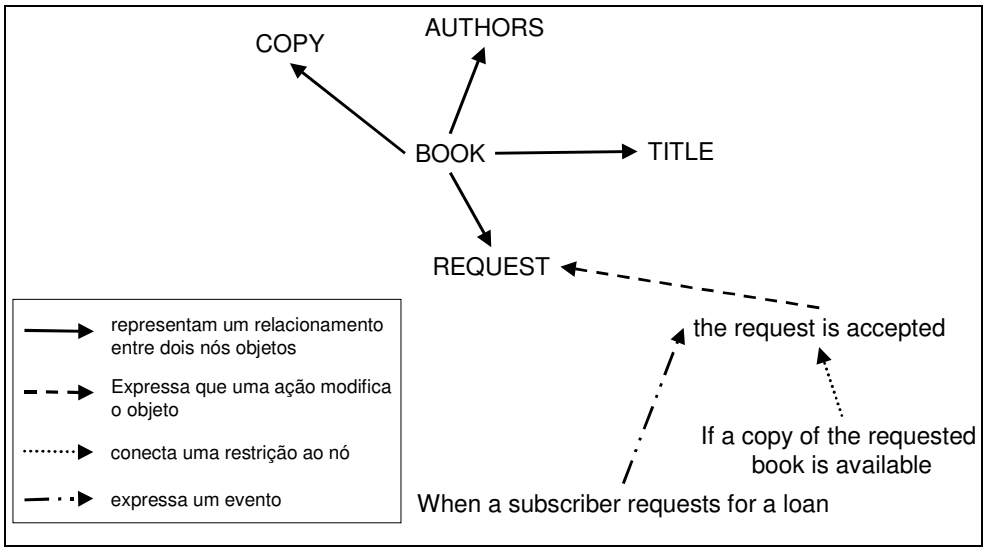


Figura 4.7: Um exemplo de rede semântica do *OICSI.*

4.5 Técnica Lingüística Proposta por Li *et al.* (2004)

Com base no argumento de que o processo da Engenharia de Requisitos é um processo inerentemente centrado na natureza humana, os autores propõem uma técnica algorítmica de análise e reestruturação de texto em linguagem natural implementada em uma ferramenta semi-automatizada. O objetivo é aplicar processamento de linguagem natural diretamente na elicitação de requisitos. Os elementos orientados a objeto a elicitar incluem as classes, os objetos, os métodos, as propriedades e os relacionamentos. O processo inicia após a identificação léxica da palavra, onde é alocado um POS, *part of speech,* de cada palavra, sendo nesta etapa identificados os substantivos, verbos e pronomes. Na fase 2 as sentenças são formatadas em uma estrutura padrão do tipo *sujeito-verbo-objeto*, de acordo com regras previamente definidas. Na fase 3 é gerado o diagrama de classe de acordo com as descrições simplificadas. Nas fases 4 e 5 ocorre o envolvimento do usuário a fim de que sejam confirmados os pronomes e os atores utilizados.

Os autores propõem mapeamentos entre os elementos da linguagem natural, que são os substantivos, pronomes, pronomes pessoais e verbos, e os elementos da linguagem de modelagem orientada a objeto, que são as classes, métodos e propriedades. Os objetos não estão incluídos por serem os mesmos instâncias concretas de uma classe, pois o foco está no nível abstrato. Os mapeamentos são os seguintes:

MoE 1: traduzir *substantivos* em *classes*.

MoE 2: traduzir *substantivos-substantivos* em *classe-propriedade* de acordo com a posição.

MoE 3: traduzir a estrutura *sujeito-verbo-objeto* para um diagrama de classe com o *sujeito* e o *objeto* como *classes* e o *verbo* como candidato a *método*.

MoE 4: traduzir o *verbo* lexical de um substantivo não pessoal em um método deste substantivo.

MoE 5: traduzir o *verbo* lexical de um substantivo pessoal em um caso de uso, ou parte do mesmo, vinculado a um ator definido por um nome.

MoE 6: equivaler um *substantivo* a um *pronome pessoal* tal como o *substantivo* na sentença anterior.

MoE 7: para a estrutura “S-V-O”, converter para “S-V-OO”, ou seja, tratar substantivos compostos como uma entidade.

4.6 Técnica Lingüística Proposta no Projeto *Color-X*

O projeto *Color-X* é um acrônimo de *COnceptual Linguistically based Object oriented Representation language for Information and Communication System,* onde *ICS* é abreviada para *X* (BURG e VAN DE RIET, 1995a, 1995b e 1996).

Color-X implementa uma técnica de modelagem de objeto baseada em lingüística com o objetivo de estreitar o vazio entre os documentos de requisitos escritos em linguagem natural e os modelos conceituais estáticos e dinâmicos. No projeto ocorre uma combinação de técnica de modelagem formal, por meio da linguagem *CPL (Conceptual Prototyping Language)*, sendo a mesma baseada em lingüística, com técnicas de modelagem gráfica. Considerando o foco do presente trabalho, será abordado o modelo *Color-X Static Object Model (CSOM)* que mostra os objetos, classes e relacionamentos. A *CPL* permite especificar restrições em uma linguagem bem próxima da natural.

A principal razão da utilização de conhecimento lingüístico é fazer com que as palavras sejam empregadas de forma mais consistente nos modelos por meio de regras. Exemplos de regras básicas incluem: *substantivos podem ser nomes de classes* e *verbos devem ser relacionamentos*. Exemplos de regras que relacionam significados de palavras incluem: *certos tipos de relacionamentos requerem certos tipos de classes* e *certos tipos de classes não podem relacionar-se*. Vantagens adicionais incluem a atribuição de mais expressividade às técnicas de modelagem e a possibilidade de geração de sentenças em linguagem natural, a partir dos modelos gerados, a fim de permitir a verificação da consistência dos mesmos junto aos requisitos.

O modelo *CSOM* inclui os seguintes conceitos: classes, objetos, relacionamentos e restrições. A classe é denotada por um substantivo, sendo composta de atributos, com os respectivos tipos, e de métodos. Um objeto é uma instância de uma classe e seus atributos possuem valores. Relacionamentos são identificados pelos verbos e são categorizados em relacionamentos estáticos padrões, que contém relações especiais da CPL, tais como, *is_a-, has_a-, exists-, value-of, etc,* e os relacionamentos *user-defined*, que são definidos pelo usuário diretamente dos documentos de requisitos.

O processo de especificação de requisitos do *Color-X* inclui a utilização de técnicas existentes acrescidas de uma técnica proposta pelo projeto, tal qual estão descritas a seguir.

1. *Análise de linguagem natural.* Método introduzido para produzir um modelo entidade-relacionamento (CHEN, 1983) que consta de transformar sentenças em *sentenças elementares* com apenas um verbo, encontrar substantivos comuns que geram entidades,

encontrar verbos transitivos que, comumente, indicam ação, identificar adjetivos que, usualmente, representam atributos e identificar advérbios que, normalmente, representam atributos de um relacionamento.

4. *Análise gramatical.* Método utilizado no projeto KISS (HOPPENBROUWERS *et al.*, 1997), que possui o suporte gráfico da ferramenta *GRAMMALIZER.* O método consiste basicamente em identificar as categorias gramaticais das palavras e frases do texto selecionado, transformando as sentenças selecionadas em sentenças estruturadas *KISS*; a partir destas, ocorre o julgamento do analista para decidir sobre quais informações contidas são relevantes. Os modelos são criados a partir do texto selecionado, sendo que as alterações nas estruturas das sentenças permitem a geração do modelo apropriado. O método é interativo e semi-automático.

3. *Análise semântica.* Método particularmente inserido no projeto *Color-X*, que permite vincular o significado das palavras ao modelo conceitual. O método tenta resolver problemas de homônimos, bem como fornece mais informações sobre os conceitos, tais como ontológico (*emprestar* significa *dar algo temporariamente*), estrutural (verbos transitivos requerem a especificação de um sujeito e um objeto) e entre conceitos (*emprestar* tem um antônimo complementar *retornar*).

A CPL permite formalizar a representação interna para o modelo conceitual, tornando-a disponível para utilização por uma ferramenta automatizada, tal como um gerador de código, e a verificação lingüística dos conteúdos dos modelos.

O *Color-X* possui um editor gráfico, denominado *Tool for Static Object Model* (TSOD), que permite a realização de duas técnicas de validação: a primeira, transforma o significado dos modelos em sentenças em linguagem natural completamente compreensíveis por seres humanos e, a segunda, transforma o modelo conceitual em programas de simulação executáveis, a fim de verificar o comportamento do sistema.

4.7 Considerações

O presente capítulo apresentou seis técnicas lingüísticas com o objetivo de dar suporte às atividades da Engenharia de Requisitos.

A técnica apresentada por Block *et al.* (1990) é limitada, permitindo a extração de objetos e seus relacionamentos do texto. A técnica de Overmyer *et al.* (2001) tem como ponto forte a definição de um processo heurístico com apoio de ferramentas gráficas bastante interativas de suporte ao analista, tais como um analisador gramatical, um editor

de modelo e um descritor dos conceitos do domínio do problema. A técnica descrita por Juristo *et al.* (1999) relaciona, de forma justificada, as estruturas da linguagem natural e os conceitos orientados a objeto; o método restringe o escopo da linguagem a um subconjunto, em virtude da extensão e da riqueza da mesma, e usa conceitos matemáticos para formalmente estabelecer o relacionamento entre o referido subconjunto da linguagem natural e os conceitos orientados a objeto, sendo criada para tal uma *linguagem de utilidade.* A fundamentação matemática do método consta de traduzir as estruturas lingüísticas da linguagem de utilidade em representações matemáticas e os modelos orientados a objetos em estruturas matemáticas, sendo estabelecidas equivalências matemáticas entre ambas as representações, o que garante que as estruturas orientadas a objeto modelam corretamente as estruturas da linguagem natural, pois as respectivas representações são equivalentes. A técnica proposta por Rolland e Proix (1992) é composta de um processo que: inclui a análise, a lingüística e o mapeamento; aplica a análise sintática para identificar a estrutura da sentença; e equipara os conceitos identificados na árvore com modelos lingüísticos padronizados. Li *et al.* (2004) propõem regras de mapeamento entre os elementos da linguagem natural e elementos de modelagem orientados a objeto. O projeto *Color-X* inclui a implementação de uma técnica de modelagem de objeto baseada em lingüística com o objetivo de estreitar a solução de continuidade entre os documentos de requisitos escritos em linguagem natural e os modelos conceituais estáticos e dinâmicos; no projeto ocorre uma combinação de técnica de modelagem formal, por meio da linguagem *CPL*, sendo esta baseada em lingüística, com técnicas de modelagem gráfica.

Tabela 4.5: Estudo comparativo entre as técnicas lingüísticas.

Técnica Lingüística	Automação	Análise sintática	Análise semântica	Análise de discurso	Gramática Gerativa
Block, 1990	Sim	Sim	Sim	Não	Não
Overmyer, 2001	Sim	Não	Não	Não	Não
Juristo *et al.*, 1999	Não	Sim	Sim	Não	Não
Rolland e Proix, 1992	Sim	Sim	Sim	Não	Sim
Li, 2004	Sim	Sim	Não	Não	Não
Color-X	Sim	Sim	Não	Não	Não

A tabela 4.5 ilustra um estudo comparativo entre as técnicas estudadas no presente capítulo. Conceitualmente, uma técnica lingüística para análise de texto envolve a análise sintática, semântica e de discurso, bem como a adoção de uma gramática gerativa, a fim de permitir a verificação da correção sintática das sentenças analisadas. O aspecto automação é fundamental no suporte às atividades da Engenharia de Requisitos.

Capítulo 5

Assistente Inteligente para Extração de Elementos Orientados a Objeto de Discurso

5.1 Definição do Problema

A ação de extrair os requisitos ocorre por meio de técnicas que incluem entrevistas, questionários, descrições textuais de caso de uso, entre outras, tendo, usualmente, como produtos finais, especificações de requisitos em linguagem natural. A partir das referidas especificações, a modelagem orientada a objeto permite a construção de sistemas de software complexos, onde o desenvolvedor deve abstrair diferentes vistas do sistema, construir modelos usando notações precisas, verificar se os modelos satisfazem os requisitos do sistema e, gradualmente, adicionar detalhes a fim de que os modelos possam ser implementados.

No contexto da geração de um modelo conceitual orientado a objeto, as tarefas de identificação de classes, atributos, associações e multiplicidades são usualmente processadas manualmente e dirigidas por heurísticas que o analista adquire por meio de experiência, o que caracteriza uma solução de continuidade de automação entre a identificação de requisitos em linguagem natural e a posterior modelagem conceitual, sendo esta lacuna de automação o problema a ser solucionado por esta tese.

Uma solução para o referido problema inclui a especificação e a implementação de um *assistente inteligente*, i.e., uma ferramenta automatizada com suporte baseado em conhecimento (FALBO, 1998), que permita gerar um modelo conceitual orientado a objeto a partir de requisitos em linguagem natural. A figura 5.1 ilustra o esquema geral do problema a ser pesquisado.

Figura 5.1: Esquema geral do tema de pesquisa proposto.

5.2 Objetivos

O objetivo geral desta tese de doutorado é automatizar a geração de modelo conceitual orientado a objeto a partir de especificações de requisitos em linguagem natural. O referido objetivo inclui a construção de um assistente inteligente, i.e., uma ferramenta automatizada com suporte baseado em conhecimento, denominado de *Assistente Inteligente para Extração de Elementos Orientados a Objeto de Discurso (ASIEEOOD).*

Os objetivos específicos incluem:

• aplicar métodos da engenharia de software integrados com técnicas da inteligência artificial;

• especificar uma técnica lingüística a ser implementada no *ASIEEOOD* como suporte ao processo da Engenharia de Requisitos;

• considerar as atividades de elicitação, modelagem e validação do processo da Engenharia de Requisitos, de acordo com Hoffman e Lehner (2001), não sendo considerada a atividade de verificação;

• gerar o modelo conceitual orientado a objeto de acordo com a notação definida pelo metamodelo diagrama de classe da UML, considerando que os elementos orientados a objeto incluem as classes, os atributos, as associações e as multiplicidades;

• definir uma arquitetura para o sistema de PLN, i.e., o *ASIEEOOD*;

• permitir o processamento de qualquer linguagem natural que possua estruturas sintagmáticas semelhantes ao português, e.g., inglês.

De acordo com Pressman (2002), os métodos de elicitação incluem entrevistas, grupos de trabalho, equipes de discussão, cenários de casos de uso, sendo que o resultado alcançado da identificação dos requisitos gera, entre outros, uma relação de requisitos agrupados por funcionalidade e respectivas restrições. Hoffman e Lehner (2001) afirmam, também, que a elicitação pode ser realizada por meio de diversos métodos, e.g., entrevistas, questionários, *brainstorming,* grupos de discussão, análise de planos de negócios, estudos de mercado, e destacam que na fase posterior, i.e., a modelagem, especialistas têm proposto métodos e linguagens de especificação com o intuito de tornar os requisitos mais precisos e consistentes, sendo os referidos métodos tradicionalmente separados de acordo com o que os autores denominam aspectos de requisitos, i.e., dados, funcionais e comportamentais.

Baseado nos conceitos de Pressman (2002) e Hoffman e Lehner (2001), o *ASIEEOOD* é uma ferramenta de suporte ao Processo da Engenharia de Requisitos*,* ilustrado na figura

5.2, a ser empregada por um analista, ou engenheiro de conhecimento, na edição de sentenças que representam as *especificações de requisitos de dados* em linguagem natural, tendo como foco o modelo conceitual e as respectivas restrições, e.g., associações e multiplicidades. As referidas sentenças serão processadas por uma técnica lingüística implementada pelo *ASIEEOOD*, cujos resultados serão os elementos orientados a objeto. Os elementos citados permitirão a execução da atividade posterior do processo da engenharia de requisitos, i.e., a modelagem. A validação será realizada por meio de inspeção no modelo conceitual orientado a objeto gerado por um editor gráfico do *ASIEEOOD.* O conjunto de sentenças que representam as especificações de dados é denominado de *discurso* no domínio lingüístico.

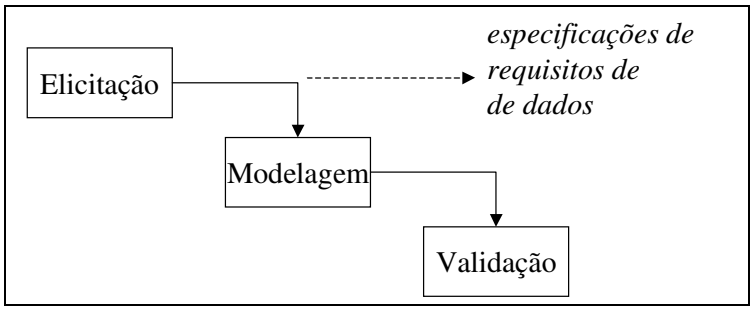


Figura 5.2: Processo adotado da Engenharia de Requisitos.

5.3 Técnica Lingüística

5.3.1 Mecanismo Lingüístico Intuitivo Usado por Analistas

Rolland e Proix (1992) apresentam a seguinte introdução ao comportamento de solução de problema analítico durante a modelagem conceitual:

> "Um analista ouve a sentença *um assinante tem um nome e um endereço"* e introduz no modelo conceitual a entidade *assinante* com dois atributos, *nome* e *endereço.* A segunda sentença é *"o colydrema tem um pedistylus e um folicul",* sendo identificada a entidade *colydrema* com dois atributos, *pedistylus* e *folicul*".

O referido analista não sabe o significado das palavras na segunda sentença, mas encontra um modelo para a situação descrita que, entretanto, é correta. No caso, o raciocínio do analista está baseado no reconhecimento de um padrão particular de sentença, coloquial para o mesmo.

5.3.2 Hipótese para Solução do Problema Proposto

A hipótese para solução do problema proposto tem como fundamento o citado comportamento e inclui o desenvolvimento de uma técnica lingüística que permita extrair conceitos orientados a objeto de sentenças em linguagem natural baseado em constituintes sintáticos, tais como estrutura sintática e categoria gramatical, independentemente do significado da palavra propriamente dito.

A presente técnica possui os seguintes requisitos lingüísticos básicos:

• adotar um sistema de gramática composto por uma *gramática*, como o algoritmo formal que permite a geração das seqüências gramaticais, e um método de reconhecimento de palavra (HAUSSER, 2001);

• permitir o processamento de qualquer linguagem natural, cujo alfabeto e estruturas sintáticas sejam semelhantes ao Português;

• aplicar lingüística computacional nas análises morfológica, sintática e semântica no contexto de discurso;

• adotar uma linguagem de representação do conhecimento que permita gerar bases de conhecimento;

• representar o conhecimento por meio de uma rede semântica, núcleo do sistema;

• explorar, de forma intensa, a *abordagem declarativa* no nível de conhecimento do sistema;

• permitir a edição gráfica da gramática adotada;

• automatizar a geração do modelo conceitual, considerando que o diagrama a ser gerado pelo presente assistente não é um modelo de informação estrito, mas uma ferramenta de comunicação que permite compreender, de acordo com Larman (2000), os conceitos importantes do domínio do problema, bem como seus relacionamentos;

• restringir, considerando a existência de um número infinito de estruturas sintagmáticas (CHOMSKY, 2002), as referidas estruturas na edição dos discursos especificados em linguagem natural por meio de uma linguagem controlada (SOWA, 2002a) e (LUISA *et al.*, 2003);

• reconhecer as sentenças sintaticamente corretas, e.g., *o professor leciona banco de dados*, não determinando a má formação de uma determinada sentença;

• pressupor que o analista, que especifica os requisitos em linguagem natural, edita sentenças semanticamente corretas dentro do domínio do problema, evitando sentenças tais como *o médico foi atendido por um caminhão*, estando esta última sentença sintaticamente

correta, mas semanticamente incorreta, tendo efeito no modelo conceitual e passível de identificação pelo analista durante a atividade de validação;

• permitir a correção de erros ortográficos e palavras desconhecidas pelo engenheiro do conhecimento, responsável pela manutenção da base de dados e da base de conhecimento.

5.3.3 Sistema de Gramática

O sistema de gramática adotado para o assistente inteligente proposto inclui:

• a *gramática de estrutura sintagmática, ou PSG-grammar (PSG)*, como o algoritmo formal que permite a geração das seqüências gramaticais (CHOMSKY, 2002);

• a *forma de palavra* como o método de reconhecimento de palavra automático, o que permite a implementação de um algoritmo de reconhecimento simples por meio da busca da palavra completa como chave no dicionário analisado (HAUSSER, 2001).

A presente proposta adota a independência da gramática em relação à semântica explicada por Chomsky (2002), i.e., a gramática é autônoma e independente do significado. Considerando a conceituação de gramática como o conjunto de regras que permitem organizar as palavras de uma língua em frases, a análise sintática proposta inclui a construção de uma ferramenta que permita produzir a estrutura gramatical das sentenças sem fundamentos semânticos, e.g., referência, significado, sinonímia, não sendo adotado o princípio da composicionalidade apresentado na seção 2.2.2.3.1. A relação sintaxe e semântica é definida de forma independente e somente após a determinação da correção da estrutura sintática ocorre a extração semântica das sentenças analisadas.

Outro requisito adotado é a extensão da *PSG* a fim de que a mesma inclua toda a linguagem diretamente, o que é possível segundo Chomsky (2002), gerando, entretanto, a perda da simplicidade de uma *PSG* limitada, bem como o desenvolvimento transformacional. As justificativas para adoção do referido método incluem:

• a capacidade declarativa da base de conhecimento permite um número infinito de estruturas sintáticas, de modo que o sistema "aprende" de acordo com o *corpus* de cada domínio de problema;

• a limitação das estruturas sintáticas em função da linguagem controlada adotada;

• a simplicidade na extração semântica dos elementos orientados a objeto das sentenças analisadas.

Considerando que uma gramática de uma linguagem *L* é essencialmente uma teoria de *L* (ver seção 2.2.1), a presente técnica tem como requisito uma teoria de estrutura

lingüística que permite, inicialmente, determinar por um procedimento de decisão se uma gramática de uma linguagem atende a um determinado *corpus*, ou, no caso, um discurso; caso negativo, a gramática é refinada por meio de um procedimento de descoberta de regras sintagmáticas a partir do citado *corpus*, tal qual ilustrado na Figura 5.3, i.e., a gramática cresce em função das novas estruturas identificadas nos *corpora*.

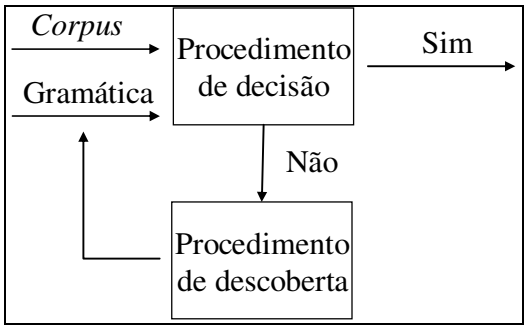


Figura 5.3: Teoria da estrutura lingüística adotada (CHOMSKY, 2002).

5.3.4 Linguagem de Representação do Conhecimento

A linguagem de representação do conhecimento permite a formulação do conhecimento por meio de representações simbólicas que capacitam um sistema raciocinar. A referida linguagem tem como principal aspecto uma noção bem definida de vínculo lógico, pois, de acordo com Brachman e Levesque (2004), o assim chamado *trabalho de raciocinar*, adequado também ao presente contexto, significa computar os vínculos lógicos de uma base de conhecimento.

A lógica de primeira ordem (LPO) é a mais amplamente utilizada, estudada e implementada versão da lógica, sendo que alguns lógicos modernos limitam o expressivo poder da LPO a um subconjunto mais facilmente computável, e.g., cláusulas definidas de Horn (BRACHMAN e LEVESQUE, 2004) (SOWA, 2000). As representações do conhecimento em cláusulas definidas de Horn podem ser usualmente separadas em dois tipos: fatos e regras (ver seção 2.4). Os fatos determinam verdades básicas do domínio, enquanto que as regras são utilizadas para expressar novos relacionamentos. As proposições consideradas como argumentos verdadeiros são denominadas hipóteses ou axiomas, e as proposições, que buscam a conseqüência lógica a partir dos referidos axiomas, são denominadas *teoremas*. A partir desses conceitos, aparece a atividade denominada de *prova de teorema*, cujo objetivo é derivar uma conseqüência lógica a partir dos *axiomas* (CLOCKSIN e MELLISH, 2003).

Baseado nos argumentos acima descritos, esta técnica lingüística adota a cláusula definida de Horn em notação Prolog como a linguagem de representação do conhecimento. A linguagem Prolog inclui *fatos e regras* que são aceitos como um conjunto de *axiomas* e o *questionamento* do usuário como um t*eorema* presumido, sendo a mesma um *provador de teorema* com uma bem definida noção de vínculo lógico baseado no procedimento de resolução *SLD* (CLOCKSIN e MELLISH, 2003) e (BRATKO, 2001).

5.3.5 Estrutura de Representação do Conhecimento

A rede semântica é uma estrutura para representação do conhecimento cujo modelo inclui nós e arcos interconectados; os nós representam conceitos de entidades, atributos, eventos ou estados e os arcos, conexões entre os conceitos (SOWA *et al.*, 1991). Considerando a conceituação apresentada, destaca-se o seguinte questionamento: a rede semântica pode ser utilizada como suporte à geração de conhecimento e ao raciocínio automatizado? A presente pesquisa tem como resposta "sim", com base nas seguintes fontes:

• Stuart Shapiro declara que uma estrutura em rede pode suportar importantes tipos de raciocínios subconscientes que não são diretamente representados por meio de uma forma lógica linear (SOWA *et al.*, 1991);

• Shastri afirma que, de uma forma geral, é possível traduzir uma rede semântica em uma linguagem não gráfica e vice-versa (SHASTRI, 1991);

• Russel e Norvig (2004) consideram a rede semântica um sistema especialmente projetado para organizar e raciocinar com categorias, oferecendo auxílio gráfico para visualização de uma base de conhecimento;

• Sowa (2002b) declara que a rede semântica é uma representação gráfica declarativa que pode ser utilizada para representação do conhecimento, bem como suporte ao raciocínio automatizado.

No escopo do relacionamento conceitual da UML com a rede semântica, destacam-se dois questionamentos: algum diagrama da UML pode ser utilizado como rede semântica? caso positivo, qual a semântica de um diagrama UML? A primeira questão tem como respaldo científico a afirmativa apresentada por Sowa (2002b), que classifica alguns diagramas da UML como uma rede semântica e justifica declarando que central à UML existe uma definição de rede de tipos de objetos e uma outra rede, tal qual um gráfico relacional, que permite uma representação de meta-informação. A segunda questão, i.e., o entendimento semântico da UML, requer a compreensão da especificação da UML por

meio da hierarquia de um metamodelo em quatro níveis (OMG-I, 2004), tal qual ilustra a figura 3.3 na seção 3.2.1.

Do exposto, esta técnica lingüística adota a notação do Diagrama de Classe da UML, que corresponde ao nível M2 da figura 3.3, como a estrutura de representação do conhecimento, i.e., a rede semântica, o que permitirá ao assistente inteligente raciocinar com instâncias de objetos que comporão axiomas a serem inseridos na base de conhecimento de acordo com a sintaxe da linguagem de representação do conhecimento adotada.

5.4 Nível Lingüístico Morfológico

A análise morfológica especificada para o *ASIEEOOD* tem como finalidade implementar as funcionalidades relacionadas com a fragmentação do discurso em sentenças, a busca de palavras compostas e a realização da etiquetação morfológica dos vocábulos.

5.4.1 Reconhecimento Automático de Palavra

No contexto do sistema de gramática adotado, o reconhecimento automático de palavra adotado é o método de *forma de palavra* (ver seção 2.2.2.1.1), cujo algoritmo simplesmente equipara toda a superfície da forma de palavra desconhecida na correspondente chave na análise do léxico, o que exige um léxico de grandes dimensões (HAUSSER, 2001). As justificativas para adoção do referido método incluem:

- simplicidade do algoritmo de reconhecimento automático de palavra;
- aplicação de uma linguagem controlada, o que reduz significativamente o léxico utilizado pelo referido assistente, e.g., verbo na terceira pessoa;
- não necessidade de reconhecimento de neologismos pelo assistente;
- intervenção do engenheiro de conhecimento na etiquetação morfológica de palavra não reconhecida pelo analisador léxico, o que possibilita uma atualização constante do léxico e a viabilização de sempre gerar o modelo conceitual especificado textualmente pelo analista;
- não necessidade de criação de regras para identificação de regularidades e irregularidades morfológicas da linguagem natural adotada;
- a interpretação semântica ao nível morfológico, tal qual ocorre no método de reconhecimento de forma de palavra por meio de alomorfe, não se faz necessária na

presente técnica lingüística, em virtude da mesma adotar o conceito da independência entre a gramática adotada e a semântica, tal qual descrito na seção 2.2.2.3.1.

A análise morfológica propriamente dita inicia com a busca de palavras compostas, o que permite a desnecessária ambigüidade dos seus componentes bem como a sua utilização na análise sintática. As estruturas com hífens são facilmente reconhecidas, outras, entretanto, por não o possuírem, exigem uma busca mais complexa, sendo o algoritmo adotado baseado na solução oferecida por Bick (2000), i.e., a verificação ocorre a partir de grupos formados de 4 palavras, tal qual ilustra a figura 5.4. As palavras que não fazem parte de uma determinada palavra composta são analisadas por um processo de etiquetação para palavra simples.

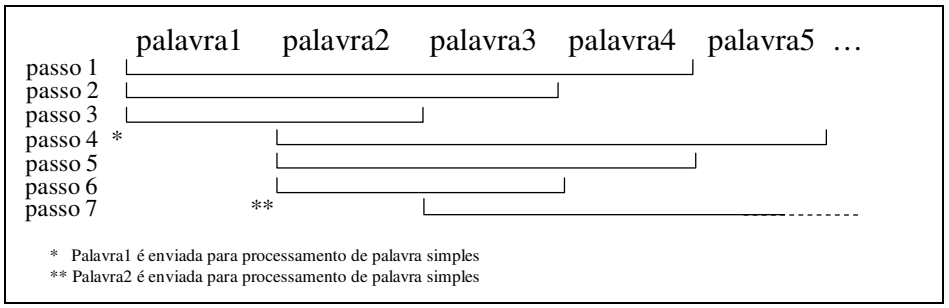


Figura 5.4: Algoritmo de busca de palavra composta.

5.4.2 Etiquetação Morfológica

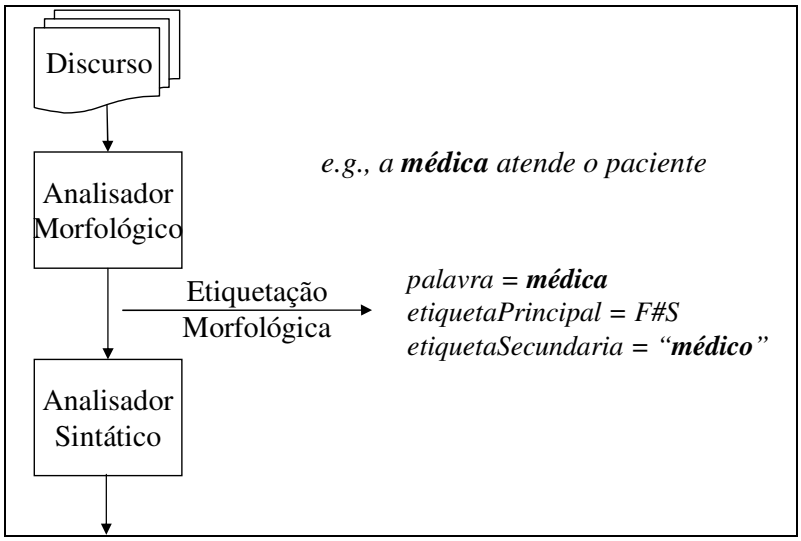


Figura 5.5: Processo de etiquetação morfológica.

A etiquetação implementada pelo *Analisador Morfológico* do *ASIEEOOD* consiste de uma etiquetação principal, i.e., *tag* principal, que inclui a flexão de gênero e a flexão de número, e uma etiquetação secundária, i.e., *tag* secundária, para palavras da categoria gramatical "substantivo" e "adjetivo", permitindo, se for o caso, a determinação do

vocábulo correspondente no singular e no masculino. A figura 5.5 ilustra, por meio de uma representação do tipo *pipeline*, o processo de etiquetação morfológica do *ASIEEOOD*, relacionando as etiquetações às principais funcionalidades do sistema.

As figuras 5.6 e 5.7(a) ilustram o padrão de etiquetação adotado pela presente técnica lingüística. Cada palavra é instanciada por um objeto da classe *CAnalisadorMorfologico* que determina as etiquetas principal e secundária de cada vocábulo. Uma lista de palavras forma uma sentença e uma lista destas, um discurso. Os espaços em branco de uma palavra composta sem hífen são substituídos pelo caracter "_", a fim de permitir o processamento da mesma pelo motor de inferência da base de conhecimento, i.e., Prolog.

```
classe CAnalisadorMorfologico
        String * palavra;
        String * etiquetaPrincipal;      (#gênero#número)
        String * etiquetaSecundaria;     (representação ou regularização)
```

Figura 5.6: Classe Analisador Morfológico.

O processo de etiquetação para o *ASIEEOOD* considera as seguintes premissas básicas:

• a denominação de uma classe ou de um atributo tem a flexão de número na forma do *singular*, exceto quando a palavra é invariável, e a flexão de gênero no *masculino*, exceto quando a palavra é uniforme, i.e., comum aos dois gêneros;

• os elementos orientados a objeto *classe* e *atributo* são identificados por vocábulos da categoria gramatical substantivo (ver seção 5.6.1).

```
{ palavra="a" etiquetaPrincipal="#f#s#" etiquetaSecundaria="o" }
{ palavra="médica" etiquetaPrincipal="#f#s#" etiquetaSecundaria="médico" }
{ palavra="atende" etiquetaPrincipal="#u#s#" etiquetaSecundaria="#" }
{ palavra="o" etiquetaPrincipal="#m#s#" etiquetaSecundaria="#" }
{ palavra="paciente" etiquetaPrincipal="#u#s#" etiquetaSecundaria="#" }
                                        (a)

        Palavra              "médica"
        EtiquetaPrincipal    "#f#s#"
        EtiquetaSecundaria   "médico"
                                (b)
```

Figura 5.7: Etiquetação morfológica do *ASIEELOO.*

Em conseqüência das premissas citadas, a etiqueta denominada *tag secundária*, exemplificada na figura 5.7(b), aplica-se aos seguintes casos:

• *regularização em gênero e número* dos vocábulos da classe de palavra *substantivo*, pois os mesmos podem aparecer em diferentes pontos do texto e serem referentes de uma mesma classe ou atributo, e.g., o substantivo *alunas* é regularizado para *aluno*;

• *representação semântica* para as palavras da categoria gramatical *adjetivo* considerados *instâncias* de possíveis atributos no domínio do problema, e.g., a palavra adjetiva *amarelo* está associada ao vocábulo substantivo *cor* que, por sua vez, pode ser um atributo de classe.

Em resumo, a regularização determina que uma palavra é sempre processada semanticamente no *singular*, ou como invariável, e no masculino, ou como uniforme. A tabela 5.1 mostra as flexões utilizadas na presente técnica lingüística. A etiquetação de número será necessária na análise semântica, particularmente na extração das multiplicidades.

Cabe destacar que a etiquetação da categoria gramatical não está implementada pelo *Analisador Morfológico* em virtude da possibilidade de ocorrência da ambigüidade gramatical de um vocábulo, onde uma ou mais categorias gramaticais podem ser associadas a uma determinada palavra. A referida ambigüidade é resolvida pelo *Analisador Sintático.*

Tabela 5.1: Etiquetas de flexão adotadas pelo *ASIEEOOD*.

Flexão	Tipo	Etiqueta Adotada
Gênero	Masculino	M
	Feminino	F
	Uniforme ou comum de dois gêneros	U
Número	Singular	S
	Plural	P
	Invariável	I

5.5 Nível Lingüístico Sintático

A presente técnica lingüística tem como requisito o processamento de qualquer língua que tenha estruturas sintagmáticas idênticas ao português, e.g., inglês. Entretanto, considerando a necessidade da realização de um estudo de caso para a hipótese proposta, priorizou-se, por ser esta a língua natural dos estudos desenvolvidos, a implementação das estruturas sintagmáticas do português (ver seção 2.2.2.2.1).

5.5.1 Predicação Gramatical

A tabela 5.2 ilustra as categorias gramaticais, e respectivos predicados e argumentos, utilizados pelas regras e fatos sintáticos inseridos na base de conhecimento, adotadas pelo *ASIEEOOD*. A tabela 5.3 detalha o grupo dos determinantes e a tabela 5.4, o grupo dos pronomes.

Tabela 5.2: Categorias gramaticais adotadas pelo *ASIEEOOD*.

Categoria Gramatical	Predicado	Argumento
Adjetivo	adjetivo	Adj
Advérbio	adverbio	Adv
Conjunção	conjunção	Conj
Determinante	determinante	Det
Preposição	preposição	Prep
Preposição_ (combinação e contração)	preposição_	Prep_
Pronome	pronome	Pron
Substantivo comum	substantivo	Subst
Substantivo próprio	substantivo_próprio	Prop
Verbo principal	verbo	Verbo
Verbo auxiliar	verbo_aux	Verbo_aux

Tabela 5.3: Grupo dos determinantes (AZEREDO, 2003).

Determinante	Elementos
Artigo definido	o, a , os, as.
Pronome demonstrativo	este, estes, esta, estas, isto, esse, esses, essa, essa, isso, aquele, aqueles, aquela, aquelas, aquilo.
Pronome possessivo	meu(s), minha(s), teu(s), tua(s), seu(s), sua(s) nosso(s), vosso(s), vossa(s), seu(s), sua(s).
Pronome indefinido	todo(s), muito(s), pouco(s), bastante, próprio, tanto(s), quanto(s), mais, vários, cada, qualquer, quaisquer, algum(ns), nenhum, um(ns) , outro(s), demais, etc.
Numerais cardinais	um, dois, três, quatro, etc.

Tabela 5.4: Grupo dos pronomes (AZEREDO, 2003).

Pronome	Elementos
Pessoais	eu, tu, ele, ela, nós, vós, eles, elas (retos) me, te, o, a, se, lhe, nos, vos, os, as, se, lhes (oblíquo)
Relativos	que, o(s) qual(is), a(s) qual(is), cujo(s), cuja(s), quem, onde, quanto(s), quantas, quando, como
Interrogativos	que, quem, quanto, qual

Esta técnica simplifica as duas formas de ligações das preposições a outras classes gramaticais, i.e., combinação e contração, tal como exemplifica a tabela 5.5, propondo a categoria gramatical *preposição+*. Na combinação ocorre a manutenção de todos os fonemas quando da união da preposição com outra palavra. Na contração, ocorre a modificação da estrutura fonológica da preposição ao unir-se a outra palavra. O motivo da adoção da referida simplificação gramatical, i.e., *preposição+*, está no fato de que não justifica o esforço computacional de processar as combinações e as contrações de forma separada e.g., *do (preposição+)* equivale a *de + o (preposição + artigo)*, pois, em termos sintáticos, o conceito *preposição+* facilita a estruturação sintática das frases e, em termos semânticos, permite auxiliar na identificação das multiplicidades.

Tabela 5.5: Exemplos de combinações e contrações (INFANTE, 2001).

Preposição+	Exemplos
Combinações	ao, aos (preposição + artigo).
Contrações	do, dos, da, das, num, nuns, numa, numas, disto, disso, daquilo, naquele, naqueles, naquela, naquelas, à, às, àquele, àquilo, etc.

A *Predicação Gramatical*, gerada pelo *Analisador Sintático* durante a análise sintática, inclui a associação dos vocábulos do léxico com as respectivas categorias gramaticais, permitindo a geração de fatos a serem inseridos na *base de conhecimento sintática*, tal qual ilustra a figura 5.8. A associação de mais de uma categoria gramatical a um determinado vocábulo no início da análise sintática permite a resolução de ambigüidades morfológicas, i.e., palavras com mais de uma categoria gramatical no léxico, mas em posições sintáticas distintas. Como exemplo, a palavra *amarelo* na sentença *o amarelo é uma cor suave* é um substantivo cujo processamento sintático exige o predicado

gramatical *substantivo(singular, substantivo(amarelo)) --> [amarelo])* e na sentença *o tênis é amarelo*, exige o predicado *adjetivo(singular, adjetivo (amarelo)) --> [amarelo]).*

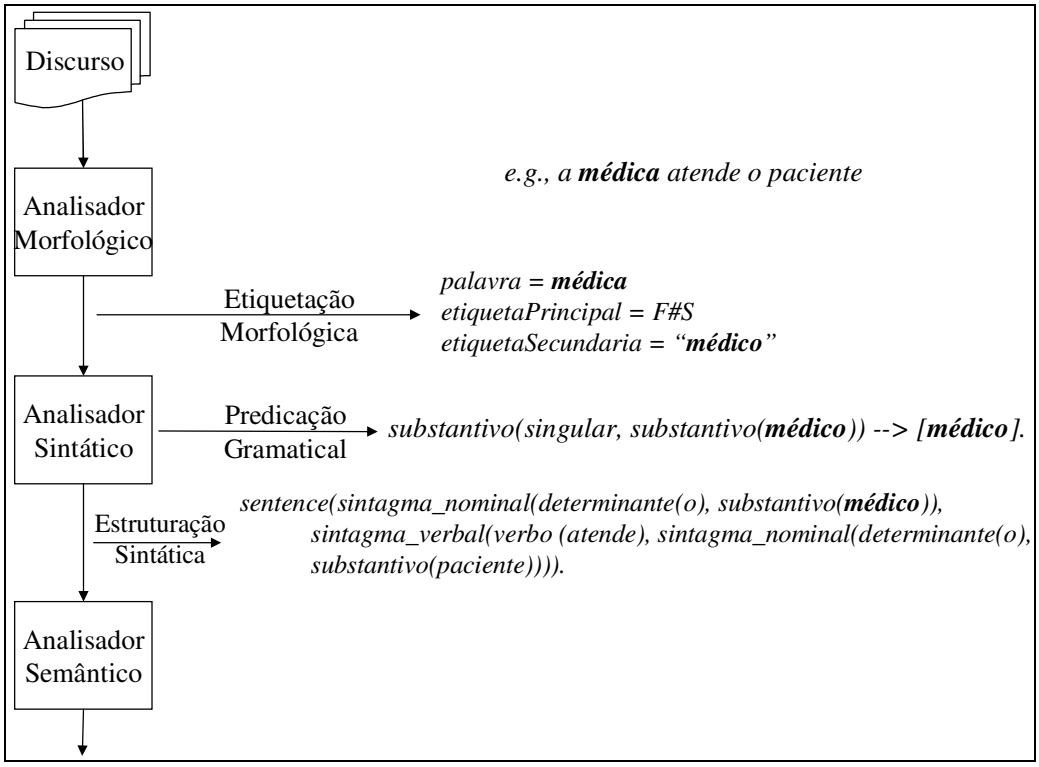


Figura 5.8: Predicação gramatical.

A *Predicação Gramatical* permite, juntamente com as regras sintáticas inseridas na *base de conhecimento sintática,* a geração da estrutura sintática plana da sentença analisada, e.g., *sentence(sintagma_nominal(determinante(o), substantivo(médico)), sintagma_verbal(verbo (atende), sintagma_nominal(determinante(o), substantivo(paciente))))*.

5.5.2 Desenho da Árvore Sintática

A tabela 5.6 ilustra as estruturas sintáticas adotadas pelo *ASIEEOOD*. A coluna predicado corresponde à denominação utilizada na composição das regras sintáticas a serem inseridas na base de conhecimento, e.g. , *sentence(Numero, sentence(SN, SV))--> sintagma_nominal(Numero, SN), sintagma_verbal(Numero,SV),!*. As variáveis têm como finalidade servir de parâmetros nas referidas regras e, quando instanciadas, concretizam os vínculos lógicos existentes em uma determinada *base de conhecimento sintática*.

O sistema *ASIEEOOD* tem uma funcionalidade, tal qual ilustra a figura 5.9, que permite gerar a estrutura sintática plana de uma determinada sentença a partir do desenho da respectiva árvore sintática, sendo esta uma atribuição do engenheiro de conhecimento.

Tabela 5.6: Etiquetas de estruturas sintáticas.

Sintagma	Predicado	Variável
Sintagma nominal	sintagma_nominal	SN
Sintagma verbal	sintagma_verbal	SV
Sintagma preposicionado	sintagma_preposicionado	SP
Sintagma adjetivo	sintagma_adjetivo	SADJ
Sintagma adverbial	sintagma_adjverbial	SADV
Sentença	sentence	S

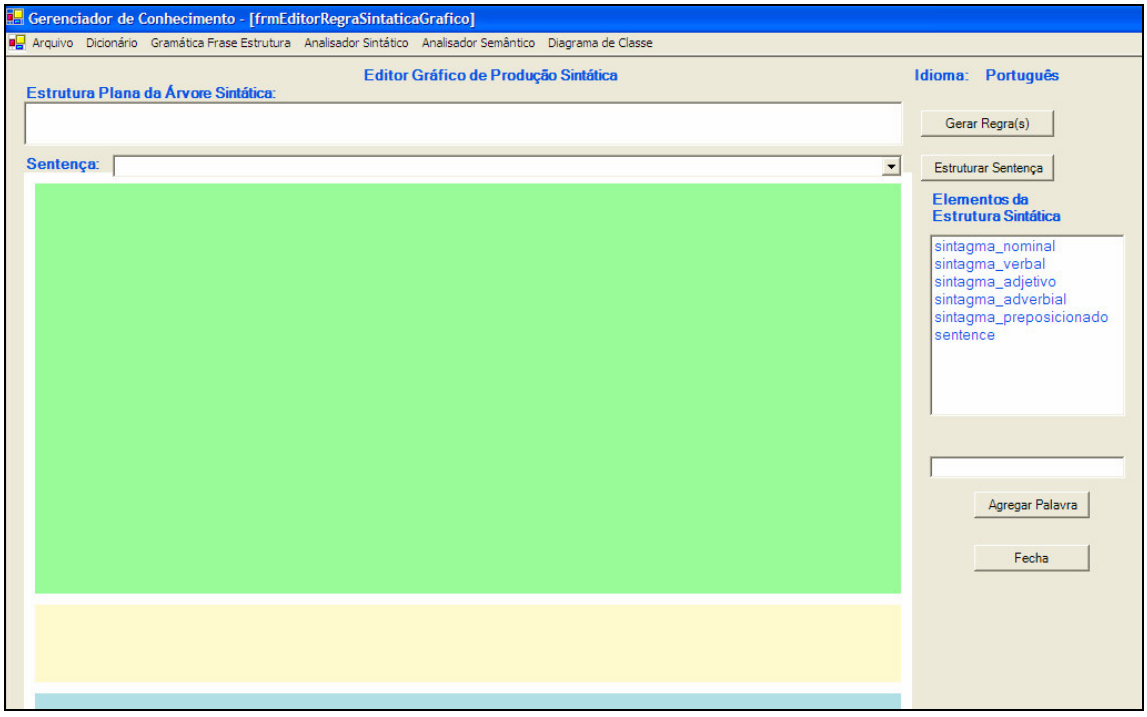


Figura 5.9: Editor gráfico de regras sintáticas.

O desenho da árvore sintática deve seguir os fundamentos relacionados com a estruturação sintática do português apresentados na seção 2.2.2.2.1, bem como a regra de ouro das estruturas sintáticas, i.e., os modificadores estão sempre vinculados à frase que os mesmos modificam. Um erro muito comum a ser evitado é exemplificado por meio da expressão *muito rapidamente* e ilustrado na figura 5.10, i.e., o sintagma adverbial (SAdv) *muito* é modificado pelo advérbio *rapidamente* resultando na fórmula $SAdv \rightarrow SAdv, Adv$ (CARNIE, 2002). Embora não afirmado pela referência, a citada regra permite a recursão.

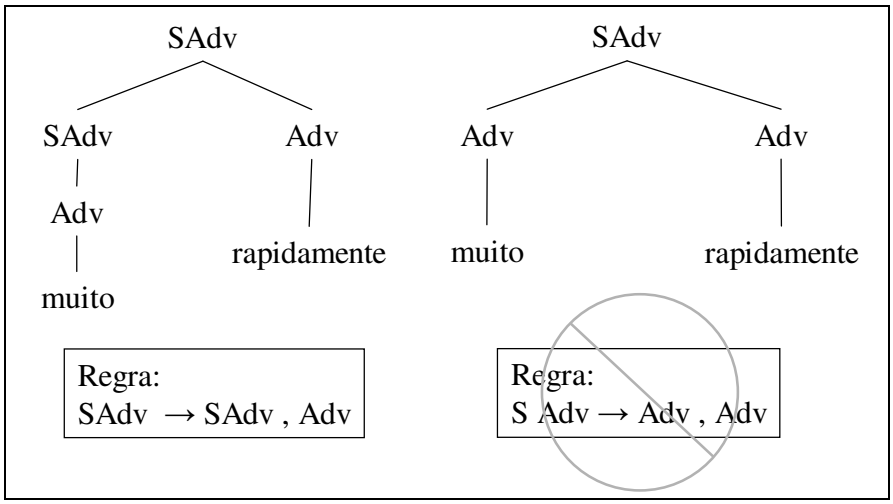


Figura 5.10: Exemplo de regra de ouro.

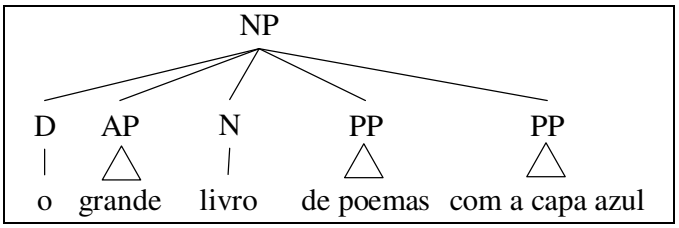


Figura 5.11: Exemplo de estrutura plana.

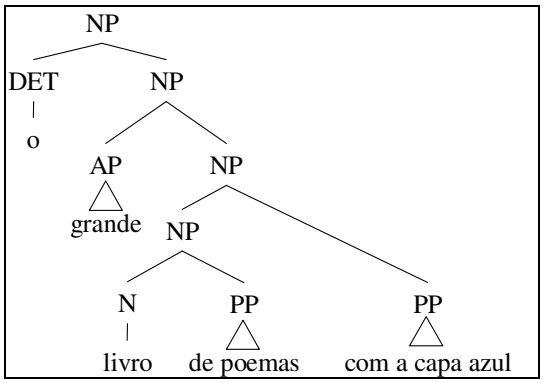


Figura 5.12: Exemplo de estrutura articulada.

Um outro aspecto importante, baseado na teoria X-bar apresentada na seção 2.2.2.2.4, a ser considerado no desenho de árvores sintáticas é a busca de estruturas mais articuladas, i.e., menos planas, que permitam justificar as associações entre itens nas sentenças analisadas. A figura 5.11 ilustra uma estrutura plana e a figura 5.12 ilustra a mesma estrutura mais articulada, sendo que o triângulo representa uma simplificação de estrutura sintagmática. Comparando-se ambas as figuras com as estruturas apresentadas pela teoria *X-bar* nas figuras 2.30 e 2.31, a presente teoria não utiliza as categorias intermediárias, e.g., N' (N-bar), V' (V-bar), mantendo as categorias padrões, e.g., NP, VP, com aplicação do conceito da definição de estruturas mais bem articuladas. Cabe ressaltar que as

substituições propostas na teoria *X-bar* baseiam-se, entre outras, em *one-replacement* para NP e *do-so* (ou *did-so*) para VP, não sendo as mesmas aplicáveis ao idioma português.

A estrutura sintagmática articulada permite a utilização mais intensa do conceito da recursividade quando implementada em uma linguagem lógica tal qual Prolog, i.e., regras maiores são compostas por regras menores permitindo um maior numero de sentenças a serem geradas com as mesmas combinações de regras. O desenho ilustrado na figura 5.13 permite exemplificar o referido conceito; inicialmente, o engenheiro de conhecimento desenha a árvore, sendo as respectivas regras sintáticas geradas, tal qual ilustra a figura 5.14. O resultado do processamento da sentença *o paciente tem uma identidade, um nome e um endereço* gera a estrutura sintagmática plana mostrada na figura 5.15 e o resultado para a sentença *o paciente tem uma identidade, um CRM, um nome e um endereço* é ilustrado na figura 5.16, sendo a diferença entre ambas o sintagma nominal *um CRM*. A recursividade permitiu que ambas as sentenças fossem processadas por uma mesma base de conhecimento sintática composta pelas regras e fatos sintáticos gerados a partir da estrutura desenhada na figura 5.13.

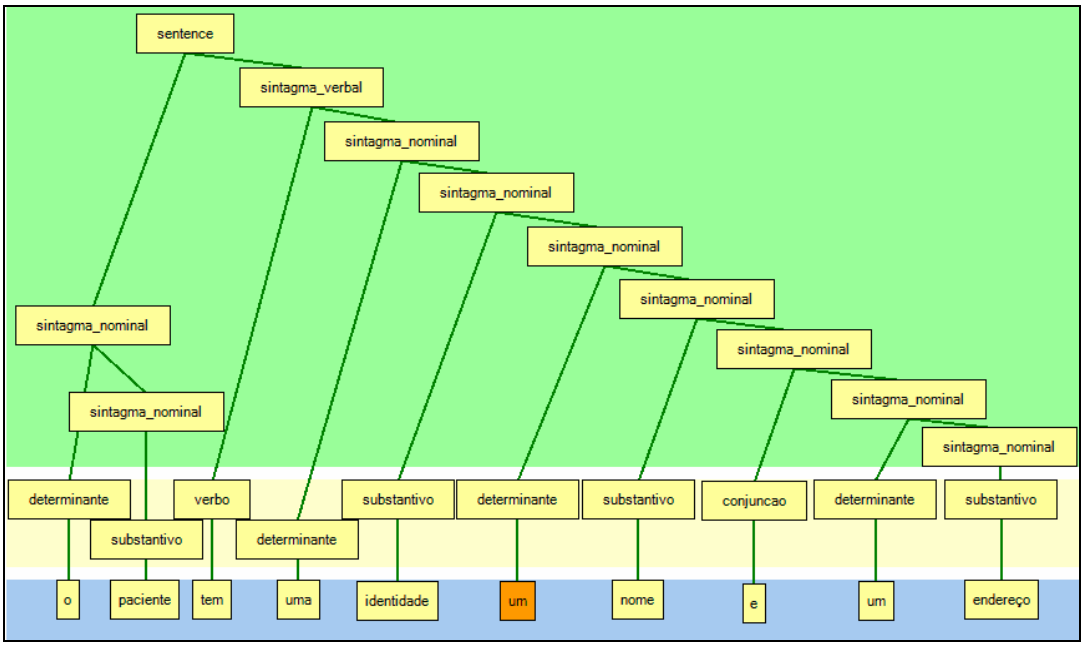


Figura 5.13: Exemplo de estrutura recursiva.

```
sentence(Number,sentence(SN,SV)) --> sintagma_nominal(Number,SN),sintagma_verbal(Number,SV),!.
sintagma_nominal(Number,sintagma_nominal(Subst)) --> substantivo(Number,Subst).
sintagma_nominal(Number,sintagma_nominal(Conj,SN)) --> conjuncao(Conj),sintagma_nominal(Number,SN).
sintagma_verbal(Number,sintagma_verbal(Verbo,SN)) --> verbo(Number,Verbo),sintagma_nominal(Number,SN).
sintagma_nominal(Number,sintagma_nominal(Det,SN)) --> determinante(Number,Det),sintagma_nominal(Number,SN).
sintagma_nominal(Number,sintagma_nominal(Subst,SN)) --> substantivo(Number,Subst),sintagma_nominal(Number,SN).
```

Figura 5.14: Exemplos regras sintáticas.

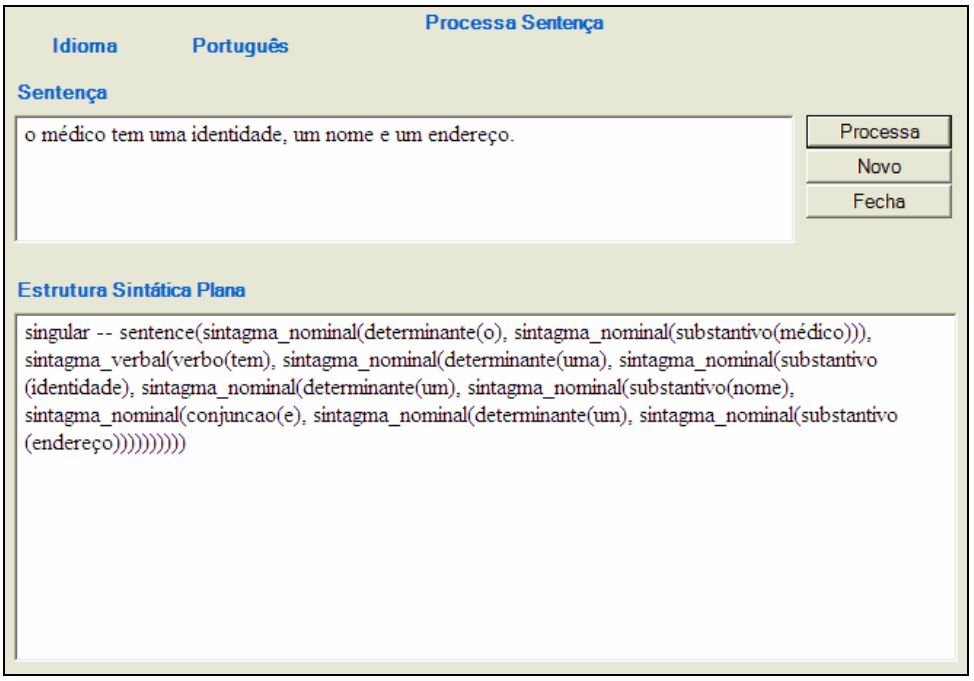


Figura 5.15: Exemplo de estrutura sintática plana com três atributos.

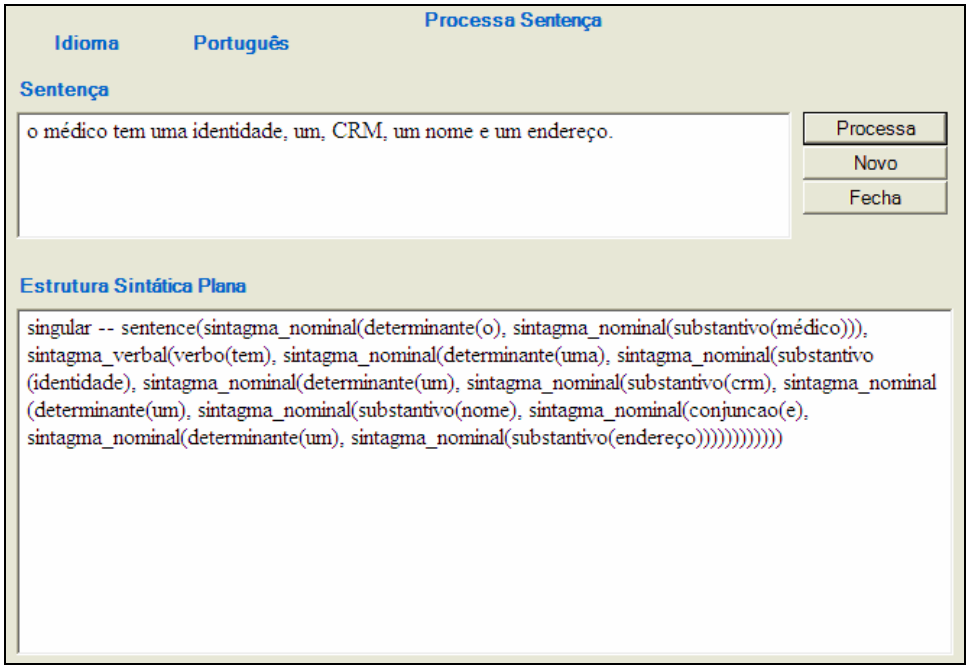
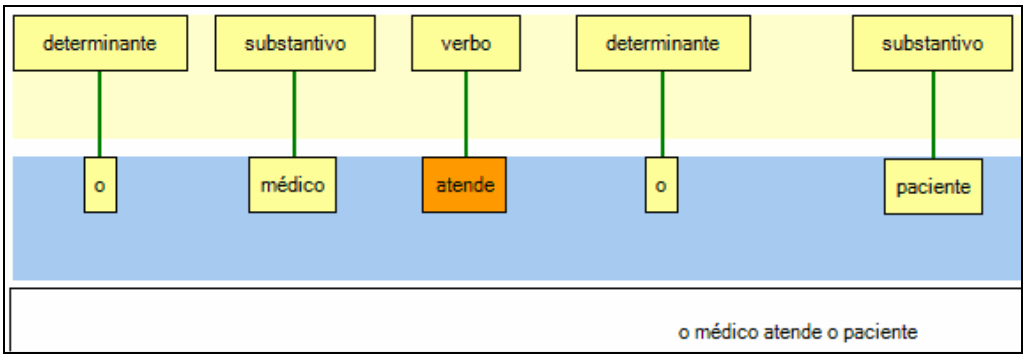


Figura 5.16: Exemplo de estrutura sintática plana com quatro atributos.

O desenho de uma árvore sintática exige prática e pode ser realizado de duas formas: *botton-up* e *top-down*. A técnica *botton-up* é mais indicada para engenheiros de conhecimento menos experientes, ao contrário da *top-down* (CARNIE, 2002).

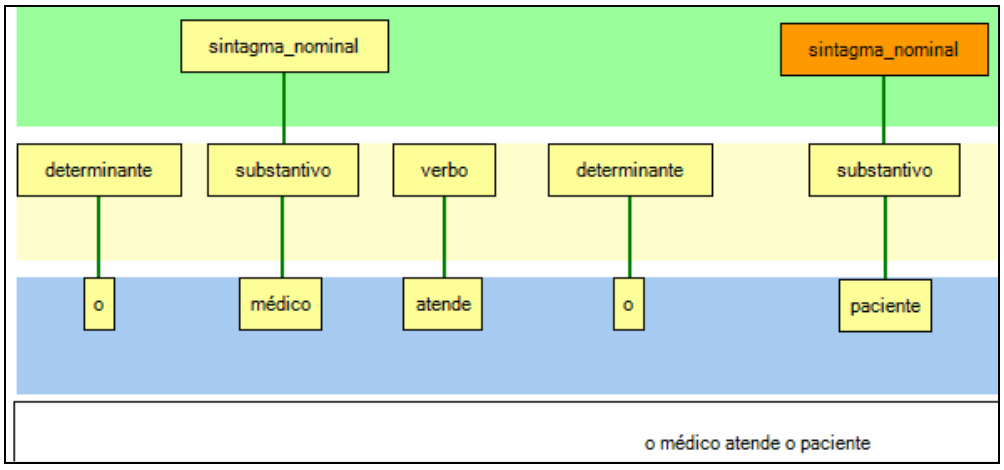
A seguir serão listadas instruções passo-a-passo para o desenho da árvore sintática de acordo com o método *botton-up*.

a) Desconsiderando a pontuação, escrever a sentença e identificar a *PoS* de cada palavra.

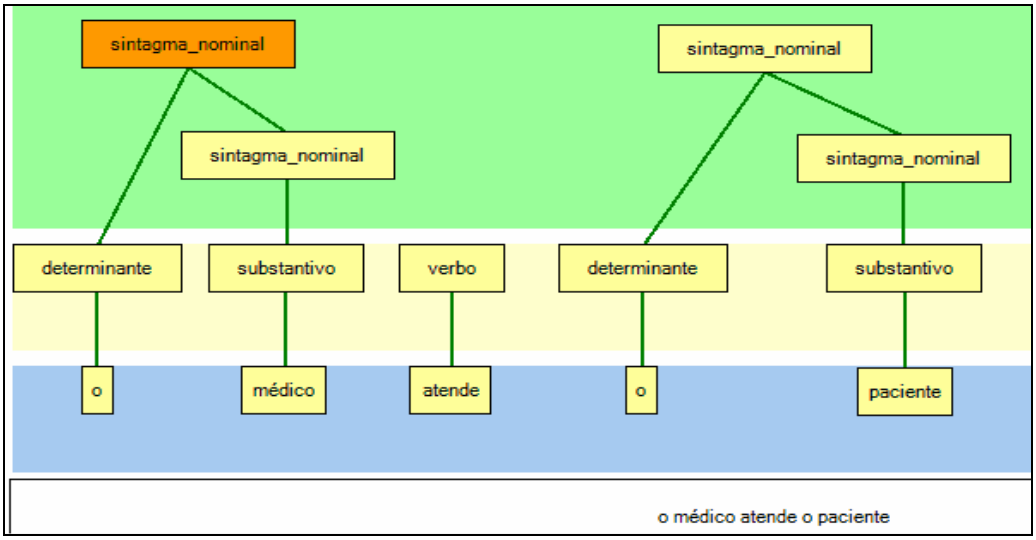


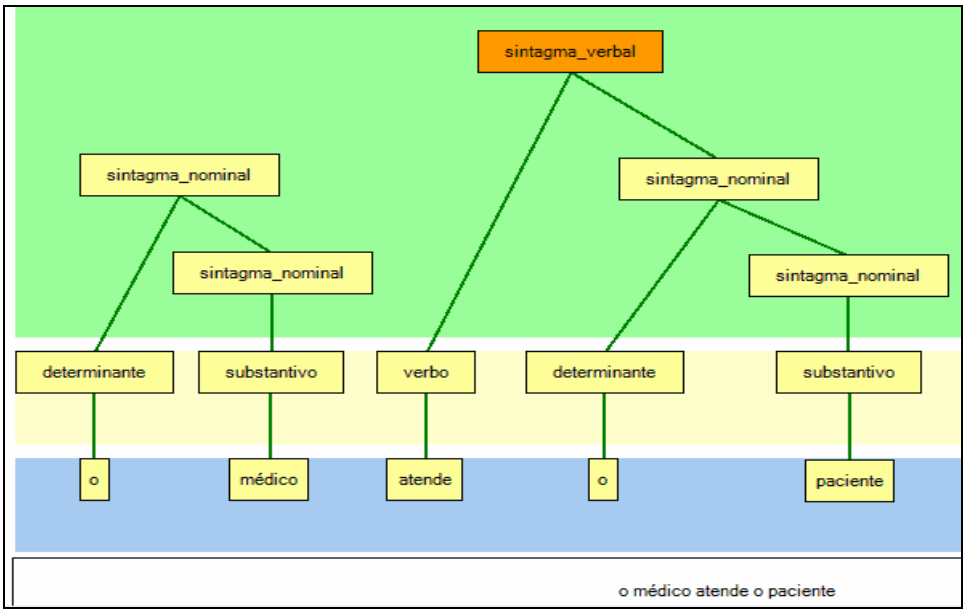
b) Identificar o que modifica o que, de modo que se um modificador modifica algo, ambos fazem parte do mesmo sintagma.

c) Iniciar ligando itens que modificam um ao outro, e.g., se um modificador está modificando um substantivo, então deve ser gerada uma regra do tipo *sintagma nominal*.

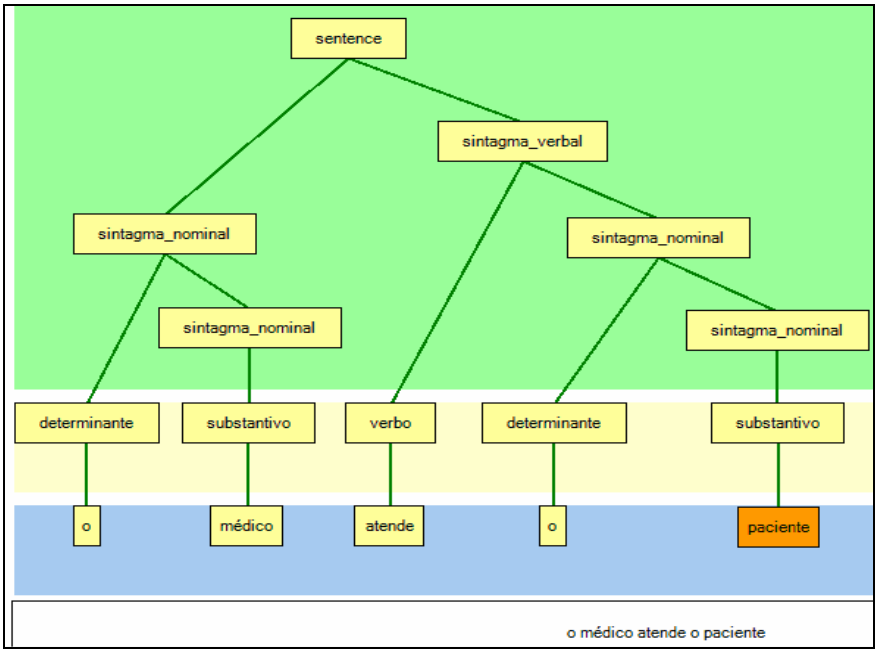


d) Tentar compor as regras na seguinte ordem: *sintagma adverbial*, *sintagma nominal* e *sintagma preposicionado*, *sintagma verbal* e *sentença*. Se uma sentença tem mais de uma oração, sugere-se iniciar pela sentença à direita e caso exista sentença embutida, iniciar por esta.

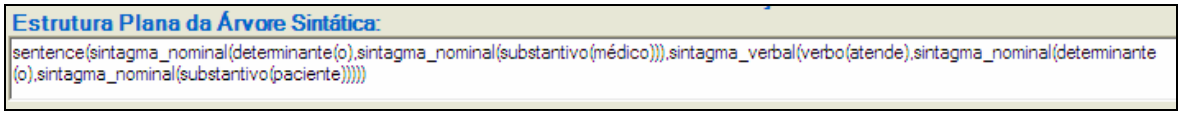




e) Quando estiverem definidos os sintagmas SN e SV, aplicar a regra S (sentença).



f) Verificar na caixa de texto *Estrutura Plana da Árvore Sintática* se a estrutura plana gerada pela árvore sintática está correta.



Algumas observações importantes:

• todos os símbolos terminais e não terminais devem estar ligados na estrutura;

• cada símbolo deve possuir uma única linha imediatamente acima do mesmo, sendo que para baixo é possível existir mais de uma linha;

• não cruzar linhas.

5.5.3 Teoria Sintática

No contexto do nível lingüístico sintático e alinhado aos requisitos determinados na seção 5.4, a teoria sintática adotada para o analisador sintático, ou *parser sintático,* do *ASIEEOOD* inclui os seguintes requisitos lingüísticos:

• a gramática adotada é a gramática de estrutura sintagmática, ou *PSG*, tal qual definida na seção 2.2.1.2;

• a linguagem de representação do conhecimento adotada inclui a extensão notacional Prolog denominada *Definite Clause Grammar (DCG)*, por permitir uma aplicação direta da referida linguagem lógica como analisador sintático, facilitando a implementação formal da *PSG* e a decomposição da sentença em seus constituintes, bem como a implementação do conceito de *dependência do contexto*, onde uma sentença depende do contexto onde ela ocorre, tal como o número de concordância (BRATKO, 2001), tendo em vista que este conceito auxilia na identificação de multiplicidades;

• a notação do Diagrama de Classe da UML como estrutura de representação do conhecimento, i.e., a rede semântica do analisador sintático.

A análise sintática proposta possui dois objetivos:

• verificar a correção da sintaxe das sentenças analisadas do discurso;

• gerar a estrutura sintática plana da sentença analisada para posterior extração dos constituintes sintáticos, i.e., a estrutura sintática e a categoria gramatical, dos vocábulos que compõem as referidas sentenças.

A teoria sintática para o analisador sintático do *ASIEEOOD* tem como fundamento a Teoria do Modelo, i.e., o estudo das interpretações dos sistemas formais, que enfocam as relações entre teorias e modelos. Um conjunto de axiomas juntos com todos os teoremas deriváveis dos mesmos é chamado de uma *teoria*, i.e., um conjunto de declarações que possuem vínculos decorrentes de conseqüências lógicas. Um modelo para uma teoria requer um domínio abstrato e estruturado e a interpretação para todas as expressões primitivas da teoria naquele domínio; de acordo com determinada interpretação, todas as declarações na teoria tornam-se verdadeiras para aquele modelo naquela interpretação (PARTEE *et al.*, 1990). A partir destes fundamentos, é proposto o modelo descrito a seguir.

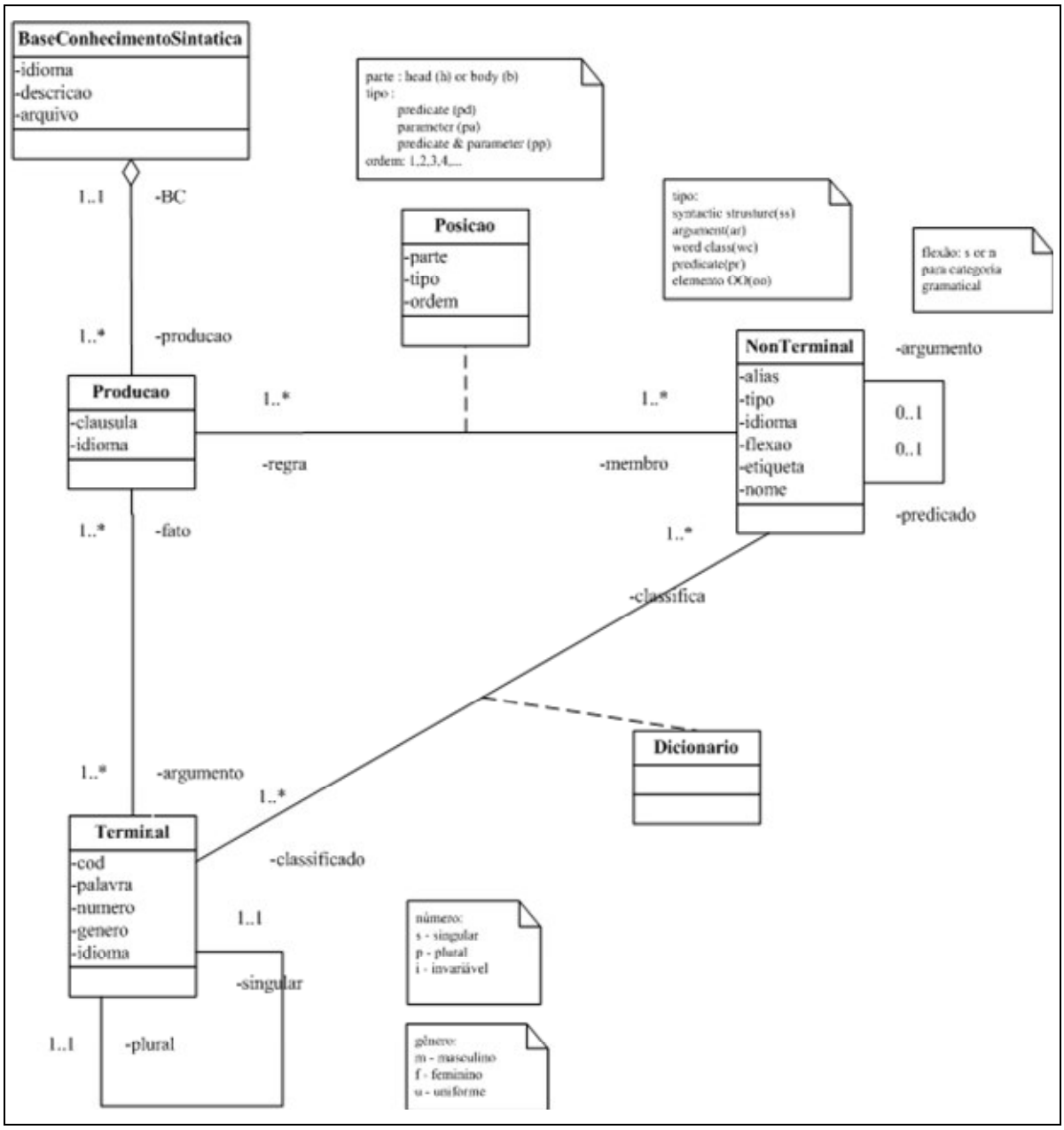


Figura 5.17: Rede semântica do *MESU.*

Definição 1: Um *Modelo de Estrutura Sintática baseado em UML (MESU)* para uma gramática *G* é composto pela estrutura *‹T, N, P, S*> (HAUSSER, 2001), (GRISHMAN, 1999) e (SILVA e CARVALHO, 2006a) onde:

i) *T* é o vocabulário terminal, i.e., todos os vocábulos e símbolos de um determinado idioma;

ii) *N* é o vocabulário não terminal que inclui os símbolos usados na especificação da gramática, e.g., *sintagma_nominal*;

iii) *P* é um conjunto de axiomas, i.e., produções, representados por regras sintáticas, e.g., *sentence(Number, sentence(NP, VP)) --> sintagma_nominal(Number, NP), sintagma_verbal(Number, VP),* e fatos sintáticos, e.g., *determinante(determinante(a)) --> [a],* que comporão a base de conhecimento;

iv) *S* é o símbolo inicial;

v) A estrutura de representação do conhecimento do *MESU* está ilustrada na figura 5.17.

Definição 2: Uma produção *P* é um axioma do tipo regra sintática representada por predicados *n-ários* e argumentos extraídos de *N* com a seguinte sintaxe em BNF:

<regraSintatica> ::= <predicadoCabeca> "→" <listaPredicadoCorpo>"."

<predicadoCabeca> ::= <atomoEstruturaSintatica>(<listaTermoSintaticoCabeca>)

<listaPredicadoCorpo> ::=

<atomoEstruturaSintatica>(<listaTermoSintaticoCorpo>) ,

<listaPredicadoCorpo> | <atomoCategoriaGramatical>

(<listaTermoSintaticoCorpo>), <listaPredicadoCorpo>

<listaTermoSintaticoCabeca> ::=<atomoNumero> , predicadoCabeca |

<listaTermoVariavelSintatica>

<listaTermoSintaticoCorpo> ::=<atomoNumero> ,

<termoVariavelEstruturaSintatica>| <atomoNumero> ,

<atomoVariavelCategoriaGramatical> |

<atomoVariavelCategoriaGramatical>

<listaTermoVariavelSintatica> ::=

<termoAliasEstruturaSintatica>,<listaTermoVariavelSintatica> |

<atomoAliasCategoriaGramatical>,<listaTermoVariavelSintatica>

<termoVariavelEstruturaSintatica> ::= <atomoVariavelEstruturaSintatica>

|<atomoVariavelEstruturaSintatica> <numeral>

< atomoEstruturaSintatica > ::= sentence | sintagma_nominal | ... | sintagma_verbal

<atomoVariavelEstruturaSintatica> ::= SN | SAdj | SP | SAdv | SV

<atomoVariavelCategoriaGramatical> ::= Subst | Verbo | Adj | Adv | ...

< atomoCategoriaGramatical >::= substantivo | adjetivo | ... | verbo

<atomoNumero> ::= Number | Number<numeral>

<numeral> ::= <digito> | <numeral><digito>

Definição 3: Uma produção *P* é um axioma do tipo fato sintático representada por um predicado com dois argumentos cuja composição inclui elementos extraídos de *T* e *N* com a seguinte sintaxe em BNF:

<fatoSintatico> ::= <predicadoCabeca> "→" [<atomoTerminal>]"."

<predicadoCabeca> ::= <atomoCategoriaGramatical>(<atomoNumero>, <predicadoCategoriaGramatical>)

< predicadoCategoriaGramatical > ::= <atomoCategoriaGramatical> (<atomoTerminal>)

< atomoCategoriaGramatical >::= substantivo | adjetivo | verbo | ...

(<atomoTerminal>) ::= vocábulos do léxico

<atomoNumero> ::= singular | plural

Definição 4: O *TeSe* é um teorema sintático, i.e., teorema presumido a ser provado, a partir dos axiomas regras e fatos sintáticos, com o seguinte subconjunto da Gramática *G* e sintaxe em BNF:

<teoremaSintatico> ::= sentence(X, ParseTree, [<atomoTerminal>, <listaAtomoTerminal>], []) "."

<listaAtomoTerminal> ::= <atomoTerminal> | <atomoTerminal>,<listaAtomoTerminal>

<atomoTerminal> ::= vocábulos do léxico

As definições 2 e 3 do modelo *MESU* permitem a geração dos axiomas que irão compor a base de conhecimento sintática, i.e., as regras e fatos sintáticos, sendo, de acordo com Sowa (2000), um modelo computacional com método declarativo, i.e., baseado em axiomas. O *MESU* utiliza técnicas de prova de teoremas na derivação de suas conseqüências lógicas (CLOCKSIN e MELLISH, 2003), tendo os referidos teoremas a sintaxe determinada por meio da definição 4. A prova de um teorema para uma determinada instância de um modelo MESU significa a ocorrência de uma interpretação, i.e., determinadas regras e fatos sintáticos tornam-se verdadeiros para aquele modelo naquela interpretação. Uma base de conhecimento sintática é uma instância do modelo MESU.

5.5.4 Linguagem Controlada

As frases de uma língua não são catalogáveis por que são infinitas tanto em número como, teoricamente, em extensão. As palavras, sim, podem ser listadas em dicionários e mesmo assim muitas se criam ou são modificadas (AZEREDO, 2003). Tal assertiva é corroborada por Carnie (2005) que afirma ser a linguagem um sistema produtivo infinito. Osborne e Macnish (1996) descrevem os numerosos problemas associados ao processamento de uma linguagem natural irrestrita, i.e., uma linguagem usada para comunicação entre humanos, sem considerar as restrições determinadas pela máquina, tais como a possibilidade de falha do léxico por não conter entradas para todas as palavras, a associação duas ou mais estruturas sintáticas a uma determinada sentença e a falha na análise semântica por ter que considerar todas as construções sintáticas geradas. Os autores destacam, também, que o impacto dos referidos problemas pode ser reduzido por meio da restrição da linguagem utilizada.

Esta técnica lingüística adota o conceito de linguagem controlada, i.e., limitar a estrutura frasal utilizada pelo analista que realiza a especificação textual do modelo conceitual, a fim de permitir uma otimização do processamento computacional do discurso analisado, particularmente na redução da quantidade de estruturas sintáticas possíveis. A presente pesquisa está alinhada com Schwritter e Fuchs (1996), que destacam a dificuldade da derivação de especificações formais a partir de requisitos informais, tendo em vista a disparidade entre os mundos conceituais do domínio da aplicação e do desenvolvimento de software. Os referidos autores propõem uma linguagem denominada *Attempto* como um subconjunto da linguagem natural que possa ser processada por um computador.

Cabe destacar que o núcleo de conhecimento principal da presente pesquisa de doutorado está na especificação da teoria lingüística implementada pelo analisador semântico, sendo a definição de uma linguagem controlada necessária para a realização de um estudo analítico que permita avaliar a hipótese proposta para a solução do problema pesquisado. No contexto das restrições relacionadas com o uso de uma linguagem natural irrestrita e da necessidade de realização da referida experimentação, a linguagem controlada do *ASIEEOOD* inclui as regras descritas a seguir, e respectivas justificativas, que possibilitam restringir as estruturas sintáticas do Português a serem empregadas pelos analistas.

Restrição	Justificativas / Exemplos
(1) Sentença declarativa afirmativa.	• O tipo declarativo afirmativo pode combinar-se com toda e qualquer oração, sem sofrer outras alterações que não a transformação obrigatória de concordância e aquelas de cunho meramente estilístico, e.g., deslocamentos (CHOMSKY, 2002); • Elimina os subtipos interrogativos, imperativos e exclamativos (SOUZA E SILVA e KOCK, 2004); • Aplicação de sentenças declarativas (SCHWRITTER e FUCHS, 1996).
(2) Sentença com sujeito, verbo e predicado	• Associações aparecem freqüentemente como verbos nas declarações dos problemas (RUMBAUGH e BLAHA, 2005); • O verbo é um determinante para as associações entre sujeitos e predicados , e.g., o médico (*sujeito*) atende o paciente (*predicado*).
(3) Não utilizar nomes próprios.	• Nomes próprios são considerados *rigid designation*, sendo sua referência uma pessoa, no caso não se referencia a uma entidade que é uma referência (GAMUT, 1991).
(4) Tempo presente.	• Em lógica, proposições são tradicionalmente independentes de tempo e lugar; de modo que uma sentença no passado refere a um momento no tempo anterior ao provido pelo contexto, o que requer dois contextos (GAMUT, 1991).
(5) Pronomes nominativos, acusativo, anafórico.	• O significado do discurso está relacionado a outro vocábulo próximo no discurso, e.g., para ele também..., aquele (a quem nos referimos) comprou.... (CARNIE, 2002).

Restrição	Justificativas / Exemplos
(6) Não usar sujeito indeterminado.	• Idem restrição (2); • e.g., alguém ..., precisa-se de ... (SOUZA e SILVA E KOCK , 2004).
(8) Não usar pronome reflexivo.	• Idem restrição (5); • Elimina transformação reflexiva, e.g., a menina enfeitava-se ..., ...para si (SOUZA E SILVA e KOCK , 2004).
(9) Não usar regionalismos nem coloquialismos.	• Inté, p'ra.
(10) Usar abreviações substantivadas.	• Instância de um atributo, e.g., CREA (como atributo de uma classe engenheiro) (BICK, 2002).
(11) Uma relação deve estar escrita em ambas as direções.	• Considerando que a multiplicidade especifica o número de instâncias de uma classe que pode se relacionar com uma única instância de uma classe associada (RUMBAUGH e BLAHA, 2005), é necessário garantir as multiplicidades em ambas as direções; • e.g., o médico atende os pacientes / o paciente é atendido pelos médicos.
(12) Um atributo sempre é uma característica de uma classe e deve estar sempre após o verbo, i.e., no predicado.	• e.g., o médico tem um CRM, um nome e um endereço.
(13) Tratamento na terceira pessoa	• e.g., o médico atende o paciente.
(14) Não usar pronome de tratamento.	• Idem restrição (5); • e.g., V. Exª, V.Sª.

Restrição	Justificativas / Exemplos
(15) Artigos, numerais, pronomes ou advérbios devem ser utilizados nas frases para definição das multiplicidades.	• e.g., o médico atende o paciente (1..1); • e.g., o médico atende os pacientes (1..N); • e.g., o médico atende vários pacientes (1..N); • e.g., o médico atende um ou mais pacientes (1..N); • e.g., o médico atende um ou muitos pacientes (1..N); • e.g., o médico atende zero ou muitos pacientes (0..N).

5.5.5 Ambigüidade Gramatical

Considerando o vocábulo *verde,* a figura 5.18 mostra que a predicação gramatical gera dois predicados em virtude do mesmo ter duas categorias gramaticais relacionadas, e.g., na sentença *a grama é verde*, o vocábulo *verde* é um adjetivo, sendo que na frase *o verde representa a natureza,* a mesma palavra é categorizada como *substantivo.*

A resolução da ambigüidade gramatical, i.e., um vocábulo com mais de uma categoria gramatical, é solucionada pelo analisador sintático por meio da posição sintática do vocábulo, sendo a categoria gramatical definida quando da prova do teorema e a conseqüente geração da estrutura sintática plana.

Predicação gramatical do vocábulo **verde:**

*substantivo(singular, substantivo(**verde**)) --> [**verde**].*
*adjetivo(singular, adjetivo(**verde**)) --> [**verde**].*

? A grama é verde. (teorema a ser provado)

*sentence(sintagma_nominal(determinante(a),substantivo(grama)),sintagma_verbal (verbo(é),**sintagma_adjetivo(adjetivo(verde))**))*

? O verde representa a natureza. (teorema a ser provado)

*sentence(**sintagma_nominal**(determinante(o), **substantivo(verde)**), sintagma_verbal(verbo(representa), sintagma_nominal(determinante(a), substantivo(natureza))))*

Figura 5.18: Resolução de ambigüidade morfológica.

As restrições impostas pela linguagem controlada na presente pesquisa permitem uma análise sintática mais simplificada, e.g., os verbos não são referentes aos elementos orientados a objeto, facilitando, também, o processo de desambigüidade morfológica. Bick

(2000) especifica as principais ambigüidades morfológicas que estão ilustradas na figura 2.15 (seção 2.2.2.1.3), sendo as mesmas avaliadas a seguir:

Homonímia	Exemplos	Avaliação
(1a) lexical (morfema livre): diferente forma base e mesma categoria.	- *foi*, "ir" VPS 3S; - *foi*, "ser" V PS 3S.	A referida homonímia aplica-se somente à categoria gramatical "verbo", o que não tem significância para a presente técnica lingüística em virtude da citada categoria não gerar elementos OO.
(1b) lexical: mesma forma base e diferente categoria.	- *complementar* V; - *complementar* ADJ.	Homonímia resolvida na análise sintática, pois a posição sintática está relacionada com a categoria gramatical, e.g., um verbo não ocupa a posição sintática de um adjetivo em uma determinada regra de estrutura sintagmática.
(1c) lexical: mesma forma base, diferente em uma ou mais categorias de lexemas.	- guarda F (grupo); - guarda M (vigilante).	Homonímia resolvida na atividade de validação do modelo conceitual.
(2) flexional (morfema amarrado): mesma forma base, diferente em uma ou mais categorias de forma de palavra.	- amamos PR 1P; - amamos PS 1P.	Homonímia não considerada na presente técnica tendo em vista que para a categoria "verbo" não existem elementos orientados a objeto relacionados.
(3) lexico-flexional (morfema livre e amarrado): diferente forma base, diferente em categoria de lexema e de forma de palavra.	- busca N F S - busca V PR 3S	Idem avaliação apresentada em (1b).

5.5.6 Estudo de Caso de Análise Sintática

O engenheiro de conhecimento do sistema *ASIEEOOD* dispõe de um componente editor gráfico, tal qual ilustrado na figura 5.19, que permite automatizar a diagramação da árvore sintática de uma determinada estrutura sintática, e.g., *o médico atende o paciente*. O referido componente gera a seguinte estrutura sintática plana, i.e., em forma textual, relativa à estrutura desenhada: *sentence(sintagma_nominal(determinante(o), sintagma_nominal(substantivo (médico))), sintagma_verbal(verbo(atende), sintagma_nominal(determinante(o), sintagma_nominal(substantivo(paciente)))))*.

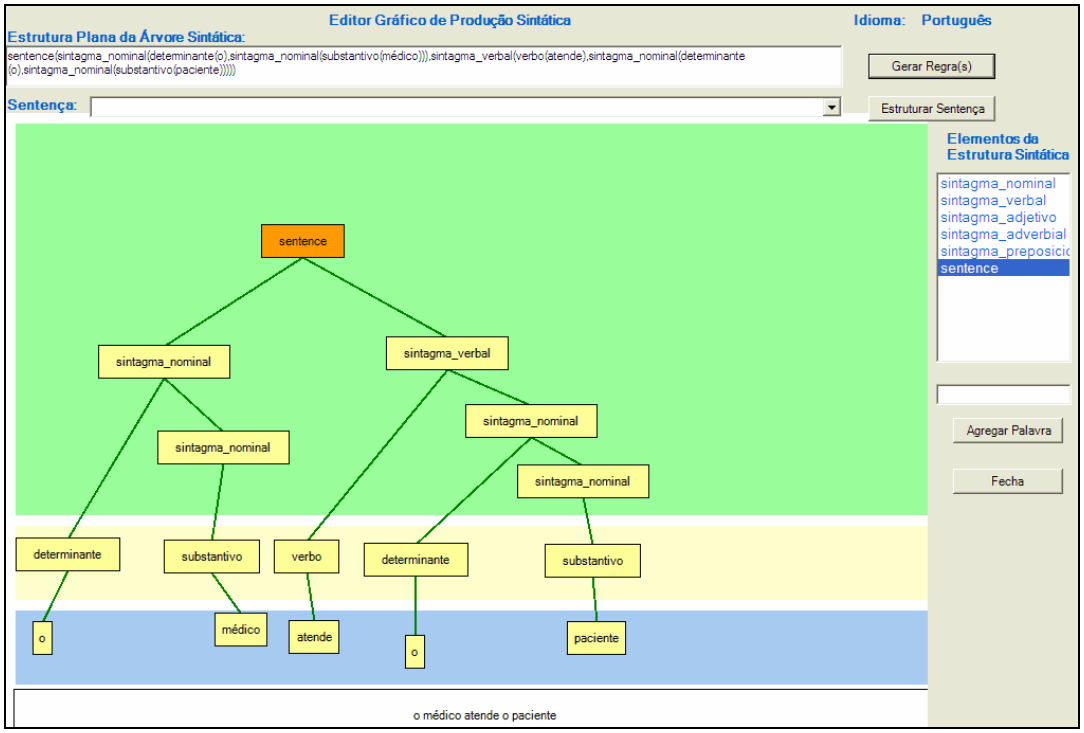


Figura 5.19: Editor gráfico do *ASIEEOOD*.

O método *GerarRegraSintaticaDeGrafico* do componente *CAnalisadorSintatico*, cuja interface está descrita na figura 5.20, implementa um algoritmo que extrai as seguintes regras sintáticas, em notação Prolog DCG, da citada estrutura frasal:

Regra 1: *sentence(Number,sentence(SN,SV)) -->*
sintagma_nominal(Number,SN),sintagma_verbal(Number,SV),!.

Regra 2: *sintagma_verbal(Number,sintagma_verbal(Verbo,SN)) -->*
verbo(Number,Verbo),sintagma_nominal(Number,SN).

Regra 3: *sintagma_nominal(Number,sintagma_nominal(Subst)) -->*
substantivo(Number,Subst).

Regra 4: *sintagma_nominal(Number,sintagma_nominal(Det,SN)) -->*
determinante(Number,Det),sintagma_nominal(Number,SN).

```
public __gc class CAnalisadorSintatico
{
private:
        String * sentenca;
        String * idioma;
        FILE * stream;
        ArrayList *listaRegras ;
public:
        CAnalisadorSintatico(void);
        CAnalisadorSintatico(String * idiom);
        CAnalisadorSintatico(String * sent, String * idiom);
        String * ProcessarSentenca(void);
        bool GerarRegraSintaticaDeGrafico(String * regraAS);
        ArrayList * LerProducaoBD(String * idioma, int idEstrSint);
        bool ArmazenarSentencaTemp(String * sentenca);
private:
        bool CriarBCProlog(String * path);
        int InicializarBC(int idEstrSint, String * idioma);
        int InserirBCProlog(String * path, ArrayList * arrProducao, bool criaBC);
        char * StringToChar(String * strAux);
        int CriarFatosLexicosDicionario(String * path, int qteArq);
        String * ProcessarSentencaProlog(ArrayList * arrPalavra, String * path);
        ArrayList * OrdenarRegrasProducoes(ArrayList * arrProd);
        String * FormatarFatoSintatico(String * palavra, String * catGram, String * flexao, String * numero);
        ArrayList * ExtrairElementosRegraSintatica(String * strSource);
        ArrayList * ListarElementosExtraidosRegraSintatica(String * strSource);
        String * EstruturarRegraSintatica(ArrayList * listaRegra);
        String * FormatarElementoDeRegra(String * elemento, String * &argumento, String * extensao);
        ArrayList *  ArmazenarRegraSintaticaGrafica(String * strSource, String * idioma);
        bool GravarRegra(String * idioma, ArrayList * arrProducao, int idSentenca);
        String * ExtrairPrimeiroElementoSubClausula(ArrayList * listaRegra);
        ArrayList * SepararElementosRegraSintatica(String * strSource);
        ArrayList * AjustarNumeroElementosRegraSintatica(ArrayList * arrRegras);
        ArrayList * SepararStringRegraSintatica(String * str);
        ArrayList * AjustarFlexaoNumeroRegrasSintaticas(ArrayList * arrRegra);
        String * AcrescentarCaracterNumero(String * elemento, int limite, ArrayList * arrRegra, String * &ultExt);
        String * AjustarParametrosRegra(String * str);
        String *  RenomearParametro(String * str, String * palavra, int qteParIgual);
        String * MontarProducaoDePosicao(ArrayList * arrPos);
        String * RetirarVirgulaSentenca(String * str);
        String * ObterExtensaoSintagma(String * str);
        String * DeterminarNumeroSintagma(String * str, ArrayList * arr);
        ArrayList * DeterminarNumeroEstrSintatica(ArrayList * arr);
};
```

Figura 5.20: Interface do componente *Analisador Sintático.*

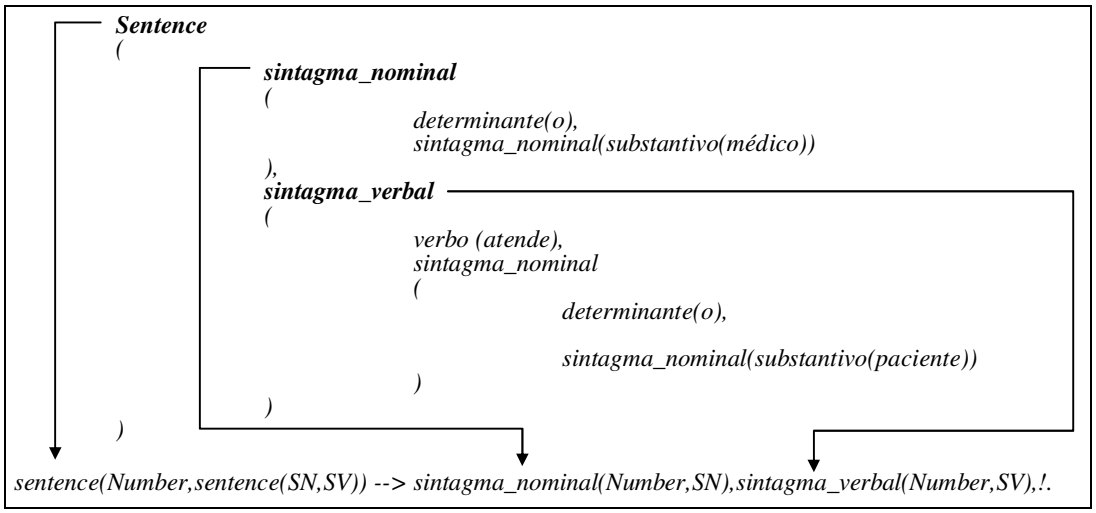


Figura 5.21: Esquema da geração da regra 01.

As figuras 5.21 e 5.22 ilustram os esquemas que, a partir da estrutura sintática plana gerada da sentença analisada e de acordo com a definição 2 do *MESU*, geram as regras sintáticas 1 e 2 anteriormente descritas. A figura 5.23 ilustra as referidas regras sintáticas em uma *base de conhecimento sintática* instanciada em tempo de execução, cabendo destacar que cada instância de uma base corresponde a uma estrutura sintagmática determinada pelo engenheiro de conhecimento, i.e., uma estrutura possível de interpretação.

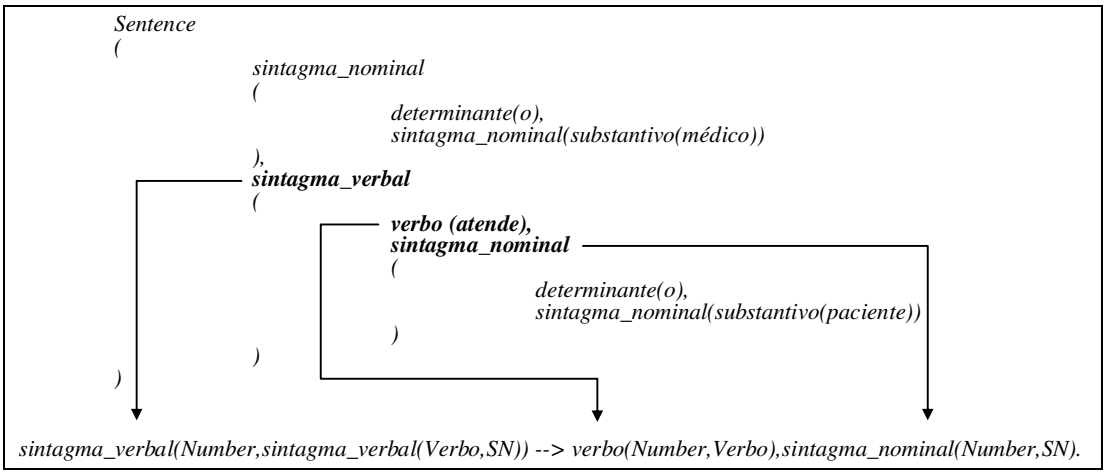


Figura 5.22: Esquema da geração da regra 02.

A figura 5.23 exemplifica um modelo instanciado a partir do *MESU*, i.e., as regras e fatos sintáticos, sendo os fatos resultantes da predicação gramatical apresentada na seção 5.5.1. Considerando a instanciação citada, a interpretação para a sentença *o médico atende o paciente* é verdadeira de modo que os fatos sintáticos relacionados com os cinco vocábulos da sentença, juntamente com as regras, são verdadeiros, cabendo ao motor de inferência Prolog o estabelecimento do vínculo lógico a partir do teorema a ser provado. Na figura 5.24, e de acordo com a sintaxe apresentada na definição 4 do *MESU*, o teorema a ser provado é *sentence(X, ParseTree, [o, médico, atende, o, paciente],[]),* cuja interpretação atribui à variável *X* o valor *singular*, i.e., a sentença tem a flexão de número predominantemente no singular, e a variável *ParseTree* o valor *sentence(sintagma_nominal(determinante(o), substantivo(médico)), sintagma_verbal (verbo (atende), sintagma_nominal (determinante(o), substantivo(paciente))))*, i.e., a estrutura sintática plana da sentença.

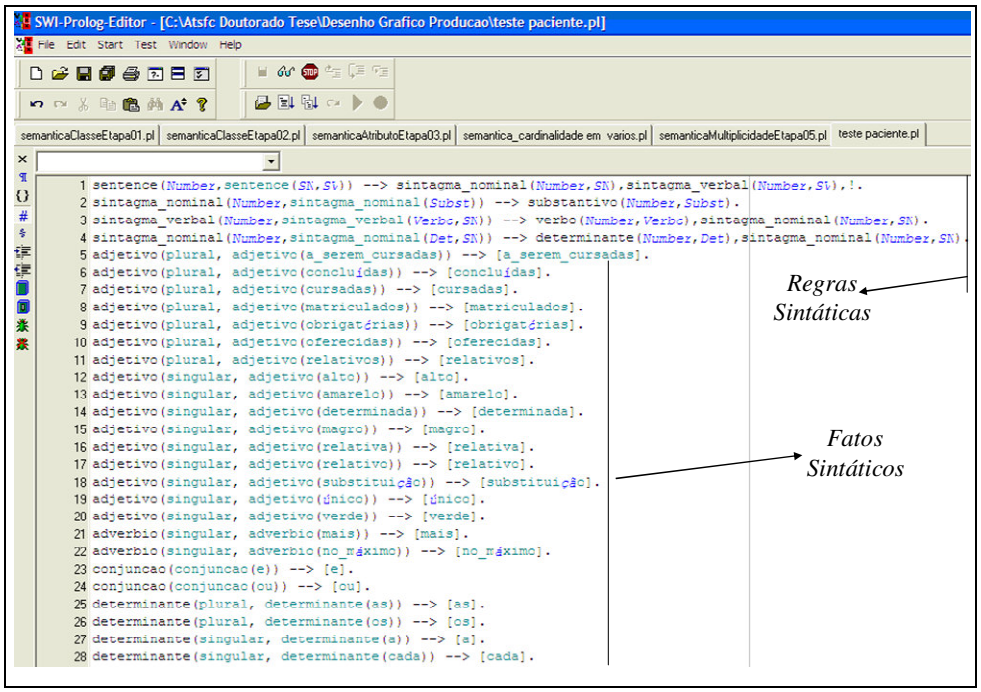


Figura 5.23: Exemplo de *Modelo MESU* para o estudo de caso sintático.

?- sentence(X, ParseTree, [o, médico, atende, o, paciente], []).

X = singular

ParseTree = sentence(sintagma_nominal(determinante(o), substantivo(médico)),
sintagma_verbal(verbo(atende), sintagma_nominal(determinante(o), substantivo(paciente)))) ;

Figura 5.24: Exemplo de teorema sintático *MESU* para o estudo de caso sintático.

5.6 Nível Lingüístico Semântico

De acordo com Gamut (1991), o núcleo de uma teoria lingüística está em sua teoria semântica, sendo o ponto de conexão entre sintaxe e semântica um dos aspectos mais delicados do estudo lingüístico. A presente tese está alinhada com a premissa citada, sendo, portanto, o núcleo desta pesquisa a teoria semântica.

O objetivo primário da análise semântica proposta é extrair o significado do discurso, i.e., os seguintes elementos orientados a objeto: as classes, os atributos, as associações e as multiplicidades.

5.6.1 Categorias Gramaticais *versus* Elementos Orientados a Objeto

Um aspecto importante do assistente proposto é o relacionamento das categorias gramaticais com os elementos orientados a objeto foco da presente pesquisa. A seguir são

apresentados conceitualmente os elementos orientados a objeto, buscando-se identificar as associações existentes entre estes e as categorias gramaticais.

Classe. Descreve um grupo de objetos com as mesmas propriedades (atributos), comportamento (operações), tipos de relacionamentos e semântica (RUMBAUGH e BLAHA, 2005). As principais fontes para identificação de classes incluem os substantivos comuns (RUMBAUGH e BLAHA, 2005), (LARMAN, 2000) e (AMBLER, 2005) e os sintagmas nominais (RUMBAUGH e BLAHA, 2005) e (LARMAN, 2000), e.g., pessoa, processo.

Atributo. Nome de uma propriedade de uma classe que descreve um *valor* contido por cada objeto da classe (RUMBAUGH e BLAHA, 2005). Os referidos autores destacam a seguinte analogia: objeto está para classe assim como valor está para atributo. As principais fontes para identificação de atributos incluem os adjetivos, e.g., o valor "urgente" pode conduzir ao atributo *prioridade*, a abstração de valores típicos, e.g., o valor "1.000 km" pode conduzir ao atributo *distância* (RUMBAUGH e BLAHA, 2005) e substantivos podem ser candidatos a atributos (LARMAN, 2000).

Associação. Descrição de um grupo de vínculos, i.e., conexões físicas entre objetos, com estrutura e semântica comuns. Uma associação descreve um conjunto de potenciais vínculos da mesma forma que uma classe descreve um conjunto de potenciais objetos. Associações freqüentemente aparecem como *verbos* nas declarações dos problemas (RUMBAUGH e BLAHA, 2005).

Multiplicidade. Especifica o número de instâncias de uma classe que pode relacionar-se com uma única instância de uma classe associada. A literatura freqüentemente descreve multiplicidade como sendo "um" ou "muitos", normalmente é um subconjunto de números não negativos. A UML especifica multiplicidades com um intervalo, tais como "1" (exatamente um), "1..*" (um ou mais), ou "3..5" (três a cinco, inclusive). As multiplicidades devem ser definidas após a determinação das classes e associações (RUMBAUGH e BLAHA, 2005). O presente trabalho adota como multiplicidades possíveis para o desenho gráfico os seguintes valores: numerais, e.g., 1 (um), ou *, para muitos.

Destaca-se, no estudo gramatical, a oposição existente entre substantivo e adjetivo. Os substantivos nomeiam seres e conceitos, sendo palavras ligadas à capacidade humana de perceber os dados da realidade e denominá-los, individualizando-os. Os adjetivos atuam sempre em função dos substantivos, caracterizando-os. Observa-se que os adjetivos são fontes para identificação de atributos, e.g., o valor *amarelo* pode conduzir ao atributo *cor*

(RUMBAUGH e BLAHA, 2005), como conseqüência, os mesmos não permitem gerar diretamente os elementos orientados a objeto denominados *atributos*.

O *ASIEEOOD* possui uma funcionalidade, ilustrada na figura 5.25, que relaciona substantivos e adjetivos a fim de permitir a extração do elemento orientado a objeto *atributo,* da categoria *substantivo* correspondente a uma instância da classe *adjetivo*, por meio de regras semânticas. A etiquetação secundária apresentada na seção 5.4.2 permite relacionar as referidas categorias quando da análise sintática da sentença na qual o vocábulo pertence.

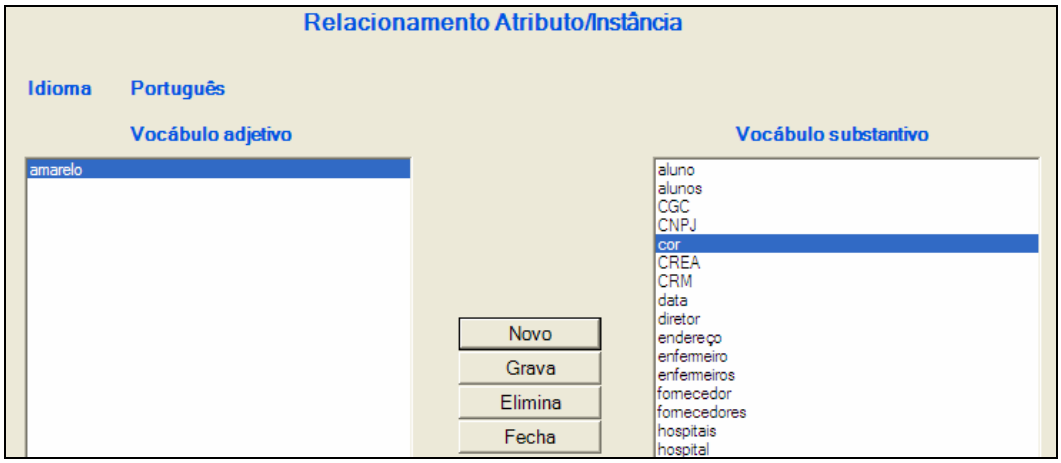


Figura 5.25: Funcionalidade do *ASIEEOOD* que relaciona atributo e instância.

5.6.2 Teoria Semântica

O núcleo de uma teoria lingüística está em sua teoria semântica (GAMUT, 1991), sendo que neste contexto a presente pesquisa está conforme com:

- a teoria semântica apresentada por Kamp e Reyle (1993) que destaca a existência de uma sistemática conexão entre significado e forma lingüística, tal como uma conexão baseada em regras, sendo que a noção de significado inclui extrair conteúdo de tais formas;
- a Teoria do Significado de Frege (ver seção 2.2.2.3.2), cuja interpretação semântica permite determinar que o significado de um vocábulo se refere a um elemento orientado a objeto (GAMUT, 1991) (HAUSSER, 2001);
- o princípio da não-composicionalidade no relacionamento entre a sintaxe e a semântica durante a interpretação de uma sentença (HAUSSER, 2001);
- a existência de uma independência entre a gramática e a semântica (CHOMSKY, 2002);
- a semântica lógica, ou semântica da teoria do modelo (GAMUT, 1991).

A conexão entre significado e forma lingüística está descrita no mecanismo lingüístico intuitivo usado por analistas apresentado na seção 5.3.1. Nesse contexto, esta teoria semântica tem como premissa básica que a extração do significado, i.e, os elementos orientados a objeto, ocorre em função dos constituintes sintáticos dos vocábulos, i.e., estrutura sintática e categoria gramatical, independentemente do significado da palavra propriamente dito.

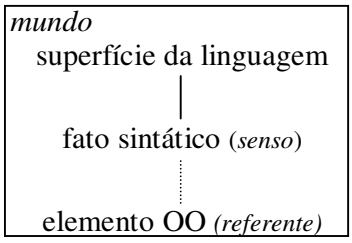


Figura 5.26: Interpretação semântica adotada.

A figura 5.26 ilustra a adequação da presente teoria à Teoria do Significado de Frege, i.e., da interpretação semântica adotada. Cada palavra, em uma determinada frase analisada, é traduzida em uma linguagem lógica, a partir dos constituintes sintáticos gerados pelo analisador sintático, em uma forma denominada de *fato sintático*, que é o *modo de apresentação*, i.e., *sense* na teoria de Frege, adotado como critério pelo qual o referente, i.e., *referent* na teoria de Frege, é determinado (GAMUT, 1991) (HAUSSER, 2001). Como exemplifica a figura 5.27, a palavra *paciente* na sentença *o doutor atende o paciente* tem como modo de apresentação o fato sintático *constituinteSintatico (sintagma_verbal, sintagma_verbal(sintagma_nominal), substantivo, paciente, singular, s1)*, i.e., um predicado que inclui dois posicionamentos na estrutura sintática da frase, sendo o primeiro relacionado com o sujeito ou predicado, a categoria gramatical, a flexão em número e o identificador de sentença no contexto do discurso. O referente do vocábulo *paciente* é o elemento orientado a objeto *classe*.

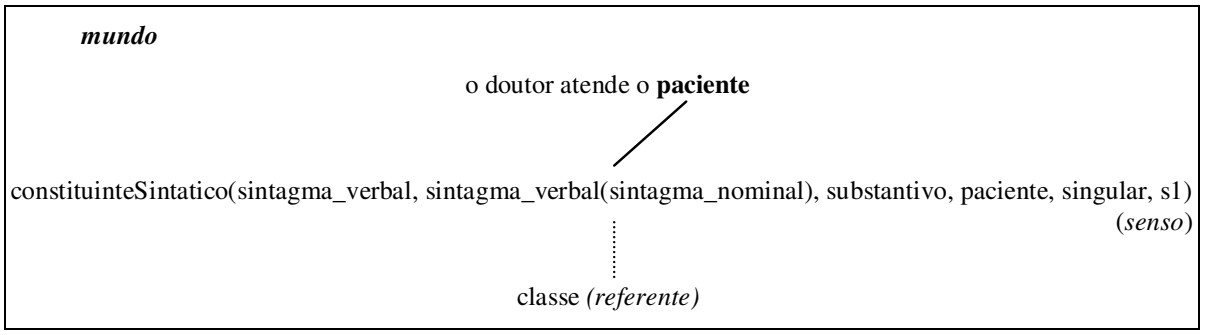


Figura 5.27: Exemplo da interpretação semântica adotada.

O princípio da composicionalidade, apresentado por Hausser (2001), estabelece um relacionamento entre sintaxe e semântica durante a interpretação de uma sentença, i.e., a interpretação de uma frase é obtida por meio da interpretação de seus componentes (ver seção 2.2.2.3.1). Para esta técnica lingüística, a extração do conteúdo consiste em analisar sintaticamente as sentenças e derivar os significados do todo, i.e., do discurso. A caracterização desta técnica como não-composicional justifica-se pela inexistência de uma relação direta entre as expressões analisadas e a realidade, existindo um nível intermediário de representação semântica onde as informações geradas pelo discurso são armazenadas na forma de fatos sintáticos, sendo estes especificados pelos constituintes sintáticos dos vocábulos, i.e., estrutura sintática e categoria gramatical.

Em lingüística, de acordo com Hausser (2001), a semântica é um componente da gramática que deriva representações de significado de superfícies naturais analisadas sintaticamente. A presente técnica adota a independência da gramática proposta por Chomsky (2002), i.e., autônoma e independente do significado. A principal justificativa está na não-composicionalidade da mesma.

A presente técnica está fundamentada, também, em semântica lógica, ou semântica da teoria do modelo, de acordo com uma visão lógica semântica apresentada por Gamut (1991) e Hausser (2001) (ver seção 2.2.2.3.3). No nível I, ilustrado na figura 5.28, as superfícies da linguagem são traduzidas em proposições, denominadas de *fatos sintáticos*, por meio da linguagem lógica de representação do conhecimento adotada, i.e., *cláusula definida de Horn*. Os algoritmos de atribuições, denominados de *regras semânticas*, permitem a conexão entre o nível I e o II, sendo este o modelo que representa o estado do mundo. Os referidos algoritmos permitem, também, as deduções lógicas e, como conseqüência e dentro da perspectiva desta teoria, a completa interconexão entre semântica e lógica.

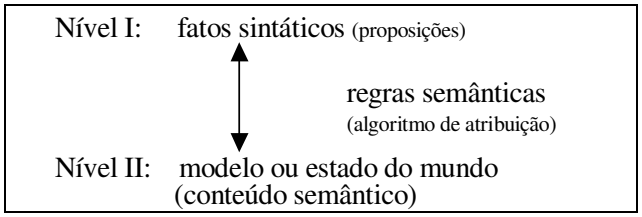


Figura 5.28: Interpretação semântica adotada.

No contexto ainda da semântica da teoria do modelo e considerando a definição formal de uma linguagem interpretada apresentada por Tarski (ver seção 2.2.2.3.3) , esta teoria semântica tem como linguagem objeto, i.e., a linguagem a ser interpretada semanticamente, e metalinguagem, i.e., a linguagem que formula as definições semânticas

(HAUSSER,2001), a *cláusula definida de Horn* em notação Prolog padrão, pois, embora os termos *linguagem objeto* e *metalinguagem* refiram-se a linguagens com diferentes funcionalidades, podem ser implementadas por uma única linguagem (GAMUT, 1991). A figura 5.29 exemplifica ambas as linguagens, cabendo destacar que as proposições em linguagem objeto são denominadas de *fatos sintáticos* e as definições semânticas formuladas por meio da metalinguagem são denominadas de *regras semânticas*.

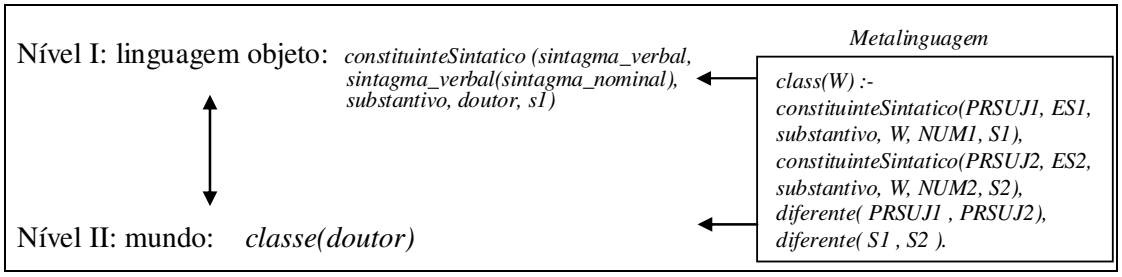


Figura 5.29: Exemplos de linguagem objeto e metalinguagem.

5.6.2.1 Modelo de Discurso baseado em UML

A presente técnica lingüística aplica a semântica lógica orientada a discurso, i.e., sobre uma seqüência coerente de sentenças em vez de uma frase isolada (KAMP e REYLE, 1993). A unidade semântica primária é o vocábulo cujo significado depende basicamente de seus constituintes sintáticos, i.e., categoria gramatical e estrutura sintática. Os referidos constituintes contribuem com a vinculação lógica da *base de conhecimento semântica* que determina os referentes do discurso, i.e., os elementos orientados a objeto que permitem a modelagem conceitual.

A distinção entre sintaxe e semântica, na tradição lógica, está proximamente vinculada à distinção entre os sistemas formais e suas interpretações. A Teoria do Modelo, i.e., o estudo das interpretações dos sistemas formais, enfoca as relações entre teorias e modelos. Um conjunto de axiomas juntos com todos os teoremas deriváveis dos mesmos é chamado de uma *teoria*, i.e., um conjunto de declarações vinculadas por meio de conseqüências lógicas. Um modelo para uma teoria requer um domínio abstrato e estruturado e a interpretação para todas as expressões primitivas da teoria naquele domínio, de modo que para a referida interpretação todas as declarações na teoria tornam-se verdadeiras para aquele modelo naquela interpretação (PARTEE *et al.*, 1990). Com base nos referidos conceitos e utilizando a estrutura de representação do conhecimento adotada, i.e., a rede semântica, o modelo adotado, que permite a geração dos axiomas que irão compor a *base de conhecimento semântica*, está estruturado de acordo com as definições descritas a seguir.

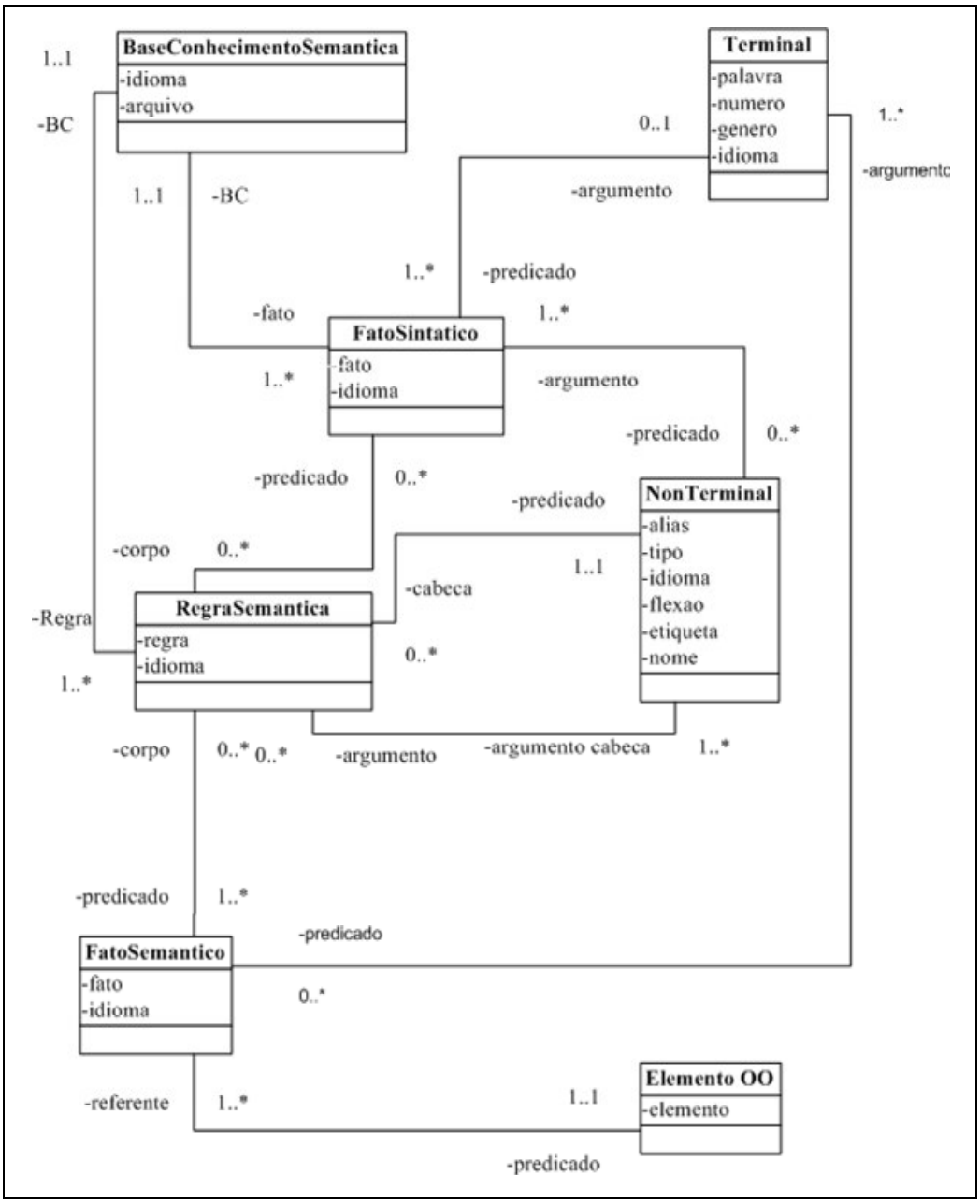


Figura 5.30: Rede semântica do *MDU*.

Definição 1: Um *Modelo de Discurso baseado em UML (MDU)* para uma linguagem natural *L* é composto pela estrutura *‹D, cFaSi, cFaSe, cReSe, cTeSe›*, onde (SILVA e CARVALHO, 2006b):

i) *D* é o domínio do *MDU*, i.e., todos os vocábulos, categorias gramaticais e estruturas sintáticas de uma determinada linguagem natural *L*, e.g., Português;

ii) *cFaSi* é um conjunto de axiomas sintáticos denominados de *fatos sintáticos (FaSi)*;

iii) *cFaSe* é um conjunto de axiomas semânticos denominados de *fatos semânticos (FaSe)*;

iv) *cReSe* é um conjunto de axiomas semânticos denominados de *regras semânticas (ReSe)*;

v) *cTeSe* é um conjunto de *teoremas semânticos (TeSe)*;

vi) *FaSi*, *FaSe, ReSe e TeSe* são gerados a partir da estrutura de representação do conhecimento ilustrada na figura 5.30.

Definição 2: O axioma *FaSi* é um predicado com seis argumentos com o seguinte subconjunto do vocabulário *V* em sintaxe BNF:

<fatoSintatico> ::= <predicado> "."

<predicado> ::= constituinteSintatico(<listaTermosSintaticos>)

<listaTermosSintaticos> ::= <atomoEstruturaSintatica>,
<atomoCategoriaGramatical>, <atomoLexico>, <atomoSentenca>,
<atomoNumero>

< atomoEstruturaSintatica > ::= sintagma_nominal | … | sintagma_verbal

< atomoCategoriaGramatical >::= substantivo | adjetivo | … | verbo

< atomoLexico> ::= < atomoPequeno> | '<string>'

< atomoPequeno> ::= <letraMinuscula> | <atomoPequeno> <letraMinuscula>

<string> ::= <caracter> | <string><caracter>

< atomoSentenca>::= s <numeral>

<numeral> ::= <digito> | <numeral><digito>

<atomoNumero> ::= <singular> | <plural>

Definição 3: O axioma *ReSe* implementa a heurística que permite extrair os referentes do discurso, i.e., os elementos OO, sendo um predicado n-ário com o seguinte subconjunto do vocabulário *V* em sintaxe BNF:

<regraSemantica> ::= <cabecaPredicadoSemantico> :-
<listaCorpoPredicadoSemantico>"."

< cabecaPredicadoSemantico> ::= <atomoSemanticoCabeca>
(<listaTermoSemantico>)

< listaCorpoPredicadoSemantico> ::= <corpoPredicadoSemantico> |
<listaCorpoPredicadoSemantico>, <corpoPredicadoSemantico>

< corpoPredicadoSemantico::= < atomoSemanticoCorpo >
(<listaTermoSemantico>) | <atomoSemanticoCorpo>
(<listaTermoSemantico>) | <atomoEstruturaSintatica>|
<atomoCategoriaGramatical>

< atomoSemanticoCabeca > ::= class | association | atributte | cardinality | inheritance

< atomoSemanticoCorpo> ::= constituinteSintatico

<listaTermoSemantico> ::= <variavelSemantica> | <listaTermoSemantico>, <variavelSemantica > | <atomo> | <listaTermoSemantico>, <atomo>

< variavelSemantica> ::= < letraMaiusculaSemantica> [<numeral>]

<letraMaiusculaSemantica> ::= C | R | A | M | WC | SS | W | S

<atomo> ::= <pequenoAtomo> | '<string>'

<pequenoAtomo> ::= <letraMinuscula> | <pequenoAtomo> < letraMinuscula >

<string> :: <caracter> | <string>< caracter>

<numeral> ::= <digito> | <numeral><digito>

**** C – classe, R – associação, A – atributo, M – multiplicidade,*

**** WC – categoria gramatical, SS – estrutura sintática, W – palavra, S – sentença.*

Definição 4: O *FaSe* é um axioma para um elemento orientado a objeto representado por um predicado n-ário gerado por uma regra semântica com o seguinte subconjunto do vocabulário *V* em sintaxe BNF:

<fatoSemantico> ::= <predicado> "."

<predicado> ::= <atomoSemantico>(<listaTermoSemantico>)

<atomoSemantico> ::= classe | associacao | atributo | multiplicidade |

<listaTermoSemantico> ::= <atomoLexico> | <listaTermoSemantico>, <atomoLexico>

<atomoLexico> ::= <pequenoAtomo> | '<string>'

<pequenoAtomo> ::= < letraMinuscula > | <pequenoAtomo> <letraMinuscula>

<string> :: <caracter> | <string><caracter>

Definição 5: O *TeSe* é um teorema semântico, i.e., teorema presumido a ser provado a partir dos axiomas FaSi, FaSe e ReSe com o seguinte subconjunto do vocabulário *V* em sintaxe BNF:

<fatoSemantico> ::= <predicado> "."

<predicado> ::= <atomoSemantico>(<listaTermoSemantico>)

< atomoSemantico > ::= class | association | attribute | cardinality | inheritance

< listaTermoSemantico > ::= <variavelSemantica> | < listaTermoSemantico>, < variavelSemantica>

< variavelSemantica > ::= X | Y | Z | ..

A representação de conhecimento *MDU* permite a geração de uma *base de conhecimento semântica* para o assistente inteligente proposto. Brachman e Levesque (2004) destacam que o significado é tipicamente capturado por interpretações específicas, i.e., uma interpretação mapeada sobre um domínio, e apresentam o seguinte questionamento: *como seria possível dar uma interpretação a um sistema que poderia envolver (talvez infinitos) conjuntos de excelentes objetos tais como países e animais?* A resposta inclui a conexão entre sistemas baseados em conhecimento, tal qual o assistente proposto, e o vínculo lógico. Considere *S* um conjunto de sentenças e *α* uma sentença qualquer: se *S = {α1,..., αn}*, então $S \models \alpha$ se e somente se a sentença [{α1 ^…^ αn} ^ α] é válida, i.e., S vincula a sentença *α*. As sentenças não vinculadas podem ou não ser verdadeiras, mas o sistema baseado em conhecimento pode seguramente concluir quais vínculos existem. Baseado no conceito de vínculo lógico, onde *FaSi* e *FaSe* são fatos do discurso, as regras *ReSe* permitem expressar a conexão entre símbolos não lógicos envolvidos, cabendo aos teoremas *TeSe* a definição do conjunto de sentenças vinculadas próximas ao conjunto de sentenças verdadeiras em uma determinada interpretação. O cálculo do referido vínculo é uma forma de raciocínio, por meio de inferência dedutiva, que permite a extração do significado dos termos envolvidos. Baseado na referência (SOWA, 2000), o MDU é um modelo computacional com um método declarativo, i.e., baseado em axiomas, que utiliza técnicas de prova de teoremas na derivação de suas conseqüências lógicas.

5.6.2.2 Predicação Sintática

A figura 5.31 ilustra a atividade *predicação sintática* gerada pelo *Analisador Semântico* durante a análise semântica. Esta atividade permite a geração dos axiomas fatos sintáticos de acordo com a sintaxe apresentada na definição 2 do *MDU*, i.e., um predicado com seis argumentos. Cada palavra do discurso analisado tem um predicado *constituinteSintatico* que, de acordo com a interpretação semântica adotada (ver seção 5.6.2), corresponde ao nível intermediário *senso, ou modo de apresentação,* que permite extrair os significados, ou *referentes*, desejados do discurso.

A referida predicação, com dois exemplos ilustrados na figura 5.32, ocorre a partir da estrutura sintática plana gerada da análise sintática. O primeiro argumento do predicado sintático identifica a posição *sujeito* ou *predicado* na sentença, e.g., médico está no *sintagma_nominal*, que indica o sujeito, e paciente, no *sintagma_verbal*, que indica o predicado. O segundo argumento identifica a posição sintática do vocábulo na frase, e.g., o

médico está no *sintagma_nominal* e paciente no *sintagma_verbal(sintagma_nominal)*. O terceiro argumento identifica a categoria gramatical, o quarto, o vocábulo analisado, o quinto, a flexão em número, e o sexto, a identificação da sentença no discurso considerado.

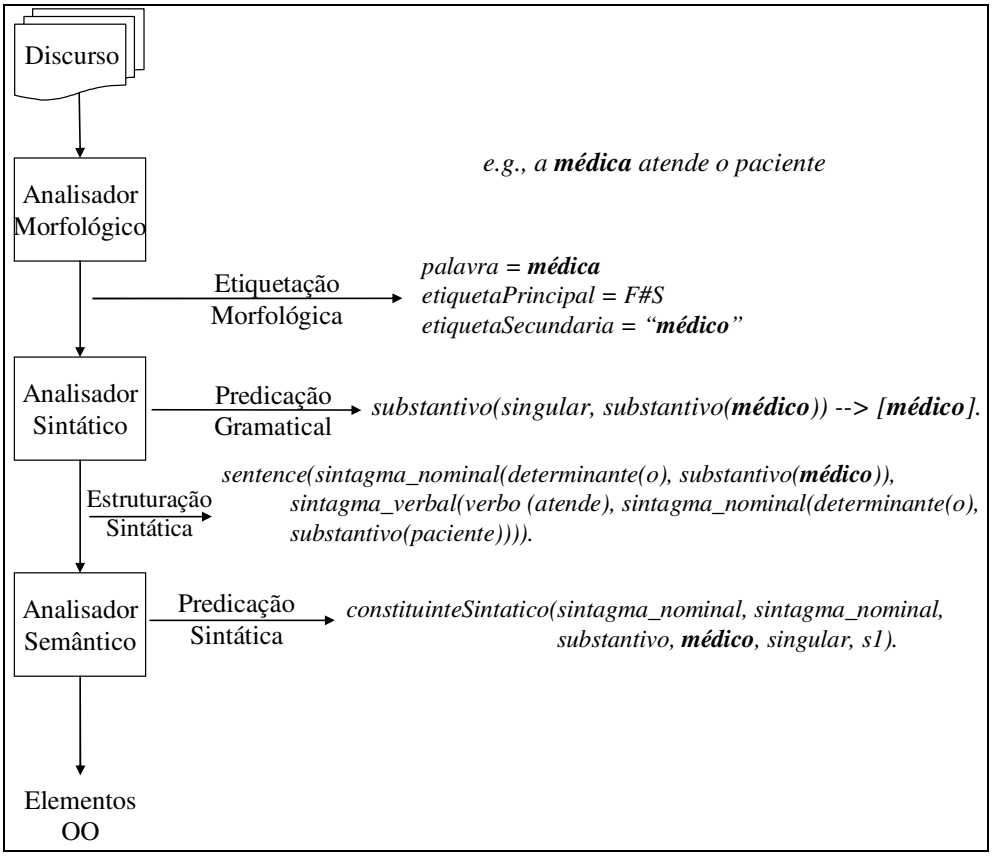


Figura 5.31: Exemplo de predicação sintática.

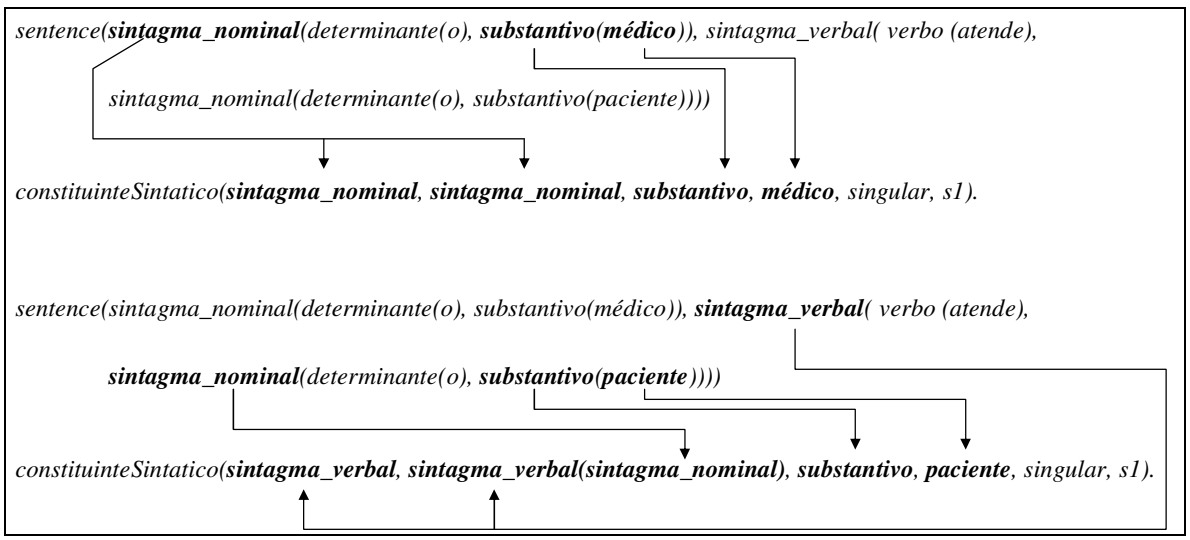


Figura 5.32: Exemplos de geração de predicados sintáticos.

Os métodos *GerarFatosSintaticos, FormataFatosSintaticos e InicializarBC* do *Componente Analisador Semântico*, descritos na figura 5.33, implementam as funcionalidades que permitem a extração dos dados da estrutura sintática plana, a

formatação e a inserção dos referidos *fatos sintáticos*, juntamente com as regras semânticas, na *base de conhecimento semântica*, respectivamente.

```
public __gc class CAnalisadorSemantico
    {
     private:
                String * discurso;
                String * idioma;
                String * file;
                FILE * stream;
                PlEngine * engine;…
     public:
                CAnalisadorSemantico(void);
                CAnalisadorSemantico(String * discourse, String * idiom);
                ArrayList * ProcessarDiscurso(ArrayList * arrConstituinte);
    private:
                bool InicializarBC(void);
                ArrayList * GerarFatosSintaticos(ArrayList * arrConstituinte);
                ArrayList * ObterConstituentesSintaticos(String * strSource, int nrSentenca);
                ArrayList * FormataFatosSintaticos(ArrayList *lista, String * sentenca, String * sujPred );
                bool CriarBCProlog(void);
                bool InserirBCProlog(ArrayList * arrProducao);
                ArrayList * ExtrairClasses(void);
                ArrayList * ExtrairAtributos(void);
                ArrayList * ExtrairRelacoes(void);
                ArrayList * ExtrairCardinalidades(void);
                char  * StrChr(String * strAux);
                ArrayList * FormatarFatoOO(ArrayList * arrFato);
                String * BuscarPalavraSingular(int cod);
                String * InverterPalavrasRelacao(String * str);
                ArrayList * VerificarRelacaoRepetida(ArrayList * arr);
                String * InverterPalavrasMultiplicidade(String * str);
                String * RetirarMultiplicidade(String * str);
                bool CompararRelacao(String * regra1, String * regra2);
                ArrayList * DefinirCardinalidade(ArrayList * arrInicial);
                ArrayList * InverterNomeMultiplicidade(ArrayList * arrInicial);
                String * BuscarCardinalidadeMinima(String * str);
                String * BuscarCardinalidadeMaxima(String * str);
                String * SelecionarCardinalidade(String * regra1, String * regra2);
                ArrayList * DefinirMultiplicidadeFinal(ArrayList * arrInicial);
                ArrayList * VerificarMultiplicidadeBinariaRepetida(ArrayList * arr);
                String * IsolarNomeAssociacao(String * str);
                String * FormatarNovaAssociacao(String * str);
                ArrayList * RedefinirAssociacao(ArrayList * arr);
                ArrayList * AjustarArrayFatosSintaticos(ArrayList *lista );
};
```

Figura 5.33: Interface do componente *Analisador Semântico*.

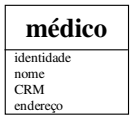
5.6.2.3 Regras Semânticas

As regras semânticas (*ReSe*) implementam as heurísticas que permitem extrair os referentes do discurso, i.e., os elementos orientados a objeto, cuja sintaxe está descrita na definição 3 do *MDU* (ver seção 5.6.2.1). Cabe destacar que a rede semântica do domínio semântico permite a representação das regras especificadas pelo engenheiro de conhecimento.

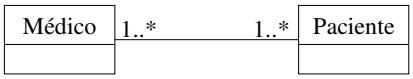
A seguir estão descritas as regras que permitem extrair os elementos orientados a objeto, requisitos para o modelo conceitual a ser gerado posteriormente.

Regra Semântica 1	
Elemento OO	Classe.
Heurística	***Se*** (*W* é substantivo) e (*W* está em sentenças distintas) e (*W* está em posições sintáticas distintas) ***então*** (*W* é uma classe).
Regra	*class(W) :-* *constituinteSintatico(PRSUJ1, ES1, substantivo, W, NUM1, S1),* *constituinteSintatico(PRSUJ2, ES2, substantivo, W, NUM2, S2),* *diferente(PRSUJ1 , PRSUJ2), diferente(S1 , S2).*
Exemplo	o ***médico*** atende os pacientes. o hospital contrata os ***médicos.*** **médico**

Regra Semântica 2	
Elemento OO	Classe.
Heurística	***Se*** (*W* é substantivo) e (*W* está no sujeito) e (*W* e *W1* estão em uma mesma sentença) e (*W1* é uma classe) e (*W1* está no predicado) ***então*** (*W* é uma classe).
Regra	*class(W) :-* *constituinteSintatico(sintagma_nominal, ES1, substantivo , W , NUM1, S1),* *constituinteSintatico(PRSUJ2, ES2 , substantivo , W1 , NUM2, S1),* *diferente(sintagma_nominal , PRSUJ2),diferente(W, W1), classe(W1).*
Exemplo	o *médico* atende os pacientes. o *paciente* é atendido pelo *médico.* o ***hospital*** contrata os *médicos.* **hospital**

Regra Semântica 3	
Elemento OO	Atributo.
Heurística	Se (*X* é classe) e (*W* é substantivo) e (*W* está no predicado) e (*W* não é uma classe) e (*W* não está no sujeito) e (*X* e *W* estão na mesma sentença) então (*W* é um atributo da classe *X*).
Regra	*attribute(X, W) :-* *constituinteSintatico(PRSUJ1, ES1 , substantivo , X , NUM1,S1),* *constituinteSintatico(PRSUJ2, ES2 , substantivo , W , NUM2,S1),* *not(classe(W)), classe(X), diferente(W , X), diferente(ES1, ES2),* *diferente(PRSUJ2 , sintagma_nominal).*
Exemplo	o *médico* atende os pacientes. o hospital contrata os *médicos.* o *médico* tem uma ***identidade***, um ***nome***, um ***CRM*** e um ***endereço.*** 

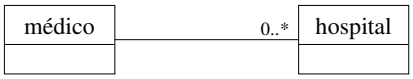
Regra Semântica 4	
Elemento OO	Associação.
Heurística	Se (*W* é classe) e (*X* é classe) e (*W* e *X* estão na mesma sentença) e (*W* e *X* estão em estruturas sintáticas diferentes) então (*W* tem associação com X).
Regra	*association(W,X) :-* *constituinteSintatico(PRSUJ1, ES1, substantivo, W, NUM1, S1),* *constituinteSintatico(PRSUJ2, ES2 , substantivo , X , NUM2, S1),* *classe(X), diferente(ES1,ES2), classe(W).*
Exemplo	o *hospital* contrata os *médicos.* 

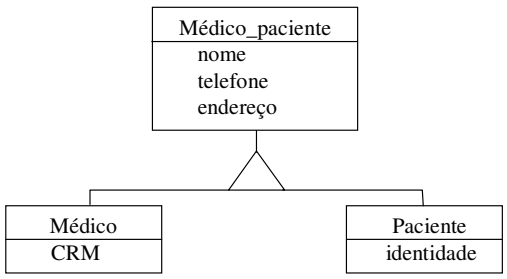
Regra Semântica 5	
Elemento OO	Cardinalidade.
Heurística	**Se** (*R* relaciona *W e X*) e (*W* e *X* estão na mesma sentença) **então** (multiplicidade mínima de *X* é *1*) e (multiplicidade máxima de *X* é a flexão em número de *X* na sentença).
Regra	*cardinality(R,W,X, 1 ,NUM2) :-* *constituinteSintatico(PRSUJ1, ES1 , substantivo , W , NUM1, S1),* *constituinteSintatico(PRSUJ2, ES2 , substantivo , X , NUM2, S1),* *associacao(R,W,X), diferente(ES1,ES2),diferente(PRSUJ1,PRSUJ2),* *classe(X), classe(W).*
Exemplo	o *médico* atende os *pacientes.* o *paciente* é atendido pelos *médicos.* 

Regra Semântica 6	
Elemento OO	Cardinalidade.
Heurística	**Se** (*R* relaciona *W e X*) e (*W* e *X* estão na mesma sentença) e (existe um *numeral* e o advérbio *mais* no predicado) **então** (multiplicidade mínima de *X* é o *numeral*) e (multiplicidade máxima de *X* é *muitos*)* ** representado pelo valor zero*
Regra	*cardinality(R, W, X, MIN, 0) :-* *constituinteSintatico(sintagma_nominal, ES1, substantivo, W, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES2 , substantivo , X , NUM2, S1),* *constituinteSintatico(sintagma_verbal, ES3 , numeral , MIN, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES4 , adverbio , mais, NUM1, S1),* *associacao(R,W,X), diferente(ES1,ES2), classe(W), classe(X).*
Exemplo	*o professsor leciona duas ou mais disciplinas* 

Regra Semântica 7	
Elemento OO	Cardinalidade.
Heurística	**Se** (*R* relaciona *W e X*) e (*W* e *X* estão na mesma sentença) e (existe um numeral e o advérbio *vários* no predicado) **então** (multiplicidade mínima de *X* é o *numeral*) e (multiplicidade máxima de *X* é *muitos*)* ** representado pelo valor zero*
Regra	*cardinality(R, W, X, MIN, 0) :-* *constituinteSintatico(sintagma_nominal, ES1, substantivo, W, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES2, numeral, MIN, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES3, substantivo, X , NUM2, S1),* *constituinteSintatico(sintagma_verbal, ES4, pronome, vários, NUM2, S1),* *associacao(R,W,X), diferente(ES1,ES2), classe(W), classe(X).*
Exemplo	*o médico atende um ou vários pacientes.* 

Regra Semântica 8	
Elemento OO	Cardinalidade.
Heurística	**Se** (*R* relaciona *W e X*) e (*W* e *X* estão na mesma sentença) e (existe o advérbio *no máximo* seguido de um *numeral* no predicado) **então** (multiplicidade mínima de *X* é *zero*) e (multiplicidade máxima de *X* é o *numeral*). ** representado pelo valor zero*
Regra	*cardinality(R, W, X, 0, MAX) :-* *constituinteSintatico(sintagma_nominal, ES1, substantivo, W, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES2, adverbio, no_máximo, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES3, numeral, MAX, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES4 , substantivo, X , NUM2, S1),* *associacao(R,W,X), diferente(ES1,ES2), classe(W), classe(X).*
Exemplo	*o professor leciona no máximo cinco disciplinas.* 

Regra Semântica 9	
Elemento OO	Cardinalidade.
Heurística	**Se** (*R* relaciona *W e X*) e (*W* e *X* estão na mesma sentença) e (existe a preposição *em* seguida do pronome *vários* no predicado) **então** (multiplicidade mínima de *X* é *zero*) e (multiplicidade máxima de *X* é o *numeral*). ** representado pelo valor zero*
Regra	*cardinality(R, W, X, 0, 0) :-* *constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES2, preposicao, em , NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES3, pronome, vário , NUM2, S1),* *constituinteSintatico(sintagma_verbal, ES4 , substantivo , X , NUM2, S1),* *associacao(R,W,X), diferente(ES1,ES3), classe(W), classe(X).*
Exemplo	um *médico* trabalha em vários *hospitais*. 

Regra Semântica 10	
Elemento OO	Herança.
Heurística	**Se** (*X* é classe) e (*Y* é classe) e (*W* é atributo de *X*) e (*W* é atributo de *Y*) e (*X* e *Y* estão em sentenças distintas) e (*W* está no predicado) **então** (*W* é atributo comum de *X* e *Y*).
Regra	*inheritance(X, Y, W) :-* *constituinteSintatico(PRSUJ1, ES1 , substantivo , X , NUM1,S1),* *constituinteSintatico(PRSUJ2, ES2 , substantivo , Y , NUM1,S2),* *constituinteSintatico(sintagma_verbal, ES3 , substantivo , W , NUM2,S1),* *constituinteSintatico(sintagma_verbal, ES4 , substantivo , W , NUM2,S2),* *atributo(X, W), atributo(Y, W), classe(X), classe(Y), diferente(X, Y).*
Exemplo	*o médico tem um CRM, um nome, um telefone e um endereço.* *o paciente tem uma identidade, um nome, um telefone e um endereço.*  *obs: número de atributos comuns >= 3.*

5.6.2.4 Predicação Semântica

A predicação semântica, ilustrada na figura 5.34, permite gerar os fatos semânticos que, por sua vez, representam os significados do discurso, i.e., os elementos orientados a objeto, foco desta teoria semântica. A linguagem objeto dos fatos citados é, também, a cláusula definida de Horn em notação Prolog e a sintaxe é a estabelecida na definição 4 do *MDU*.

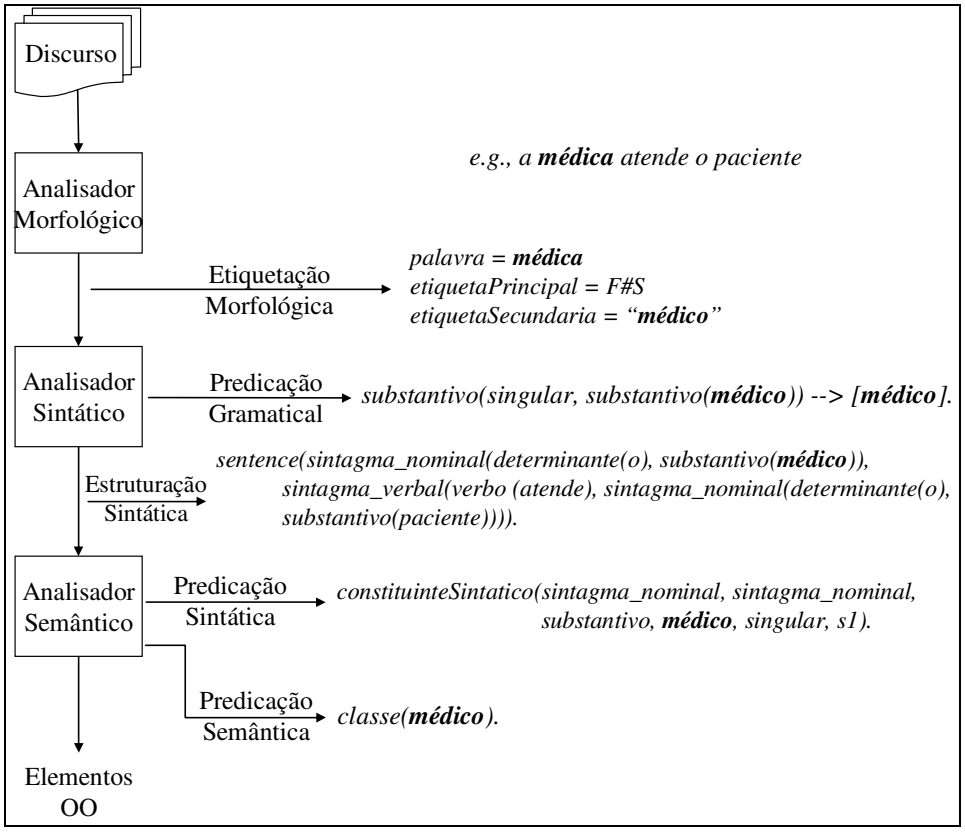


Figura 5.34: Exemplo de predicação semântica.

Com exceção da *regra semântica 01*, todas as demais incluem um predicado do tipo *fato semântico*. A justificativa para as inclusões citadas está na dependência existente entre os elementos orientados a objeto, e.g., uma *associação* depende de serem ambos os elementos relacionados do tipo *classe*.

A figura 5.35 ilustra as dependências por meio de um diagrama de atividades simplificado para o método *ProcessarDiscurso* do *Componente Analisador Semântico* do *ASIEEOOD.* Uma *base de conhecimento semântica* é instanciada em tempo de execução, tal qual ocorre com a *base de conhecimento sintática*, ao final de cada atividade ilustrada na figura 5.35, e.g., após a extração das classes a base citada é novamente instanciada com os predicados das respectivas classes, i.e., os *fatos semânticos*. A figura 5.36 exemplifica uma *base de conhecimento semântica* gerada em tempo de execução pelo *ASIEEOOD* cuja

linguagem objeto, dos fatos sintáticos e semânticos, e metalinguagem, das regras semânticas, são a mesma, i.e., cláusulas definidas de Horn em notação Prolog padrão.

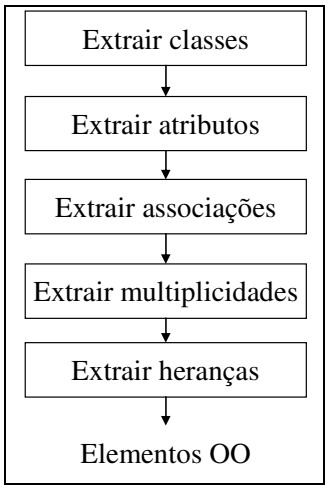


Figura 5.35: Diagrama de atividades do processamento do discurso.

```
%Base de Conhecimento Semantica :
diferente(X , Y) :- (X  =  Y), !, fail; true.
classe(xxxx).
class(W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1,S1 ),
            constituinteSintatico( PRSUJ2, ES2 , substantivo , W , NUM2, S2 ),
            diferente( PRSUJ1 , PRSUJ2), diferente( S1 , S2 ).
attribute(X, W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , X , NUM1,S1 ),
                   constituinteSintatico( PRSUJ2, ES2 , substantivo , W , NUM2,S1 ),
                   not(classe(W)), classe(X), diferente(W ,  X), diferente(ES1, ES2),
                   diferente(PRSUJ2 ,  sintagma_nominal).
association(W,X) :- constituinteSintatico(PRSUJ1, ES1 , substantivo , W , NUM1, S1 ),
                    constituinteSintatico(PRSUJ2, ES2 , substantivo , X , NUM2, S1 ),
                    classe(X),  diferente(ES1,ES2),  classe(W).

cardinality(R,W,NUM1,X,NUM2) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1, S1),
                                constituinteSintatico( PRSUJ2, ES2 , substantivo , X , NUM2, S1 ),
                                associacao(R,W,X),  diferente(ES1,ES2).


class(W) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1 ),
            constituinteSintatico( PRSUJ2, ES2 , substantivo , W1 , NUM2, S1 ),
            diferente(sintagma_nominal , PRSUJ2),diferente( W, W1 ), classe(W1).

constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, o, singular, s1).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, médico, singular, s1).
constituinteSintatico( sintagma_verbal, sintagma_verbal,verbo, atende, singular, s1).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, o, plural, s1).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, paciente, plural, s1).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, o, singular, s2).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, hospital, singular, s2).
constituinteSintatico( sintagma_verbal, sintagma_verbal,verbo, contrata, singular, s2).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, o, plural, s2).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, médico, plural, s2).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, o, singular, s3).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, paciente, singular, s3).
constituinteSintatico( sintagma_verbal, sintagma_verbal,verbo_aux, é, singular, s3).
constituinteSintatico( sintagma_verbal, sintagma_verbal,verbo, atendido, singular, s3).
```

Figura 5.36: Exemplo de *base de conhecimento semântica*.

5.6.3 Estudo de Caso de Análise Semântica

A figura 5.37 descreve o processo e respectivas atividades do método lingüístico proposto. O objetivo do estudo de caso é, a partir de um pequeno discurso, apresentar o resultado de cada atividade até a extração dos elementos orientados a objeto, foco da referida técnica.

Discurso em linguagem natural:

- "o médico atende um ou mais pacientes.";
- "o hospital contrata os médicos.";
- "o paciente é atendido pelos médicos.";
- "o médico tem um CRM, um nome, um telefone e um endereço.";
- "o paciente tem uma identidade, um nome, um telefone e um endereço.";
- "o hospital contrata os enfermeiros.";
- "o enfermeiro tem um CRE, um nome e um endereço.";
- "o hospital possui um CNPJ e um endereço.";
- "um médico trabalha em vários hospitais.".

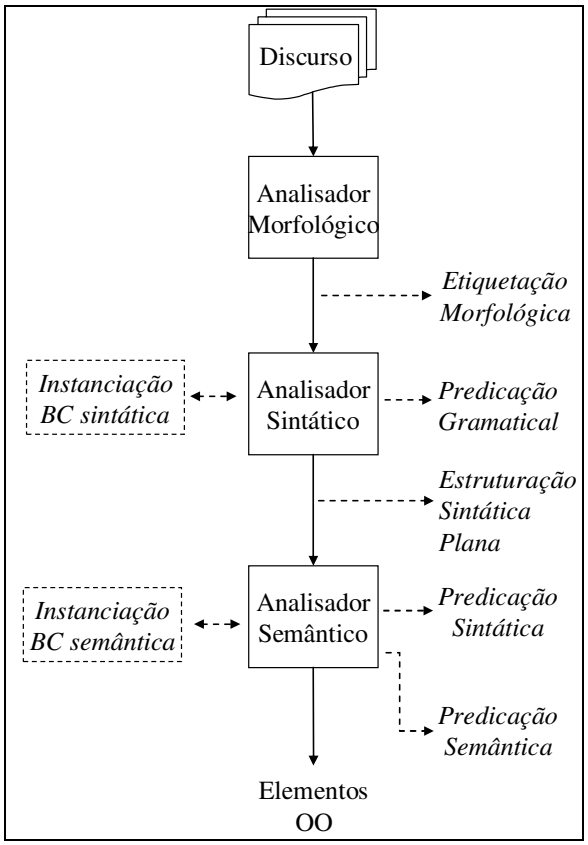


Figura 5.37: Processo lingüístico.

Atividade Etiquetação Morfológica

O Analisador Morfológico (ver seção 5.4.2) realiza a etiquetação morfológica para cada vocábulo do discurso, tal como exemplifica a etiquetação da frase *o médico atende um ou mais pacientes* ilustrada na figura 5.38.

Atividade Predicação Gramatical

A Predicação Gramatical (ver seção 5.5.1) permite a formatação, de acordo com a sintaxe estabelecida pela definição 3 do modelo *MESU,* dos vocábulos que compõem o

léxico armazenado na base de dados em predicados denominados *fatos sintáticos* que comporão a *base de conhecimento sintática*. A figura 5.39 ilustra uma vista parcial dos fatos sintáticos gerados.

```
Palavra          "o"
EtiquetaPrincipal          "#m#s#"
EtiquetaSecundaria         "#"

Palavra          "médico"
EtiquetaPrincipal          "#m#s#"
EtiquetaSecundaria         "#"

Palavra          "atende"
EtiquetaPrincipal          "#u#s#"
EtiquetaSecundaria         "#"

Palavra          "um"
EtiquetaPrincipal          "#m#s#"
EtiquetaSecundaria         "#"

Palavra          "ou"
EtiquetaPrincipal          "#u#i#"
EtiquetaSecundaria         "#‘

Palavra          "mais"
EtiquetaPrincipal          "#u#i#"
EtiquetaSecundaria         "#"

Palavra          "pacientes"
EtiquetaPrincipal          "#u#p#"
EtiquetaSecundaria         "paciente"
```

Figura 5.38: Etiquetação morfológica.

```
adjetivo(singular, adjetivo(magro)) --> [magro].
adverbio(singular, adverbio(mais)) --> [mais].
adverbio(singular, adverbio(no_máximo)) --> [no_máximo].
conjuncao(conjuncao(e)) --> [e].
conjuncao(conjuncao(ou)) --> [ou].
determinante(singular, determinante(a)) --> [a].
determinante(singular, determinante(o)) --> [o].
determinante(singular, determinante(um)) --> [um].
numeral(numeral(um)) --> [um].
preposicao(preposicao(em)) --> [em].
preposicao(preposicao(por)) --> [por].
substantivo(plural, substantivo(enfermeiros)) --> [enfermeiros].
substantivo(plural, substantivo(hospitais)) --> [hospitais].
substantivo(plural, substantivo(médicos)) --> [médicos].
substantivo(plural, substantivo(pacientes)) --> [pacientes].
substantivo(singular, substantivo(código)) --> [código].
substantivo(singular, substantivo(cor)) --> [cor].
substantivo(singular, substantivo(crm)) --> [crm].
substantivo(singular, substantivo(enfermeiro)) --> [enfermeiro].
substantivo(singular, substantivo(hospital)) --> [hospital].
substantivo(singular, substantivo(identidade)) --> [identidade].
substantivo(singular, substantivo(médico)) --> [médico].
verbo(plural, verbo(atendem)) --> [atendem].
verbo(singular, verbo(atende)) --> [atende].
verbo(singular, verbo(tem)) --> [tem].
verbo(singular, verbo(trabalha)) --> [trabalha].
verbo_aux(singular, verbo_aux(é)) --> [é].
...
```

Figura 5.39: Exemplos de fatos sintáticos.

Atividade Instanciação de Base de Conhecimento Sintática

A instanciação de uma *base de conhecimento sintática* corresponde à geração de um modelo *MESU* em tempo de execução pelo *Componente Analisador Sintático* do *ASIEEOOD,* i.e., para cada estrutura sintagmática de frase gerada pelo engenheiro de conhecimento, as regras sintáticas são geradas a partir da estrutura em árvore apresentada na seção 5.5.2. A figura 5.40 exemplifica uma *base de conhecimento sintática* instanciada a partir das regras e fatos sintáticos.

```
1  sentence(Number,sentence(SN,SV)) --> sintagma_nominal(Number,SN),sintagma_verbal(Number,SV),!.
2  sintagma_verbal(Number,sintagma_verbal(Verbo,SN)) --> verbo(Number,Verbo),sintagma_nominal(Number,SN).
3  sintagma_nominal(Number,sintagma_nominal(Det,Subst)) --> determinante(Number,Det),substantivo(Number,Subst).
4  sintagma_adverbial(Number,sintagma_adverbial(Num,Conj,Adv)) --> numeral(Num),conjuncao(Conj),adverbio(Number,Adv).
5  sintagma_nominal(Number,sintagma_nominal(SAdv,Subst)) --> sintagma_adverbial(Number,SAdv),substantivo(Number1,Subst).
6
7  adjetivo(plural, adjetivo(a_serem_cursadas)) --> [a_serem_cursadas].
8  adjetivo(plural, adjetivo(concluídas)) --> [concluídas].
9  adjetivo(plural, adjetivo(cursadas)) --> [cursadas].
10 adjetivo(plural, adjetivo(matriculados)) --> [matriculados].
11 adjetivo(plural, adjetivo(obrigatórias)) --> [obrigatórias].
12 adjetivo(plural, adjetivo(oferecidas)) --> [oferecidas].
13 adjetivo(plural, adjetivo(relativos)) --> [relativos].
14 adjetivo(singular, adjetivo(alto)) --> [alto].
15 adjetivo(singular, adjetivo(amarelo)) --> [amarelo].
16 adjetivo(singular, adjetivo(determinada)) --> [determinada].
17 adjetivo(singular, adjetivo(magro)) --> [magro].
18 adjetivo(singular, adjetivo(relativa)) --> [relativa].
19 adjetivo(singular, adjetivo(relativo)) --> [relativo].
20 adjetivo(singular, adjetivo(substituição)) --> [substituição].
21 adjetivo(singular, adjetivo(único)) --> [único].
22 adjetivo(singular, adjetivo(verde)) --> [verde].
23 adverbio(singular, adverbio(mais)) --> [mais].
24 adverbio(singular, adverbio(no_máximo)) --> [no_máximo].
25 conjuncao(conjuncao(e)) --> [e].
26 conjuncao(conjuncao(ou)) --> [ou].
27 determinante(plural, determinante(as)) --> [as].
28 determinante(plural, determinante(os)) --> [os].
29 determinante(singular, determinante(a)) --> [a].
30 determinante(singular, determinante(cada)) --> [cada].
31 determinante(singular, determinante(o)) --> [o].
32 determinante(singular, determinante(um)) --> [um].
33 determinante(singular, determinante(uma)) --> [uma].
34 numeral(numeral(cinco)) --> [cinco].
35 numeral(numeral(dois)) --> [dois].
36 numeral(numeral(um)) --> [um].
```

Figura 5.40: Exemplo de uma instância de *base de conhecimento sintática*.

Atividade Estruturação Sintática Plana

A sentença *médico atende um ou mais pacientes* terá uma interpretação verdadeira ao ser instanciado o modelo de *base de conhecimento sintática* ilustrado na figura 5.40 e sob o questionamento do teorema descrito na figura 5.41. A interpretação verdadeira para o referido teorema, cuja sintaxe é determinada pela definição 4 do modelo *MESU*, tem como resultado a estrutura sintática plana em negrito, mostrada, juntamente com o resultado do processamento das demais sentenças do discurso, na figura 5.42.

sentence(X,ParseTree,[o, médico, atende, um, ou, mais, pacientes],[]).

Figura 5.41: Exemplo de teorema a ser provado.

*"**sentence(sintagma_nominal(determinante(o), substantivo(médico)), sintagma_verbal(verbo(atende), sintagma_nominal(sintagma_adverbial(numeral(um), conjunção (ou), adverbio(mais)), substantivo(pacientes))))**";*

"sentence(sintagma_nominal(determinante(o), substantivo(hospital)), sintagma_ verbal(verbo(contrata), sintagma_nominal(determinante(os), substantivo(médicos))))";

"sentence(sintagma_nominal(determinante(o), substantivo(paciente)), sintagma_ verbal(verbo_aux(é), verbo(atendido), sintagma_nominal(preposicao_(pelos), substantivo (médicos))))";

"sentence(sintagma_nominal(determinante(o), substantivo(médico)), sintagma_ verbal(verbo(tem), sintagma_nominal(determinante(um), substantivo(crm), sintagma_ nominal(determinante(um), substantivo(nome), sintagma_nominal(determinante(um), substantivo(telefone), sintagma_nominal(conjuncao(e), determinante(um), substantivo (endereço)))))))" ;

"sentence(sintagma_nominal(determinante(o), substantivo(paciente)), sintagma_ verbal(verbo(tem), sintagma_nominal(determinante(uma), substantivo(identidade), sintagma_nominal(determinante(um), substantivo(nome), sintagma_nominal (determinante(um), substantivo(telefone), sintagma_nominal(conjuncao(e), determinante (um), substantivo(endereço)))))))" ;

"sentence(sintagma_nominal(determinante(o), substantivo(hospital)), sintagma_ verbal(verbo(contrata), sintagma_nominal(determinante(os), substantivo(enfermeiros))))";

"sentence(sintagma_nominal(determinante(o), substantivo(enfermeiro)), sintagma_ verbal(verbo(tem), sintagma_nominal(determinante(um), substantivo(cre), sintagma_ nominal(determinante(um), substantivo(nome), sintagma_nominal(conjuncao(e), determinante(um), substantivo(endereço))))))";

"sentence(sintagma_nominal(determinante(o), substantivo(hospital)), sintagma_ verbal(verbo(possui), sintagma_nominal(determinante(um), substantivo(cnpj), sintagma_ nominal(conjuncao(e), determinante(um), substantivo(endereço)))))";

"sentence(sintagma_nominal(determinante(um), substantivo(médico)), sintagma_ verbal(verbo(trabalha), sintagma_preposicionado(preposicao(em), sintagma_nominal (pronome(vários), substantivo(hospitais))))).

Figura 5.42: Estruturas sintáticas planas.

Atividade Predicação Sintática

A predicação sintática (ver seção 5.6.2.2) ocorre para cada vocábulo do discurso de acordo com a sintaxe especificada pela definição 2 do modelo *MDU*. O predicado gerado é composto por seis termos: a estrutura sintática de sujeito ou de predicado, a estrutura sintática completa, a categoria gramatical, o vocábulo, a flexão em número e o indicador de sentença. A figura 5.43 exemplifica a predicação dos vocábulos da sentença *o médico atende um ou mais pacientes.*

*constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, **o**, singular, s1).*

*constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, **médico**, singular, s1).*

*constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, **atende**, singular, s1).*

*constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_adverbial)),numeral, **um**, singular, s1).*

*constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_adverbial)),conjuncao, **ou**, singular, s1).*

*constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_adverbial)),adverbio, **mais**, singular, s1).*

*constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, **paciente**, plural, s1).*

Figura 5.43: Exemplo de predicação sintática.

```
diferente(X , Y) :- (X = Y), !, fail; true.
classe(xxxx).
class(W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1,S1 ), constituinteSintatico( PRSUJ2, ES2 ,
        substantivo , W , NUM2, S2 ),  diferente( PRSUJ1 , PRSUJ2), diferente( S1 , S2 ).
attribute(X, W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , X , NUM1,S1 ), constituinteSintatico( PRSUJ2, ES2 ,
        substantivo , W , NUM2,S1 ), not(classe(W)), classe(X), diferente(W ,  X), diferente(ES1, ES2),
diferente(PRSUJ2 ,  sintagma_nominal).
association(W,X) :-    constituinteSintatico(PRSUJ1, ES1 , substantivo , W , NUM1, S1 ), constituinteSintatico(PRSUJ2,
        ES2 , substantivo , X , NUM2, S1 ), classe(X),  diferente(ES1,ES2),  classe(W).
class(W) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1 ), constituinteSintatico( PRSUJ2,
        ES2 , substantivo , W1 , NUM2, S1 ), diferente(sintagma_nominal , PRSUJ2),diferente( W, W1 ), classe(W1).
cardinality(R, W, X, MIN, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1),
        constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_adverbial)) , numeral , MIN,
        NUM1, S1), constituinteSintatico(sintagma_verbal, ES2 , substantivo , X , NUM2, S1 ),
        constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_adverbial)) , adverbio ,
        muitos, NUM2, S1), associacao(R,W,X),   diferente(ES1,ES2), classe(W), classe(X), not(classe(MIN)).
cardinality(R, W, X, MIN, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1),
        constituinteSintatico(sintagma_verbal, ES2 , substantivo , X , NUM2, S1 ), constituinteSintatico(sintagma_verbal,
        ES3 , numeral , MIN, NUM1, S1), constituinteSintatico(sintagma_verbal, ES4 , adverbio , mais, NUM1,
        S1),associacao(R,W,X),  diferente(ES1,ES2), classe(W), classe(X), not(classe(MIN)).
cardinality(R,W,X,NUM1,NUM2) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1, S1),
        constituinteSintatico( PRSUJ2, ES2 , substantivo , X , NUM2, S1 ), associacao(R,W,X),
        diferente(ES1,ES2),diferente(PRSUJ1,PRSUJ2), classe(X), classe(W), not(classe(NUM1)), not(classe(NUM2)).
cardinality(R, W, X, MIN, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1),
        constituinteSintatico(sintagma_verbal, ES2 , numeral , MIN, NUM1, S1), constituinteSintatico(sintagma_verbal,
        ES3 , substantivo , X , NUM2, S1 ),constituinteSintatico(sintagma_verbal, ES4 , adverbio , vários, NUM2, S1),
        associacao(R,W,X),  diferente(ES1,ES2), classe(W), classe(X), not(classe(MIN)).
cardinality(R, W, X, MIN, MAX) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1),
        constituinteSintatico(sintagma_verbal,
        sintagma_verbal(sintagma_nominal(sintagma_adverbial(sintagma_adverbial))) , numeral , MIN, NUM1, S1),
        constituinteSintatico(sintagma_verbal, ES2 , substantivo , X , NUM2, S1 ), constituinteSintatico(sintagma_verbal,
        sintagma_verbal(sintagma_nominal(sintagma_adverbial)) , numeral , MAX, NUM1, S1), associacao(R,W,X),
        diferente(ES1,ES2), classe(W), classe(X), not(classe(MIN)).
cardinality(R, W, X, 0, MAX) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1),
        constituinteSintatico(sintagma_verbal, ES2, adverbio , no_máximo, NUM1, S1),
        constituinteSintatico(sintagma_verbal, ES2, numeral , MAX, NUM1, S1), constituinteSintatico(sintagma_verbal,
        ES3 , substantivo , X , NUM2, S1 ),associacao(R,W,X),  diferente(ES1,ES2), classe(W), classe(X),
        not(classe(MAX)).
inheritance(X, Y, W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , X , NUM1,S1 ),constituinteSintatico( PRSUJ2,
        ES2 , substantivo , Y , NUM1,S2 ),constituinteSintatico( sintagma_verbal, ES3 , substantivo , W , NUM2,S1 ),
        constituinteSintatico( sintagma_verbal, ES4 , substantivo , W , NUM2,S2 ), atributo(X, W), atributo(Y, W),
        classe(X), classe(Y), diferente(X, Y).
```

Figura 5.44: Regras semânticas.

Atividade Instanciação de Base de Conhecimento Semântica

A instanciação de uma *base de conhecimento semântica* corresponde à geração de um modelo *MDU* em tempo de execução pelo componente analisador semântico do *ASIEEOOD.* Inicialmente, a base citada é composta pelas regras semânticas, geradas pelo engenheiro de conhecimento de acordo com a definição 3 do *MDU* apresentada na seção 5.6.2.3 e ilustradas na figura 5.44, e pelos fatos sintáticos gerados a partir da predicação sintática e exemplificados na figura 5.43.

Atividade Predicação Semântica

A predicação semântica (ver seção 5.6.2.4) inclui a geração dos predicados ilustrados na figura 5.45, sendo os mesmos armazenados na base de dados como *fatos semânticos*. Posteriormente são processados pelo *Editor Gráfico do Diagrama de Classe,* cujo processo permite a geração do modelo desenhado na figura 5.46.

heranca(médico_paciente, médico).
classe(médico).
classe(médico_paciente).
classe(paciente).
multiplicidade(hospital/enfermeiro, enfermeiro, 1, 0).
multiplicidade(hospital/enfermeiro, hospital, 1, 1).
multiplicidade(hospital/médico, hospital, 1, 0).
multiplicidade(hospital/médico, médico, 1, 0).
multiplicidade(médico/paciente, médico, 1, 0).
multiplicidade(médico/paciente, paciente, 1, 0).
heranca(médico_paciente, paciente).
classe(enfermeiro).
classe(hospital).
atributo(paciente, identidade).
atributo(enfermeiro, nome).
associacao(médico/paciente, médico, paciente).
atributo(enfermeiro, cre).
atributo(enfermeiro, endereço).
atributo(hospital, cnpj).
atributo(hospital, endereço).
associacao(hospital/enfermeiro, hospital, enfermeiro).
atributo(médico, crm).
atributo(médico_paciente, endereço).
atributo(médico_paciente, nome).
atributo(médico_paciente, telefone).
associacao(hospital/médico, hospital, médico).

Figura 5.45: Predicados referentes aos elementos orientados a objeto extraídos.

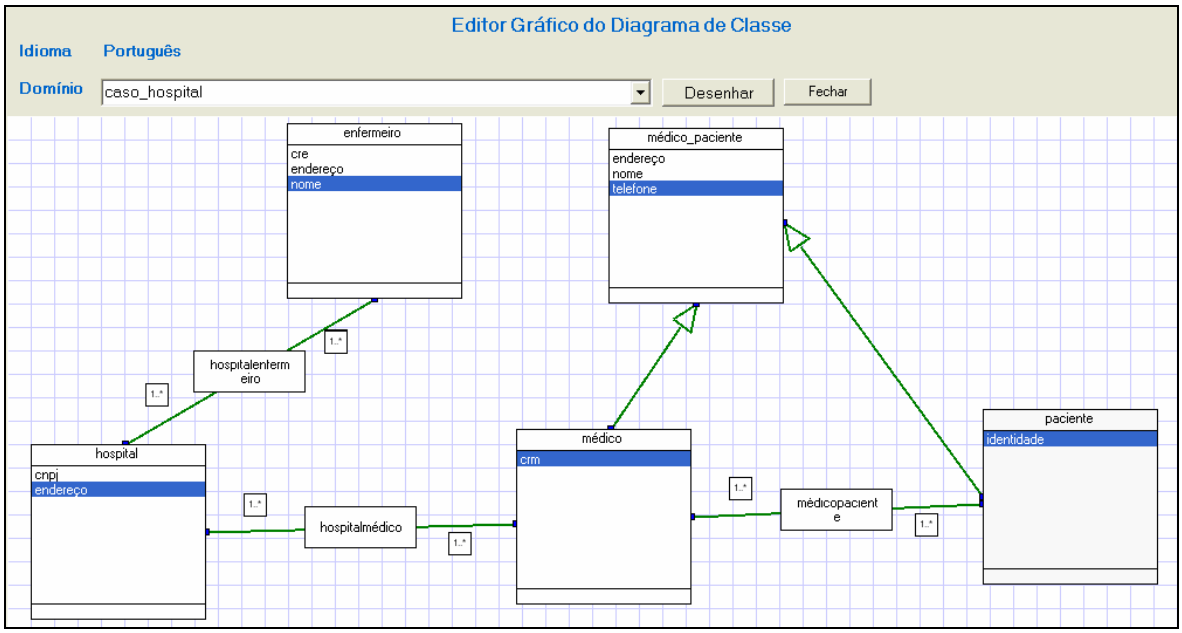


Figura 5.46: Modelo conceitual do estudo de caso.

5.7 Arquitetura do Assistente Inteligente Baseado em Conhecimento

O *ASIEEOOD* é um sistema baseado em conhecimento com dois atores: o engenheiro de conhecimento e o analista. As principais responsabilidades do engenheiro de conhecimento incluem a criação e a manutenção da base de conhecimento, ou seja, a criação de regras sintáticas e semânticas. A figura 5.47 ilustra o esquema do sistema onde estão definidas as principais interfaces funcionais dos referidos atores.

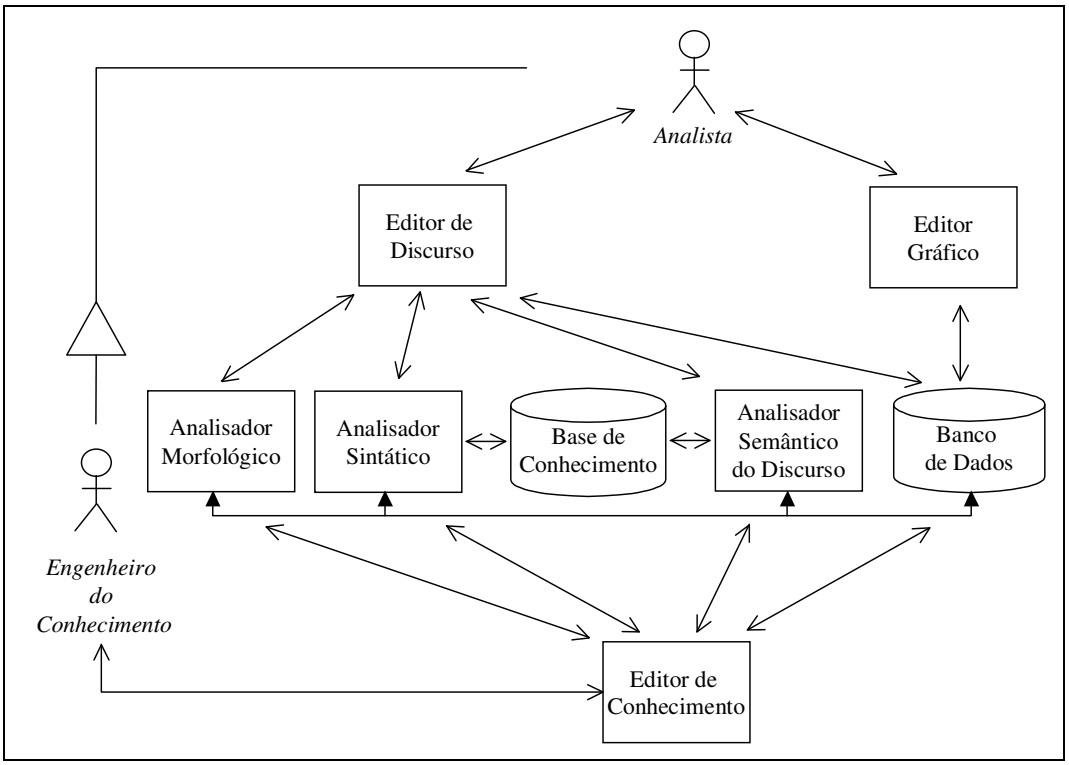


Figura 5.47: Esquema geral do assistente inteligente.

5.7.1 Descrição da Arquitetura

A arquitetura proposta conforma a *Descrição de Arquitetura do Processo Unificado* (JACOBSON *et al.*,2000) com base nas seguintes premissas: prover a fundação pela qual o sistema será construído (AMBLER, 2004), sendo uma ponte entre os requisitos e a implementação (GARLAN, 2000); e destacar os principais diagramas em notação *UML* que permitam a implementação do conceito de que construir uma arquitetura em software significa enquadrar as atividades de análise e projeto em um maior e mais coerente *framework* (ALBIN, 2003).

Com base nas referidas premissas, a descrição de arquitetura proposta inclui as seguintes vistas e respectivos diagramas (SILVA e CARVALHO, 2006c):

• vista de caso de uso, modelada por meio do diagrama de pacote de casos de uso;

• vista de análise e projeto, modelada por meio do diagrama de pacote da estrutura estática;

• vista de implementação, modelada por meio do diagrama de pacote do sistema e do diagrama de componentes;

• vista de *implantação,* modelada por meio do diagrama de implantação.

5.7.2 Vista de Caso de Uso

A *Vista de Caso de Uso* inclui o pacote de caso de uso ilustrado na figura 5.48 que define o contexto do sistema. O referido pacote permite a organização, em um único diagrama, de vários diagramas de caso de uso. O ator *analista* acessa o pacote *Edição Gráfica do Diagrama de Classe* e *Processamento de Discurso,* enquanto que o ator *engenheiro de conhecimento*, além de ser uma especialização de *analista*, tem o privilégio sobre os pacotes *Gerenciamento de Base de Conhecimento* e *Edição Gráfica Árvore Sintática.*

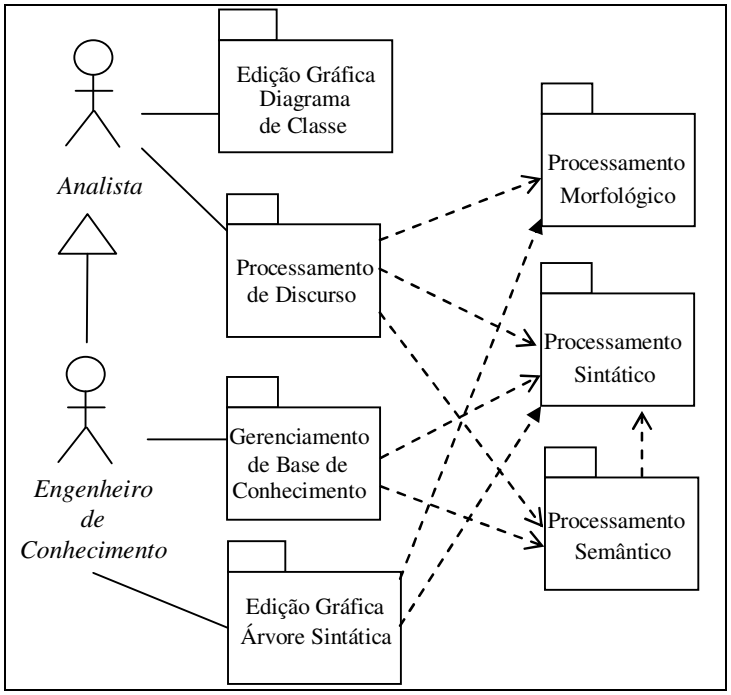


Figura 5.48: Pacote Diagrama de Caso de Uso.

O pacote *Processamento de Discurso*, ilustrado na figura 5.49, inclui os casos de uso cujas funcionalidades permitem o processamento das sentenças em linguagem natural que compõem o discurso, o pacote *Edição Gráfica Diagrama de Classe* inclui as funcionalidades relacionadas ao desenho do modelo conceitual, o pacote *Gerenciamento*

de Base de Conhecimento inclui as funcionalidades relativas à gestão dos axiomas das bases de conhecimento, i.e., regras e fatos. O pacote *Processamento Morfológico* é composto pelos casos de uso que tratam dos vocábulos e respectivas etiquetações. O pacote *Processamento Sintático* contém os casos de usos que permitem a análise sintática das sentenças e a extração dos constituintes sintáticos, i.e., a categoria gramatical e a estrutura sintática. O pacote *Processamento Semântico*, desenhado na figura 5.50, inclui as funcionalidades do analisador semântico do discurso possibilitando a extração dos elementos orientados a objeto a partir dos constituintes sintáticos.

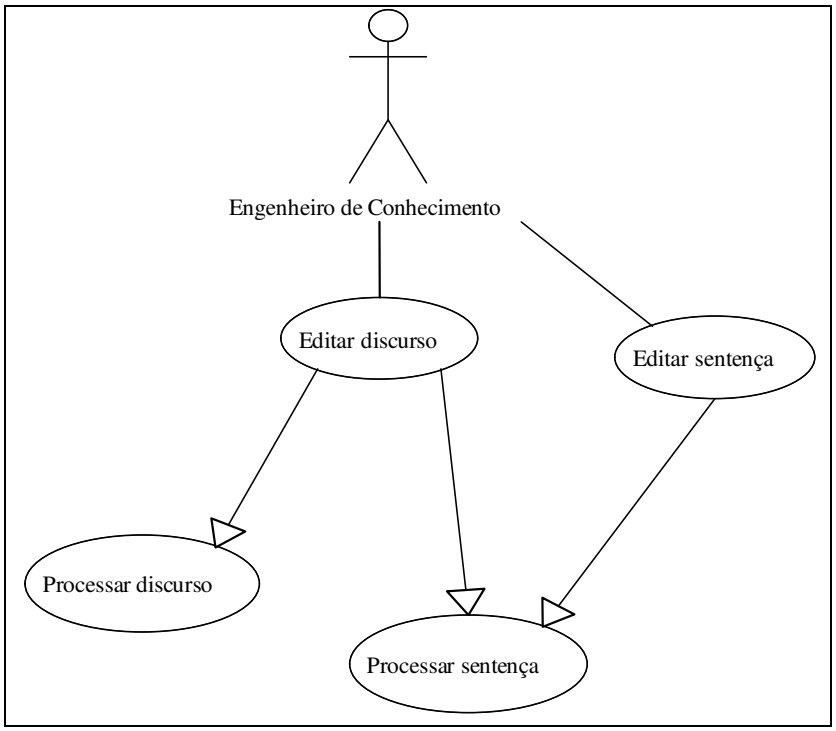


Figura 5.49: Pacote de caso de uso Processamento de Discurso.

5.7.3 Vista de Análise e Projeto

A *Vista de Análise e Projeto* encapsula duas vistas originais do *Processo Unificado*, sendo esta decisão de projeto fundamentada nas seguintes premissas: a arquitetura deve produzir as vistas úteis para o sistema (CLEMENTS, 2005); e os modelos diagramas de classe têm a mesma notação desde a análise até o projeto, i.e., os modelos desenvolvidos durante o ciclo de vida do processo de desenvolvimento de software (JACOBSON *et al.*, 2000).

O pacote de *Estrutura Estática*, ilustrado na figura 5.51, permite a organização, em um único diagrama, dos diagramas de classes existentes no sistema de PLN proposto. O pacote *Domínio Sintático* contém todas as classes utilizadas pelo analisador sintático e o pacote

Domínio Semântico, as classes necessárias ao analisador semântico. O pacote *DB Persistência* inclui todas as classes de acesso à base de dados do sistema, representando a camada de persistência do mesmo. O pacote *SWI-CPP* é uma interface C++ para Prolog, tal como um *commercial off-the-shelf (COTS).* A figura 5.17 ilustra o diagrama de classe do *Domínio Sintático* e a figura 5.30, do *Domínio Semântico.*

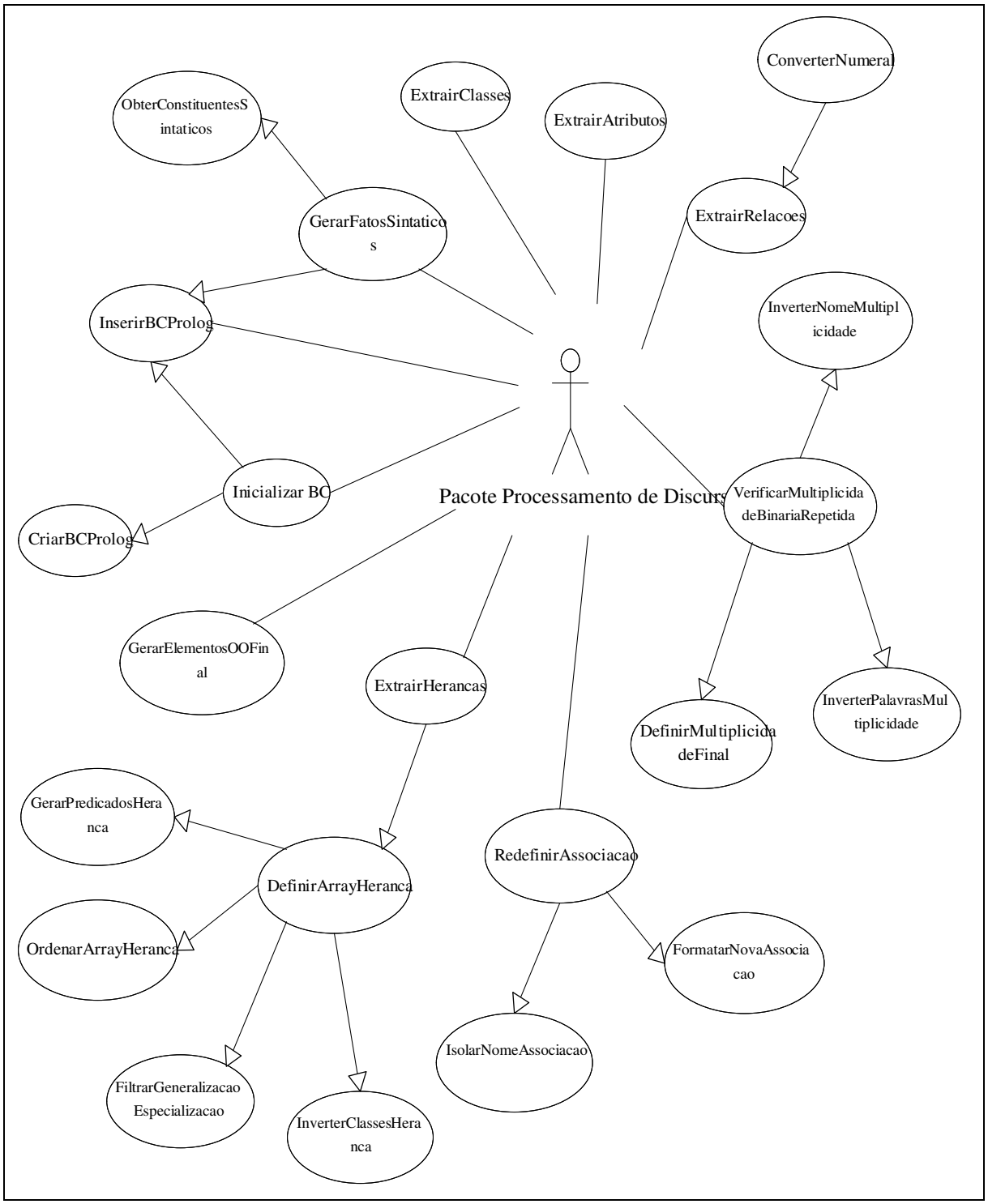


Figura 5.50: Pacote de caso de uso Processamento Semântico.

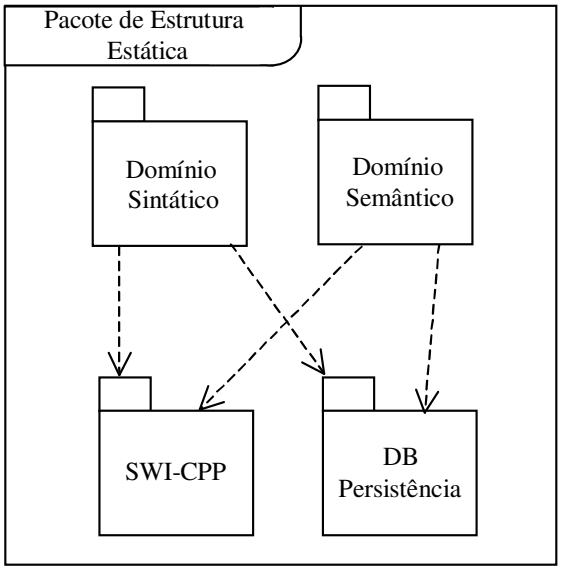


Figura 5.51: Pacote de Estrutura Estática.

5.7.4 Vista de Implementação

A *Vista de Implementação* inclui o pacote *Sistema* e o diagrama de componentes. O pacote *Sistema*, usado como pacote estereotipado, descreve os dois módulos principais, i.e., os dois subsistemas que representam a decomposição de alto nível, do sistema PLN em estudo, tal qual mostra a figura 5.52.

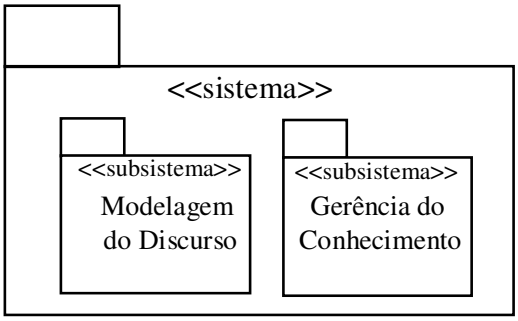


Figura 5.52: Pacote Sistema.

O subsistema *Gerência do Conhecimento* inclui as funcionalidades que permitem a manipulação das bases de conhecimento pelo engenheiro de conhecimento e o subsistema *Modelagem do Discurso*, as funcionalidades relacionadas ao processamento do discurso, a extração dos elementos orientados a objeto do mesmo e o desenho gráfico do modelo conceitual orientado a objeto. Na UML o classificador *subsistema* é uma versão especializada do classificador componente (BELL, 2004), sendo os referidos subsistemas os classificadores de mais alto nível do sistema.

O *Diagrama de Componentes* define a modelagem de alto nível dos componentes de software e as interfaces entre os referidos componentes (AMBLER, 2004), sendo o mesmo ilustrado na figura 5.53 em notação *UML 2.0.*

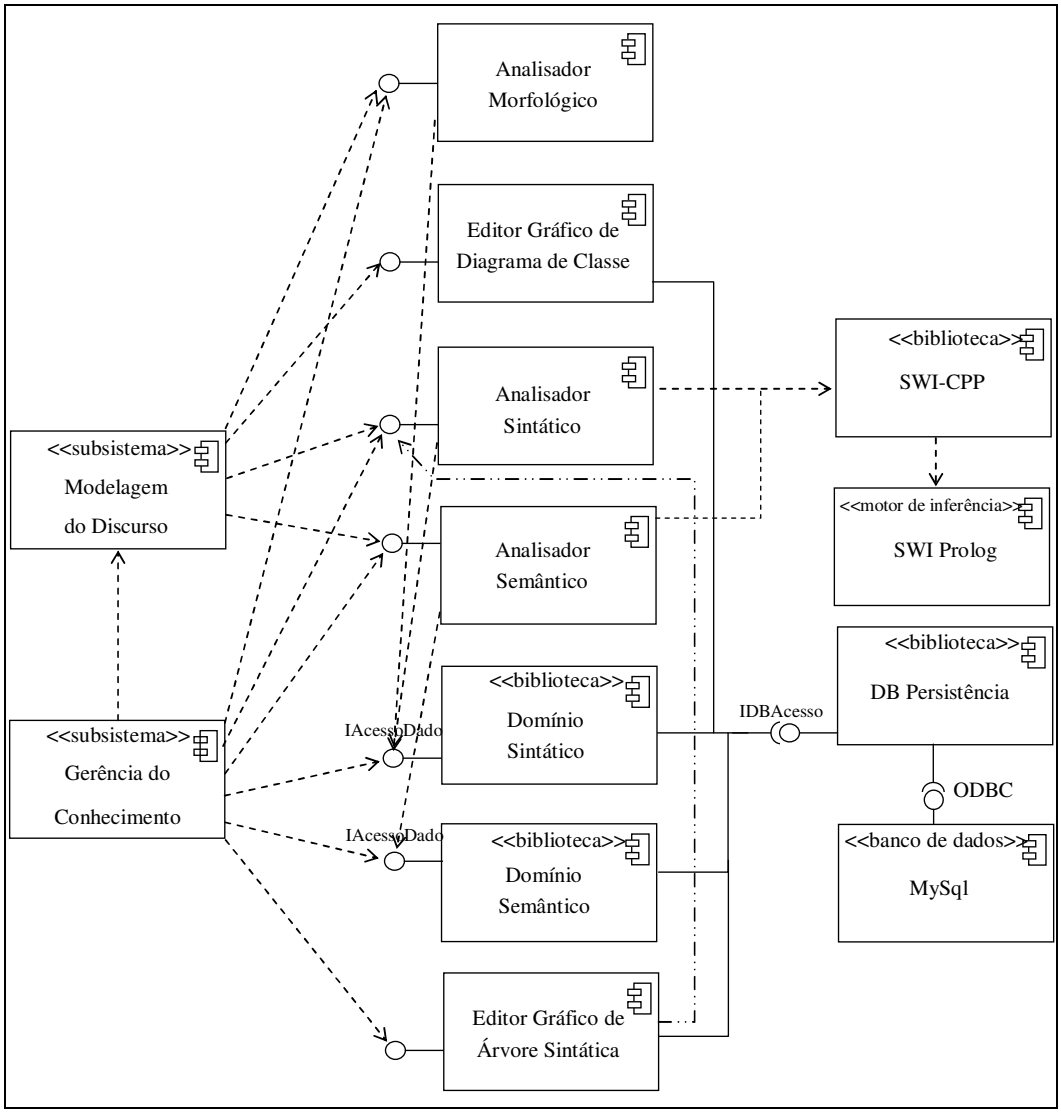


Figura 5.53: Diagrama de Componente.

As bibliotecas *Domínio Sintático* e *Domínio Semântico* implementam a interface *IAcessoDado*, descrita na figura 5.54 (a), de modo que todas as classes são acessadas por meio dos métodos da interface. Os componentes *Analisador Sintático, Analisador Semântico, Editor Gráfico de Diagrama de Classe* e *Editor Gráfico de Árvore Sintática* implementam suas próprias interfaces por meio de métodos públicos, e.g., a implementação do *Analisador Sintático* na figura 5.20. A figura 5.54 (b) ilustra a interface *IDBAcesso*, implementada pelo componente *DB Persistência.*

```
public __gc __interface IAcessoDado
{
   public:
            bool Inserir(void);
            bool Alterar(void);
            bool Eliminar(void);
            ArrayList* Buscar(void);
            ArrayList* BuscarLista(void);
}

public __gc class Terminal : public IAcessoDado
{...
     public:
                ...
                bool Inserir(void);
                bool Alterar(void);
                bool Eliminar(void);
                ArrayList* Buscar(void);
                ArrayList* BuscarLista(void);
};
```

(a)

```
public __gc __interface IDBAcesso
{
   public:
            bool Inserir(void);
            bool Alterar(void);
            bool Eliminar(void);
            ArrayList* Buscar(void);
            ArrayList* BuscarLista(void);
}

public __gc class DBDicionario : public IDBAcesso, public DBConexao
{...
    public:
            ...
            bool Inserir(void);
            bool Alterar(void);
            bool Eliminar(void);
            ArrayList* Buscar(void);
            ArrayList* BuscarLista(void);
            bool Conectar(void);
};
```

(b)

Figura 5.54: Exemplos de classes de interface

O *Diagrama de Componentes* determina exatamente quais os artefatos de alto nível a serem desenvolvidos, permitindo o reuso de componentes, e.g., o componente *Analisador Sintático* é requerido pelos subsistemas *Gerência do Conhecimento* e *Modelagem do Discurso*.

5.7.5 Vista de Implantação

A *Vista de Implantação* inclui o *Diagrama de Implantação* ilustrado na figura 5.55, sendo a mesma uma vista de configuração em tempo de execução dos nós de processamento e dos componentes que serão executados nos referidos nós. O referido diagrama define dois tipos de hardwares *clientes*, determinados pelos atores do sistema, e um hardware *servidor*, onde estão alocados o *servidor de aplicação*, o *servidor de banco de dados* e o *servidor de conhecimento*. Justifica-se o projeto de um único servidor pelo fato de não ser o banco de dados *MySql* um software corporativo que exija um servidor dedicado.

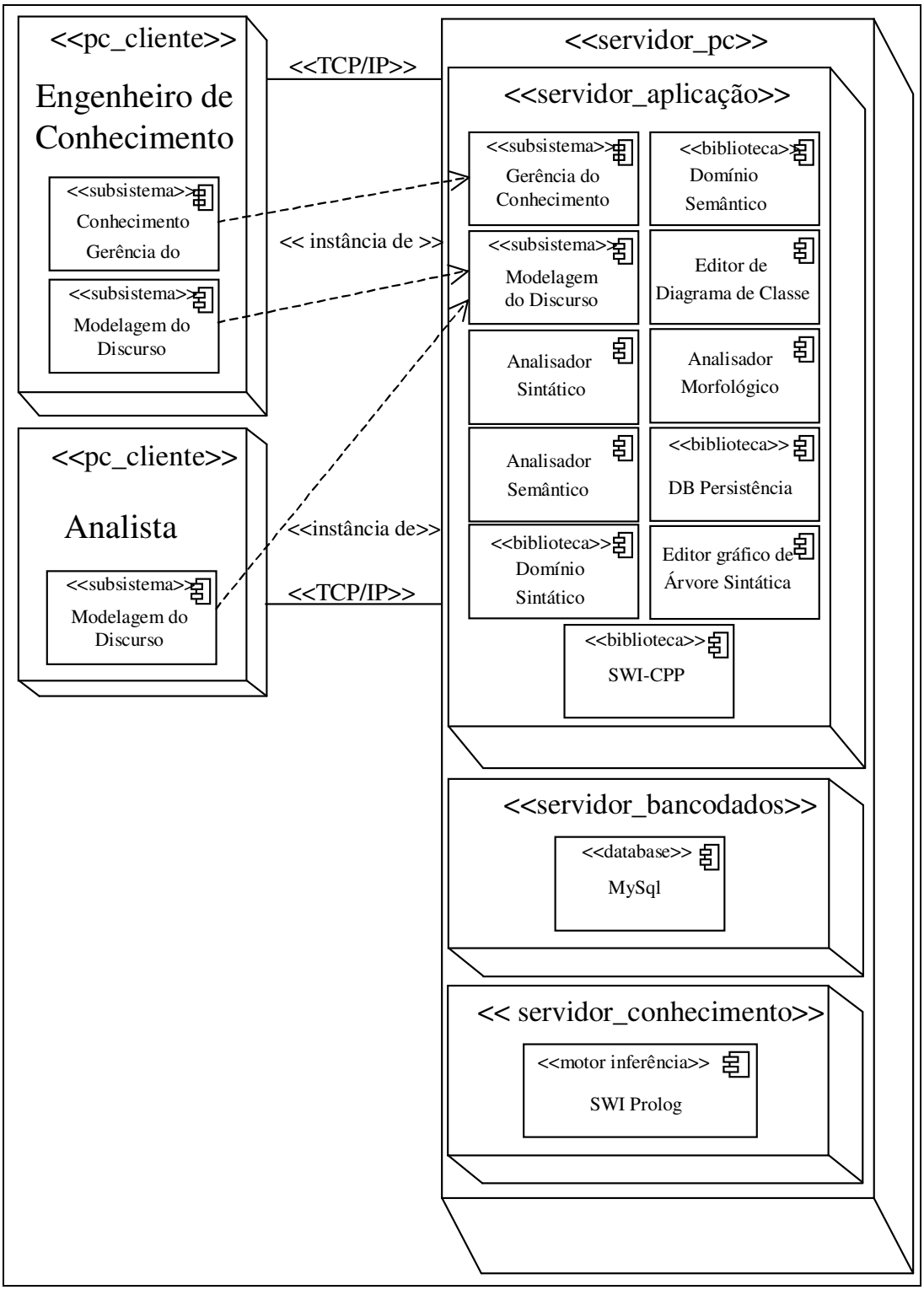


Figura 5.55: Diagrama de Implantação.

5.8 Considerações

O presente capítulo, núcleo deste trabalho, descreveu a técnica lingüística, objeto desta pesquisa, e a arquitetura de software do assistente inteligente *ASIEEOOD*. A técnica

lingüística proposta considerou a definição de um sistema de gramática, uma linguagem de representação do conhecimento, uma estrutura de representação do conhecimento e de níveis lingüísticos.

O sistema de gramática proposto é composto pela *gramática de estrutura sintagmática*, ou *PSG-grammar (PSG)*, como o algoritmo formal que permite a geração das seqüências gramaticais (CHOMSKY, 2002) e a *forma de palavra* como o método de reconhecimento de palavra automático, por permitir a implementação de um algoritmo de reconhecimento simples por meio de busca da palavra completa como chave no dicionário analisado (HAUSSER, 2001).

A linguagem de representação do conhecimento adotada é a cláusula definida de Horn em notação Prolog, por ser uma linguagem de programação lógica usada, entre outras aplicações, em análise de linguagem natural (RUSSEL e NORVIG, 2004). O nível de significado declarativo da mesma permite informar ao computador o que é verdade e, a partir das referidas verdades, obter conclusões.

A estrutura de representação do conhecimento, i.e., a rede semântica adotada, é a notação do diagrama de classe da UML. A referida escolha justifica-se na premissa de que a rede semântica é um sistema especialmente projetado para organizar e raciocinar com categorias, oferecendo auxílio gráfico para visualização de uma base de conhecimento (RUSSEL e NORVIG, 2004).

O *ASIEEOOD,* por ser baseado em conhecimento, implementa uma *abordagem declarativa*, o que garante ao engenheiro de conhecimento flexibilidade na especificação das regras semânticas, sendo as mesmas independentes de domínios ou procedimentos.

Capítulo 6

Estudo Empírico

6.1 Apresentação

Os estudos empíricos da engenharia de software, e.g., estudo de caso e experimento, são fundamentais na avaliação de processos e atividades humanas, sendo benéficos, também, quando existe a necessidade de avaliar o uso de produtos de software ou ferramentas (WHOLIN *et al.*, 2000).

Com a finalidade de avaliar a técnica lingüística implementada no *ASIEEOOD*, a presente tese adota o estudo empírico denominado de *estudo de caso*.

6.2 Contexto

O objetivo geral deste estudo empírico é avaliar a técnica lingüística proposta, que permite extrair os elementos orientados a objeto, implementada no *ASIEEOOD*.

Os objetivos específicos do referido estudo empírico incluem:

- avaliação da linguagem controlada proposta a partir de especificações em linguagem natural de dois documentos estruturados;
- avaliação das regras semânticas estabelecidas;
- comparação dos modelos gerados por alunos a partir das especificações em linguagem natural com os modelos gerados por um engenheiro de conhecimento.

A perspectiva para o estudo empírico é dupla. Inicialmente, a perspectiva é relativa ao analista, sendo limitada à edição das especificações em linguagem natural a partir dos estudos de caso; posteriormente, a perspectiva é relativa ao engenheiro de conhecimento, identificando as novas estruturas sintagmáticas e as novas regras semânticas.

O ambiente do estudo empírico é o acadêmico, i.e., dois documentos estruturados distribuídos a alunos de graduação e pós-graduação na área de informática, sendo a solução realizada de forma individual e a remessa dos resultados por meio eletrônico. O primeiro

estudo trata de um histórico acadêmico e o segundo, de uma nota fiscal. O Anexo A contém o material entregue aos alunos para realização destes estudos de caso.

Com o objetivo de permitir uma avaliação das sentenças processadas, as mesmas foram classificadas, por observação, em cores cujos significados estão descritos na tabela 6.1.

Tabela 6.1: Padrão de cores das sentenças processadas.

Cor	Significado	Exemplo
Vermelha	• A sentença necessita de uma nova estrutura sintagmática devido a não existência na base de conhecimento da respectiva estrutura.	• Aluno concluiu o ensino médio em uma escola em data determinada.
Verde	• O processamento sintático da sentença ocorre devido à existência na base de conhecimento da respectiva estrutura sintagmática.	• Nota fiscal possui um número, um identificador único e uma data de emissão.
Azul	• A sentença conduz à geração de uma nova regra semântica.	• Os códigos das disciplinas são relativos a departamentos.
Vinho	• A sentença necessita ser reescrita.	• *Produto* possui *código do produto*, *descrição dos produtos*, situação tributária, unidade, quantidade, valor unitário, valor total, alíquota de ICMS.
Preta	• A sentença tem uma ênfase funcional.	• O valor total do imposto a ser recolhido é registrado na nota fiscal.
Laranja	• A sentença não se aplica como especificação.	• Um tipo pode ser entrada ou saída.

6.3 Resultados

A fim de avaliar a linguagem controlada proposta a partir de especificações em linguagem natural de dois documentos estruturados, o primeiro dos três objetivos específicos deste estudo empírico, a figura 6.1 ilustra os resultados do estudo de caso *Histórico Escolar* e a figura 6.2, do estudo de caso *Nota Fiscal*, sendo ambos os estudos realizados por oito alunos.

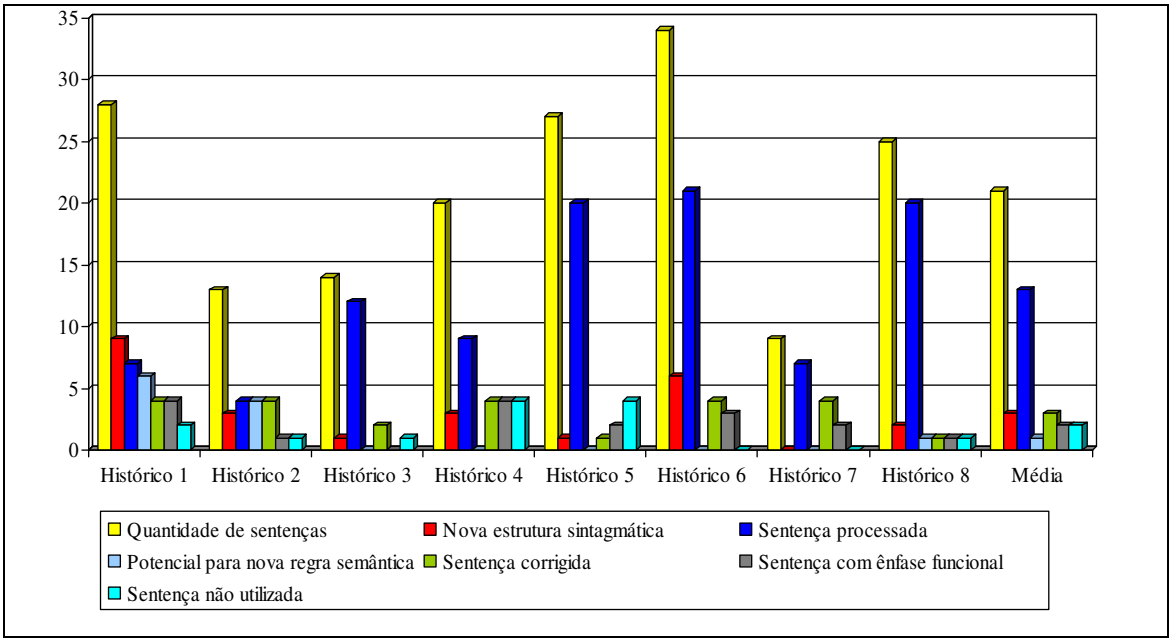


Figura 6.1: Estudo de caso *Histórico Escolar.*

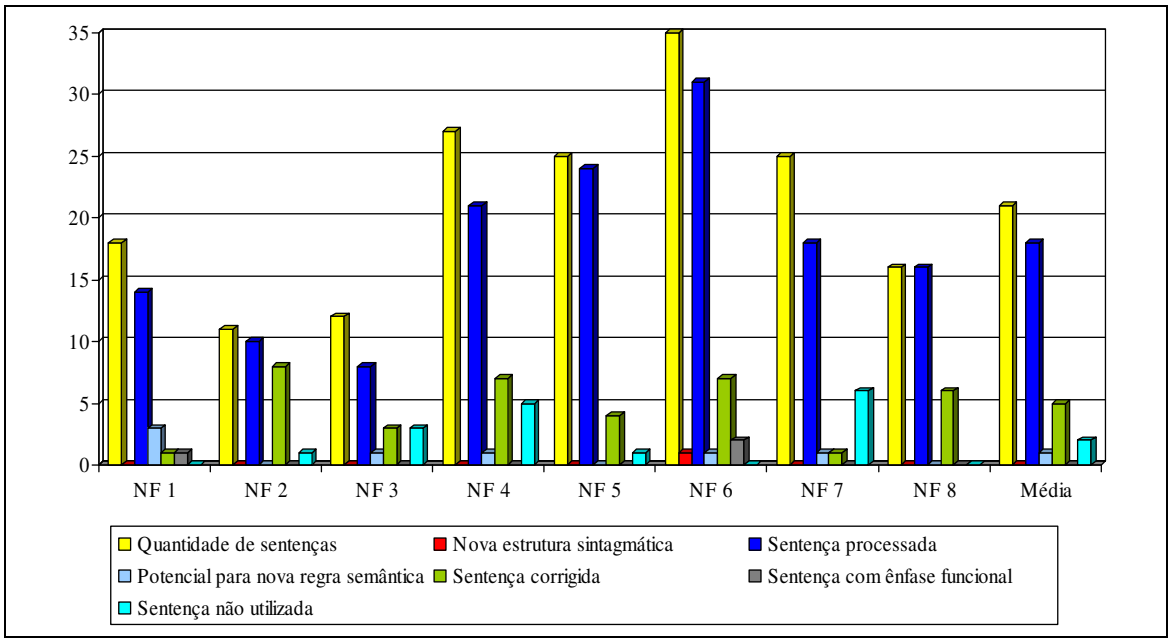


Figura 6.2: Estudo de caso *Nota Fiscal.*

A fim de avaliar as regras semânticas estabelecidas, o segundo dos três objetivos específicos deste estudo empírico, a figura 6.3 descreve o discurso gerado por um engenheiro de conhecimento a partir do documento estruturado *NOTA FISCAL*, estando a respectiva modelagem ilustrada na figura 6.4. A figura 6.5 descreve o discurso gerado por um aluno, sendo este o melhor documento estruturado *NOTA FISCAL* avaliado entre os oito alunos, estando a respectiva modelagem ilustrada na figura 6.6. O texto pior avaliado está ilustrado na figura 6.7, estando a respectiva modelagem representada na figura 6.8. O

Anexo B descreve a *base de conhecimento semântica* instanciada quando do processamento do discurso incluso na figura 6.3.

Uma nota fiscal tem um número, uma data limite para emissão, uma data de emissão, uma data saída, uma data entrada, uma hora da saída, um CNPJ, uma inscrição estadual, uma inscrição estadual substituto tributário, um CFOP.
Uma nota fiscal tem uma operação.
Uma operação está associada a uma ou várias notas fiscais.
Uma operação tem um nome.
Uma nota fiscal tem um endereço.
Um endereço é associado a uma ou varias notas fiscais.
Um endereço é composto por uma rua, um número, um complemento, um bairro, uma cidade, uma UF e um telefone.
Uma nota fiscal é gerada para um destinatário remetente.
Um destinatário remetente recebe uma ou várias notas fiscais.
Um destinatário remetente possui endereço.
Um endereço é associado a um destinatário remetente.
Um destinatário remetente possui nome razão social e inscrição estadual.
Uma nota fiscal tem um ou vários itens produto.
Um item produto é vendido em uma nota fiscal.
Um item produto está associado a um produto.
Um produto está associado a zero ou vários itens produto.
Um item produto possui uma quantidade, uma situação tributária e uma alíquota de ICMS.
Um produto tem um código, uma descrição, uma unidade e um valor unitário.
Uma nota fiscal tem um cálculo do imposto.
Um cálculo do imposto está associado a uma nota fiscal.
Um cálculo do imposto possui uma base de cálculo ICMS, um valor do ICMS, uma base ICMS substituição, um valor ICMS substituição, um valor total, um valor do frete, um valor do seguro, uma outras despesas acessórias, um valor total IPI e um valor total nota.
Uma nota fiscal tem transportador volumes transportados.
Um transportador volumes transportados está associado a uma nota fiscal.
Um transportador volumes transportados possui um nome, uma razão social, um frete para conta, uma placa do veículo, um CNPJ CPF, uma quantidade, uma espécie, uma marca, um número, um peso bruto, um peso líquido e uma inscrição estadual.
Um transportador volumes transportados possui um endereço.
Um endereço está associado a um transportador volumes transportados.
Uma nota fiscal possui dado adicional.
Dado adicional está associado a uma nota fiscal.
Dado adicional inclui informação complementar, filial, pedido número, vendedor número, número de autenticação e rota.

Figura 6.3: Discurso elaborado por engenheiro de conhecimento.

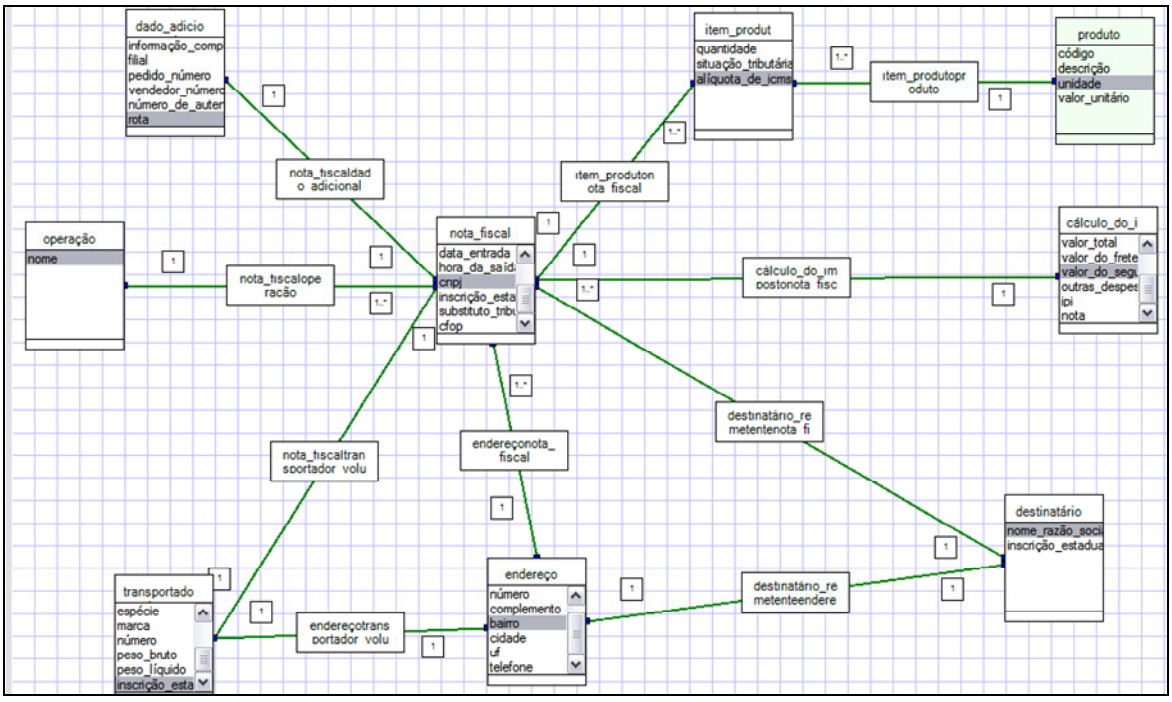


Figura 6.4: Modelagem de discurso elaborado por engenheiro de conhecimento.

Produto tem nome, código, preço.
Empresa tem endereço, CNPJ, Inscrição Estadual.
Empresa vende Produto.
Natureza da operação possui Natureza, CFOP, Inscrição estadual substituto tributário , CNPJ, Inscrição estadual.
Destinatário remetente tem nome razão social, CPF CNPJ, endereço, bairro ou distrito, CEP, município, fone ou fax, UF, inscrição estadual.
Empresa possui nota fiscal.
Empresa possui transportadora.
Transportador volumes transportados possui nome, endereço, frete, placa, UF, CNPJ CPF, inscrição estadual, município, quantidade, espécie, marca, número, peso bruto, peso líquido.
Nota Fiscal possui natureza da operação.
Nota Fiscal possui dado adicional.
Nota Fiscal possui cálculo de imposto.
Nota Fiscal possui transportador volumes transportados.
Nota Fiscal possui produtos.
Cálculo de imposto possui base de cálculo ICMS, valor do ICMS, base de cálculo substituição, valor ICMS substituição, frete, seguro, despesa acessória, valor do IPI.
Produto possui código, descrição, situação tributária, unidade, quantidade, valor unitário, valor total, alíquota de ICMS.
Dado adicional possui informação complementar, filial, pedido, vendedor, número de autenticação, rota, reservado ao fisco.

Figura 6.5: Discurso melhor avaliado.

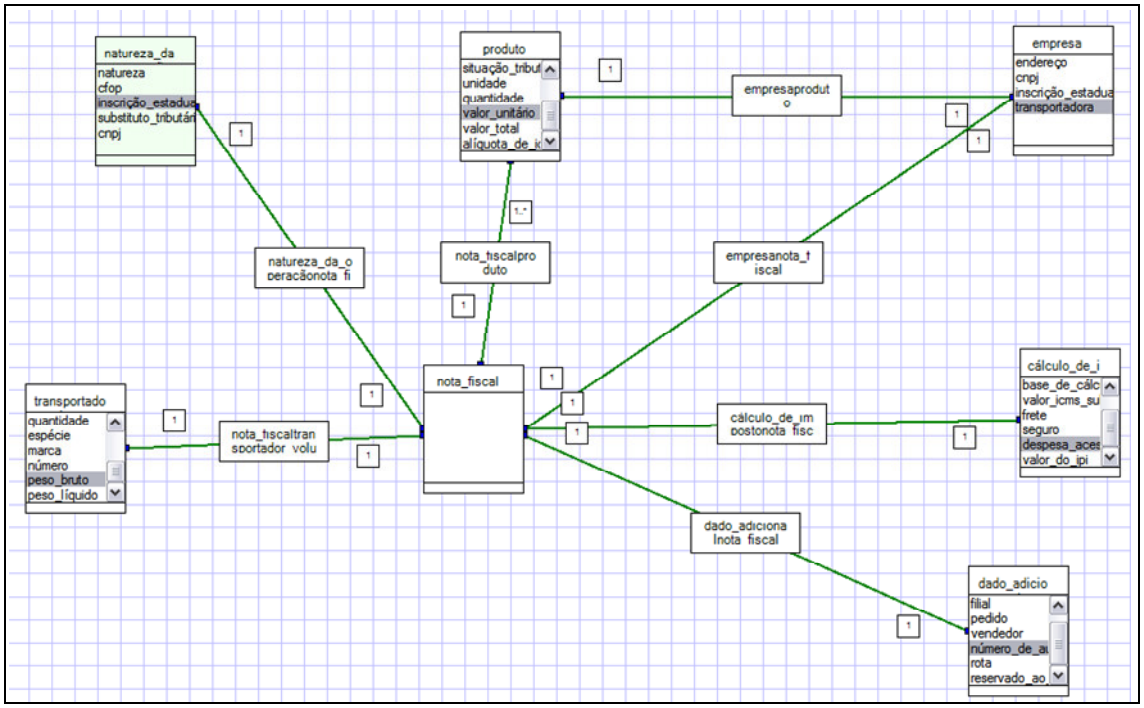


Figura 6.6: Modelagem do discurso melhor avaliado.

A nota fiscal possui um cnpj.
A nota fiscal possui uma natureza da operação.
A nota fiscal possui um CFOP.
A nota fiscal possui uma inscrição estadual substituto tributário e uma inscrição estadual.
A nota fiscal possui um número.
A nota fiscal possui um tipo.
A nota fiscal possui um destinatário.
A nota fiscal possui um ou vários produtos.
A nota fiscal possui um cálculo de imposto.
A nota fiscal possui zero ou um transportador.
A nota fiscal possui zero um dados adicionais.
A nota fiscal possui uma data de emissão.
A nota fiscal possui uma data de tipo.
A nota fiscal possui zero ou uma hora de saída.
O destinatário possui um nome razão social.
O destinatário possui zero ou um cnpj cpf.
O destinatário possui um endereço.
O destinatário possui um bairro ou distrito.
O destinatário possui um município.
O destinatário possui um fone ou fax.
O destinatário possui uma uf.
O destinatário possui zero ou um uma inscrição estadual.
A nota fiscal possui um ou mais produtos.
O produto possui um código.
O produto possui uma descrição.

Figura 6.7: Discurso pior avaliado.

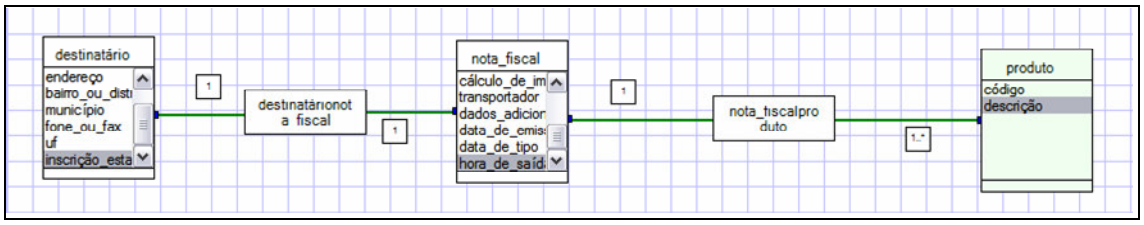


Figura 6.8: Modelagem do discurso pior avaliado.

A fim de permitir a comparação dos modelos gerados pelos alunos a partir das especificações em linguagem natural com o modelo gerado por um engenheiro de conhecimento, o terceiro dos três objetivos específicos deste estudo empírico, o gráfico da figura 6.9 relaciona as classes geradas, a figura 6.10, os atributos, a figura 6.11, as associações e a figura 6.12, as multiplicidades, sendo as identificações de todos os elementos orientados a objeto classificadas em corretas e incorretas.

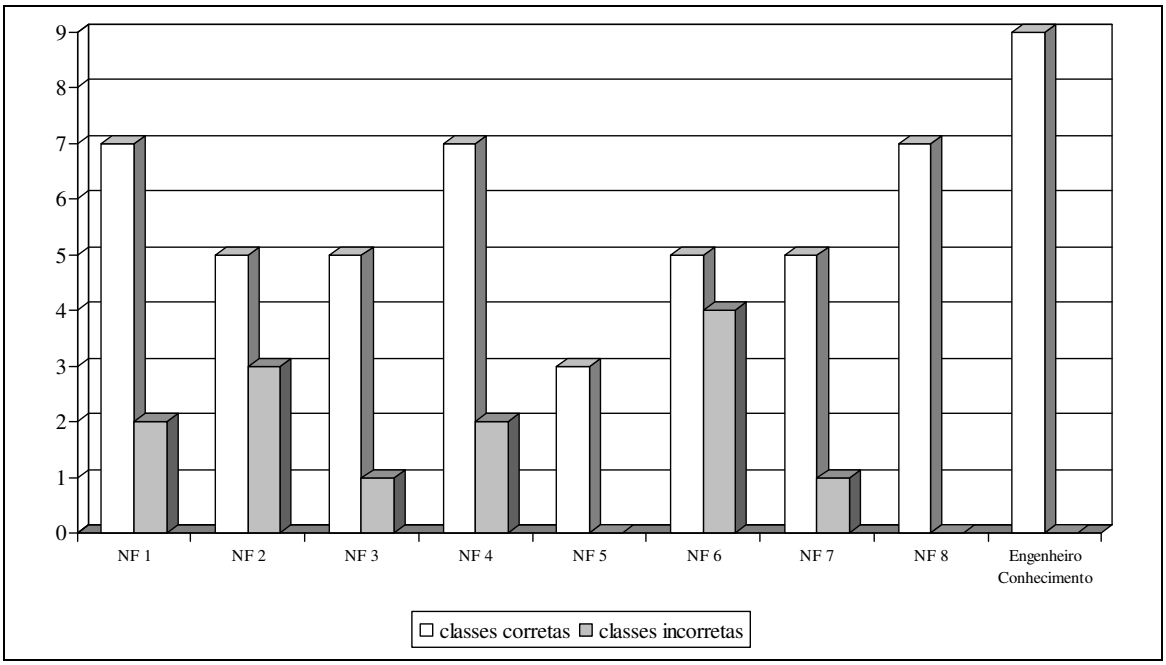


Figura 6.9: Gráfico comparativo do elemento orientado a objeto *Classe.*

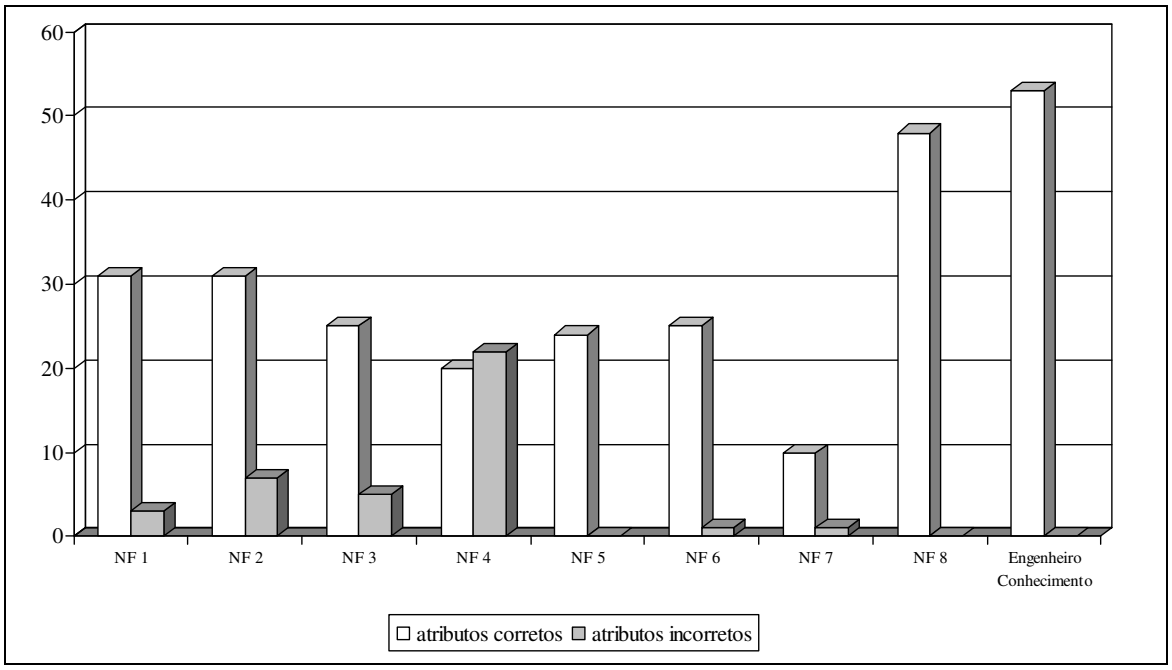


Figura 6.10: Gráfico comparativo do elemento orientado a objeto *Atributo.*

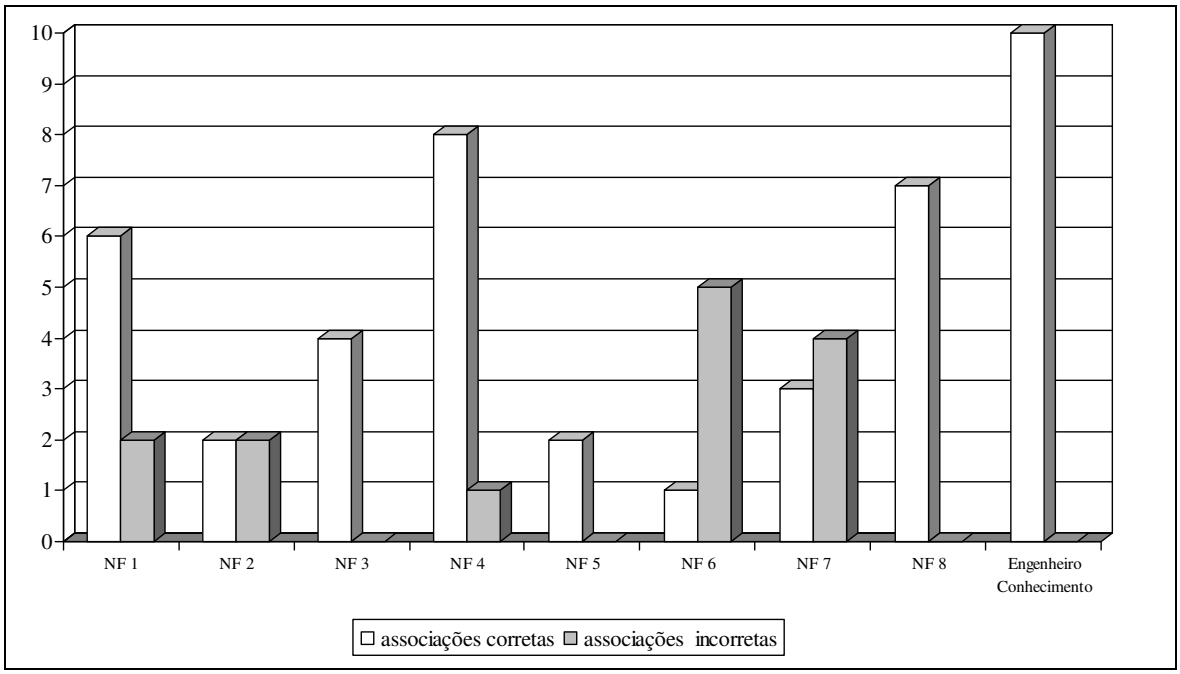


Figura 6.11: Gráfico comparativo do elemento orientado a objeto *Associação.*

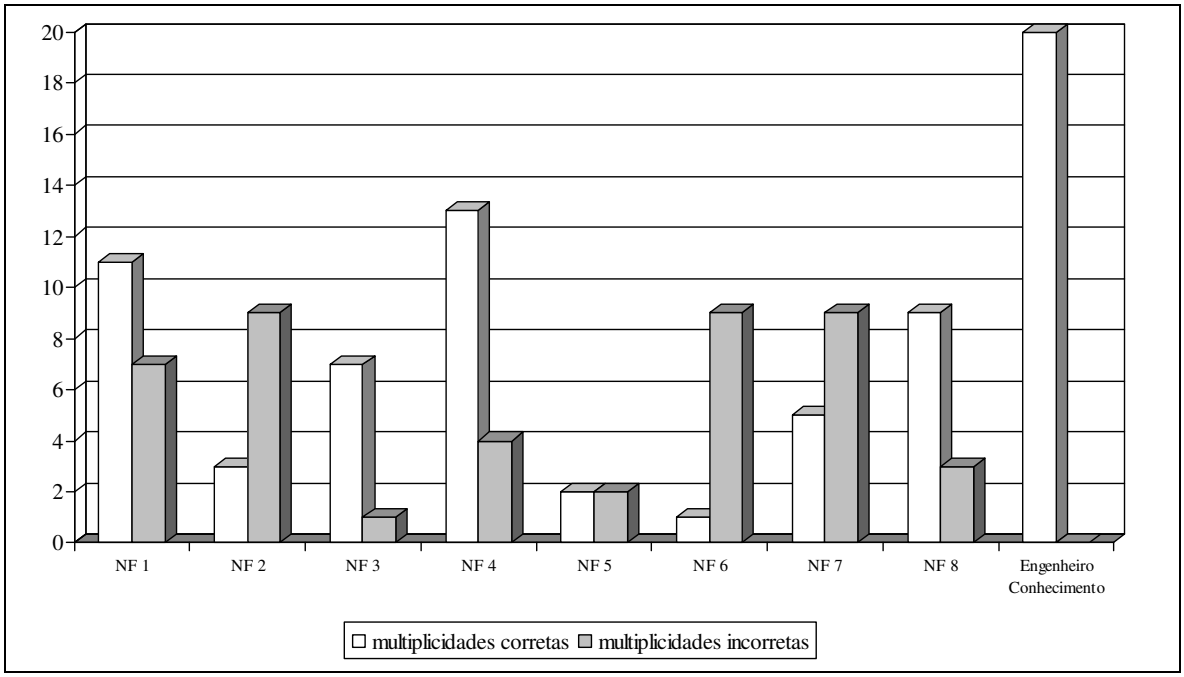


Figura 6.12: Gráfico comparativo do elemento orientado a objeto *Multiplicidade.*

A partir das informações obtidas nas figuras 6.1 a 6.12, destacam-se as seguintes observações:

• a técnica lingüística implementada pelo *ASIEEOOD* atende o objetivo proposto para a presente pesquisa, i.e., gerar um modelo conceitual orientado a objeto a partir de especificações de requisitos em linguagem natural;

• o acréscimo de sentenças processadas gera uma diminuição significativa da necessidade de novas estruturas sintagmáticas, reflexo da exeqüibilidade da linguagem

controlada em restringir as estruturas da linguagem, facilitando o processamento das sentenças;

• o uso de palavras, ou expressões, semelhantes para um mesmo conceito, e.g., data de saída e data saída, gera a necessidade de que a sentença seja corrigida pelo engenheiro de conhecimento, podendo, também, o conceito final ser definido durante a validação do modelo conceitual no editor gráfico;

• a aplicação do método declarativo na geração de regras sintáticas, representativas de estruturas sintagmáticas, e de regras semânticas flexibiliza a determinação de regras por parte do engenheiro de conhecimento;

• a técnica lingüística proposta é independente de domínio do problema;

• as regras semânticas são flexíveis e permitem a geração dos elementos orientados a objeto determinados pelos analistas que especificam textualmente o problema em estudo, e.g., a *nota fiscal*;

• a modelagem é dependente da especificação, i.e., discurso, elaborado por um analista, de modo que para um discurso bem elaborado (figura 6.5) é gerado um modelo conceitual (figura 6.6) semelhante ao modelo gerado por um engenheiro de conhecimento (figura 6.4), caso contrário é especificado um discurso (figura 6.7) cujo modelo (figura 6.8) é bem diferente do gerado pelo citado engenheiro;

• o entendimento da aplicação da linguagem controlada, por parte do analista, é essencial à geração de um modelo conceitual orientado a objeto representativo do problema em estudo.

O exemplo a seguir ilustra a flexibilidade para elaboração de uma regra semântica pelo engenheiro de conhecimento durante o processamento semântico de um determinado discurso especificado por um analista. Durante o processamento do discurso descrito na figura 6.3 não foi possível identificar corretamente, quando da análise semântica realizada pelo *ASIEEOOD*, a multiplicidade gerada pela sentença *uma operação está associada a uma ou várias notas fiscais*, pois entre as regras semânticas existentes somente o pronome *vários* é considerado. A partir da necessidade de uma nova regra semântica que considere o pronome *várias*, foi criada, por meio do editor de regras do *ASIEEOOD*, a regra 11 a seguir, tendo como resultado a multiplicidade "*1..**" ilustrada na figura 6.4 entre as classes *operação* e *nota_fiscal*.

Regra Semântica 11	
Elemento OO	Cardinalidade.
Heurística	**Se** (*R* relaciona *W e X*) e (*W* e *X* estão na mesma sentença) e (existe um numeral e o advérbio *várias* no predicado) **então** (multiplicidade mínima de *X* é o *numeral*) e (multiplicidade máxima de *X* é *muitos*)* ** representado pelo valor zero*
Regra	*cardinality(R, W, X, MIN, 0) :-* *constituinteSintatico(sintagma_nominal, ES1, substantivo, W, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES2, numeral, MIN, NUM1, S1),* *constituinteSintatico(sintagma_verbal, ES3, substantivo, X , NUM2, S1),* *constituinteSintatico(sintagma_verbal, ES4, pronome, vária, NUM2, S1),* *associacao(R,W,X), diferente(ES1,ES2), classe(W), classe(X).*

6.4 Considerações

Os dois estudos empíricos do tipo *estudo de caso* apresentados no presente capítulo permitiram avaliar a técnica lingüística implementada no *ASIEEOOD* por meio de uma linguagem controlada, não sendo, entretanto, a referida linguagem objeto desta pesquisa.

Dos referidos estudos, conclui-se que a técnica lingüística implementada pelo *ASIEEOOD* atende o objetivo proposto para a presente pesquisa, i.e., gerar um modelo conceitual orientado a objeto a partir de especificações de requisitos em linguagem natural.

Capítulo 7

Conclusão e Perspectivas Futuras

Este capítulo conclui a presente tese de doutorado, destacando os resultados obtidos, apresentando as principais contribuições e sugerindo perspectivas futuras a fim de que seja dada continuidade ao trabalho.

7.1 Conclusão

O objetivo geral desta tese foi automatizar a geração de modelo conceitual orientado a objeto a partir de especificações de requisitos em linguagem natural.

A solução do referido problema incluiu a especificação e a implementação de um *assistente inteligente*, i.e., uma ferramenta automatizada com suporte baseado em conhecimento (FALBO, 1998), denominado de *Assistente Inteligente para Extração de Elementos Orientados a Objeto de Discurso (ASIEEOOD)*. Este assistente implementa uma técnica lingüística que permite extrair os conceitos orientados a objeto de sentenças em linguagem natural em função dos constituintes sintáticos dos vocábulos do discurso, i.e., estrutura sintática e categoria gramatical, independentemente do significado da palavra propriamente dito. Os elementos orientados a objeto considerados incluem as classes, os atributos, as associações e as multiplicidades.

O trabalho considerou as seguintes atividades do processo da Engenharia de Requisitos: elicitação, modelagem e validação. A elicitação ocorre, em uma primeira etapa, por meio da textualização de requisitos contidos em documentos de especificações diversos, e.g., documento estruturado, entrevista, sendo, em uma segunda etapa, os referidos textos, ou discurso, processados pelo assistente inteligente e cujos resultados incluem os elementos orientados a objeto em forma de predicados semânticos. A modelagem ocorre a partir dos referidos predicados. A validação é realizada por inspeção visual do analista em um editor gráfico.

Os requisitos lingüísticos básicos da técnica implementada no *ASIEEOOD* incluem a definição de um sistema de gramática, de uma linguagem de representação do conhecimento, de uma estrutura de representação do conhecimento e de níveis lingüísticos, sendo considerados os níveis lingüísticos morfológico, sintático e semântico.

O sistema de gramática proposto é composto pela *gramática de estrutura sintagmática*, ou *PSG-grammar (PSG)*, como o algoritmo formal que permite a geração das seqüências gramaticais (CHOMSKY, 2002) e a *forma de palavra* como o método de reconhecimento de palavra automático, por permitir a implementação de um algoritmo de reconhecimento simples por meio de busca da palavra completa como chave no dicionário analisado (HAUSSER, 2001).

A linguagem de representação do conhecimento adotada é a cláusula definida de Horn em notação Prolog, por ser uma linguagem de programação lógica usada, entre outras aplicações, em análise de linguagem natural (RUSSEL e NORVIG, 2004). O nível de significado declarativo da mesma permite informar ao computador o que é verdade e, a partir das referidas verdades, obter conclusões.

A estrutura de representação do conhecimento, i.e., a rede semântica adotada, é a notação do diagrama de classe da UML. A referida escolha justifica-se na premissa de que a rede semântica é um sistema especialmente projetado para organizar e raciocinar com categorias, oferecendo auxílio gráfico para visualização de uma base de conhecimento (RUSSEL e NORVIG, 2004).

A análise morfológica tem como principais funcionalidades a fragmentação do discurso em sentenças, a busca de palavras compostas e a realização da etiquetação morfológica dos vocábulos.

A análise sintática objetiva a verificação da correção da sintaxe das sentenças analisadas do discurso e da geração das respectivas estruturas sintáticas planas para posterior extração dos constituintes sintáticos, i.e., a estrutura sintática e a categoria gramatical, dos vocábulos componentes das referidas sentenças.

Alinhado com a premissa de Gamut (1991) de que *o núcleo de uma teoria lingüística está em sua teoria semântica*, o presente trabalho tem como núcleo uma teoria semântica fundamentada na teoria estabelecida por Kamp e Reyle (1993), no método indireto de interpretação semântica proposto por Montague (GAMUT, 1991), na interpretação semântica baseada na Teoria do Significado de Frege, na semântica lógica, ou semântica da Teoria do Modelo, de acordo com a visão lógica semântica apresentada por Gamut (1991) e Hausser (2001), na não-composicionalidade da gramática e na semântica lógica

orientada a discurso. Em síntese, a superfície da linguagem natural é traduzida para uma linguagem lógica na forma de predicado, cujos argumentos mais significativos estão relacionados com a posição sintática do vocábulo analisado na sentença e a respectiva categoria gramatical. Este predicado é a *forma de apresentação* que permite extrair o significado, i.e., o *referente*. O resultado da referida predicação são os predicados que compõem a base de conhecimento semântica na forma de *fatos sintáticos*, sendo a referida base complementada por *regras semânticas*. As regras semânticas implementam as heurísticas que, por meio de vínculos lógicos com os fatos sintáticos, permitem extrair os referentes do discurso, i.e., os elementos orientados a objeto, cuja sintaxe está descrita nas definições do *Modelo de Discurso Baseado em UML* (*MDU)*, cabendo ao engenheiro de conhecimento a geração das mesmas.

A realização de dois estudos empíricos do tipo *estudo de caso* permitiram avaliar a técnica lingüística implementada no *ASIEEOOD*. Da referida avaliação, conclui-se que o objetivo proposto para a presente pesquisa de Doutorado foi atingido.

A tabela 6.1 apresenta um comparativo da técnica lingüística implementada no *ASIEEOOD* com as principais técnicas pesquisadas.

Tabela 6.1: Comparativo entre técnicas lingüísticas.

Técnica Lingüística	Automação	Análise sintática	Análise semântica	Análise de discurso	Gramática Gerativa
Block, 1990	Sim	Sim	Sim	Não	Não
Overmyer *et al.*, 2001	Sim	Não	Não	Não	Não
Juristo *et al.*, 1999	Não	Sim	Sim	Não	Não
Rolland e Proix, 1992	Sim	Sim	Sim	Não	Sim
Li, 2004	Sim	Sim	Não	Não	Não
Color-x	Sim	Sim	Não	Não	Não
Kamp e Reyle, 1993	Não	Sim	Sim	Sim	Sim
ASIEEOOD	*Sim*	*Sim*	*Sim*	*Sim*	*Sim*

Do exposto, pode-se destacar que as principais contribuições deste trabalho incluem:

• o desenvolvimento de uma técnica lingüística fundamentada em lógica semântica que simula o raciocínio de um analista na identificação de elementos orientados a objeto a partir de especificações de requisitos em linguagem natural;

• o uso intensivo do método declarativo, garantindo flexibilidade ao engenheiro de conhecimento na definição de regras que compõem as bases de conhecimentos sintática e semântica;

• a exeqüibilidade da adoção de uma linguagem controlada que não limite as regras sintáticas nem as regras semânticas, garantindo ao sistema capacidade de interpretação

independente de domínio, i.e., a linguagem citada pode ser aplicada a diferentes problemas;

- a adoção de redes semânticas como estruturas de representação de conhecimento para as bases de conhecimento sintática e semântica, particularmente pelo uso da notação do digrama de classe da UML na representação das citadas redes, permitindo, também, a modelagem da estrutura estática do sistema;
- a definição de uma arquitetura para o sistema, servindo de base para trabalhos futuros;
- a independência da gramática em relação à semântica, permitindo que, em trabalhos futuros, seja utilizado um outro sistema de gramática;
- a automação da técnica lingüística proposta, viabilizando a industrialização do *ASIEEOOD*.

7.2 Perspectivas Futuras

Um esforço considerável de pesquisa foi realizado no intuito de especificar uma técnica lingüística fundamentada em lógica semântica com o objetivo de extrair os elementos orientados a objeto de um determinado discurso, esforço este coroado com a implementação da mesma pelo *ASIEEOOD*. No entanto, a riqueza do tema permite que muitas outras pesquisas sejam realizadas, tais como:

- especificação de uma linguagem controlada, pois a linguagem apresentada neste trabalho serviu apenas para validação da técnica lingüística proposta, por não ser a mesma objeto desta pesquisa, sendo, entretanto, provada a sua exeqüibilidade em restringir as estruturas sintagmáticas existentes na base de conhecimento sintática a fim de facilitar o processamento lingüístico das sentenças;
- especificação detalhada de uma gramática de estrutura sintagmática para o Português, incluindo a automação da construção de árvores sintáticas por meio de técnicas de inteligência artificial;
- aplicação de ontologia de domínio, a fim de que seja definido um vocabulário específico para cada domínio, bem como o reuso de especificações em domínios semelhantes;
- estudo de uma técnica lingüística que contemple a dimensão funcional ou dinâmica do diagrama de classe, i.e., definição dos métodos;
- integração do *ASIEEOOD* com o TABA;

• verificação da adoção de outro sistema de gramática, e.g., gramática *L-grammar* e gramática de restrição, considerando a independência da gramática;

• construção de um analisador morfológico para identificação de neologismos, busca de palavras sinônimas e a identificação da raiz desejada para cada vocábulo a fim de que a mesma seja associada a um elemento orientado a objeto;

• construção de uma interface inteligente de modo que as sentenças de um determinado discurso possam ser analisadas, de forma automática, quanto à correção de sua estrutura sintagmática e correções sensíveis ao contexto possam ser propostas;

• aplicação da técnica lingüística proposta na elicitação de requisitos em *casos de uso, entrevistas* e *questionários;*

• interface do editor gráfico com outras ferramentas *case*;

• implementação da atividade de *verificação* do processo da engenharia de requisitos, tal como o parafraseamento dos modelos conceituais gerados;

• aplicação, em complementação à técnica lingüística proposta, de técnica probabilística aplicada ao processamento de linguagem natural.

Referências Bibliográficas

ABBOTT, R., 1983, "Program Design by Informal English Description, *Communication ACM 16*, pp. 882-894.

ALBIN, T., 2003, *The Art of Software Architecture*. Indianapolis, Wiley Publishing, Inc.

ALLEN, James F., FRISCH Alan M., 1982, "What's in a semantic network?" In: *Proceedings of the 20th conference on Association for Computational Linguistics* (June), pp. 19-27.

ALUÍSIO, S.M., AIRES, R.V., 2000, "Etiquetação de um Corpus e Construção de um Etiquetador de Português", Relatórios Técnicos do ICMC-USP, 107 (NILC-TR-00-2).

AMBLER, S., 2004, *The Object Primer: Agile Model-Driven Development with UML 2.* 3rd ed. New York, Cambridge University Press.

ANALYTI, Anastasia, SPYRATOS, Nicolas, CONSTANTOPOULOS, Panos, 1998, "On The Semantics of a Semantic Network", *Fundamenta Informaticae,* v. 36, ed. 2-3 (Nov), pp. 109-144.

AZEREDO, José Carlos, 2003, *Iniciação à Sintaxe do Português.* 9ª ed. Brasil, Jorge Zahar Editor.

BICK, Eckhard, 2000, *The Parsing System "Palavras"*. Denmark, Aarthus University Press.

BELL, D., 2004, "UML Basics: The Component Diagram", *IBM Global Service,* IBM, (Dec).

BLOCK, Carl, MCMILLAN, M., MARTIN, J., MONARCHI, D., 1990, "A Prototype System for Extracting Objects from Software Specifications". In: *Proceedings of the 1990 ACM SIGBDP Conference on Trends and directions in expert systems*, (Oct/Nov), pp. 367-376.

BRACHMAN, Ronald J., LEVESQUE, Hector J., 2004, *Knowledge Representation and Reasoning*. San Francisco, Morgan Kaufmann Publishers.

BRATKO, Ivan, 2001, *PROLOG – Programming for Artificial Intelligence*. 3rd ed. Boston, Addison-Wesley Publishers.

BURG, J. F. M., VAN DE RIET, R. P., 1995a, "Color-X: Object Modeling Profits from Linguistic". In: *Very Large Knowledge Bases: Knowledge* Building & *Knowledge Sharing (KB&KS`95),* The Netherlands, IOS Press, Amsterdam, pp. 204-214.

BURG, J. F. M., VAN DE RIET, R. P., 1995b, "The Impact of Linguistics on Conceptual Models: Consistency and Understandability". In: *Proceedings of the First International Workshop on Applications of Natural Language to Databases (NLDB'95)*, Versailles, France, pp. 183-197.

BURG, J. F. M., VAN DE RIET, R. P., 1996, "Analyzing Informal Requirements Specifications: A first step towards conceptual modeling". *Applications of Natural Language to Information Systems.* In: *Proceedings of the Second International Workshop (NLDB`96)*. Netherlands, IOS Press, Amsterdam, pp. 15-27.

CARNIE, Andrew, 2002, *Syntax A Generative Introduction.* 2nd ed. Oxford, Blackwell Publishing.

CHEN, P., 1983, "English Sentence and Entity-Relationship Diagrams"*, Information Systems,* pp. 127-149.

CLEMENTS, P., 2005, "Comparing the SEI's views and beyond approach for documenting software architecture with ANSI-IEEE 1471-2000", Technical Note CMU/SEI-2005-TN-017, (Jul).

CLOCKSIN, William F., MELLISH, Christopher, 2003, *Programming in Prolog*. 5th ed. Germany, Spring-Verlag.

CHOMSKY, Noam, 2002, *Syntactic Structures*. 2nd ed. Berlin, Mouton de Gruyter.

CHOWDHURY, Gobinda, 2003, "Natural Language Processing", *Annual Review of Information Technology*, v. 37, pp. 51-59.

DELIYANNI, Amaryllis, KOWALSKI, Robert A., 1979, "Logic and semantic networks"*, Communications of the ACM*, v. 22, issue 3, (Mar), pp. 184-192.

DUKE, Roger, ROSE, Gordon, 2000, *Formal Object-Oriented Especification using Object-Z*, Cornerstones of Coputing, MacMillan Press Limited.

EVANS, Andy S., 1998, "Reasoning with UML Class Diagrams". In: *Proceedings of the Second IEEE Workshop on Industrial Strength Formal Specification Techniques*, (Oct), p. 102.

FALBO, Ricardo A., 1998, *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software,* Tese de Doutorado, COPPE, UFRJ.

FRANCE, Robert B., 1999, "A Problem-Oriented Analysis of Basic UML Static Requirements Modeling Concepts", *ACM SIGPLAN Notices,* v. 34, issue 10 (Oct), pp. 57-69.

FUCHS, N. E., SCHWERTEL, U., TORGE, S., 2000, "A Natural Language Front-End to Model Generation", *Journal of Language and Computation*, v. 1, n. 2, pp. 199-214.

GAL, A., Lapalme, G., SAINT-DIZIER, P., SOMMERS, H, 1991, *Prolog for Natural Language Processing*. England, John Wiley & Sons.

GAMUT, L. T. F., 1991, *Logic, Language and Meaning: Intensional Logic and Logic Grammar Vol 2*. Chicago, The University of Chicago Press.

GARLAN D., 2000, "Software architecture: a roadmap". In: *IEEE-CS Procedure of the Conference on the Future of Software Engineering*, ACM Press, pp.91-100.

GRISHMAN, Ralph, 1999, *Computational Linguistic: an Introduction*. Cambridge University Press.

HAREL, David, RUMPE, Bernhard, 2000, "Modeling Languages: Syntax, Semantics and all That Stuff. Part I: The Basic Stuff", *Technical Report MCS00-16, Mathematics & Computer Science, Weizmann Institute of Science*.

HAUSSER, R., 2001, *Foundations of Computational Linguistic.* Berlin, Springer-Verlag.

HICKEY, Ann M., DAVIS, Alan M., 2002, "Requirements Elicitation Technique Selection: A Model for Knowledge-Intensive Software Development Processes". In: *Proceedings of the 36th Hawaii International Conference on System Sciences*, *IEEE*, p.96a.

HILBURN, Thomas B., HIRMANPOUR, Iraj, KHAJENOORI, Soheil, TURNER, Richard, QASEN, Abir, 1999, "A Software Engineering Body of Knowledge Version 1.0", *Technical Report CMU/SEI-99-TR-004*.

HOFFMAN, H. F., LEHNER, F., 2001, "Requirements Engineering as a Success Factor in Software Projects", v. 18, issue 4, (Jul/Aug), pp. 58-66.

HOPPENBROUWERS, J., VAN DER VOS, B., HOPPENBROUWERS, S., 1997, "NL structures and conceptual modeling: Grammalizing for KISS", *Data & Knowledge Engineering,* v. 23, n. 1 (Jun), pp. 79-92.

INFANTE, Ulisses, 2001, *Curso de Gramática Aplicada aos Textos.* Brasil, Editora Scipione.

JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH, James, 2000, *El Proceso Unificado de Desarrollo de Software*. Madrid, Pearson Educación S.A..

JURISTO, Natalia, MORANT, J. L., MORENO, Ana Maria, 1999, "A Formal Approach for Generating OO Specifications From Natural Language"*, Journal of Systems and Software*, v.48, n. 2, (Oct) , pp. 139-153.

JURISTO, Natalia, MORENO, Ana Maria, 2000, "How to Use Linguistic Instruments for Object-Oriented Analysis"*, IEEE Software*, May/June, pp. 80-89.

KAMP, Hans, REYLE, Uwe, 1993, *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Netherlands, Kluwer Academic Publishers.

KOF, Leonid, 2004, "Natural Language Processing for Requirements Engineering: Applicability to Large Requirements Documents", *Fakultat fur Informatik, Technische Universitat Munchen*, (Aug).

LARMAN, Craig, 2004, *Utilizando UML e Padrões*. Brasil, Ed. Bookman.

LI, Ke, DEWAR, R. G., POOLEY, R. J., 2004, "Requirements Capture in Natural Problem Statements", *Department of Computer Science, Heriot-Watt University*, Edimburgh, UK, www.macs.hw.ac.uk (Oct).

LUISA, M., MARIANGELA, F., PIERLUIGI, N., 2003, "Market research for requeriments analysis using linguistic tools", *Springer-Verlag London Limited*.

MAKSIMCHUK, Robert A., NAIBURG, 2005, *UML for Mere Mortals*. Addison-Wesley Professional.

MORENO, Ana Maria, JURISTO, Natalia, VAN de RIET, R. P., 2000, "Formal Justification in Object-Oriented Modelling: A Linguistic Approach*", Data & Knowledge Engineering* v. 33, n. 1 (Apr), pp. 25-47.

MORGAN, Richard, LOLITA Group, 1995, "Description of the LOLITA System as Used in MUC-6". In: *Proceedings of the 6th Conference on Message Understanding MUC6´95*, pp. 71-85.

NEWELL, Allen, 1981, "The Knowledge Level", *Artificial Intelligence Magazine*, (Summer).

NUSEIBEH, Bashar, EASTEBROOK, Steve, 2000, "Requirements Engineering: A Roadmap", *ACM, Future of Software Engineering*, pp. 35-46.

OMG-I, 2004, *Object Management Group: UML 2.0 Infrastructure*, OMG document ptc/04-10-14.

OMG-OCL, 2003, Object Management Group: *UML 2.0 OCL Specification Adopted Specification*, OMG document ptc/03-10-14, http://www.omg.org/cgi-bin/doc?ptc/2003-10-14.

OMG-S, 2003, Object Management Group: *UML 2.0 Superestructure – Final Adopted Specification*, OMG document ad/03-08-02, http://www.omg.org/docs/ad/03-08-02.pdf.

OSBORNE, Miles, MACNISH, C. K., 1996, "Processing Natural Language Software Requirement Specifications". In: *Proceedings of the 2nd International Conference on Requirements Engineering*, IEEE, p. 229.

OVERMYER, Scott P. *et al.*, 2001, "Conceptual Modeling through Linguistic Analysis Using LIDA". In: *Proceedings of the 23rd International Conference on Software Engineering*, pp. 401-410.

PARTEE, B.H., MEULEN, A., WALL, R. E., 1990*, Mathematical Methods in Linguistics.* Netherlands, Kluwer Academics Publishers.

PRESSMAN, Roger S., 2002, *Ingeniería de Software: Un Enfoque Práctico*, 5ª edición, Editorial Mc Graw Hill.

ROLLAND, C., PROIX, C., 1992, "A Natural Language Approach for Requirements Engineering". In: *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering,* v. 593 of Lecture Notes in Computer Science, pp. 257-277.

RUMBAUGH, James, BLAHA, Michael, 2005, *Object-Oriented Modeling and Design with UML.* 2nd ed. Pearson Prentice Hall.

RUSSEL, Stuart, NORVIG, Peter, 2004, *Inteligência Artificial*. 2ª ed. Brasil, Editora Campus.

SCHWRITTER, R., FUCHS, N. E., 1996, "Attempto From Specifications in Controlled Natural language towards Executable Specifications", Apresentado em *EMISA Workshop* (May).

SELIC, Bran V., 2004, "On The Semantic Foundations of Standard UML 2.0", LNCS 3185, Springer-Verlag, pp. 181-199.

SHAPIRO, Stuart C., 1978, "Path-Based and Node-Based Inference in Semantic Networks". In: *Proceedings of the Theoretical Issues in natural Language Processing*, (Jul), pp. 219-225.

SHAPIRO, Stuart C., 2000, "An Introduction to SNePS 3". *Bernhard Ganter & Guy W. Mineau, Eds.* Conceptual Structures: Logical, Linguistic, and Computational Issues. *Lecture Notes in Artificial Intelligence 1867,* Springer-Verlag, Berlin, pp. 510-524.

SHAPIRO, Stuart C., The SNePS Implementation Group, 2003, "SNePS 3 User´s Manual", Department of Computer Science and Center of Cognitive Science, (Mar).

SHAPIRO, Stuart C., RAPAPORT, William J., 1990, "The SNePS Family", *Department of Computer Science and Center of Cognitive Science*, (Sep).

SHASTRI, L., 1991, *Why Semantic Networks? Principles of Semantic Networks, Explorations in the Representation of Knowledge*. CA, Morgan Kaufmann Publishers Inc.

SILVA, Alberto T., CARVALHO, L. A., 2006a, "A Knowledge Representation Semantic Network to a Natural Language Syntactic Analyser Based on UML". *In: IFIP - World Computer Congress, 2006, Santiago. Professional Practice in Artificial Intelligence. Boston : Springer Verlag - Boston, v. 218, pp. 237-246.*

SILVA, Alberto T., CARVALHO, L. A., 2006b, "A Linguistic Technique for a Semantic Analyser based on Logical Semantics". In: *Proceedings Of The Second IASTED International Conference on Computational Intelligence, v. 1. pp. 48-53*

SILVA, Alberto T., CARVALHO, L. A., 2006c, "Architecture for an Intelligent Assistant for the Generation of an Object-Oriented Conceptual Model from Natural Language". In: *3er Congresso Internacional de la Region Andina, 2006,* QUITO. IEEE - ANDESCON 2006.

SMITH, M. H., GARIGLIANO, R., MORGAN, 1994, "Generation in the LOLITA System: An Engineering Approach". *Laboratory of Natural Language Engineering, University of Durham.*

SOMMERVILLE, Ian, 2005, "Integrated Requirements Engineering: a Tutorial", *IEEE Software,* (Jan/Feb) , vol. 22, n. 1, pp. 16-23.

SOUZA, João N., 2002, *Lógica para Ciência da Computação*. Brasil, Editora Campus.

SOUZA E SILVA, Maria C. P., KOCH, Ingedore V., 2004, *Lingüística Aplicada ao Português: Sintaxe.* 12ª ed. Brasil, Cortez Editora.

SOWA, John F. *et al.*, 1991, *Principles of Semantic Networks. Explorations in the Representation of Knowledge.* San Mateo, CA, Morgan Kaufmann Publishers, Inc.

SOWA, John F., 2000, *Knowledge Representation. Logical, Philosophical and Computational Foundations*. CA, Brooks/Cole Thomson Learning.

SOWA, John F., 2002a, "Architectures for Intelligent Systems", Special Issue on Artificial Intelligence of the IBM Systems Journal, v. 41, n. 3, pp. 331-349.

SOWA, John F., 2002b, "Semantic Network", http://www.jfsowa.com/pubs/semnet.htm.

SPIVEY, J. Michael, 1998, "The Z Notation: A Reference Manual", J. M. Spivey Oriel College, Oxford.

WARMER, Jos, KLEPPE, A., 2003, *The Object Constraint Language. Getting your Model Ready for MDA*. 2nd ed. Boston, MA, Addison-Wesley.

WEISCHEDEL, Ralph, CARBONELL, Jaime, GROSZ, Barbara, LEHNERT, Wendy, MARCUS, Mitchell, PERRAULT, Raymond e WILENSKY, Robert, 1999, "White Paper on Natural Language Processing". In: *Proceedings of the Workshop On Speech and Natural Language*, ACM, pp. 481-493.

WHOLIN, Claes, RUNESON, Per, HÖST, Martin, OHLSSON, Magnus C., REGNELL, Björn, WESSLÉN, Anders, 2000, *Experimentation in Software Engineering An Introduction.* Norwell, MA, Kluwer Academic Publisher.

Anexo A

Estudo Empírico

Contexto

O presente trabalho inclui uma experimentação do tipo *estudo de caso* para a tese de doutorado intitulada "*Assistente Inteligente para Geração de Modelo Conceitual Orientado a Objeto"*.

O referido assistente é uma ferramenta semi-automatizada que implementa uma técnica lingüística, com suporte baseado em conhecimento, cujo objetivo é apoiar o processo de Engenharia de Requisitos, particularmente nas fases de elicitação, modelagem e validação. A partir de documentos escritos em linguagem natural, o analisador semântico proposto permite a elicitação de elementos orientados a objeto, i.e., classes, atributos, associações e multiplicidades, sendo a modelagem automatizada a partir dos referidos conceitos; a validação ocorre por inspeção visual em um editor gráfico. A notação do modelo a ser gerado é a definida pelo metamodelo diagrama de classe da *Unified Modeling Language* (UML).

As técnicas de elicitação mais comuns incluem questionários, entrevistas, *use case, brainstorming*, entre outros, sendo esta última a selecionada para avaliação da técnica lingüística proposta.

Linguagem Controlada

O processamento de linguagem natural (PLN) tem como destacada restrição, entre outras, as infinitas possibilidades de estruturas sintáticas existentes em uma determinada língua, cabendo, de acordo com o objetivo do sistema PLN especificado, a definição de uma linguagem controlada a fim de restringir as possibilidades sintáticas e otimizar o respectivo processamento.

Objetivo

A presente experimentação tem como objetivo avaliar a linguagem controlada definida para o assistente inteligente proposto por meio de experimentações do tipo *estudo de caso*, bem como permitir a definição de novas regras.

Regras da Linguagem Controlada Proposta

A seguir estão listadas as regras da linguagem controlada proposta:

Regras	Exemplos
Sentença declarativa afirmativa composta por sujeito e predicado	O médico (sujeito) atende o paciente (predicado). O paciente foi atendido pelo médico? (não usar frase interrogativa) O paciente não foi atendido pelo médico. (não usar frase negativa)
Não utilizar nomes próprios	João atende o paciente.
Não usar anáfora ou pronominalização	mim, ti, ...
Pronomes nominativos, acusativo, anafórico	Para ele também, ... Aquele (a quem nos referimos) comprou... Você também vai...
Não usar sujeito indeterminado	alguém ..., precisa-se de ...
Não usar pronome reflexivo	A menina enfeitava-se ... .para si.
Não usar regionalismos nem coloquialismos	Inté, p'ra
Não usar pronomes pessoais oblíquos átonos	me, te, se, o, a, lhe, nos, vos, se, os, as, lhes
Usar abreviações substantivadas	CREA (e.g., como atributo de uma classe engenheiro)
Uma relação deve estar escrita em ambas as direções a fim de permitir a definição de multiplicidades em ambas as direções	O **médico** trabalha em vários hospitais. O hospital contrata os **médicos** em tempo parcial.
Um atributo sempre é uma característica de uma classe e deve estar sempre após o verbo, i.e., no predicado, em uma frase em forma ativa, estando os conceitos antes do verbo no singular	o médico possui um nome e um endereço
Tempo verbal	Terceira pessoa do presente

Regras	Exemplos
Artigos, numerais ou advérbios podem ser utilizados nas frases para definição das cardinalidades	o médico atende o paciente (1..1) o médico atende os pacientes (1..N) o médico atende vários pacientes o médico atende um ou mais pacientes o médico atende um ou muitos pacientes o médico atende zero ou muitos pacientes

Orientações gerais

- Especificar em linguagem natural os dois estudos de caso descritos a seguir, tendo como limitação sintática as citadas regras;
- Cada frase pode ser composta por mais de uma oração;
- A especificação de cada estudo de caso deve ser redigida de forma a descrever os elementos orientados a objeto contidos nos documentos apresentados, i.e., as classes, os atributos, as associações e as multiplicidades;
- Ambos os estudos de caso poderão estar em um mesmo arquivo "txt" e as sugestões de restrições, sempre bem vindas, incluídas no final do mesmo arquivo;
- Os textos deverão ser enviados para o endereço atsfc@cos.ufrj.br, em um arquivo identificado da seguinte forma: universidade_primeiroNome_grupo.txt, e.g., ime_Carlos_aluno.txt. Os grupos incluem *prof, aluno, alunoPG* e *outros*.

Estudos de Caso

- 01 – Histórico escolar.
- 02 – Nota fiscal.

Exemplo

Texto editado:

O médico atende os pacientes.

O hospital contrata os médicos.

O paciente é atendido pelos médicos.

O médico tem uma identidade, um nome, um CRM e um endereço.

O paciente tem uma identidade, um nome e um endereço.

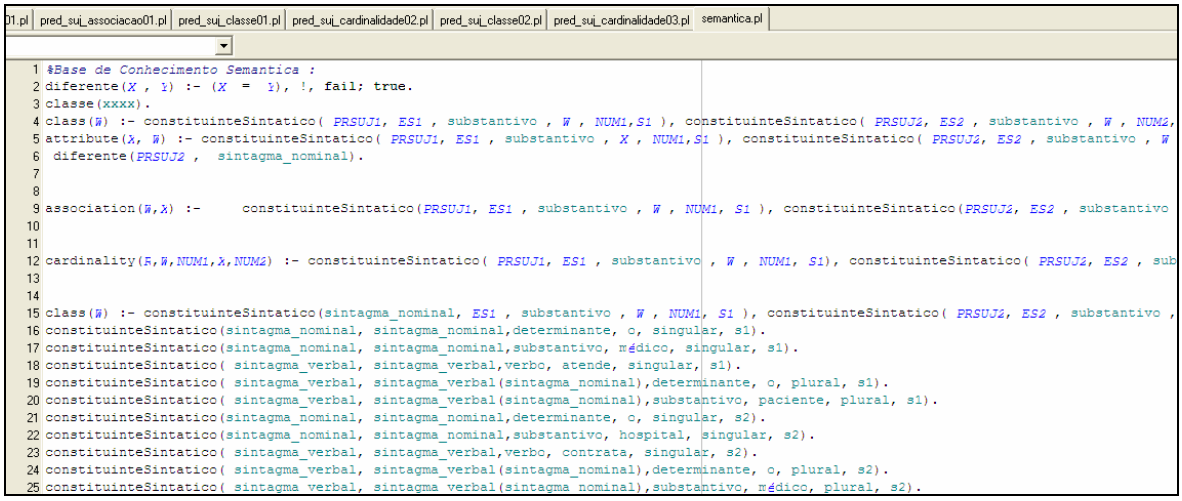
O hospital contrata os enfermeiros.

O enfermeiro tem uma identidade, um nome e um endereço.

O hospital possui um CNPJ e um endereço.

Um médico trabalha em vários hospitais.

Vista parcial da base de conhecimento gerada:



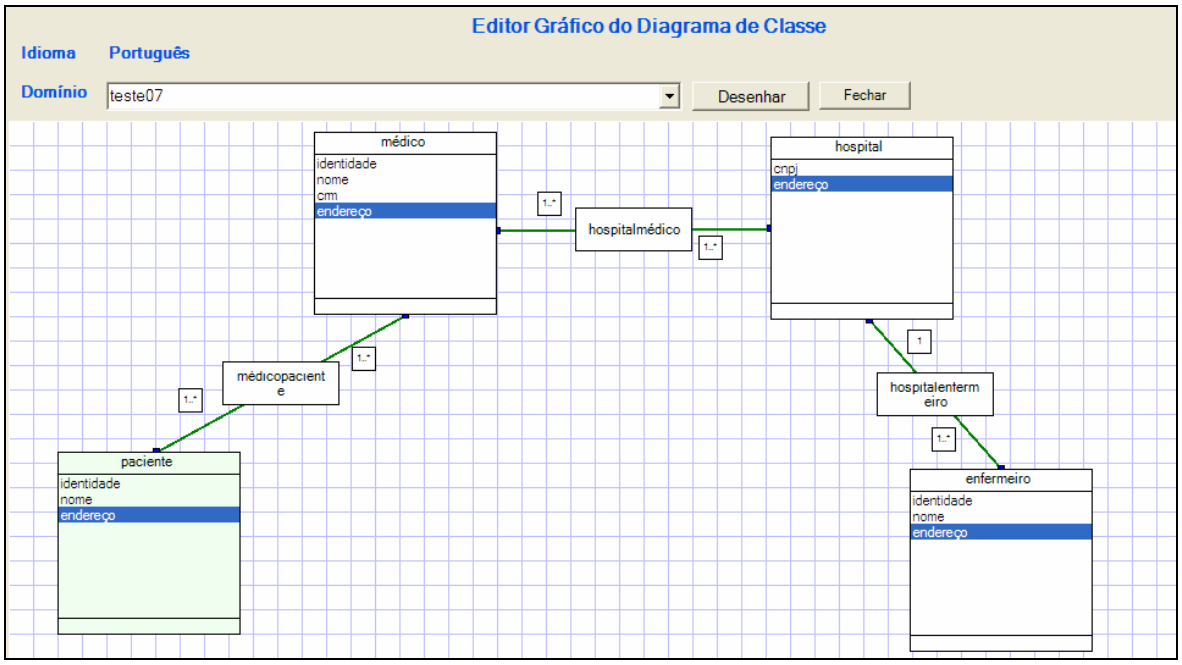
```
%Base de Conhecimento Semantica :
diferente(X , Y) :- (X = Y), !, fail; true.
classe(xxxx).
class(W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1,S1 ), constituinteSintatico( PRSUJ2, ES2 , substantivo , W , NUM2,
attribute(X, W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , X , NUM1,S1 ), constituinteSintatico( PRSUJ2, ES2 , substantivo , W
 diferente(PRSUJ2 ,  sintagma_nominal).


association(W,X) :-     constituinteSintatico(PRSUJ1, ES1 , substantivo , W , NUM1, S1 ), constituinteSintatico(PRSUJ2, ES2 , substantivo


cardinality(R,W,NUM1,X,NUM2) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1, S1), constituinteSintatico( PRSUJ2, ES2 , sub


class(W) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1 ), constituinteSintatico( PRSUJ2, ES2 , substantivo ,
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, o, singular, s1).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, médico, singular, s1).
constituinteSintatico( sintagma_verbal, sintagma_verbal,verbo, atende, singular, s1).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, o, plural, s1).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, paciente, plural, s1).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, o, singular, s2).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, hospital, singular, s2).
constituinteSintatico( sintagma_verbal, sintagma_verbal,verbo, contrata, singular, s2).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, o, plural, s2).
constituinteSintatico( sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, médico, plural, s2).
```

Diagrama de classe gerado (modelo conceitual)

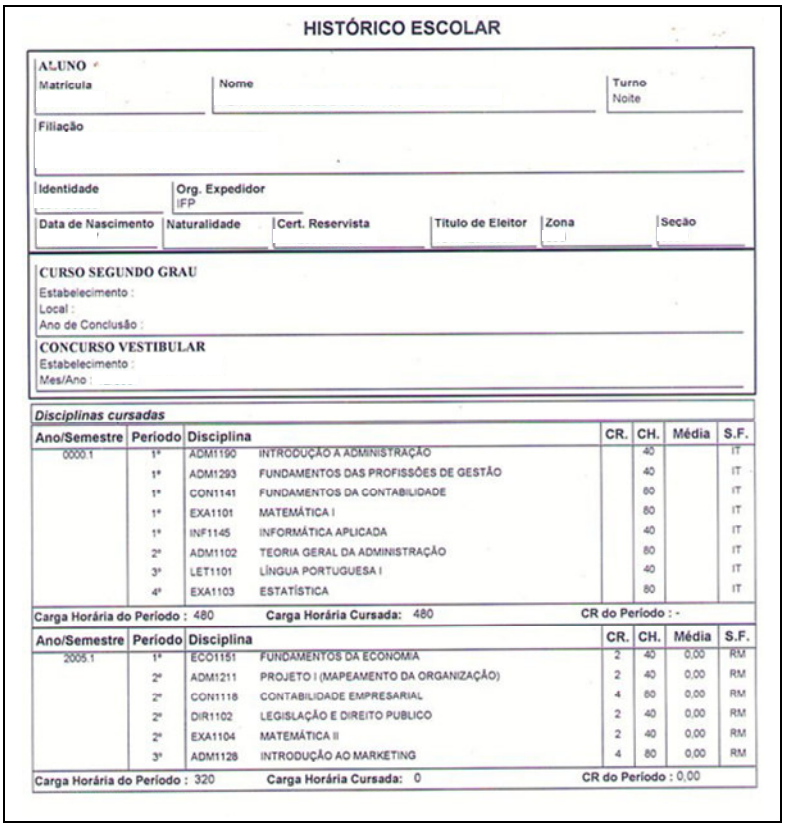


Desde já, meus agradecimentos em participar desta experimentação!

Alberto Tavares da Silva

atsfc@cos.ufrj.br

As figuras A1, A2, A3 e A4 ilustram os estudos de caso distribuídos no estudo empírico.



HISTÓRICO ESCOLAR

ALUNO

Matrícula | Nome | Turno: Noite

Filiação

Identidade | Org. Expedidor: IFP

Data de Nascimento | Naturalidade | Cert. Reservista | Título de Eleitor | Zona | Seção

CURSO SEGUNDO GRAU

Estabelecimento :
Local :
Ano de Conclusão :

CONCURSO VESTIBULAR

Estabelecimento :
Mes/Ano :

Disciplinas cursadas

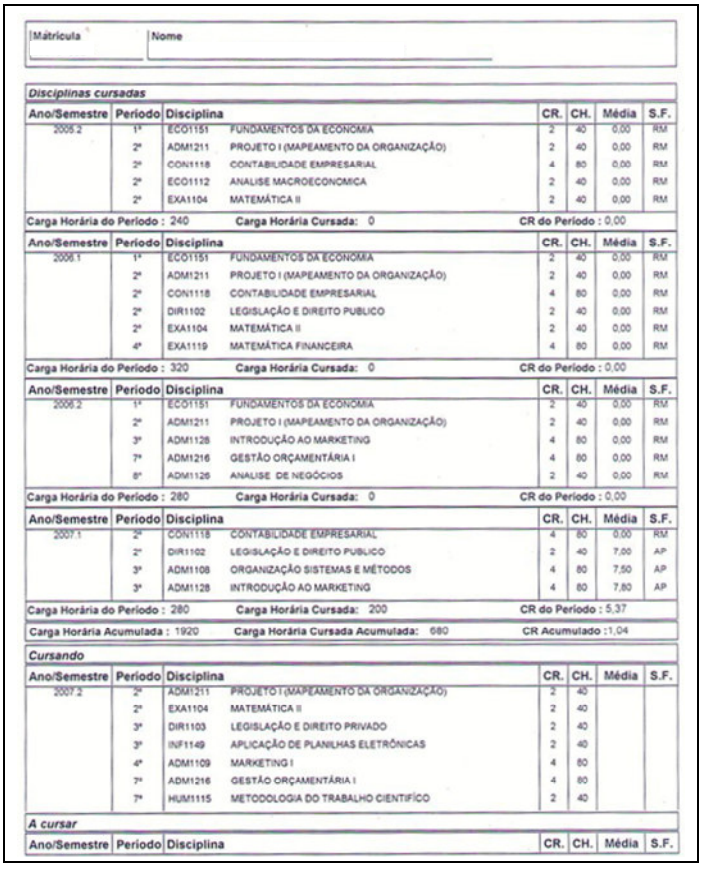
Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
0000.1	1º	ADM1190	INTRODUÇÃO A ADMINISTRAÇÃO		40		IT
	1º	ADM1293	FUNDAMENTOS DAS PROFISSÕES DE GESTÃO		40		IT
	1º	CON1141	FUNDAMENTOS DA CONTABILIDADE		80		IT
	1º	EXA1101	MATEMÁTICA I		80		IT
	1º	INF1145	INFORMÁTICA APLICADA		40		IT
	2º	ADM1102	TEORIA GERAL DA ADMINISTRAÇÃO		80		IT
	3º	LET1101	LÍNGUA PORTUGUESA I		40		IT
	4º	EXA1103	ESTATÍSTICA		80		IT

Carga Horária do Período : 480 Carga Horária Cursada: 480 CR do Período : -

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
2005.1	1º	ECO1151	FUNDAMENTOS DA ECONOMIA	2	40	0,00	RM
	2º	ADM1211	PROJETO I (MAPEAMENTO DA ORGANIZAÇÃO)	2	40	0,00	RM
	2º	CON1118	CONTABILIDADE EMPRESARIAL	4	80	0,00	RM
	2º	DIR1102	LEGISLAÇÃO E DIREITO PUBLICO	2	40	0,00	RM
	2º	EXA1104	MATEMÁTICA II	2	40	0,00	RM
	3º	ADM1128	INTRODUÇÃO AO MARKETING	4	80	0,00	RM

Carga Horária do Período : 320 Carga Horária Cursada: 0 CR do Período : 0,00

Figura A1: Folha 1 do histórico escolar.



Matrícula | Nome

Disciplinas cursadas

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
2005.2	1º	ECO1151	FUNDAMENTOS DA ECONOMIA	2	40	0,00	RM
	2º	ADM1211	PROJETO I (MAPEAMENTO DA ORGANIZAÇÃO)	2	40	0,00	RM
	2º	CON1118	CONTABILIDADE EMPRESARIAL	4	80	0,00	RM
	2º	ECO1112	ANALISE MACROECONOMICA	2	40	0,00	RM
	2º	EXA1104	MATEMÁTICA II	2	40	0,00	RM

Carga Horária do Período : 240 Carga Horária Cursada: 0 CR do Período : 0,00

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
2006.1	1º	ECO1151	FUNDAMENTOS DA ECONOMIA	2	40	0,00	RM
	2º	ADM1211	PROJETO I (MAPEAMENTO DA ORGANIZAÇÃO)	2	40	0,00	RM
	2º	CON1118	CONTABILIDADE EMPRESARIAL	4	80	0,00	RM
	2º	DIR1102	LEGISLAÇÃO E DIREITO PUBLICO	2	40	0,00	RM
	2º	EXA1104	MATEMÁTICA II	2	40	0,00	RM
	4º	EXA1119	MATEMÁTICA FINANCEIRA	4	80	0,00	RM

Carga Horária do Período : 320 Carga Horária Cursada: 0 CR do Período : 0,00

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
2006.2	1º	ECO1151	FUNDAMENTOS DA ECONOMIA	2	40	0,00	RM
	2º	ADM1211	PROJETO I (MAPEAMENTO DA ORGANIZAÇÃO)	2	40	0,00	RM
	3º	ADM1128	INTRODUÇÃO AO MARKETING	4	80	0,00	RM
	7º	ADM1216	GESTÃO ORÇAMENTÁRIA I	4	80	0,00	RM
	8º	ADM1126	ANALISE DE NEGÓCIOS	2	40	0,00	RM

Carga Horária do Período : 280 Carga Horária Cursada: 0 CR do Período : 0,00

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
2007.1	2º	CON1118	CONTABILIDADE EMPRESARIAL	4	80	0,00	RM
	2º	DIR1102	LEGISLAÇÃO E DIREITO PUBLICO	2	40	7,00	AP
	3º	ADM1108	ORGANIZAÇÃO SISTEMAS E MÉTODOS	4	80	7,50	AP
	3º	ADM1128	INTRODUÇÃO AO MARKETING	4	80	7,80	AP

Carga Horária do Período : 280 Carga Horária Cursada: 200 CR do Período : 5,37

Carga Horária Acumulada : 1920 Carga Horária Cursada Acumulada: 680 CR Acumulado :1,04

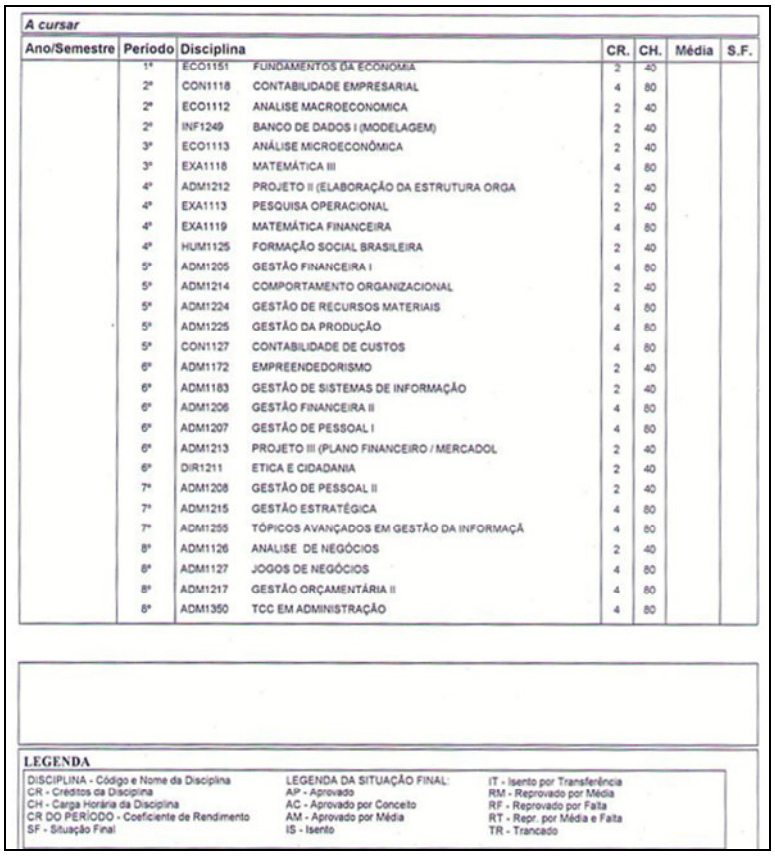
Cursando

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
2007.2	2º	ADM1211	PROJETO I (MAPEAMENTO DA ORGANIZAÇÃO)	2	40		
	2º	EXA1104	MATEMÁTICA II	2	40		
	3º	DIR1103	LEGISLAÇÃO E DIREITO PRIVADO	2	40		
	3º	INF1149	APLICAÇÃO DE PLANILHAS ELETRÔNICAS	2	40		
	4º	ADM1109	MARKETING I	4	80		
	7º	ADM1216	GESTÃO ORÇAMENTÁRIA I	4	80		
	7º	HUM1115	METODOLOGIA DO TRABALHO CIENTIFICO	2	40		

A cursar

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.

Figura A2: Folha 2 do histórico escolar.



A cursar

Ano/Semestre	Período	Disciplina		CR.	CH.	Média	S.F.
	1º	ECO1151	FUNDAMENTOS DA ECONOMIA	2	40		
	2º	CON1118	CONTABILIDADE EMPRESARIAL	4	80		
	2º	ECO1112	ANALISE MACROECONOMICA	2	40		
	2º	INF1249	BANCO DE DADOS I (MODELAGEM)	2	40		
	3º	ECO1113	ANÁLISE MICROECONÔMICA	2	40		
	3º	EXA1118	MATEMÁTICA III	4	80		
	4º	ADM1212	PROJETO II (ELABORAÇÃO DA ESTRUTURA ORGA	2	40		
	4º	EXA1113	PESQUISA OPERACIONAL	2	40		
	4º	EXA1119	MATEMÁTICA FINANCEIRA	4	80		
	4º	HUM1125	FORMAÇÃO SOCIAL BRASILEIRA	2	40		
	5º	ADM1205	GESTÃO FINANCEIRA I	4	80		
	5º	ADM1214	COMPORTAMENTO ORGANIZACIONAL	2	40		
	5º	ADM1224	GESTÃO DE RECURSOS MATERIAIS	4	80		
	5º	ADM1225	GESTÃO DA PRODUÇÃO	4	80		
	5º	CON1127	CONTABILIDADE DE CUSTOS	4	80		
	6º	ADM1172	EMPREENDEDORISMO	2	40		
	6º	ADM1183	GESTÃO DE SISTEMAS DE INFORMAÇÃO	2	40		
	6º	ADM1206	GESTÃO FINANCEIRA II	4	80		
	6º	ADM1207	GESTÃO DE PESSOAL I	4	80		
	6º	ADM1213	PROJETO III (PLANO FINANCEIRO / MERCADOL	2	40		
	6º	DIR1211	ETICA E CIDADANIA	2	40		
	7º	ADM1208	GESTÃO DE PESSOAL II	2	40		
	7º	ADM1215	GESTÃO ESTRATÉGICA	4	80		
	7º	ADM1255	TÓPICOS AVANÇADOS EM GESTÃO DA INFORMAÇÃ	4	80		
	8º	ADM1126	ANALISE DE NEGÓCIOS	2	40		
	8º	ADM1127	JOGOS DE NEGÓCIOS	4	80		
	8º	ADM1217	GESTÃO ORÇAMENTÁRIA II	4	80		
	8º	ADM1350	TCC EM ADMINISTRAÇÃO	4	80		

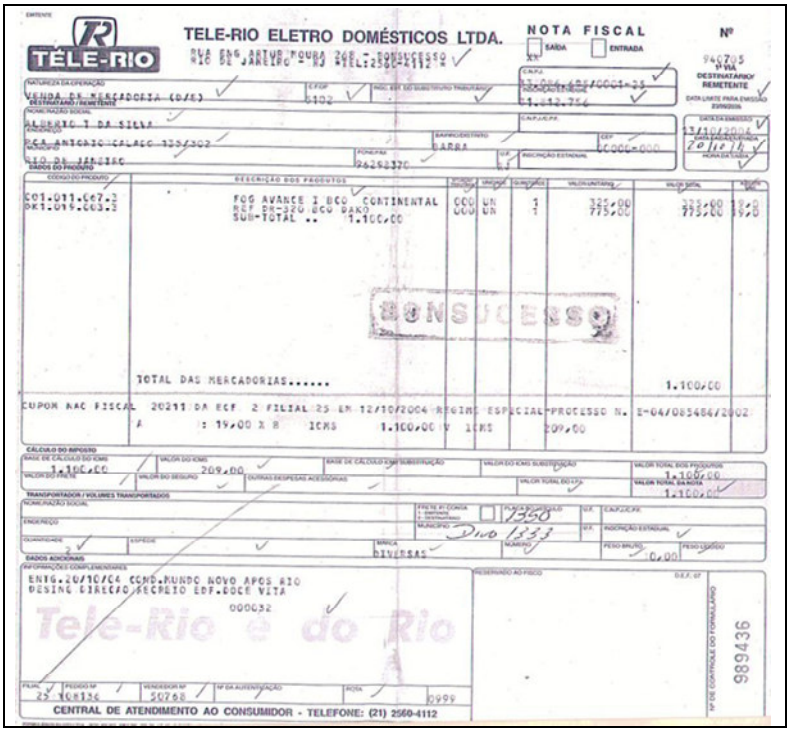
LEGENDA

DISCIPLINA - Código e Nome da Disciplina
CR - Créditos da Disciplina
CH - Carga Horária da Disciplina
CR DO PERÍODO - Coeficiente de Rendimento
SF - Situação Final

LEGENDA DA SITUAÇÃO FINAL:
AP - Aprovado
AC - Aprovado por Conceito
AM - Aprovado por Média
IS - Isento

IT - Isento por Transferência
RM - Reprovado por Média
RF - Reprovado por Falta
RT - Repr. por Média e Falta
TR - Trancado

Figura A3: Folha 3 do histórico escolar.



TELE-RIO ELETRO DOMÉSTICOS LTDA.

NOTA FISCAL

Nº

TELE-RIO

Tele-Rio é do Rio

CENTRAL DE ATENDIMENTO AO CONSUMIDOR - TELEFONE: (21) 2560-4112

Figura A4: Nota fiscal.

Anexo B

Instância de Base de Conhecimento Semântica

```
diferente(X , Y) :- (X  =  Y), !, fail; true.
classe(xxxx).
class(W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W , NUM1,S1 ),
    constituinteSintatico( PRSUJ2, ES2 , substantivo , W , NUM2, S2 ),  diferente( PRSUJ1
    , PRSUJ2), diferente( S1 , S2 ).
attribute(X, W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , X , NUM1,S1 ),
    constituinteSintatico( PRSUJ2, ES2 , substantivo , W , NUM2,S1 ), not(classe(W)),
    classe(X), diferente(W ,  X), diferente(ES1, ES2),
diferente(PRSUJ2 ,  sintagma_nominal).
association(W,X) :-     constituinteSintatico(PRSUJ1, ES1 , substantivo , W , NUM1, S1 ),
    constituinteSintatico(PRSUJ2, ES2 , substantivo , X , NUM2, S1 ), classe(X),
    diferente(ES1,ES2),  classe(W).
class(W) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1 ),
    constituinteSintatico( PRSUJ2, ES2 , substantivo , W1 , NUM2, S1 ),
    diferente(sintagma_nominal , PRSUJ2),diferente( W, W1 ), classe(W1).
inheritance(X, Y, W) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , X , NUM1,S1
    ),constituinteSintatico( PRSUJ2, ES2 , substantivo , Y , NUM1,S2
    ),constituinteSintatico( sintagma_verbal, ES3 , substantivo , W , NUM2,S1 ),
    constituinteSintatico( sintagma_verbal, ES4 , substantivo , W , NUM2,S2 ), atributo(X,
    W), atributo(Y, W), classe(X), classe(Y), diferente(X, Y).
cardinality(R,W,X,NUM1,NUM2) :- constituinteSintatico( PRSUJ1, ES1 , substantivo , W
    , NUM1, S1), constituinteSintatico( PRSUJ2, ES2 , substantivo , X , NUM2, S1 ),
    associacao(R,W,X),  diferente(ES1,ES2), diferente(PRSUJ1,PRSUJ2), classe(X),
    classe(W).
cardinality(R, W, X, MIN, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo
    , W , NUM1, S1), constituinteSintatico(sintagma_verbal, ES2 , substantivo , X , NUM2,
    S1 ),constituinteSintatico(sintagma_verbal, ES3 , numeral , MIN, NUM1,
    S1),constituinteSintatico(sintagma_verbal, ES4 , adverbio , mais, NUM1, S1),
    associacao(R,W,X),  diferente(ES1,ES2), classe(W), classe(X).
cardinality(R, W, X, MIN, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo
    , W , NUM1, S1), constituinteSintatico(sintagma_verbal, ES2 , numeral , MIN, NUM1,
    S1), constituinteSintatico(sintagma_verbal, ES4 , pronome , vário, NUM2, S1),
    constituinteSintatico(sintagma_verbal, ES3 , substantivo , X , NUM2, S1 ),
    associacao(R,W,X),  diferente(ES1,ES2), classe(W), classe(X).
cardinality(R, W, X, 0, MAX) :- constituinteSintatico(sintagma_nominal, ES1 ,
    substantivo , W , NUM1, S1), constituinteSintatico(sintagma_verbal, ES2, adverbio ,
    no_máximo, NUM1, S1), constituinteSintatico(sintagma_verbal, ES3, numeral , MAX,
    NUM1, S1), constituinteSintatico(sintagma_verbal, ES4 , substantivo , X , NUM2, S1 ),
    associacao(R,W,X), diferente(ES1,ES2), classe(W), classe(X).
```

cardinality(R, W, X, 0, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1), constituinteSintatico(sintagma_verbal, ES2, preposicao, em , NUM1, S1), constituinteSintatico(sintagma_verbal, ES3, pronome, vários , NUM2, S1), constituinteSintatico(sintagma_verbal, ES4 , substantivo , X , NUM2, S1), associacao(R,W,X), diferente(ES1,ES3), classe(W), classe(X).

cardinality(R, W, X, 0, 0) :- constituinteSintatico(sintagma_nominal, ES1 , substantivo , W , NUM1, S1), constituinteSintatico(sintagma_verbal, ES2, preposicao, em , NUM1, S1), constituinteSintatico(sintagma_verbal, ES3, pronome, vário , NUM2, S1), constituinteSintatico(sintagma_verbal, ES4 , substantivo , X , NUM2, S1), associacao(R,W,X), diferente(ES1,ES3), classe(W), classe(X).

cardinality(R, W, X, MIN, 0) :- constituinteSintatico(sintagma_nominal, ES1, substantivo, W, NUM1, S1),constituinteSintatico(sintagma_verbal, ES2, numeral, MIN, NUM1, S1),constituinteSintatico(sintagma_verbal, ES3, substantivo, X , NUM2, S1),constituinteSintatico(sintagma_verbal, ES4, pronome, várias, NUM2, S1),associacao(R,W,X), diferente(ES1,ES2), classe(W), classe(X).

constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s1).

constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, nota_fiscal, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, um, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),substantivo, número, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),determinante, uma, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))),substantivo, data_limite_para_emissão, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))),determinante, uma, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))),substantivo, data_de_emissão, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))),determinante, uma, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))))),substantivo, data_saída, singular, s1).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),determinante, uma, singular, s1).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))))),substantivo, data_entrada, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),determinante, uma, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),substantivo, hora_da_saída, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),determinante, um, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),substantivo, cnpj, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),determinante, uma, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),substantivo, inscrição_estadual, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),determinante, uma, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),substantivo, inscrição_estadual, singular, s1).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta

gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))))))))),substantivo, substituto_tributário, singular, s1).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))))))))))),determinante, um, singular, s1).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))))))))))), substantivo, cfop, singular, s1).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s2).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, nota_fiscal, singular, s2).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s2).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, uma, singular, s2).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, operação, singular, s2).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s3).
constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, operação, singular, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associada, singular, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial)),numeral, uma, singular, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial(sintagma_adverbial))),conjuncao, ou, singular, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial(sintagma_adverbial(sintagma_adverbial)))),pronome, vária, plural, s3).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, nota_fiscal, plural, s3).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s4).

constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, operação, singular, s4).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s4).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, um, singular, s4).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, nome, singular, s4).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s5).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, nota_fiscal, singular, s5).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s5).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, um, singular, s5).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, endereço, singular, s5).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s6).
constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, endereço, singular, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, é, singular, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial)),numeral, uma, singular, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial(sintagma_adverbial))),conjuncao, ou, singular, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial(sintagma_adverbial(sintagma_adverbial)))),pronome, , plural, s6).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, nota_fiscal, plural, s6).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s7).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, endereço, singular, s7).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, é, singular, s7).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, composto, singular, s7).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, por, singular, s7).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, uma, singular, s7).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo, rua,
singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal))),det
erminante, um, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal))),sub
stantivo, número, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal)))),determinante, um, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal)))),substantivo, complemento, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal))))),determinante, um, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal))))),substantivo, bairro, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))),determinante, uma, singular,
s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))),substantivo, cidade, singular,
s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))),determinan
te, uma, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomin
al)))))))),conjuncao, e, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomin
al(sintagma_nominal)))))))),determinante, um, singular, s7).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomin
al(sintagma_nominal)))))))),substantivo, telefone, singular, s7).

constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s8).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, nota_fiscal, singular, s8).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, é, singular, s8).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, gerada, singular, s8).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, para, singular, s8).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, um, singular, s8).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo, destinatário_remetente, singular, s8).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s9).
constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, destinatário_remetente, singular, s9).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, recebe, singular, s9).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, uma, singular, s9).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),conjuncao, ou, singular, s9).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),pronome, vária, plural, s9).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))),substantivo, nota_fiscal, plural, s9).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s10).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, destinatário_remetente, singular, s10).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s10).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, endereço, singular, s10).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s11).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, endereço, singular, s11).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, é, singular, s11).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s11).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s11).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, um, singular, s11).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo,
destinatário_remetente, singular, s11).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular,
s12).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo,
destinatário_remetente, singular, s12).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s12).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo,
nome_razão_social, singular, s12).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal)),conjuncao, e, singular, s12).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),substantiv
o, inscrição_estadual, singular, s12).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular,
s13).
constituinteSintatico(sintagma_nominal,
sintagma_nominal(sintagma_nominal),substantivo, nota_fiscal, singular, s13).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s13).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante,
um, singular, s13).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal)),conjuncao, ou, singular, s13).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),pronome,
vário, plural, s13).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal)))),substantivo, item_produto, plural, s13).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular,
s14).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, item_produto,
singular, s14).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, é, singular, s14).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, vendido,
singular, s14).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado),preposicao, em, singular, s14).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, uma,
singular, s14).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo,
nota_fiscal, singular, s14).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular,
s15).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, item_produto,
singular, s15).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s15).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s15).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s15).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, um, singular, s15).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo, produto, singular, s15).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s16).
constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, produto, singular, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial)),numeral, zero, singular, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial(sintagma_adverbial))),conjuncao, ou, singular, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_adverbial(sintagma_adverbial(sintagma_adverbial)))),pronome, vário, plural, s16).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, item_produto, plural, s16).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s17).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, item_produto, singular, s17).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s17).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, uma, singular, s17).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, quantidade, singular, s17).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),determinante, uma, singular, s17).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),substantivo, situação_tributária, singular, s17).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),conjuncao, e, singular, s17).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal)))),determinante, uma, singular, s17).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal)))),substantivo, alíquota_de_icms, singular, s17).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular,
s18).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, produto, singular,
s18).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s18).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante,
um, singular, s18).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo,
código, singular, s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal)),determinante, uma, singular,
s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal)),substantivo, descrição,
singular, s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),determina
nte, uma, singular, s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),substantiv
o, unidade, singular, s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal)))),conjuncao, e, singular, s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal))))),determinante, um, singular, s18).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal))))),substantivo, valor_unitário, singular, s18).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular,
s19).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, nota_fiscal,
singular, s19).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s19).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante,
um, singular, s19).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo,
cálculo_do_imposto, singular, s19).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular,
s20).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo,
cálculo_do_imposto, singular, s20).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s20).

constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s20).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s20).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, uma, singular, s20).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo, nota_fiscal, singular, s20).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s21).
constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, cálculo_do_imposto, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, uma, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),substantivo, base_de_cálculo_icms, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),determinante, um, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))),substantivo, valor_do_icms, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))),determinante, uma, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))),substantivo, base_icms_substituição, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))),determinante, um, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))),substantivo, valor_icms_substituição, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))),determinante, um, singular, s21).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))),substantivo, valor_total, singular, s21).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal))))))))))),determinante, um,
singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))))),substa
ntivo, valor_do_frete, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal))))))))))))),determinante, um, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal)))))))))))))),substantivo, valor_do_seguro, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal))))))))))))))),determinante, uma, singular,
s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))))))))),substantivo,
outras_despesas_acessórias, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))
)))))))),determinante, um, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal)))))))))))))))))),substantivo, valor_total, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal))))))))))))))))))),substantivo, ipi, singular, s21).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal))))))))))))))))))),conjuncao, e,
singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))))))))))))),
determinante, um, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomin
al))))))))))))))))))),substantivo, valor_total, singular, s21).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomin
al(sintagma_nominal))))))))))))))))))),substantivo, nota, singular, s21).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular,
s22).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, nota_fiscal,
singular, s22).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, tem, singular, s22).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo,
transportador_volumes_transportados, singular, s22).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular,
s23).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo,
transportador_volumes_transportados, singular, s23).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s23).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo,
associado, singular, s23).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s23).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, uma,
singular, s23).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo,
nota_fiscal, singular, s23).

constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s24).
constituinteSintatico(sintagma_nominal, sintagma_nominal(sintagma_nominal),substantivo, transportador_volumes_transportados, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, um, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),substantivo, nome, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),determinante, uma, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),substantivo, razão_social, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),substantivo, um_frete_para_conta, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),determinante, uma, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),substantivo, placa_do_veículo, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))),determinante, um, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))),substantivo, cnpj_cpf, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))),determinante, uma, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))),substantivo, quantidade, singular, s24).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))))))),determinante, uma, singular, s24).

constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal))))))))))))),substantivo, espécie, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal)))))))))))))),determinante, uma, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal)))))))))))))),substantivo, marca, singular,
s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))))))),determinante
, um, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))))))))
))))))),substantivo, número, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal)))))))))))))))),determinante, um, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal)))))))))))))))),substantivo, peso_bruto, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta
gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nomi
nal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintag
ma_nominal(sintagma_nominal(sintagma_nominal)))))))))))))))),determinante, um,
singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_no
minal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sinta

gma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))))))))))))))))))),
substantivo, peso_líquido, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal))))))))))))))))))))),conjuncao, e, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal)))))))))))))))))))))),determinante, uma, singular, s24).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_nominal))))))))))))))))))))))),substantivo,
inscrição_estadual, singular, s24).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s25).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, transportador_volumes_transportados, singular, s25).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s25).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),determinante, um, singular, s25).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, endereço, singular, s25).
constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, um, singular, s26).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, endereço, singular, s26).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s26).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s26).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s26).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, um, singular, s26).
constituinteSintatico(sintagma_verbal,
sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo,
transportador_volumes_transportados, singular, s26).

constituinteSintatico(sintagma_nominal, sintagma_nominal,determinante, uma, singular, s27).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, nota_fiscal, singular, s27).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, possui, singular, s27).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, dado_adicional, singular, s27).
,substantivo, dado_adicional, singular, s28).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo_aux, está, singular, s28).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_verbal),verbo, associado, singular, s28).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado),preposicao, a, singular, s28).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),determinante, uma, singular, s28).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_preposicionado(sintagma_nominal)),substantivo, nota_fiscal, singular, s28).
constituinteSintatico(sintagma_nominal, sintagma_nominal,substantivo, dado_adicional, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal,verbo, inclui, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal),substantivo, informação_complementar, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal)),substantivo, filial, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal))),substantivo, pedido_número, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal)))),substantivo, vendedor_número, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))),substantivo, número_de_autenticação, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))),conjuncao, e, singular, s29).
constituinteSintatico(sintagma_verbal, sintagma_verbal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal(sintagma_nominal))))),substantivo, rota, singular, s29).
classe(nota_fiscal).
classe(operação).
classe(endereço).
classe(destinatário_remetente).
classe(item_produto).
classe(produto).
classe(cálculo_do_imposto).
classe(transportador_volumes_transportados).

classe(dado_adicional).
atributo(nota_fiscal, número).
atributo(nota_fiscal, data_limite_para_emissão).
atributo(nota_fiscal, data_de_emissão).
atributo(nota_fiscal, data_saída).
atributo(nota_fiscal, data_entrada).
atributo(nota_fiscal, hora_da_saída).
atributo(nota_fiscal, cnpj).
atributo(nota_fiscal, inscrição_estadual).
atributo(nota_fiscal, substituto_tributário).
atributo(nota_fiscal, cfop).
atributo(operação, nome).
atributo(endereço, rua).
atributo(endereço, número).
atributo(endereço, complemento).
atributo(endereço, bairro).
atributo(endereço, cidade).
atributo(endereço, uf).
atributo(endereço, telefone).
atributo(destinatário_remetente, nome_razão_social).
atributo(destinatário_remetente, inscrição_estadual).
atributo(item_produto, quantidade).
atributo(item_produto, situação_tributária).
atributo(item_produto, alíquota_de_icms).
atributo(produto, código).
atributo(produto, descrição).
atributo(produto, unidade).
atributo(produto, valor_unitário).
atributo(cálculo_do_imposto, base_de_cálculo_icms).
atributo(cálculo_do_imposto, valor_do_icms).
atributo(cálculo_do_imposto, base_icms_substituição).
atributo(cálculo_do_imposto, valor_icms_substituição).
atributo(cálculo_do_imposto, valor_total).
atributo(cálculo_do_imposto, valor_do_frete).
atributo(cálculo_do_imposto, valor_do_seguro).
atributo(cálculo_do_imposto, outras_despesas_acessórias).
atributo(cálculo_do_imposto, ipi).
atributo(cálculo_do_imposto, nota).
atributo(transportador_volumes_transportados, nome).
atributo(transportador_volumes_transportados, razão_social).
atributo(transportador_volumes_transportados, um_frete_para_conta).
atributo(transportador_volumes_transportados, placa_do_veículo).
atributo(transportador_volumes_transportados, cnpj_cpf).
atributo(transportador_volumes_transportados, quantidade).
atributo(transportador_volumes_transportados, espécie).
atributo(transportador_volumes_transportados, marca).
atributo(transportador_volumes_transportados, número).
atributo(transportador_volumes_transportados, peso_bruto).
atributo(transportador_volumes_transportados, peso_líquido).
atributo(transportador_volumes_transportados, inscrição_estadual).

atributo(dado_adicional, informação_complementar).
atributo(dado_adicional, filial).
atributo(dado_adicional, pedido_número).
atributo(dado_adicional, vendedor_número).
atributo(dado_adicional, número_de_autenticação).
atributo(dado_adicional, rota).
atributo(nota_fiscal, número).
atributo(nota_fiscal, data_limite_para_emissão).
atributo(nota_fiscal, data_de_emissão).
atributo(nota_fiscal, data_saída).
atributo(nota_fiscal, data_entrada).
atributo(nota_fiscal, hora_da_saída).
atributo(nota_fiscal, cnpj).
atributo(nota_fiscal, inscrição_estadual).
atributo(nota_fiscal, substituto_tributário).
atributo(nota_fiscal, cfop).
atributo(operação, nome).
atributo(endereço, rua).
atributo(endereço, número).
atributo(endereço, complemento).
atributo(endereço, bairro).
atributo(endereço, cidade).
atributo(endereço, uf).
atributo(endereço, telefone).
atributo(destinatário_remetente, nome_razão_social).
atributo(destinatário_remetente, inscrição_estadual).
atributo(item_produto, quantidade).
atributo(item_produto, situação_tributária).
atributo(item_produto, alíquota_de_icms).
atributo(produto, código).
atributo(produto, descrição).
atributo(produto, unidade).
atributo(produto, valor_unitário).
atributo(cálculo_do_imposto, base_de_cálculo_icms).
atributo(cálculo_do_imposto, valor_do_icms).
atributo(cálculo_do_imposto, base_icms_substituição).
atributo(cálculo_do_imposto, valor_icms_substituição).
atributo(cálculo_do_imposto, valor_total).
atributo(cálculo_do_imposto, valor_do_frete).
atributo(cálculo_do_imposto, valor_do_seguro).
atributo(cálculo_do_imposto, outras_despesas_acessórias).
atributo(cálculo_do_imposto, ipi).
atributo(cálculo_do_imposto, nota).
atributo(transportador_volumes_transportados, nome).
atributo(transportador_volumes_transportados, razão_social).
atributo(transportador_volumes_transportados, um_frete_para_conta).
atributo(transportador_volumes_transportados, placa_do_veículo).
atributo(transportador_volumes_transportados, cnpj_cpf).
atributo(transportador_volumes_transportados, quantidade).
atributo(transportador_volumes_transportados, espécie).

atributo(transportador_volumes_transportados, marca).
atributo(transportador_volumes_transportados, número).
atributo(transportador_volumes_transportados, peso_bruto).
atributo(transportador_volumes_transportados, peso_líquido).
atributo(transportador_volumes_transportados, inscrição_estadual).
atributo(dado_adicional, informação_complementar).
atributo(dado_adicional, filial).
atributo(dado_adicional, pedido_número).
atributo(dado_adicional, vendedor_número).
atributo(dado_adicional, número_de_autenticação).
atributo(dado_adicional, rota).
associacao(nota_fiscal/operação, nota_fiscal, operação).
associacao(operação/nota_fiscal, operação, nota_fiscal).
associacao(nota_fiscal/endereço, nota_fiscal, endereço).
associacao(endereço/nota_fiscal, endereço, nota_fiscal).
associacao(nota_fiscal/destinatário_remetente, nota_fiscal, destinatário_remetente).
associacao(destinatário_remetente/nota_fiscal, destinatário_remetente, nota_fiscal).
associacao(destinatário_remetente/endereço, destinatário_remetente, endereço).
associacao(endereço/destinatário_remetente, endereço, destinatário_remetente).
associacao(nota_fiscal/item_produto, nota_fiscal, item_produto).
associacao(item_produto/nota_fiscal, item_produto, nota_fiscal).
associacao(item_produto/produto, item_produto, produto).
associacao(produto/item_produto, produto, item_produto).
associacao(nota_fiscal/cálculo_do_imposto, nota_fiscal, cálculo_do_imposto).
associacao(cálculo_do_imposto/nota_fiscal, cálculo_do_imposto, nota_fiscal).
associacao(nota_fiscal/transportador_volumes_transportados, nota_fiscal,
    transportador_volumes_transportados).
associacao(transportador_volumes_transportados/nota_fiscal,
    transportador_volumes_transportados, nota_fiscal).
associacao(transportador_volumes_transportados/endereço,
    transportador_volumes_transportados, endereço).
associacao(endereço/transportador_volumes_transportados, endereço,
    transportador_volumes_transportados).
associacao(nota_fiscal/dado_adicional, nota_fiscal, dado_adicional).
associacao(dado_adicional/nota_fiscal, dado_adicional, nota_fiscal).
multiplicidade(cálculo_do_imposto/nota_fiscal, cálculo_do_imposto, 1, 1).
multiplicidade(cálculo_do_imposto/nota_fiscal, nota_fiscal, 1, 1).
multiplicidade(destinatário_remetente/endereço, destinatário_remetente, 1, 1).
multiplicidade(destinatário_remetente/endereço, endereço, 1, 1).
multiplicidade(destinatário_remetente/nota_fiscal, destinatário_remetente, 1, 1).
multiplicidade(destinatário_remetente/nota_fiscal, nota_fiscal, 1, 0).
multiplicidade(endereço/nota_fiscal, endereço, 1, 1).
multiplicidade(endereço/nota_fiscal, nota_fiscal, 1, 0).
multiplicidade(endereço/transportador_volumes_transportados, endereço, 1, 1).
multiplicidade(endereço/transportador_volumes_transportados,
    transportador_volumes_transportados, 1, 1).
multiplicidade(item_produto/nota_fiscal, item_produto, 1, 0).
multiplicidade(item_produto/nota_fiscal, nota_fiscal, 1, 1).
multiplicidade(item_produto/produto, item_produto, 1, 0).
multiplicidade(item_produto/produto, produto, 1, 1).

```
multiplicidade(nota_fiscal/dado_adicional, dado_adicional, 1, 1).
multiplicidade(nota_fiscal/dado_adicional, nota_fiscal, 1, 1).
multiplicidade(nota_fiscal/operação, nota_fiscal, 1, 0).
multiplicidade(nota_fiscal/operação, operação, 1, 1).
multiplicidade(nota_fiscal/transportador_volumes_transportados, nota_fiscal, 1, 1).
multiplicidade(nota_fiscal/transportador_volumes_transportados,
    transportador_volumes_transportados, 1, 1).
```