DENIS FELIPE SUZUKI

IMPLEMENTAÇÃO DE UM SISTEMA GERADOR DE FRASES NO DOMÍNIO DE TRÁFEGO AÉREO UTILIZANDO ONTOLOGIAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Curso de Mestrado em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientadora: Eliana de Mattos Pinto Coelho

Co-orientador: Michel Gagnon

CURITIBA
2002

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno *Denis Felipe Suzuki*, avaliamos o trabalho intitulado, "*Implementação de um Sistema Gerador de Frases no Domínio de Tráfego Aéreo Utilizando Ontologias*", cuja defesa foi realizada no dia 09 de dezembro de 2002, às onze horas, no anfiteatro B do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 09 de dezembro de 2002.

Profª. Dra. Eliana de Mattos Pinto Coelho
DINF/UFPR

Prof. Dr. Michel Gagnon
DINF/UFPR

Profª. Dra. Maria das Graças Volpe Nunes
ICMC-SC/USP

Profª. Dra. Cristina Duarte Murta
DINF/UFPR

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

SUMÁRIO

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

RESUMO

Neste trabalho, apresentamos a utilização de ontologias em um sistema gerador de frases que relatam eventos do domínio de tráfego aéreo acontecidos durante o vôo de uma ou mais aeronaves.

A vantagem desse sistema gerador é a utilização das frases geradas para a produção de relatórios por profissionais de tráfego aéreo. Atualmente, esses relatórios são feitos manualmente a partir de dados na forma de tabelas demandando bastante tempo para serem produzidos.

As ontologias exercem um papel fundamental dentro do sistema, pois elas representam o conhecimento utilizado pelos módulos do sistema, fazendo a ligação entre eles. Além disso, elas contribuem para uma modelagem dos dados mais profunda, servindo também como documentação do sistema. Outra grande vantagem da definição de ontologias é aumentar o potencial de reutilização e compartilhamento do conhecimento representado. Nesse caso, as ontologias criadas podem ser potencialmente utilizadas em outros sistemas.

ABSTRACT

In this work, we present the usage of ontologies in a phrase generating system that reports events of the air traffic domain happened during the flight of one or more aircrafts.

The advantage of this generating system is the utilization of the phrases generated for the production of reports by air traffic professionals. Currently, these reports are made manually from data in a table form, demanding much time to be produced.

Ontologies play a fundamental role in the system, because they represent the knowledge used for the modules of the system, making the plugging between them. Moreover, they contribute for a deeper modeling of the data, also serving as documentation of the system. Another great advantage of the definition of ontologies is to increase the potential of reusing and sharing of the represented knowledge. In this case, ontologies can potentially be used in other systems.

1 INTRODUÇÃO

Nesta dissertação, abordamos a questão da construção de ontologias em sistemas de processamento de língua natural. Mais especificamente, definimos e utilizamos ontologias em um sistema gerador de frases que relatam eventos de tráfego aéreo.

A geração de textos em língua natural é um subcampo da inteligência artificial e da lingüística computacional. O objetivo principal das pesquisas nessa área é a construção de sistemas que produzem textos a partir de uma representação não lingüística da informação. Esses sistemas usam conhecimento sobre a língua, sobre o mundo e sobre um domínio de aplicação para produzir documentos e relatórios.

A automação da produção de textos é importante, pois pode aliviar o ser humano da tarefa repetitiva de produzir documentos rotineiros. Cada vez mais tem se sentido a necessidade de utilizar ontologias para representação do conhecimento em sistemas de processamento de língua natural. As ontologias possibilitam diminuir o custo e o tempo de desenvolvimento de geradores de textos.

A figura 1.1 ilustra o sistema gerador de frases para o domínio de tráfego aéreo utilizando ontologias. Os dados não lingüísticos a partir dos quais as frases são geradas são dados fornecidos pelo sistema de controle de tráfego aéreo. Esses dados são obtidos principalmente dos radares que fornecem, entre outros, a altitude de uma aeronave, seu rumo e sua velocidade. Esses dados não lingüísticos correspondem à entrada do sistema gerador de frases, que é constituído por dois módulos: o programa identificador de eventos e o programa gerador de frases. O primeiro módulo identifica os eventos a partir dos dados do sistema de controle de tráfego aéreo e produz uma representação dos eventos identificados. O segundo módulo tem como entrada a representação dos eventos de tráfego aéreo e tem como saída as frases que relatam os eventos. Esse sistema utiliza ontologias em todo o processo de geração de frases.

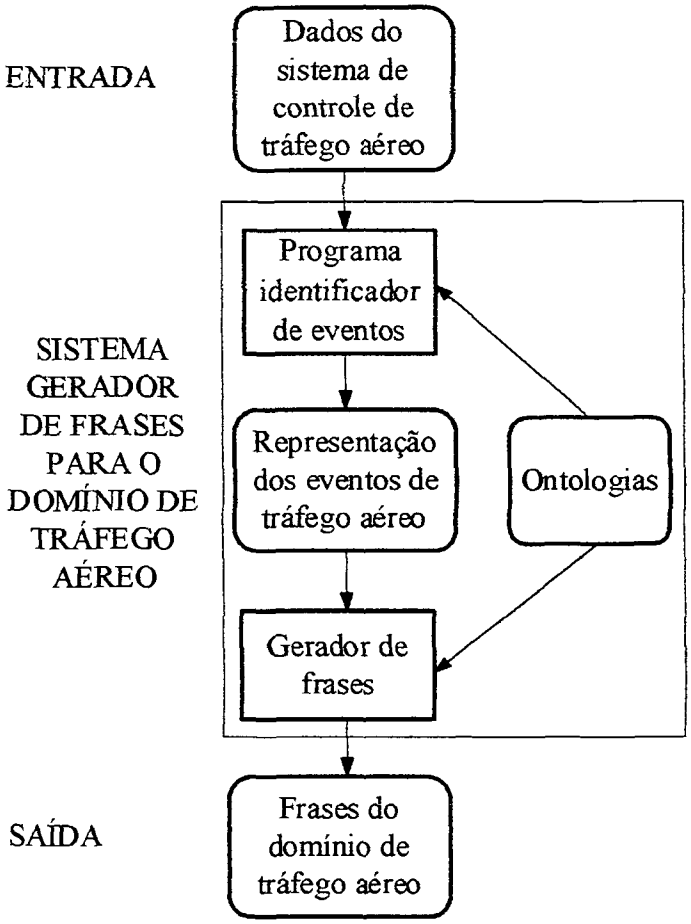


Figura 1.1 Visão geral do sistema gerador de frases para o domínio de tráfego aéreo

A seguir, mostramos exemplos de frases produzidas pelo sistema. As frases relatam eventos associados ao vôo de uma ou mais aeronaves:

- A aeronave VRG1111 decolou da pista três do aeródromo Afonso Pena às 10h 20min 00s.
- A aeronave PTPPP começou a subir do nível de vôo 030 para o nível de vôo 050 às 10h 20 min 10s.
- O radar Sorocaba detectou a aeronave VRG1111 às 10h 22min 10s.

A vantagem desse sistema gerador de frases é a utilização das frases geradas para a produção de relatórios por profissionais da área de tráfego aéreo, poupando o trabalho de interpretar os dados vindos diretamente dos radares.

As ontologias exercem um papel fundamental dentro do sistema, pois elas representam o conhecimento utilizado pelos módulos do sistema, fazendo a ligação entre eles. Além disso, elas contribuem para uma modelagem dos dados mais profunda, servindo também como documentação do sistema. Outra grande vantagem da definição de ontologias é aumentar o potencial de reutilização e compartilhamento do conhecimento representado. Nesse caso, as ontologias criadas podem ser potencialmente utilizadas em outros sistemas do domínio de tráfego aéreo.

Estruturamos esta dissertação em sete capítulos. No próximo capítulo, apresentamos o conceito de ontologia, a classificação das ontologias, as vantagens de sua utilização, os princípios metodológicos para construção de ontologias e as linguagens para definição de ontologias.

No capítulo 3, apresentamos a ontologia de tráfego aéreo, mostrando sua taxonomia e a definição dos seus conceitos e relações na linguagem Ontolingua.

O tema principal do capítulo 4 são os eventos de tráfego aéreo. Inicialmente, será apresentado o conceito de *papéis temáticos*. Também serão apresentados naquele capítulo, os conceitos e relações da ontologia de papéis temáticos e da ontologia de eventos de tráfego aéreo. Finalizamos o capítulo, apresentando o funcionamento do programa identificador de eventos.

No capítulo 5, apresentamos os fundamentos básicos para definição de gramáticas para processamento de língua natural.

No capítulo 6, ilustramos o funcionamento do módulo gerador de frases, o qual utiliza uma gramática para produção de frases para o domínio de tráfego aéreo.

Finalmente, o capítulo 7 descreve as contribuições e também as perspectivas para continuação do nosso trabalho.

2 ONTOLOGIAS

Neste capítulo, apresentamos o conceito de ontologia, algumas vantagens da utilização de ontologias no desenvolvimento de sistemas, alguns princípios para construção de ontologias e as linguagens para definição de ontologias.

2.1 CONCEITO DE ONTOLOGIA

A palavra **ontologia** vem do grego *ontos* (ser), e *logos* (estudo) e é definida no Dicionário Aurélio como "a parte da filosofia que trata do ser enquanto ser, isto é, do ser concebido como tendo uma natureza comum que é inerente a todos e a cada um dos seres".

Atualmente, as ontologias estão sendo pesquisadas também na área da inteligência artificial, onde esse termo significa uma "especificação explícita de uma conceituação" [Gru93].

Uma **conceituação** corresponde à escolha de conceitos e de relações entre esses conceitos para representar uma parte do mundo. Em relação a um domínio, várias conceituações são possíveis. Diferentes conceituações correspondem a visões diversas de uma mesma parte da realidade. Para um mesmo domínio, diferentes conceituações vão permitir a resolução de diferentes problemas. Uma conceituação é considerada melhor do que outra se ela possibilita a resolução de um problema mais facilmente ou se ela torna mais fácil a compreensão de um domínio [Gen87].

Uma ontologia também pode ser vista como um consenso sobre uma conceituação compartilhada. Assim, uma ontologia exprime um entendimento comum de um conjunto de agentes ou sistemas sobre uma conceituação.

Vejamos um exemplo de uma ontologia de informação bibliográfica. Essa ontologia pode ser usada para troca de informações entre diferentes bases de dados bibliográficas ou para integração de bases de dados bibliográficas com outras bases

de dados. Ela contém, entre outros, os seguintes conceitos: documento, autor, tese e livro. A seguir, apresentamos uma parte dessa ontologia expressa em língua natural para definir o conceito *autor*:

Um autor é uma pessoa que tem exatamente um nome associado e tem que ter escrito pelo menos um documento

Em geral, os conceitos estão descritos em uma taxonomia na ontologia. Veja como exemplo a taxonomia da ontologia de informação bibliográfica:

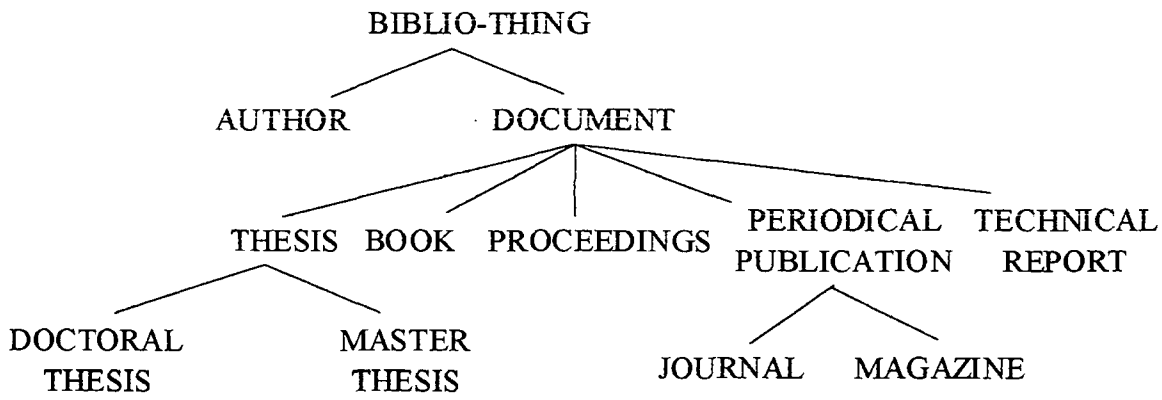


Figura 2.1 Taxonomia da ontologia de informação bibliográfica

Como podemos observar na figura 2.1, os conceitos seguem uma organização hierárquica, onde, por exemplo, os conceitos *author* e *document* estão no mesmo nível hierárquico.

Além dessa organização hierárquica, cada conceito também é descrito com seus atributos e relações. Dessa forma, uma ontologia expressa uma interpretação semântica do conhecimento. Tomemos como exemplo a descrição do conceito autor dado acima em língua natural definido na linguagem Ontolingua [Gru92,Gru93]:

```
(define-class AUTHOR (?x)
  :def (and (person ?x)
            (has-one ?x author.name)
            (value-type ?x author.name biblio-name)
            (has-some ?x author.documents)))
```

Essa representação define o conceito *autor* como uma classe. A cláusula *def* define as condições necessárias para os membros da classe, ou seja, descreve o que deve ser verdadeiro para as instâncias da classe. O primeiro termo da conjunção (*person ?x*) define que o autor tem que ser uma pessoa. O segundo termo (*has-one ?x author.name*) define que o autor só tem um nome associado. O terceiro termo (*value-type ?x author.name biblio-name*) define que o nome do autor deve ser da classe *biblio-name*. O quarto termo (*has-some ?x author.documents*) define que o autor deve ter escrito pelo menos um documento.

Uma ontologia é uma representação de um vocabulário, não sendo o vocabulário que a qualifica como ontologia, mas as conceituações que os termos no vocabulário pretendem representar [Cha99]. Assim, traduzir os termos em uma ontologia de uma língua para outra, por exemplo do português para o inglês, não muda a ontologia conceitualmente, pois existe um mapeamento, por exemplo, de *autor* em português para *author* em inglês. Esse mapeamento é uma interpretação semântica mesmo que seja na mesma língua. Por exemplo, suponha que um sistema disponibilize *códigos de endereçamento postal* e outro sistema disponibilize *CEPs*. Fica claro que trata-se do mesmo conceito escrito de formas diferentes. Através da utilização de uma ontologia para interoperação entre esses sistemas, eles poderiam trocar informações, pois teriam o mapeamento das informações equivalentes semanticamente.

Uma ontologia expressa um **engajamento ontológico** (*ontological commitment*), isto é, um acordo sobre uma determinada conceituação. Engajamentos

ontológicos são necessários para restringir possíveis interpretações de uma ontologia. Quando agentes ou sistemas utilizam uma ontologia, eles estão engajados ontologicamente com essa ontologia, de forma que eles estão de acordo com as definições da ontologia e que eles usarão essas definições de forma consistente. O significado dos termos são fixados de forma que todos entendem a mesma coisa quando o conceito é usado.

As ontologias são diferentes das bases de conhecimento, pois as ontologias são conjuntos de conceitos e relações em um domínio, enquanto que as bases de conhecimento contém instâncias dos conceitos e relações em um domínio. Se fizermos uma analogia com o esquema conceitual de banco de dados, as ontologias correspondem à representação intensional, enquanto que as bases de conhecimento correspondem à representação extensional.

2.2 CLASSIFICAÇÃO DAS ONTOLOGIAS

De acordo com [Hei95], as ontologias podem ser classificadas sob dois aspectos: tipo de estrutura da conceituação e assunto da conceituação.

2.2.1 Classificação pelo tipo de estrutura da conceituação

As ontologias são classificadas pela estrutura da conceituação em: ontologias terminológicas, ontologias de informação e ontologias de modelagem do conhecimento.

- As **ontologias terminológicas** são semelhantes a léxicos, pois especificam os termos usados para a representação do conhecimento no domínio.

- As **ontologias de informação** especificam a estrutura de registros de bases de dados. O esquema conceitual de banco de dados é um exemplo dessa classe de ontologias.

- As **ontologias de modelagem do conhecimento** especificam as conceituações do conhecimento. São mais completas que as ontologias de informação e são direcionadas para o conhecimento que descrevem.

2.2.2 Classificação pelo assunto da conceituação

As ontologias são classificadas pelo assunto da conceituação em: ontologias da aplicação, ontologias de domínio, ontologias genéricas e ontologias de representação.

- As **ontologias da aplicação** contêm todas as definições que são necessárias para modelar o conhecimento necessário para uma aplicação em particular, por exemplo, uma ontologia de planejamento de operações de guerra para uma Força Aérea [Val99].

- As **ontologias de domínio** são conceituações específicas para determinados domínios. Por exemplo, uma ontologia do domínio de tráfego aéreo.

- As **ontologias genéricas** são similares às ontologias do domínio, mas seus conceitos são mais genéricos. Normalmente, definem conceitos como estado, evento, processo, ação e componente. Por exemplo, uma ontologia do tempo com conceitos como: dia, mês, ano, hora, minuto e segundo.

- As **ontologias de representação** são ontologias criadas para facilitar a representação de outras ontologias. Por exemplo, a *Frame Ontology* de Ontolingua. Essa ontologia define os termos usados em sistemas de representação do conhecimento baseados na abordagem orientada a objetos. As ontologias genéricas e de domínio são descritas usando-se de primitivas das ontologias de representação.

No nosso projeto, definimos ontologias classificadas como ontologias de modelagem do conhecimento, segundo a classificação pelo tipo de estrutura da conceituação, e de domínio segundo a classificação pelo assunto da conceituação.

2.3 UTILIZAÇÃO DE ONTOLOGIAS

Após a definição de ontologias, apresentamos as principais vantagens da utilização de ontologias em sistemas baseados em conhecimento e sistemas em geral, e terminamos justificando a utilização de ontologias em sistemas de processamento de língua natural.

2.3.1 Representação do conhecimento

As ontologias ajudam a descrever um domínio e, portanto, possibilitam a compreensão desse domínio. Sem as ontologias, um vocabulário apenas não consegue representar o conhecimento. Assim, para planejar um sistema baseado em conhecimento é necessário fazer uma análise ontológica do domínio. A análise ontológica ajuda a esclarecer a estrutura do conhecimento. Dado um domínio, a ontologia forma o coração de qualquer sistema de representação do conhecimento para esse domínio.

Essa análise ontológica deve ser bem feita, pois se considerarmos como exemplo um domínio de várias classes de pessoas (estudantes, professores,

empregados e pessoas do sexo masculino e feminino, por exemplo) que representam os tipos da classe humana, concluímos que os estudantes podem ser empregados por algum tempo e também podem parar de ser estudantes. Uma análise criteriosa mostra que os termos *estudante* e *empregado* não descrevem categorias de humanos, mas papéis desempenhados por humanos, enquanto que pessoas do sexo masculino e feminino realmente representam subcategorias de humanos.

As ontologias também podem ser utilizadas para fazer a especificação de um sistema de software convencional, onde uma compreensão compartilhada por vários especialistas ajuda o processo de identificar requisitos do sistema, especialmente quando estão envolvidos grupos usando terminologias diferentes em um ou vários domínios [Usc96].

As estruturas de dados e os procedimentos dos sistemas convencionais implícita ou explicitamente têm um engajamento com uma ontologia do domínio, pois todos os sistemas de informação utilizam conhecimento. Nenhum programa pode ser feito sem um compromisso com um modelo do mundo com entidades, propriedades e relações nesse mundo.

A abordagem orientada a objetos também depende de uma ontologia de domínio. Os objetos, seus atributos e seus métodos mais ou menos refletem os aspectos do domínio de aplicação. Entretanto, há diferenças, pois as classes e objetos em um programa orientado a objetos são estruturas de dados. Já as classes em uma ontologia são a representação do mundo real. Por exemplo, o conceito *nome* é representado em um programa como uma cadeia de caracteres de tamanho máximo igual a vinte. O conceito *nome* em uma ontologia é uma classe que agrega todos os nomes existentes no domínio da ontologia.

A confiabilidade dos sistemas também é melhorada com o uso de ontologias, porque utilizando essas representações formais podemos fazer a checagem de consistência desse software.

Dentre as várias aplicações de ontologias para representação do conhecimento está o uso de ontologias para representar o conhecimento em

aplicações na Internet. Elas começam a ser usadas como base para proporcionar acesso semântico à Internet [Gom02]. [Ber01] propõe o uso de ontologias para aprimorar as buscas na Internet e também para relacioná-las com informações em páginas web.

2.3.2 Compartilhamento e reutilização do conhecimento

Acredita-se que as ontologias irão mudar o modo como os sistemas são desenvolvidos. Hoje, as bases de conhecimento são feitas com pouca reutilização ou compartilhamento. No futuro, existirão bibliotecas de ontologias para se desenvolver sistemas inteligentes [Swa99].

A reutilização e compartilhamento do conhecimento são importantes, pois reduzem o custo e o tempo de desenvolvimento de sistemas baseados em conhecimento, principalmente se considerarmos a aquisição do conhecimento como o "gargalo" na construção desses sistemas.

A existência de uma ontologia bem estruturada em um domínio ajuda no desenvolvimento de novos sistemas nesse domínio, pois ela serve de base para a construção desses novos sistemas [Val99].

Suponha que um programador queira fazer um programa para verificar se uma aeronave pode utilizar um aeroporto, conforme o exemplo retirado de [Swa96]. Sem uma ontologia, ele precisa criar um conceito *aeroporto*. Ele deve imaginar onde tal conceito aparecerá numa hierarquia (possivelmente exigindo a definição dos conceitos intermediários que estão faltando). Depois, ele deve acrescentar o atributo *comprimento máximo de pista* no conceito *aeroporto*. Esse atributo pode ser usado para determinar se uma aeronave pode utilizar o aeroporto.

Agora considere a mesma tarefa feita através de uma ontologia desenvolvida em colaboração. Em vez de criar o conceito *aeroporto*, o programador pode buscá-lo se ele já existe. Suponha que o programador encontre o atributo *comprimento máximo de pista* (se ele não encontrá-lo, pode adicioná-lo para outros utilizarem

mais tarde). Então, ele poderá também encontrar conceitos importantes para se saber se uma aeronave pode pousar em um aeroporto, como: *tipo de pavimento* e *auxílios à navegação*.

Para facilitar a reutilização e compartilhamento do conhecimento é necessária a representação do conhecimento em uma linguagem independente de qualquer formalismo de representação do conhecimento. O ideal é representar uma ontologia em uma interlíngua (que será explicada na seção 2.4) para facilitar a tradução da representação do conhecimento para outras linguagens. Portanto, dois ou mais sistemas podem se comunicar sem utilizar a mesma linguagem.

Com a utilização de ontologias, pode-se diminuir a barreira de comunicação entre as pessoas, organizações ou sistemas de informação [Usc96], mesmo se as pessoas e organizações têm diferentes necessidades e pontos de vista e os sistemas diferentes métodos, paradigmas, linguagens e ferramentas.

Um exemplo de reutilização de ontologias é o descrito em [Val99], onde uma ontologia do tempo com conceitos como *dia* e *hora* é reutilizada para definir uma ontologia de planejamento de operações de guerra para a Força Aérea.

Nas arquiteturas baseadas em agentes, é utilizada uma linguagem comum na qual as mensagens têm significado independente de seus agentes [Smi96]. Assim, com o uso de ontologias, os agentes são capazes de compartilhar conhecimento e isso proporciona uma coordenação de suas atividades, mesmo se os agentes forem autônomos. Utilizando uma ontologia, o sistema fica com características de um todo integrado e aparece aos usuários como se tivesse sido implementado junto, mesmo se os agentes foram feitos em tempos diferentes, em linguagens diferentes e em abordagens diferentes.

2.3.3 Utilização de ontologias em sistemas de processamento de língua natural

Os sistemas para processamento de língua natural necessitam representar grandes quantidades de conhecimento. Esses sistemas utilizam tanto conhecimentos

genéricos, como conhecimentos específicos de um domínio. Além disso, esses sistemas utilizam também conhecimentos lingüísticos tais como gramáticas e dicionários.

Cada vez mais tem se sentido a necessidade de utilizar ontologias para a representação do conhecimento nesses sistemas. As ontologias utilizadas para representar o conhecimento genérico e do domínio apresentam as seguintes vantagens:

- Representam o conhecimento de forma neutra, independente da língua, como já vimos na seção 2.1,

- Servem como uma estrutura de base e permitem uma separação clara entre o conhecimento lingüístico e o conhecimento do genérico e do domínio, pois o conhecimento lingüístico é geralmente formado por gramáticas e dicionários (capítulo 5), o conhecimento genérico é representado por ontologias genéricas (seção 2.1.2) e o conhecimento do domínio é representado por ontologias do domínio (seção 2.1.2),

- Permitem a associação de cada palavra no dicionário com um conceito na ontologia de tal forma que cada palavra tenha uma semântica bem definida e tenha também relações bem definidas com outros conceitos e, em conseqüência, com outras palavras do dicionário. Assim, por exemplo, o verbo *dormir* não pode estar relacionado ao substantivo *aeronave*, mas esse substantivo pode estar relacionado ao verbo *subir*.

Por possibilitar uma representação do conhecimento independente da língua, as ontologias facilitam a interpretação e a geração de textos em sistemas de tradução baseados em conhecimento multilíngüe. Em [Mah95,Mah96], temos o trabalho do grupo de pesquisa da *New Mexico State University* em sistemas de tradução

multilíngüe. Eles propõem o uso de ontologias para representação do conhecimento. Nesse projeto de tradução automática foi criada uma ontologia chamada de *Mikrokosmos* com aproximadamente 4500 conceitos. Essa ontologia foi elaborada para auxiliar na tradução automática de textos formando uma base de conhecimento que representa os conceitos independente da língua. A ontologia *Mikrokosmos* é uma classificação taxonômica de conceitos assim como um repositório de conhecimento de mundo expresso na forma de relações conceituais.

Outra ontologia criada para apoiar o processamento de língua natural é a *Upper Model* [Bat90]. Na *Upper Model*, a organização do conhecimento é baseada em conhecimentos lingüísticos. Entretanto, além de poder ser utilizado em sistemas de processamento de língua natural, o propósito dessa ontologia é servir como base para modelagem de outros tipos de sistemas.

2.4 PRINCÍPIOS PARA CONSTRUÇÃO DE ONTOLOGIAS

Assim como não há uma única gramática "verdadeira" para as línguas naturais, também não há uma única ontologia para qualquer domínio, pois há várias formas de fazer uma conceituação. Isso não quer dizer que uma ontologia é construída aleatoriamente, mas a sua construção deve obedecer certas diretrizes.

Em nosso trabalho, observamos principalmente as diretrizes básicas recomendadas por [Gru95] para o projeto de ontologias, apresentadas a seguir:

- Clareza: numa ontologia, as ambigüidades devem ser minimizadas, as distinções bem fundamentadas e exemplos devem ser dados para ajudar a compreensão de definições. As definições devem sempre ser documentadas em língua natural. Definir uma ontologia em linguagem formal é um meio de obter clareza.

- Coerência: uma ontologia deve ser consistente internamente, pois uma definição não pode contradizer outra definição, senão a ontologia é incoerente. A coerência também se aplica à documentação.

- Extensibilidade: uma ontologia deve ser estendida e especializada monotonicamente, isto é, se forem definidos novos termos baseados na ontologia existente, não deve haver a necessidade de revisão dos termos já existentes.

- Mínima distorção na codificação: uma ontologia deve ser codificada em uma linguagem independente de linguagens de implementação. O conhecimento definido em uma ontologia deve ser compartilhado por diferentes sistemas e estilos de representação. A definição da ontologia em uma linguagem como Ontolingua proporciona esse compartilhamento, pois uma ontologia definida em Ontolingua pode ser traduzida para diferentes linguagens de implementação.

- Engajamento ontológico mínimo (*minimal ontological commitment*): uma ontologia deve ter suas restrições semânticas minimizadas. Vamos ilustrar como minimizar as restrições semânticas através de um exemplo. Suponha uma definição do conceito *mesa* como um móvel que tem exatamente quatro pés. Entretanto, estamos restringindo desnecessariamente a definição do conceito *mesa*, pois sabemos que existem mesas com o número de pés diferentes de quatro. Dessa forma, devemos tirar essa restrição ou modificá-la para aumentar o potencial de reutilização dessa definição.

Além dessas diretrizes básicas, também estão sendo desenvolvidas metodologias para construir ontologias. Algumas delas são: a *Methontology* [Lop99]

e a metodologia baseada na formulação das questões de competência [Usc96], [Gru94].

A ***Methontology*** propõe um ciclo de construção de ontologias semelhante ao processo de desenvolvimento de software utilizando as chamadas *representações intermediárias* (tabelas, dicionários e árvores) para diminuir a "lacuna" entre o que foi pensado na aquisição do conhecimento para o que foi representado em uma linguagem formal.

Outra metodologia existente para construção de ontologias é a metodologia baseada na formulação das "**questões de competência**". Essa metodologia formal foi originada em um projeto de modelagem do conhecimento de organizações empresariais. Segundo essa metodologia, as ontologias definem os conceitos em lógica de primeira ordem, além de um conjunto de axiomas que devem representar e resolver as chamadas questões de competência.

2.5 LINGUAGENS PARA DEFINIÇÃO DE ONTOLOGIAS

Uma ontologia pode ser definida em vários tipos de linguagem. Definições em língua natural podem ser consideradas uma ontologia com seus conceitos definidos da mesma forma que o conceito *autor* apresentado no início deste capítulo. Um esquema conceitual de banco de dados ou um modelo em UML também podem ser considerados como uma ontologia. Também uma ontologia pode ser definida utilizando linguagens formais como lógica de primeira ordem.

Como vimos na seção 2.3, uma ontologia deve ser representada em uma linguagem independente de linguagens de implementação. Idealmente, uma ontologia deve ser definida em uma interlíngua. Uma **interlíngua** é uma linguagem criada para facilitar a tradução entre outras linguagens. Por exemplo, observe na figura 2.2, como é feita a tradução de linguagens sem uma interlíngua:

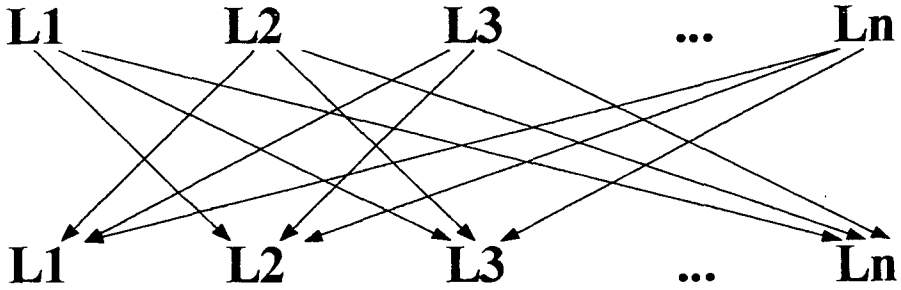


Figura 2.2 Tradução entre linguagens sem uma interlíngua

Na figura 2.2, a tradução de cada uma das linguagens é feita para cada uma das outras linguagens. Agora, observe na figura 2.3, como é feita a tradução de linguagens com a utilização de uma interlíngua:

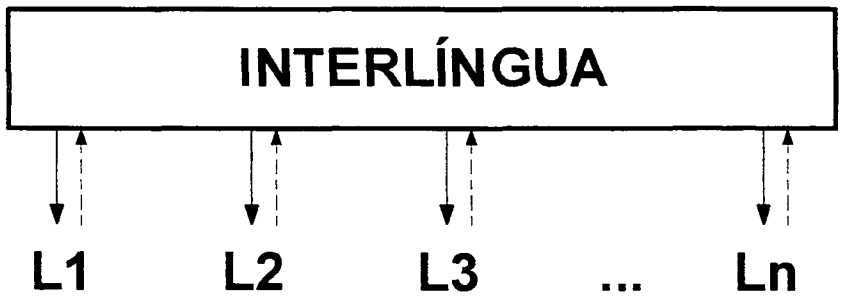


Figura 2.3 Tradução entre linguagens com o uso de uma interlíngua

Na figura 2.3, vemos que utilizando uma interlíngua, é necessário se fazer apenas a tradução de cada uma das linguagens para a interlíngua e da interlíngua para cada uma das linguagens. Dessa forma, uma interlíngua possibilita o compartilhamento de uma representação do conhecimento por várias linguagens, eliminando a necessidade de se fazer todo o processo de tradução de cada uma das linguagens para cada uma das outras linguagens.

Em nosso trabalho, decidimos utilizar Ontolingua que é uma linguagem formal que utiliza KIF (*Knowledge Interchange Format*) [Gen92]. KIF é uma extensão da lógica de primeira ordem[1] e é uma interlíngua.

Na próxima seção, apresentamos resumidamente a sintaxe e a semântica de Ontolingua.

2.5.1 Ontolingua

A linguagem Ontolingua possibilita uma organização dos conceitos baseada na abordagem orientada a objetos. Dessa forma, ela combina o poder expressivo da lógica de primeira ordem de KIF com a eficiência e os benefícios da representação orientada a objetos [Gru92].

Ontolingua foi desenvolvida para representar o conhecimento e não para fazer implementações. Uma ontologia definida em Ontolingua consiste em um conjunto de definições de classes, relações, funções e axiomas.

Classe

Uma classe é uma relação unária para agrupamento de termos semelhantes conceitualmente. Por exemplo, observe a definição da classe *book* (livro):

```
(define-class BOOK (?x)
  :def (and (document ?x)
            (has-some ?x has-author)
            (has-one ?x title-of)
            (has-one ?x publication-date-of)
            (has-one ?x publisher-of)))
```

[1] Com quantificadores universal (forall) e existencial (exists); operadores para conjunção (and), disjunção (or), implicação (=>), implicação reversa (<=), equivalência (<=>) e negação (not)

Nessa definição, um membro da classe *book* representado pela variável *?x* é um documento (*document ?x*) que tem pelo menos um autor (*has-some ?x has-author*), tem exatamente um título (*has-one ?x title-of*), uma data de publicação (*has-one ?x publication-date-of*) e um editor (*has-one ?x publisher-of*). Essa definição restringe semanticamente o termo *livro* e determina as condições necessárias para ser membro dessa classe.

Dentre os benefícios da abordagem orientada a objetos de Ontolingua está a herança. A definição da classe *document* foi herdada devido à declaração (*document ?x*) que significa também "a classe *book* é subclasse de *document*".

Uma classe também pode ser definida enumerando os valores possíveis de seus membros. Por exemplo, observe a definição em Ontolingua da classe *day-name* (nome de dia da semana):

```
(define-class DAY-NAME (?x)
    :iff-def (member ?x (setof Sunday
                               Monday
                               Tuesday
                               Wednesday
                               Thursday
                               Friday
                               Saturday)))
```

Nessa definição, um membro da classe *day-name* é um elemento (*member ?x*) do conjunto (*setof*) formado por *Sunday*, *Monday*, *Tuesday*, *Wednesday*, *Thursday*, *Friday* e *Saturday*. As relações *member* (define que o primeiro argumento é membro do segundo argumento que é um conjunto) e *setof* (define os elementos que constituem o conjunto) são predicados de KIF. A cláusula *iff-def* define as condições necessárias e suficientes para os membros da classe, ou seja, tudo aquilo que satisfaz as condições é um membro da classe. A diferença entre *def* e *iff-def* é que as

definições feitas com a cláusula *iff-def* são uma equivalência (operador *se e somente se* da lógica de primeira ordem). Por exemplo, observe a definição da classe *bachelor* (solteiro):

```
(define-class BACHELOR (?x)
  :iff-def (and (person ?x)
                (male ?x)
                (unmarried ?x)))
```

Nessa definição, um membro da classe *bachelor* é uma pessoa (*person* ?x) do sexo masculino (*male ?x*) e não casada (*unmarried ?x*). Como essa classe foi definida pela cláusula *iff-def*, não apenas todos os membros da classe *bachelor* devem ser das classes *person*, *male* e *unmarried*, mas tudo o que for das classes *person*, *male* e *unmarried* são da classe *bachelor* por definição. Se a definição da classe *bachelor* fosse feita com a cláusula *def*, então apenas todos os membros da classe *bachelor* devem ser das classes *person*, *male* e *unmarried*.

Relação

Uma relação é um conjunto de tuplas usado para descrever um relacionamento entre duas ou mais classes. Se uma relação for binária é chamada de *slot*. As relações binárias têm um domínio e um contra domínio. Por exemplo, observe a definição da relação *has-author*:

```
(define-relation HAS-AUTHOR  (?x1 ?x2)
  :def (and (document ?x1)
            (author ?x2)))
```

Nessa definição, *has-author* é uma relação que tem o domínio formado um membro da classe *document* (*document ?x1*) e o contra domínio formado por um membro da classe *author* (*author ?x2*).

Função

Uma função é uma relação entre um número *n* de termos a exatamente um outro termo, ou seja, é uma relação tal que não há duas relações de *n* termos que tenham os mesmos primeiros *n-1* termos. Por exemplo, observe a definição da função *number-of-pages-of* (número de páginas):

```
(define-function NUMBER-OF-PAGES-OF (?x1) :-> ?x2
  :def (and (document ?x1)
            (natural ?x2)))
```

Nessa definição, a função *number-of-pages-of* é um mapeamento de um membro da classe *document* (*document ?x1*) para um membro da classe dos números naturais (*natural ?x2*). Trata-se de uma função, pois cada membro da classe *document* tem o seu mapeamento para um membro da classe *natural* e esse mapeamento é determinado pelo membro da classe *document*. Uma função também pode ser definida como uma relação cujo último argumento é determinado pelos outros argumentos.

Axioma

Um axioma é uma declaração em lógica de primeira ordem assumida como verdadeira sem necessitar de prova. Por exemplo, é um axioma: "para todo par de pessoas, se eles forem irmãos, então existe alguma pessoa do sexo feminino que é mãe de ambos". Outro exemplo de axioma é a definição de *intersection-axiom*:

```
(define-axiom INTERSECTION-AXIOM
    := (=> (and (bounded ?u)
                (set ?s))
           (bounded (intersection ?u ?s)))
```

Nessa definição, *intersection-axiom* é o axioma "se existe algum *?u* que pode ser elemento de um conjunto e um conjunto *?s*, então o resultado da intersecção entre eles também pode ser elemento de um conjunto". O símbolo *:=* significa "início da definição" e *bounded* (pode ser elemento de um conjunto) é um predicado de KIF.

A linguagem Ontolingua também tem outros recursos disponíveis no endereço [Ksl02], onde há mais informações sobre essa linguagem, um editor e um repositório de ontologias baseados na linguagem Ontolingua. Utilizamos várias definições das ontologias desse repositório como exemplos nessa dissertação.

2.5.2 Ontologias de representação em Ontolingua

Como já vimos na seção 2.1.2, uma ontologia de representação é uma ontologia criada para facilitar a representação de outras ontologias. Utilizamos duas ontologias de representação em Ontolingua presentes em [Ksl02]: a *Frame Ontology* e a *Slot Constraint Sugar*.

Frame Ontology

A ontologia de representação *Frame Ontology* define os termos usados em sistemas de representação do conhecimento baseados na abordagem orientada a objetos definindo, entre outras coisas, a decomposição em subclasses disjuntas, subclasses exaustivas e partições.

As **subclasses disjuntas** são as subclasses cujos membros são disjuntos entre si. A definição da relação *disjoint-decomposition* (decomposição disjunta) da *Frame* Ontology é a seguinte:

```
(define-relation DISJOINT-DECOMPOSITION (?c ?decomp)
  :iff-def  (and (class ?c)
                 (class-partition ?decomp)
                 (=> (member ?subclass ?decomp)
                     (subclass-of ?subclass ?c))))
```

Nessa definição, uma decomposição disjunta de uma classe é um conjunto de subclasses dessa classe que são mutuamente disjuntas.

Como exemplo de utilização dessa relação da *Frame Ontology*, apresentamos a definição da classe *periodical-publication* (publicação periódica):

```
(define-class PERIODICAL-PUBLICATION (?x)
  :def (document ?x)
  :axiom-def (disjoint-decomposition
              PERIODICAL-PUBLICATION
             (setof JOURNAL
                    MAGAZINE)))
```

Nessa definição, a classe *periodical-publication* é uma subclasse de *document* que tem as subclasses disjuntas: *journal* e *magazine*. As subclasses são disjuntas, pois um jornal não pode ser uma revista e vice-versa. Outra forma de definir as subclasses disjuntas é dizer que a intersecção entre elas é vazia. Observe que as subclasses disjuntas são definidas por um axioma (*axiom-def*).

As **subclasses exaustivas** (*exhaustive-decomposition*) são as subclasses cujos membros reunidos formam a classe. A definição da relação *exhaustive-*

decomposition (decomposição em subclasses exaustivas) na *Frame Ontology* é a seguinte:

```
(define-relation EXHAUSTIVE-DECOMPOSITION (?c ?decomp)
  :def (and (class ?c)
            (=> (member ?subclass ?decomp)
                (subclass-of ?subclass ?c))))
```

Nessa definição, a relação *exhaustive-decomposition* é um relacionamento entre uma classe e sua decomposição que só é verdadeiro se qualquer subclasse que seja um membro dessa decomposição, também necessariamente seja uma subclasse da classe representada pela variável *?c*.

Como exemplo de decomposição exaustiva temos a classe *número inteiro* com as subclasses *número inteiro par* e *número inteiro ímpar*. Juntando os números inteiros pares e os números inteiros ímpares obtemos todos os números inteiros. Outro exemplo seria a classificação de *país da América do Norte* em *Estados Unidos*, *México* e *Canadá*.

Uma **partição** (*partition*) é formada pelas subclasses de uma classe que são ao mesmo tempo disjuntas e exaustivas. Por exemplo, a classificação de *animal* em *animal macho* e *animal fêmea*.

Slot Constraint Sugar

A ontologia de representação *Slot Constraint Sugar* define regras para facilitar a utilização das relações binárias (*slots*). Essas regras são conhecidas como *facetas*. A ontologia *Slot Constraint Sugar* tem, principalmente, definições de relações de restrições de cardinalidade e de restrição de tipo.

Uma relação de restrição de cardinalidade restringe a cardinalidade de outra relação. Para entender melhor uma restrição de cardinalidade, observe a definição da relação *has-one*:

```
(define-relation HAS-ONE (?instance ?binary-rel)
  :iff-def (and (instance-of ?binary-rel binary-relation)
                (cardinality ?binary-rel ?instance 1)))
```

Nessa definição, a relação *has-one* é um relacionamento entre uma instância[2] e um relação binária no qual a relação binária é uma instância da classe *binary-relation* e a cardinalidade entre essa relação binária e a instância é igual a um.

Como exemplo de utilização de relações de restrição de cardinalidade, considere a definição apresentada anteriormente da classe *book*:

```
(define-class BOOK (?x)
  :def (and (document ?x)
            (has-some ?x has-author)
            (has-one ?x title-of)
            (has-one ?x publication-date-of)
            (has-one ?x publisher-of)))
```

Nessa definição, temos a relação *has-some* restringindo a cardinalidade da relação *has-author* entre os conceitos *book* e *author* ao valor maior ou igual a um, ou seja, um livro tem pelo menos um autor. Temos também a relação *has-one* que restringe a cardinalidade das relações *title-of*, *publication-date-of* e *publisher-of* da classe *book* a um, ou seja, um membro da classe *book* está relacionado a apenas um

---

[2] Todos os termos que tem uma definição associada (classes, relações, funções) são considerados instâncias em Ontolingua. O que é comumente conhecido como instância, em Ontolingua é chamado de indivíduo (*individual*)

membro das classes dos contra domínios dessas relações. Assim, um livro só tem um título, uma data de publicação e um editor.

As restrições de tipo restringem o tipo do domínio ou do contra domínio de uma relação binária. Por exemplo, observe a definição da classe *master-thesis-reference* (referência para dissertação de mestrado):

```
(define-class MASTER-THESIS-REFERENCE (?x)
 :def (and (thesis-reference ?x)
           (value-type ref.document ?x master-thesis)
           (ref.type-of-work ?x "Master Thesis")))
```

Nessa definição, a relação *value-type* restringe o tipo do contra domínio da função *ref.document* que deve ser formado apenas por membros da classe *master-thesis*.

Existe também a restrição de cardinalidade e tipo. Por exemplo, observe a definição da relação *has-one-of-type*:

```
(define-relation HAS-ONE-OF-TYPE (?binary-rel ?inst ?type)
  :iff-def (and (instance-of ?binary-rel binary-relation)
                (instance-of ?type class)
                (cardinality ?binary-rel ?inst 1)
                (value-type ?binary-rel ?inst ?type)))
```

Nessa definição, a relação *has-one-of-type* é um relacionamento entre uma relação binária, uma instância e um tipo no qual a relação binária é uma instância da classe *binary-relation,* o tipo é uma instância de classe, a cardinalidade entre a relação binária e a instância é igual a um e o contra domínio da relação binária para a instância é formado apenas pelo tipo *?type*.

Neste capítulo, apresentamos o conceito de ontologia, sua classificação, o porquê da utilização de ontologias, as linguagens e os princípios para construção de

ontologias. Também apresentamos a linguagem Ontolingua e as ontologias de representação em Ontolingua. No próximo capítulo, apresentamos a ontologia de tráfego aéreo construída por nós que foi definida em Ontolingua.

3 ONTOLOGIA DE TRÁFEGO AÉREO

Neste capítulo, apresentamos a ontologia de tráfego aéreo que definimos. Essa ontologia é classificada como uma ontologia do domínio conforme a classificação das ontologias vista na seção 2.1. O escopo da ontologia é o tráfego aéreo civil. A ontologia de tráfego aéreo, a ontologia de eventos de tráfego aéreo e a ontologia de papéis temáticos serviram de base para a gramática do gerador de frases que será apresentado no capítulo 6.

Os conceitos da ontologia de tráfego aéreo estão de acordo com a IMA 100-12 [Ima99]. A IMA 100-12 é um documento com as regras do tráfego aéreo civil baseado nas orientações da OACI (Organização da Aviação Civil Internacional). Todos os conceitos definidos nessa ontologia e em nossas outras ontologias estão em inglês, pois é usual a definição de ontologias nessa língua. Inicialmente, explicamos a taxonomia de nível superior da ontologia e por fim apresentamos as definições dos conceitos e relações.

3.1 VISÃO GERAL DA ONTOLOGIA DE TRÁFEGO AÉREO

Para a definição da ontologia de tráfego aéreo foram incluídas as ontologias de representação *Slot Constraint Sugar* e *Frame Ontology* já apresentadas na seção 2.4.2 conforme ilustrado na figura 3.1.

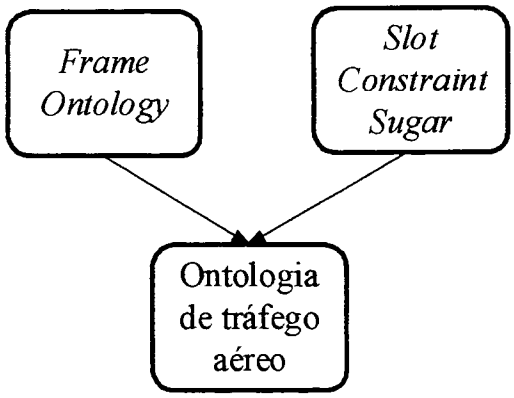


Figura 3.1 Organização hierárquica da ontologia de tráfego aéreo

A definição do nome da ontologia de tráfego aéreo em Ontolingua é a seguinte:

```
(define-ontology AIR-TRAFFIC
  (SLOT-CONSTRAINT-SUGAR FRAME-ONTOLOGY))
```

Nessa definição, o nome da ontologia de tráfego aéreo é *air-traffic* e as ontologias incluídas são: *slot-constraint-sugar* e *frame-ontology*.

Observe a taxonomia de nível superior da ontologia de tráfego aéreo mostrada na figura 3.2. Nessa figura, a raiz da taxonomia da ontologia de tráfego aéreo é a classe *thing* que é a classe usada em Ontolingua para agregar todas as outras classes. A única subclasse de *thing* é a classe *at-thing* (*air-traffic-thing*) que agrega todas as classes da ontologia de tráfego aéreo. Observe também que a classe *at-thing* se divide em duas subclasses básicas: *at-object* e *at-attribute*.

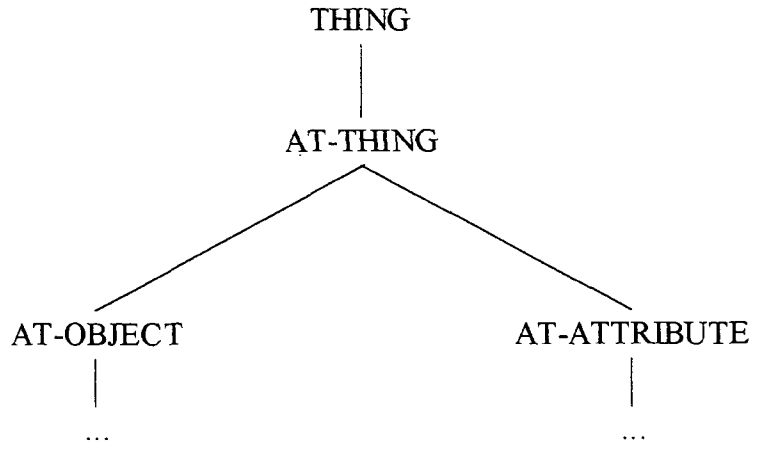


Figura 3.2 Taxonomia de nível superior da ontologia de tráfego aéreo

A classe *at-object* agrega todos os objetos da ontologia de tráfego aéreo: controle de tráfego aéreo, aeronave, plano de vôo, lugar e dispositivo eletrônico.

A outra subclasse de *at-thing* é a classe *at-attribute* que agrega todos os atributos dos objetos da ontologia de tráfego: identificador de tráfego aéreo, direção, código e valor numérico.

Na próxima seção, apresentamos mais detalhadamente a taxonomia a partir das classes *at-object* e *at-attribute*.

3.2 DEFINIÇÕES DA ONTOLOGIA DE TRÁFEGO AÉREO

Nesta seção, apresentamos as definições da classes, relações e funções da ontologia de tráfego aéreo.

Objeto da ontologia de tráfego aéreo

O conceito *objeto da ontologia de tráfego aéreo* agrega todas os objetos físicos ou mentais relacionados ao tráfego aéreo. A definição em Ontolingua da classe *at-object* (objeto da ontologia de tráfego aéreo) é a seguinte:

```
(define-class AT-OBJECT (?x)
  :def (and (at-thing ?x)
            (>= (cardinality ?x has-at-identifier) 0))
  :axiom-def (disjoint-decomposition
              AT-OBJECT
              (setof AT-CONTROL
                     FLIGHT-PLAN
                     AIRCRAFT
                     PLACE
                     ELECTRONIC-DEVICE)))
```

Nessa definição, a classe *at-object* é uma subclasse de *at-thing* que pode ou não ter identificadores. Além disso, a classe *at-object* tem as subclasses disjuntas: *at-control, flight-plan, aircraft, place* e *electronic-device.*

Atributo de objeto da ontologia de tráfego aéreo

O conceito *atributo de objeto da ontologia de tráfego aéreo* agrega todas as propriedades dos objetos da ontologia de tráfego aéreo. Por exemplo, o objeto *aeródromo* tem a propriedade *altitude*. A definição da classe *at-attribute* (atributo de objeto da ontologia de tráfego aéreo) é a seguinte:

```
(define-class AT-ATTRIBUTE (?x)
  :def (at-thing ?x)
  :axiom-def (disjoint-decomposition
              AT-ATTRIBUTE
              (setof NUMERICAL-VALUE
                     CODE
                     AT-IDENTIFIER
                     DIRECTION)))
```

Nessa definição, a classe *at-attribute* é uma subclasse de *at-thing* que tem as subclasses disjuntas: *numerical-value*, *code*, *at-identifier* e *direction*.

Identificador de objetos de tráfego aéreo

Um identificador de objetos de tráfego aéreo é tudo o que serve para identificar um objeto de tráfego aéreo. A definição da classe *at-identifier* (identificador de objeto de tráfego aéreo) é a seguinte:

```
(define-class AT-IDENTIFIER (?x)
  :def (at-attribute ?x))
```

Nessa definição, um membro da classe *at-identifier* é um atributo da ontologia de tráfego aéreo. A classe *at-identifier* agrega todos os nomes e indicativos que são usados para identificar os objetos da ontologia de tráfego aéreo.

Para relacionar os objetos aos seus identificadores, criamos a relação *has-at-identifier* cuja definição é a seguinte:

```
(define-relation HAS-AT-IDENTIFIER (?x1)  :->  ?x2
:def (and (at-object ?x1)
          (at-identifier ?x2)))
```

Nessa definição, a relação *has-at-identifier* é um mapeamento de um objeto da ontologia de tráfego aéreo para um identificador.

Nós adotamos uma uniformização em todas as nossas ontologias para o nome das relações e as funções que sempre usam o prefixo *has-* ou o sufixo *–of* mais o nome da classe relacionada, como, por exemplo, *has-at-identifier*. Outra forma

utilizada para nomear as relações e funções é representá-las pelos nomes das classes relacionadas separados por um ponto. Por exemplo, a função *airdrome.altitude.*

Aeronave

Uma aeronave é um veículo que se desloca no espaço aéreo e é apto a transportar pessoas e coisas. A definição da classe *aircraft* (aeronave) é a seguinte:

```
(define-class AIRCRAFT (?x)
  :def (and (at-object ?x)
            (has-one ?x has-at-identifier)
            (has-one ?x has-transponder)
            (can-have-one ?x has-flight-plan)))
```

Nessa definição, um membro da classe *aircraft* é um objeto da ontologia de tráfego aéreo que tem um identificador, um equipamento transponder e pode ter um plano de vôo.

A definição da função *has-transponder* é a seguinte:

```
(define-function HAS-TRANSPONDER (?x1) :-> ?x2
  :def (and (aircraft ?x1)
            (transponder ?x2)))
```

Nessa definição, a função *has-transponder* é um mapeamento de um membro da classe *aircraft* para um membro da classe *transponder.*

A definição da função *has-flight-plan* é a seguinte:

```
(define-function HAS-FLIGHT-PLAN (?x1) :-> ?x2
  :def (and (aircraft ?x1)
            (flight-plan ?x2)))
```

Nessa definição, a função *has-flight-plan* é um mapeamento de um membro da classe *aircraft* para um membro da classe *flight-plan.*

Não expandimos a hierarquia da classe *aircraft*, mas sugerimos que poderiam ser criadas as seguintes subclasses: avião, helicóptero, planador e balão, sendo que a classe avião poderia ter as seguintes subclasses: avião a hélice e avião a jato.

Dispositivo eletrônico

O conceito *dispositivo eletrônico* agrega os dispositivos eletrônicos do domínio de tráfego aéreo. A definição da classe *electronic-device* (dispositivo eletrônico) é a seguinte:

```
(define-class ELECTRONIC-DEVICE (?x)
 :def (at-object ?x)
 :axiom-def (disjoint-decomposition
             ELECTRONIC-DEVICE
            (setof RADAR
                   TRANSPONDER
                   NAVAID)))
```

Nessa definição, a classe *electronic-device* é uma subclasse de *at-object* que tem as subclasses disjuntas: *radar* (radar), *transponder* (transponder) e *navaid* (auxílio à navegação).

Equipamento transponder

As aeronaves têm um equipamento transponder que recebe sinais do radar e os responde[3]. A definição da classe *transponder* (equipamento transponder) é a seguinte:

```
(define-class TRANSPONDER (?x)
  :def (and (electronic-device ?x)
       (can-have-one ?x has-transponder-code)
       (has-one ?x has-transponder-ident)))
```

Nessa definição, um membro da classe *transponder* é um dispositivo eletrônico (*electronic-device*) que pode ter um código (*has-transponder*-code) e tem uma identificação (*has-transponder-ident*). As definições das funções *has-transponder-code* e *has-transponder-ident* são as seguintes:

```
(define-function HAS-TRANSPONDER-CODE (?x1) :-> ?x2
  :def (and (transponder ?x1)
            (transponder-code ?x2)))

(define-function HAS-TRANSPONDER-IDENT (?x1) :-> ?x2
  :def (and (transponder ?x1)
            (transponder-ident ?x2)))
```

Nessas definições, as funções *has-transponder-code* e *has-transponder-ident* são mapeamentos de um membro da classe transponder (*transponder*) para um membro das classes, respectivamente, código transponder (*transponder-code*) e identificação transponder (*transponder-ident*).

---

[3] A palavra na língua espanhola para designar o equipamento transponder é *respondedor*

Código

O conceito *código* agrega os tipos de códigos utilizados no domínio de tráfego aéreo. A definição da classe *code* (código) é a seguinte:

```
(define-class CODE (?x)
  :def (at-attribute ?x)
  :axiom-def (disjoint-decomposition
              CODE
              (setof TRANSPONDER-CODE
                     TRANSPONDER-IDENT)))
```

Nessa definição, a classe *code* é uma subclasse de *at-attribute* que tem as subclasses disjuntas: *transponder-code* (código transponder) e *transponder-ident* (identificação transponder).

Código transponder

O código transponder é um código de quatro algarismos na base octal acionado pelo piloto no equipamento transponder. O equipamento transponder transmite esse código para os radares. A definição da classe *transponder-code* (código transponder) é a seguinte:

```
(define-class TRANSPONDER-CODE (?x)
  :def (code ?x))
```

Nessa definição, um membro da classe *transponder-code*[4] é um código (*code*). Embora não esteja na definição, os valores possíveis para os membros da classe *transponder-code* são: 0000, 0001, 0002, ..., 7775, 7776, 7777.

Identificação transponder

A identificação transponder é um código transmitido pelo equipamento transponder de uma aeronave para um radar. A definição da classe *transponder-ident* (identificação transponder) é a seguinte:

```
(define-class TRANSPONDER-IDENT (?x)
  :def (and (code ?x)
            (member ?x (setof 0
                              1))))
```

Nessa definição, um membro da classe *transponder-ident* é um código (*code*) cujos valores podem ser 1 ou 0.

Lugar

O conceito *lugar* (*place*) agrega todos os objetos da ontologia de tráfego aéreo que têm uma localização espacial bem definida. A definição da classe *place* (lugar) é a seguinte:

---

[4] Um código transponder pode ser transmitido nos modos 3A, B, C, D, e S. Em nossa ontologia, a classe código transponder é a do modo 3A que é o modo usado para os códigos da aviação civil. Os modos B e D somente são usados pela aviação militar, o modo S ainda está em fase de implantação e o modo C é utilizado apenas para enviar a altitude em que a aeronave se encontra para o radar.

```
(define-class PLACE (?x)
  :def (at-object ?x)
  :axiom-def (disjoint-decomposition
               PLACE
              (setof RUNWAY
                     AIRSPACE
                     AIRDROME
                     POINT
                     ATS-ROUTE)))
```

Nessa definição, a classe *place* é uma subclasse de *at-object* que tem as subclasses disjuntas: *runway* (pista), *airspace* (espaço aéreo), *airdrome* (aeródromo), *point* (ponto geográfico) e *ats-route* (rota-ATS).

Ponto geográfico

Em tráfego aéreo, a localização espacial de objetos é feita através de pontos geográficos. Na ontologia de tráfego aéreo, a classe *point* é um ponto geográfico em coordenadas cartesianas[5] de acordo com a projeção estereográfica[6]. A definição da classe *point* (ponto geográfico) é a seguinte:

```
(define-class POINT (?x)
  :def (and (place ?x)
            (has-one ?x has-x-coordinate)
            (has-one ?x has-y-coordinate)))
```

[5] As distâncias entre essas coordenadas são medidas em milhas náuticas. Uma milha náutica é igual ao comprimento de um minuto de meridiano terrestre (1.852 metros)

[6] Representação de objetos tridimensionais em um plano

Nessa definição, um membro da classe *point* é um lugar (*place*) que tem coordenadas x e y (*has-x-coordinate* e *has-y-coordinate*). Não criamos os conceitos *coordenada x* e *coordenada y* na ontologia de tráfego aéreo, pois podemos representá-los através de números reais. Então, definimos as funções *has-x-coordinate* e *has-y-coordinate* da seguinte forma:

```
(define-function HAS-X-COORDINATE (?x1) :-> ?x2
   :def (and (point ?x1)
             (real ?x2)))

(define-function HAS-Y-COORDINATE (?x1) :-> ?x2
  :def (and (point ?x1)
            (real ?x2)))
```

Nessas definições, as funções *has-x-coordinate* e *has-y-coordinate* são mapeamentos de um membro da classe ponto geográfico (*point*) para um número real (*real*) que representa as coordenadas cartesianas.

Vários objetos na ontologia de tráfego aéreo são localizados no espaço por um ou mais pontos geográficos. O relacionamento entre esses objetos e os pontos é feita pela relação *has-point*. A definição da relação *has-point* é a seguinte:

```
(define-relation HAS-POINT (?x1 ?x2)
  :def (and (or (airspace ?x1)
                (ats-route ?x1)
                (airdrome ?x1)
                (radar ?x1)
                (navaid ?x1))
            (point ?x2)))
```

Nessa definição, a relação *has-point* é um mapeamento de um membro das classes: espaço aéreo (*airspace*), rota-ATS (*ats-route*), aeródromo (*airdrome*), radar (*radar*) ou auxílio à navegação (*navaid*) para um membro da classe ponto geográfico (*point*).

Aeródromo

Um aeródromo é uma área definida sobre a terra ou água destinada à chegada, partida e movimentação de aeronaves. Um aeródromo pode ou não ter pistas, pois um heliponto é um aeródromo que tem apenas uma área para pousos e decolagens de helicópteros. Entretanto, um aeroporto é um aeródromo que deve necessariamente ter pelo menos uma pista. A definição da classe *airdrome* (aeródromo) é a seguinte:

```
(define-class AIRDROME (?x)
  :def (and (place ?x)
            (has-some ?x has-at-identifier)
            (has-one ?x has-point)
            (has-one ?x airdrome.altitude)
            (>= (cardinality ?x has-runway) 0)))
```

Nessa definição, um membro da classe *airdrome* é um lugar (*place*) que tem pelo menos um identificador (*has-at-identifier*), tem um ponto geográfico (*has-point*), uma altitude (*airdrome.altitude*) e pode ou não ter uma ou mais pistas (*has-runway*). A definição da relação *has-runway* é a seguinte:

```
(define-relation HAS-RUNWAY (?x1 ?x2)
  :def (and (airdrome ?x1)
            (runway ?x2)))
```

Nessa definição, a relação *has-runway* é um mapeamento de um membro da classe *airdrome* para um membro da classe *runway* (pista). A definição da função *airdrome.altitude* é a seguinte:

```
(define-function AIRDROME.ALTITUDE (?x1) :-> ?x2
  :def (and (airdrome ?x1)
            (altitude ?x2)))
```

Nessa definição, a função *airdrome.altitude* é um mapeamento de um membro da classe *airdrome* para um membro da classe *altitude*.

Pista

Uma pista é uma área retangular, pavimentada ou não, destinada ao pouso e decolagem de aeronaves[7]. A definição da classe *runway* (pista) é a seguinte:

```
(define-class RUNWAY (?x)
  :def (and (place ?x)
            (has-one ?x has-at-identifier)
            (has-one ?x runway-of)))
```

Nessa definição, um membro da classe *runway* é um lugar (*place*) que tem um identificador (*has-at-identifier*) e pertence a um aeródromo (*runway-of*). A definição da função *runway-of* é a seguinte:

```
(define-function RUNWAY-OF (?x1 ?x2)
  :def (and (runway ?x1)
            (airdrome ?x2)))
```

[7] Também existem pistas de táxi (*taxiway*) utilizadas para o deslocamento das aeronaves entre o hangar ou estacionamento e a pista de pouso e decolagem

Nessa definição, a função *runway-of* é um mapeamento de um membro da classe *runway* para um membro da classe *airdrome*. Repare que *runway-of* é a relação inversa de *has-runway*.

Valor numérico

O conceito *valor numérico* agrega os valor numéricos utilizados no domínio de tráfego aéreo. A definição da classe *numerical-value* (valor numérico) é a seguinte:

```
(define-class NUMERICAL-VALUE (?x)
  :def (and (at-attribute ?x)
            (disjoint-decomposition
             NUMERICAL-VALUE
            (setof ALTITUDE
                   FLIGHT-LEVEL
                   COURSE)))
```

Nessa definição, a classe *numerical-value* é uma subclasse de *at-attribute* que tem as subclasses disjuntas: *altitude* (altitude), *flight-level* (nível de vôo) e *course* (rumo).

Altitude

Uma altitude é a distância vertical entre um objeto e o nível médio do mar medida em pés[8]. A definição da classe *altitude* (altitude) é a seguinte:

---

8 Unidade de medida que corresponde a 30,48 centímetros.

```
(define-class ALTITUDE (?x)
  :def (and (numerical-value ?x)
            (integer ?x)
            (>= ?x -1000)
            (<= ?x 24000)))
```

Nessa definição, um membro da classe *altitude* é um valor numérico (*numerical-value*) inteiro (*integer*) maior ou igual a –1.000 pés (países abaixo do nível médio do mar como a Holanda) e menor ou igual a 24.000 pés (próximo do pico do Everest). Apesar de poder ser representada apenas por um número inteiro, resolvemos criar a classe *altitude*, pois é um conceito importante no domínio de tráfego aéreo e pode ser limitada com valores máximo e mínimo.

Espaço aéreo

Um espaço aéreo é o espaço formado por uma parte da área da atmosfera onde as aeronaves voam. Esse espaço é limitado horizontalmente por pontos e verticalmente por níveis de vôo. A definição da classe *airspace* (espaço aéreo) é a seguinte:

```
(define-class AIRSPACE (?x)
  :def (and (place ?x)
            (has-some ?x has-at-identifier)
            (has-at-least ?x has-point 3)
            (has-some ?x has-flight-level)))
```

Nessa definição, um membro da classe *airspace* é um lugar (*place*) que tem pelo menos um identificador (*has-at-identifier*), tem pelo menos três pontos geográficos (*has-point*) e tem pelo menos um nível de vôo (*has-flight-level*). A definição da relação *has-flight-level* é a seguinte:

```
(define-relation HAS-FLIGHT-LEVEL (?x1 ?x2)
  :def (and (or (airspace ?x1)
                (ats-route ?x1))
       (flight-level ?x2)))
```

Nessa definição, a relação *has-flight-level* é um mapeamento de um membro das classes *airspace* ou *ats-route* para um membro da classe *flight-level*.

Nível de vôo

Um nível de vôo é uma superfície de pressão atmosférica constante, relacionada com uma determinada referência de pressão[9] e que está separada de outras superfícies análogas por determinados intervalos de pressão. Um nível de vôo é aproximadamente uma altitude em centenas de pés em relação ao nível médio do mar. Por exemplo, o nível de vôo 100 corresponde mais ou menos à altitude de 10.000 pés, pois essa conversão não é bem exata. A definição da classe *flight-level* (nível de vôo) é a seguinte:

```
(define-class FLIGHT-LEVEL (?x)
  :def (and (numerical-value ?x)
            (integer ?x)
            (<= ?x 3000)))
```

Nessa definição, um membro da classe *flight-level* é um valor numérico (*numerical-value*) inteiro (*integer*) menor ou igual a 3.000 (onde já estão os satélites).

---

[9] 1013.2 hectopascais

Rotas ATS

As rotas ATS (*Air Traffic Service*) são as "ruas do céu" fazendo uma analogia com o tráfego de veículos nas cidades. Nesse sentido, os espaços aéreos seriam as cidades e as rotas ATS as ruas. A definição da classe *ATS-route* (rota ATS) é a seguinte:

```
(define-class ATS-ROUTE (?x)
  :def (and (place ?x)
            (has-one ?x has-at-identifier)
            (has-at-least ?x has-point 2)
            (has-some ?x has-flight-level)))
```

Nessa definição, um membro da classe *ATS-route* é um lugar (*place*) que tem um identificador (*has-at-identifier*), no mínimo dois pontos geográficos (*has*-point) e pelo menos um nível de vôo (*has-flight-level*).

As rotas ATS não podem ser confundidas com o conceito de rota, pois uma rota em tráfego aéreo é sinônimo de trajetória do vôo de uma aeronave e as rotas ATS podem ou não fazer parte dessa trajetória.

Plano de vôo

Um plano de vôo é um documento contendo informações de planejamento de vôo de uma aeronave, tais como: identificador do plano de vôo, aeródromo de partida, aeródromo de destino, velocidade, nível de vôo, etc. A definição da classe *flight-plan* (plano de vôo) é a seguinte:

```
(define-class FLIGHT-PLAN (?x)
   :def (and (at-object ?x)
             (has-one ?x has-at-identifier)
             (has-some ?x flight-plan-of)
             (has-one ?x has-departure-airdrome)
             (has-one ?x has-destination-airdrome)
             (has-one ?x flight-plan.speed)
             (has-one ?x flight-plan.flight-level)))
```

Nessa definição, um membro da classe *flight-plan* é um objeto da ontologia de tráfego aéreo (*at-object*) que tem um identificador (*has-at-identifier*), pertence a uma aeronave (*flight-plan-of*), tem um aeródromo de partida (*has-departure-airdrome*), um aeródromo de destino (*has-destination-airdrome*), uma velocidade (*flight-plan.speed*) e um nível de vôo (*flight-plan.flight-level*). Não representamos todas as informações contidas em um plano de vôo real para simplificar, mas a ontologia pode ser facilmente expandida para incorporar os novos conceitos.

A definição da função *flight-plan-of* é a seguinte:

```
(define-function FLIGHT-PLAN-OF (?x1) :-> ?x2
  :def (and (flight-plan ?x1)
            (aircraft ?x2)))
```

Nessa definição, a função *flight-plan-of* é um mapeamento de um membro da classe *flight-plan* para um membro da classe *aircraft*. As definições das funções *has-departure-airdrome* e *has-destination-airdrome* são as seguintes:

```
(define-function HAS-DEPARTURE-AIRDROME (?x1) :-> ?x2
  :def (and (flight-plan ?x1)
            (airdrome ?x2)))
```

```
(define-function HAS-DESTINATION-AIRDROME (?x1) :-> ?x2
  :def (and (flight-plan ?x1)
            (airdrome ?x2)))
```

Nessas definições, as funções *has-departure-airdrome* e *has-destination-airdrome* são mapeamentos de um membro da classe *flight-plan* para um membro da classe *airdrome*. A definição da função *flight-plan.speed* é a seguinte:

```
(define-function FLIGHT-PLAN.SPEED (?x1) :-> ?x2
  :def (and (flight-plan ?x1)
            (natural ?x2)))
```

Nessa definição, a função *flight-plan.speed* é um mapeamento de um membro da classe *flight-plan* para um membro da classe dos números naturais (*natural*). Um número natural representa a velocidade, pois em tráfego aéreo não existem velocidades negativas. A definição da função *flight-plan.flight-level* é a seguinte:

```
(define-function FLIGHT-PLAN.FLIGHT-LEVEL (?x1) :-> ?x2
  :def (and (flight-plan ?x1)
            (flight-level ?x2)))
```

Nessa definição, a função *flight-plan.flight-level* é um mapeamento de um membro da classe *flight-plan* para um membro da classe *flight-level*.

Radar

Um radar é um dispositivo eletrônico localizado por um ponto geográfico[10] utilizado para detectar as aeronaves em intervalos de tempo (ciclos) constantes. A definição da classe *radar* (radar) é a seguinte:

```
(define-class RADAR (?x)
  :def (and (electronic-device ?x)
            (has-some ?x has-at-identifier)
            (has-one ?x radar.cycle)
            (has-one ?x has-point)))
```

Nessa definição, um membro da classe *radar* é um dispositivo eletrônico (*electronic-device*) que tem pelo menos um identificador (*has-at-identifier*), um ciclo de radar[11] (*radar.cycle*) e um ponto geográfico (*has-point*). A definição da função *radar.cycle* é a seguinte:

```
(define-function RADAR.CYCLE (?x1) :-> ?x2
  :def (and (radar ?x1)
            (real ?x2)))
```

Nessa definição, a função *radar.cycle* é um mapeamento de um membro da classe *radar* para um número real (*real*) que representa o ciclo de radar.

---

[10] Apesar de existirem radares móveis para uso em operações militares

[11] Os radares usados em tráfego aéreo determinam a posição das aeronaves em azimute e distância. O período de tempo que o radar leva para passar novamente pelo mesmo azimute é chamado de ciclo de radar

Fixo

As rotas ATS passam por tipos especiais de pontos geográficos que são chamados de fixos. Um fixo é um ponto geográfico com um identificador que é usado para localização e controle das aeronaves. A definição da classe *fix* (fixo) é a seguinte:

```
(define-class FIX (?x)
  :def (and (point ?x)
            (has-one ?x has-at-identifier)))
```

Nessa definição, um membro da classe *fix* é um ponto geográfico (*point*) que tem um identificador (*has-at-identifier*).

Auxílio à navegação

Um auxílio à navegação[12] é dispositivo eletrônico que está localizado em um ponto geográfico para ajudar a navegação das aeronaves e também para balizar as rotas ATS, pois serve como referência no solo. A definição da classe *navaid* (auxílio à navegação) é a seguinte:

```
(define-class NAVAID (?x)
  :def (and (electronic-device ?x)
            (has-some ?x has-at-identifier)
            (has-one ?x has-point)))
```

[12] Como exemplos de auxílios à navegação temos o VOR (radiofarol onidirecional em VHF) e o NDB (radiofarol não-direcional) que são antenas radiotransmissoras

Nessa definição, um membro da classe *navaid* é um dispositivo eletrônico (*electronic-device*) que tem pelo menos um identificador (*has-at-identifier*) e um ponto geográfico (*has-point*).

Controle de tráfego aéreo

Um controle de tráfego aéreo é a autoridade responsável pelo serviço de controle de tráfego aéreo com a finalidade de prevenir colisões de aeronaves com outras aeronaves e obstáculos e acelerar e manter ordenado o fluxo de tráfego aéreo. Para prestar esse serviço, o controle de tráfego aéreo emite autorizações para as aeronaves. A definição da classe *at-control* (controle de tráfego aéreo) é a seguinte:

```
(define-class AT-CONTROL (?x)
   :def (and (at-object ?x)
             (has-some ?x has-at-identifier)))
```

Nessa definição, um membro da classe *at-control* é um objeto da ontologia de tráfego aéreo (*at-object*) que tem pelo menos um identificador (*has-at-identifier*).

Rumo

Um rumo (*course*) é um valor numérico inteiro que define direção e sentido. Um rumo é expresso em graus, variando de zero (norte magnético) a 359, no sentido horário. A definição da classe *course* é a seguinte:

```
(define-class COURSE (?x)
  :def (and (numerical-value ?x)
            (integer ?x)
            (>= ?x 0)
            (<= ?x 359)))
```

Nessa definição, um membro da classe *course* é um valor numérico (*numerical-value*) inteiro (*integer*) maior ou igual a zero e menor ou igual a 359.

Direção

Uma direção é um sentido para o qual algum objeto está se movendo. Por exemplo, temos uma direção *esquerda* na frase "A aeronave fez uma curva à esquerda". A definição da classe *direction* (direção) é a seguinte:

```
(define-class DIRECTION (?x)
  :def (and (at-attribute ?x)
            (member ?x (setof left
                              right))))
```

Nessa definição, um membro da classe *direction* é um atributo de objeto da ontologia de tráfego aéreo (*at-attribute*) que tem os valores *left* (esquerda) ou *right* (direita).

3.3 CONSIDERAÇÕES SOBRE A ONTOLOGIA DE TRÁFEGO AÉREO

A ontologia de tráfego aéreo apresentada contém:

- Vinte e sete classes,
- Quatro relações, e
- Catorze funções.

Seguindo as recomendações vistas na seção 2.3, definimos a ontologia na linguagem Ontolingua para aumentar sua clareza e potencial de reutilização. Não

incluímos muitas restrições semânticas, visando um engajamento ontológico mínimo.

A ontologia não cobre todos os conceitos do domínio de tráfego aéreo. Decidimos não colocar todos os conceitos devido a questões de limitação de tempo e devido à complexidade do trabalho de construção de ontologias. Entretanto, acreditamos que essa ontologia possa ser facilmente estendida para incorporar novos conceitos. A ontologia de tráfego aéreo completa definida em Ontolingua está no apêndice 01.

A construção das nossas ontologias foi um processo bastante demorado, pois foram necessários vários protótipos para atingir um bom nível de estabilidade dos seus conceitos e da sua taxonomia. A escolha dos conceitos a serem incluídos nas ontologias e sua posição dentro das taxonomias foi a parte mais demorada do nosso trabalho. É muito difícil escolher os conceitos sem ter uma aplicação em vista. No nosso caso, a implementação do sistema gerador de frases nos ajudou na escolha dos conceitos a serem incluídos nas ontologias. Entretanto, tivemos a preocupação de deixar as ontologias independentes da aplicação para posterior reutilização do conhecimento representado.

4 EVENTOS DE TRÁFEGO AÉREO

Neste capítulo, descrevemos como foi feita a representação dos eventos do domínio de tráfego aéreo. Além disso, vamos mostrar como os eventos são identificados pelo programa identificador de eventos que tem como entrada os dados obtidos a partir de um sistema de controle de tráfego aéreo e como saída uma representação dos eventos identificados.

Na próxima seção, apresentamos os chamados papéis temáticos, que permitem descrever os vários argumentos associados a um evento. Em seguida, apresentamos a ontologia de papéis temáticos que construímos. Depois, mostramos a ontologia de eventos de tráfego aéreo também construída por nós que utiliza a ontologia de papéis temáticos e a ontologia de tráfego aéreo para representar os eventos do domínio de tráfego aéreo. Finalmente, apresentamos o programa identificador de eventos de tráfego aéreo que, a partir dos dados de um sistema de controle de tráfego aéreo, representa os eventos do domínio de tráfego aéreo baseando-se nas três ontologias desenvolvidas.

4.1 PAPÉIS TEMÁTICOS

Para representar semanticamente os eventos de tráfego aéreo, utilizamos os papéis temáticos (*thematic roles*) [Per96] [All95] que são um conjunto de categorias que permitem uma representação semântica dos argumentos dos eventos. Um evento está associado a um verbo, que é o constituinte central da frase. Os papéis temáticos estabelecem as relações semânticas entre o verbo e os seus argumentos. Por exemplo, na frase "A aeronave VRG1111 pousou no aeródromo Afonso Pena", o evento está associado ao verbo *pousar* que atribui o papel temático *agente* a "aeronave VRG1111" e o papel temático *localização espacial* a "aeródromo Afonso Pena".

Fazendo uma analogia com a análise sintática na qual temos, por exemplo, as funções *sujeito* e *objeto direto*, existe uma correlação do *sujeito* a um *agente* que produziu o evento e do *objeto direto* com o *tema* que foi afetado pelo evento. Porém, os papéis temáticos são independentes dos papéis sintáticos da frase. Por exemplo, considere as frases:

[1] O radar detectou a aeronave

[2] A aeronave foi detectada pelo radar

Nessas frases, o *agente* e o *tema* são os mesmos (*radar* e *aeronave* respectivamente), mas *radar* está como sujeito em [1] e *aeronave* está como sujeito em [2]. Portanto, frases diferentes sintaticamente, mas com o mesmo significado devem ter os mesmos papéis temáticos.

Os papéis temáticos usados em nosso trabalho são: agente, tema, localização espacial, beneficiário, origem, meta e localização temporal. Esses papéis temáticos serão apresentados com mais detalhes a seguir.

Agente

É o papel desempenhado pela entidade que provoca o evento. Por exemplo, *aeronave vrg1111* assume o papel temático *agente* na frase "A aeronave vrg1111 pousou no aeródromo".

Tema

É o papel que expressa a entidade diretamente afetada pelo evento. Para os verbos transitivos, normalmente, é a resposta para a pergunta "O que foi?" complementada por um verbo no particípio. Por exemplo, para saber qual é o tema

na frase "O controlador observou o avião", fazemos a pergunta "O que foi observado?".

Localização espacial

É uma referência espacial do evento. Por exemplo, *aeródromo Afonso Pena* assume o papel temático *localização espacial* na frase "A aeronave decolou do aeródromo Afonso Pena".

Beneficiário

É aquele que recebe ou usufrui de algo em um evento. Representa para quem o evento é feito. Por exemplo, *aeronave vrg1111* assume o papel temático *beneficiário* na frase "O controle autorizou o nível de vôo 150 para a aeronave vrg1111".

Origem

É a localização original de um objeto em um evento. Por exemplo, *nível de vôo 100* assume o papel temático *origem* na frase "A aeronave começou a subir do nível de vôo 100 para o nível de vôo 150".

Destino

É o destinação final de um objeto em um evento. Por exemplo, *nível de vôo 150* assume o papel temático *destino* na frase "A aeronave começou a subir do nível de vôo 100 para o nível de vôo 150".

Localização temporal

É uma referência temporal do evento. Por exemplo, *10h 20min 30s* assume o papel temático localização temporal na frase "A aeronave decolou às 10h 20min 30s.

Apresentamos alguns tipos de papéis temáticos, mas existem outros tipos e também variações na classificação dos papéis temáticos. Para mais informações sobre o assunto, recomendamos uma consulta em [Per96] e [All95].

4.2 ONTOLOGIA DE PAPÉIS TEMÁTICOS

Nesta seção, apresentamos a ontologia de papéis temáticos. Essa ontologia é bastante simples e define os papéis temáticos e as relações entre um evento e os papéis temáticos. Nosso objetivo é poder utilizar as definições dessa ontologia na representação semântica dos eventos de tráfego aéreo.

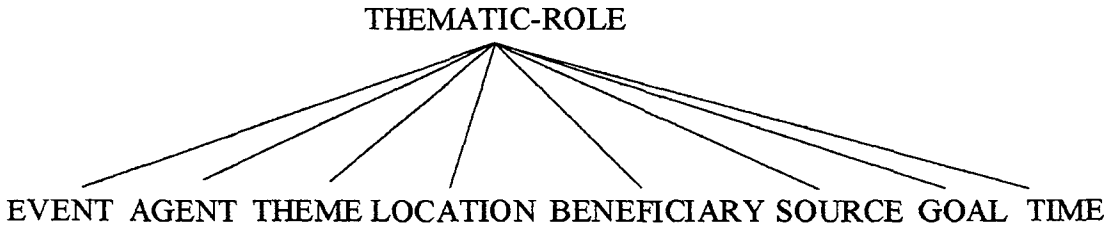


Figura 4.1 Taxonomia da ontologia de papéis temáticos

Na figura 4.1, apresentamos a taxonomia completa da ontologia de papéis temáticos. Observe que a raiz da ontologia é a classe *thematic-role.* As subclasses dessa classe correspondem ao evento e aos papéis temáticos: *agent* (agente), *theme* (tema), *location* (localização espacial), *beneficiary* (beneficiário), *source* (origem), *goal* (destino) e *time* (localização temporal).

Além dessas classes, a ontologia contém as possíveis relações entre um evento e seus papéis temáticos. Essas relações são:

- *Has-agent* que estabelece a relação entre um evento e seu agente,
- *Has-theme* que estabelece a relação entre um evento e seu tema,
- *Has-location* que estabelece a relação entre um evento e sua localização espacial,
- *Has-beneficiary* que estabelece a relação entre um evento e seu beneficiário,
- *Has-source* que estabelece a relação entre um evento e sua origem,
- *Has-goal* que estabelece a relação entre um evento e seu destino, e
- *Has-time* que estabelece a relação entre um evento e sua localização temporal.

A ontologia de papéis temáticos definida em Ontolingua está no apêndice 02.

4.3 ONTOLOGIA DE EVENTOS DE TRÁFEGO AÉREO

Nesta seção, apresentamos a ontologia de eventos de tráfego aéreo. Essa ontologia contém os eventos que são relatados nas frases geradas pelo programa gerador de frases. A semântica desses eventos é definida através da utilização das relações da ontologia de papéis temáticos apresentada na seção anterior. A ontologia de eventos de tráfego aéreo também utiliza a ontologia de tráfego aéreo (capítulo 3), pois os papéis temáticos relacionados a um evento são preenchidos por membros das classes da ontologia de tráfego aéreo. A figura 4.2 ilustra a organização hierárquica das ontologias de tráfego aéreo, papéis temáticos e eventos de tráfego aéreo.

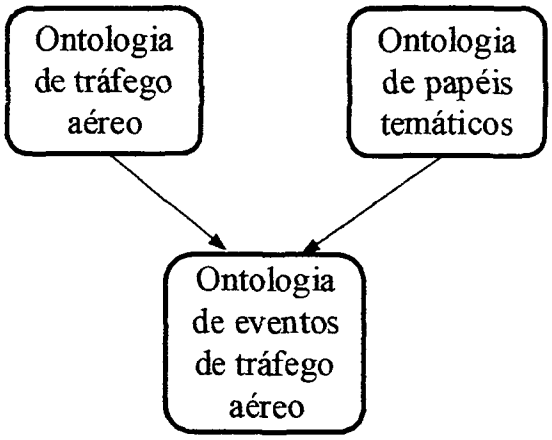


Figura 4.2 Organização hierárquica das ontologias

Na figura 4.3, apresentamos a taxonomia da ontologia de eventos de tráfego aéreo. A classe *air-traffic-event* agrega todas as outras classes da ontologia que são as subclasses disjuntas: *take-off* (decolar), *land* (pousar), *detect* (detectar), *lose* (perder), *climb* (subir), *descent* (descer), *establish* (estabilizar), *reach* (atingir), *turn* (fazer uma curva), *cross* (cruzar), *squawk* (acionar) e *clear* (autorizar). A ontologia de eventos de tráfego aéreo definida em Ontolingua está no apêndice 03.

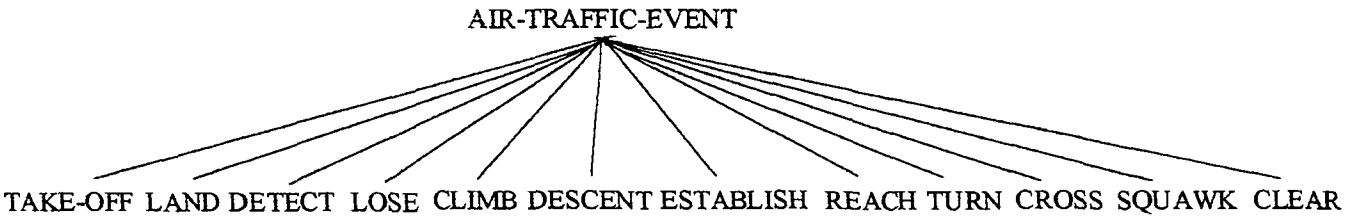


Figura 4.3 Taxonomia da ontologia de eventos de tráfego aéreo

Eventos de tráfego aéreo

A seguir, apresentamos em detalhes cada um dos eventos, quais são os papéis temáticos associados a cada evento, quais os conceitos da ontologia de tráfego aéreo que podem preencher os papéis temáticos de cada evento e, finalmente, exemplos de valores possíveis para esses papéis temáticos. Alguns eventos serão descritos juntos,

pelo fato de apresentarem os mesmos papéis temáticos e também os mesmos valores possíveis para esses papéis temáticos.

Decolar e pousar

Pela tabela 4.1, podemos ver que os eventos *decolar* e *pousar* apresentam os papéis temáticos *agent* e *location.*

Papel temático	**Conceito**	**Exemplo**
Agent	*Aircraft*	Aeronave vrg1111
Location	*Runway*	Pista dez do aeródromo Afonso Pena
	Airdrome	Aeródromo Afonso Pena

Tabela 4.1 Papéis temáticos para os eventos *decolar* e *pousar*

No caso do domínio de tráfego aéreo, um agente desses eventos deve ser uma aeronave e a localização espacial pode ser uma pista ou um aeródromo[13]. A seguir, apresentamos exemplos de frases que relatam esses eventos:

- A aeronave vrg1111 decolou.
- A aeronave vrg1111 pousou na pista dez do aeródromo Afonso Pena.
- A aeronave vrg1111 pousou no aeródromo Afonso Pena.

Detectar e perder

O evento detectar em tráfego aéreo acontece quando um radar começa a receber o retorno de um sinal enviado por ele enviado e que foi refletido por uma aeronave ou respondido pelo equipamento transponder dessa aeronave. O evento

[13] Não consideramos, por exemplo, um hidroavião que decola da água

perder acontece quando um radar pára de receber o retorno desse sinal por um tempo significativo, ou seja, um tempo maior do que o valor do ciclo do radar.

Pela tabela 4.2, podemos ver que os eventos *detectar* e *perder* apresentam os papéis temáticos *agent* e *theme*.

Papel temático	**Conceito**	**Exemplo**
Agent	*Radar*	Radar Sorocaba
Theme	*Aircraft*	Aeronave vrg1111

Tabela 4.2 Papéis temáticos para os eventos *detectar* e *perder*

No caso do domínio de tráfego aéreo, um agente desses eventos deve ser um radar[14] e o tema deve ser uma aeronave[15]. A seguir, apresentamos exemplos de frases que relatam esses eventos:

- A radar Sorocaba detectou a aeronave vrg1111.
- A aeronave vrg1111 foi detectada pelo radar Sorocaba.
- A aeronave vrg1111 foi perdida pelo radar Sorocaba.

Subir e descer

Pela tabela 4.3, podemos ver que os eventos *subir* e *descer* apresentam os papéis temáticos *agent*, *source* e *goal*.

[14] Não consideramos que satélites também podem detectar as aeronaves, pois eles ainda não são utilizados no tráfego aéreo civil

[15] Não consideramos que um radar usado em tráfego aéreo também pode detectar nuvens

Papel temático	Conceito	Exemplo
Agent	*Aircraft*	Aeronave vrg1111
Source	*Flight-level*	Nível de vôo 100
Goal	*Flight-level*	Nível de vôo 150

Tabela 4.3 Papéis temáticos para os eventos *subir* e *descer*

No caso do domínio de tráfego aéreo, um agente desses eventos deve ser uma aeronave e a origem e o destino devem ser um nível de vôo. A seguir, apresentamos exemplos de frases que relatam esses eventos:

- A aeronave vrg1111 subiu.
- A aeronave vrg1111 começou a descer para o nível de vôo 100.
- A aeronave vrg1111 subiu do nível de vôo 100 para o nível de vôo 150.
- A aeronave vrg1111 começou a descer do nível de vôo 150 para o nível de vôo 100.

Repare que a origem ou a origem e o destino fazem parte da representação semântica do evento, mas podem estar ausentes nas frases.

Estabilizar

O evento estabilizar acontece quando uma aeronave começa a manter o seu rumo constante após ter feito uma curva. Pela tabela 4.4, podemos ver que o evento *estabilizar* apresenta os papéis temáticos *agent* e *theme*.

Papel temático	Conceito	Exemplo
Agent	*Aircraft*	Aeronave vrg1111
Theme	*Course*	Rumo 100

Tabela 4.4 Papéis temáticos para o evento *estabilizar*

No caso do domínio de tráfego aéreo, um agente desse evento deve ser uma aeronave e o tema deve ser um rumo. A seguir, apresentamos um exemplo de frase que relata esse evento:

- A aeronave vrg1111 estabilizou-se no rumo 100.

Atingir

O evento atingir acontece quando uma aeronave começa a manter o seu nível de vôo constante após ter subido ou descido. Pela tabela 4.5, podemos ver que o evento *atingir* apresenta os papéis temáticos *agent* e *theme*.

Papel temático	**Conceito**	**Exemplo**
Agent	*Aircraft*	Aeronave vrg1111
Theme	*Flight-level*	Nível de vôo 150

Tabela 4.5 Papéis temáticos para o evento *atingir*

No caso do domínio de tráfego aéreo, um agente desse evento deve ser uma aeronave e o tema deve ser um nível de vôo. A seguir, apresentamos exemplos de frases que relatam esse evento:

- A aeronave vrg1111 atingiu o nível de vôo 150.
- O nível de vôo 150 foi atingido pela aeronave vrg1111.

Fazer uma curva

Pela tabela 4.6, podemos ver que o evento *fazer uma curva* apresenta os papéis temáticos *agent* e *goal*.

Papel temático	Conceito	Exemplo
Agent	*Aircraft*	Aeronave vrg1111
Goal	*Direction*	Esquerda Direita

Tabela 4.6 Papéis temáticos para o evento *fazer uma curva*

No caso do domínio de tráfego aéreo, um agente desse evento deve ser uma aeronave e o destino deve ser uma direção. A seguir, apresentamos exemplos de frases que relatam esse evento:

- A aeronave vrg1111 fez uma curva à direita.
- A aeronave vrg1111 começou a fazer uma curva.
- A aeronave vrg1111 começou a fazer uma curva à esquerda.

Cruzar

O evento cruzar acontece quando uma aeronave transpõe pontos geográficos, fixos, auxílios à navegação, aeródromos, etc. Saber quando esse evento acontece é importante para procedimentos de separação entre as aeronaves e trocas do controle de tráfego aéreo responsável pela aeronave.

Papel temático	Conceito	Exemplo
Agent	*Aircraft*	Aeronave vrg1111
Theme	*Point*	Ponto 34.5, -45.6
	Fix	Fixo Atena
	Navaid	Auxílio à navegação VOR Curitiba
	Radar	Radar Sorocaba
	Airspace	Espaço aéreo TMA Curitiba
	Airdrome	Aeródromo Afonso Pena

Tabela 4.7 Papéis temáticos para o evento *cruzar*

No contexto de nosso trabalho mostrado na tabela 4.7, consideramos que o evento *cruzar* tem os papéis temáticos *agent* e *theme*. No caso do domínio de tráfego aéreo, um agente desse evento deve ser uma aeronave e o tema deve ser um ponto geográfico, um fixo, um auxílio à navegação, um radar, um espaço aéreo ou um aeródromo. A seguir, apresentamos exemplos de frases que relatam esse evento:

- A aeronave vrg1111 cruzou o ponto 34.5, -45.6.
- O fixo atena foi cruzado pela aeronave vrg1111.

Acionar

O evento acionar[16] em tráfego aéreo acontece quando é modificado o estado do equipamento transponder de uma aeronave. Essa modificação é feita pelo piloto ao pressionar o botão da característica *ident* do transponder. O transponder então envia um sinal diferente do normal para o radar (identificação transponder). Esse sinal normalmente indica início de controle, mas também pode indicar uma falha nas comunicações. Outra possibilidade da ocorrência do evento *acionar* é a inserção[17] pelo piloto de um código no equipamento transponder da aeronave que estava em *stand-by*[18] ou com um código diferente. Utilizamos a palavra *squawk*[19] em inglês para designar esse evento.

---

[16] Em tráfego aéreo, o verbo *acionar* também pode ser usado com o significado de "pôr em funcionamento", como em "A aeronave acionou os motores", porém não corresponde ao evento *squawk* em inglês e sim *start*

[17] Por ordem do controlador de tráfego aéreo ou devido a situações como: falha de comunicações, emergência, seqüestro, etc.

[18] Modo do transponder que não envia código para o radar

[19] O verbo *squawk* em inglês usualmente significa *gritar ou grasnar* e significa *acionar* apenas no domínio do tráfego aéreo

Papel temático	Conceito	Exemplo
Agent	*Aircraft*	Aeronave vrg1111
Theme	*Transponder code*	Código transponder 3333
	Transponder ident	Identificação transponder

Tabela 4.8 Papéis temáticos para o evento *acionar*

Pela tabela 4.8, podemos ver que o evento *acionar* apresenta os papéis temáticos *agent* e *theme*. No caso do domínio de tráfego aéreo, um agente desse evento deve ser uma aeronave e o tema deve ser um código transponder ou uma identificação transponder. A seguir, apresentamos exemplos de frases que relatam esse evento:

- A aeronave vrg1111 acionou a identificação transponder.
- A aeronave vrg1111 acionou o código transponder 3333.
- O código transponder 3333 foi acionado pela aeronave vrg1111.

Autorizar

Pela tabela 4.9, podemos ver que o evento *autorizar* apresenta os papéis temáticos *agent*, *theme* e *beneficiary*.

Papel temático	Conceito	Exemplo
Agent	*At-control*	Controle Curitiba
Theme	*Flight-level*	Nível de vôo 150
	Point	Ponto 34.5, -45.6
	Fix	Fixo atena
	Navaid	Auxílio à navegação VOR Curitiba
Beneficiary	*Aircraft*	Aeronave vrg1111

Tabela 4.9 Papéis temáticos para o evento *autorizar*

No contexto de nosso trabalho, um agente desse evento deve ser um controle de tráfego aéreo, o tema deve ser um nível de vôo, um ponto geográfico, um fixo ou um auxílio à navegação e o beneficiário deve ser uma aeronave. A seguir, apresentamos exemplos de frases que relatam esse evento:

- O controle Curitiba autorizou o nível de vôo 150 para a aeronave vrg1111.
- O fixo atena foi autorizado pelo controle Curitiba para a aeronave vrg1111.

Na tabela 4.10, apresentamos uma síntese com todos os eventos e seus papéis temáticos juntamente com os conceitos da ontologia de tráfego aéreo que devem preenchê-los. Salientamos que todos os eventos em nosso trabalho têm o papel temático localização temporal que foi omitido nas tabelas apresentadas.

Evento	**Agente**	**Tema**	**Localização**	**Beneficiário**	**Origem**	**Destino**
Decolar	Aeronave		Pista Aeródromo			
Pousar	Aeronave		Pista Aeródromo			
Detectar	Radar	Aeronave				
Perder	Radar	Aeronave				
Subir	Aeronave				Nível de vôo	Nível de vôo
Descer	Aeronave				Nível de vôo	Nível de vôo
Estabilizar	Aeronave	Rumo				
Atingir	Aeronave	Nível de vôo				
Fazer uma curva	Aeronave					Direção
Cruzar	Aeronave	Ponto geográfico				

		Fixo Auxílio à navegação Radar Espaço aéreo Aeródromo				
Acionar	Aeronave	Código transponder Identificação Transponder				
Autorizar	Controle de tráfego aéreo	Nível de vôo Ponto geográfico Fixo Auxílio à navegação		Aeronave		

Tabela 4.10 Eventos de tráfego aéreo e seus respectivos papéis temáticos

4.4 PROGRAMA IDENTIFICADOR DE EVENTOS

Implementamos um programa para identificar eventos de tráfego aéreo e produzir uma representação dos eventos identificados que servem de ponto de partida para o programa gerador de frases que será apresentado no capítulo 6. A entrada do programa é um conjunto de dados de um sistema de controle de tráfego aéreo e a saída é uma representação dos eventos identificados, conforme mostra a figura 4.4. Esse programa também utiliza uma base de dados sobre tráfego aéreo baseada na ontologia de tráfego aéreo.

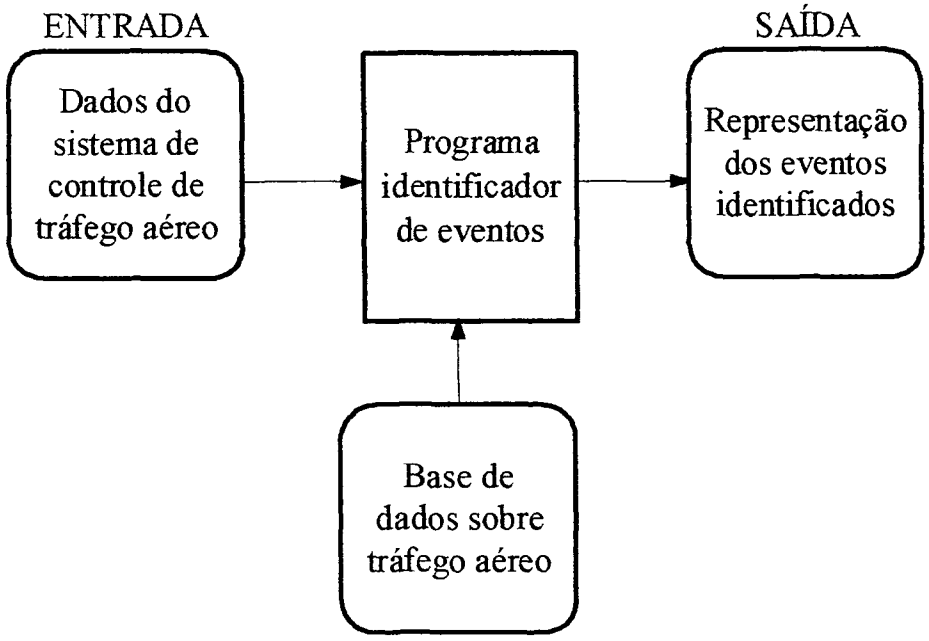


Figura 4.4 Programa identificador de eventos

O sistema de controle de tráfego aéreo tem dados sobre uma ou mais aeronaves em intervalos constantes de tempo. Em nosso trabalho, consideramos que esses dados são:

A - Identificador do plano de vôo

B - Horário (hora, minuto e segundo)

C - Coordenada cartesiana x

D - Coordenada cartesiana y

E - Velocidade

F - Rumo

G - Variação do rumo[20]

H - Código transponder

I - Nível de vôo[21]

---

[20] Quantidade de graus variados em relação à informação anterior

[21] Ou altitude em centenas de pés se uma aeronave estiver subindo ou descendo

J - Razão de subida ou descida[22]

K - Identificação transponder

L - Identificador do radar que está detectando a aeronave

M - Último ponto[23] autorizado pelo controle

N - Próximo ponto autorizado pelo controle e

O - Nível de vôo autorizado pelo controle.

Então, podemos ter para uma aeronave, conforme a tabela 4.11, o seguinte conjunto de dados:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	a1	10:20:00	0	0	180	030	0	3333	030	0	0	r1	p1	p2	040
2	a1	10:20:10	0.4	0	180	030	0	3333	030	0	0	r1	p1	p2	040
3	a1	10:20:20	0.8	0.3	180	025	-5	3333	031	6	0	r1	p1	p2	040
4	a1	10:20:30	1.2	0.2	180	020	-5	3333	032	6	0	r1	p1	p2	040
5	a1	10:20:40	1.6	0.1	180	015	-5	3333	033	6	0	r1	p1	p2	040
6	a1	10:20:50	2	0	180	010	-5	3333	034	6	1	r1	p1	p2	040
7	a1	10:21:00	2.5	0	180	005	-5	3333	035	6	1	r1	p2	p3	040
8	a1	10:21:10	3	0	180	000	-5	3333	036	6	0	r1	p2	p3	040
9	a1	10:21:20	3.5	0	180	000	0	3333	037	6	0	r1	p2	p3	040
10	a1	10:21:50	5	0	180	000	0	3333	040	6	0	r1	p2	p3	040
11	a1	10:22:00	5.5	0	180	000	0	3333	040	0	0	r1	p3	p4	040
12	a1	10:22:10	6	0	180	000	0	3333	040	0	0	r1	p3	p4	035
13	a1	10:22:20	6.5	0	170	000	0	3333	039	-6	0	r2	p3	p4	035
14	a1	10:22:30	7	0	160	000	0	4444	038	-6	0	r2	p3	p4	035
15	a1	10:22:40	7.5	0	150	000	0	4444	037	-6	0	r2	p3	p4	035

[22] Quantidade em centenas de pés que uma aeronave subiu ou desceu em um minuto

[23] Esses pontos geralmente são aeródromos, auxílios à navegação ou fixos, mas podem também ser pontos geográficos

16	a1	10:22:50	8	0	135	000	0	4444	036	-6	0	r2	p3	p4	035
17	a1	10:23:00	8.5	0	120	000	0	4444	035	-6	0	r2	p3	p4	035
18	a1	10:23:10	9	0	100	000	0	4444	035	0	0	r2	p3	p4	035

Tabela 4.11 dados do sistema de controle de tráfego aéreo

Esses dados estão associados aos conceitos da ontologia de tráfego aéreo, tais como: o identificador do plano de vôo (coluna A), as coordenadas cartesianas de um ponto geográfico (colunas C e D), o rumo (coluna F), o código transponder (coluna H), o nível de vôo (colunas I e O), a identificação transponder (coluna K) e o identificador do radar (coluna L). Repare que a informação da coluna B de cada linha (horário) é incrementada em intervalos constantes de dez segundos que normalmente correspondem ao ciclo de um radar.

Os identificadores presentes nas colunas A, L, M e N foram abreviados para a1, r1, r2, p1, p2, p3 e p4 a fim de poupar espaço. Na coluna A, o identificador do plano de vôo da aeronave a1 é "vrg1111"; na coluna L, o identificador do radar r1 é "Sorocaba" e do radar r2 é "Marília" e nas colunas M e N, o identificador do aeródromo p1 é "Afonso Pena", do fixo p2 é "atena", do fixo p3 é "maria" e do aeródromo p4 é "Bacacheri".

A seguir, mostramos um exemplo de base de dados baseada na ontologia de tráfego aéreo utilizada pelo programa identificador de eventos de tráfego aéreo:

```
Has_departure_airdrome(vrg1111,SBCT)
Has_identifier(Afonso_Pena,SBCT)
Has_identifier(Bacacheri,SBBI)
Has_altitude(SBCT,3000)
Has_altitude(SBBI,3500)
Has_runway(SBCT,30)
Has_point(TMA_Curitiba,atena)
Has_point(TMA_Curitiba,maria)
Has_point(TMA_Curitiba,varal)
```

Radar.cycle(r1,10)

Radar.cycle(r2,10)

A seguir, vamos mostrar como funciona o programa identificador de eventos e como ele representa os eventos do domínio de tráfego aéreo, ilustrando com o exemplo dos dados da tabela 4.11.

Decolar

O evento *decolar*, em geral, é identificado na primeira linha do conjunto de dados exemplificado na tabela 4.11, comparando o nível de vôo (coluna I) com a altitude do aeródromo previsto de decolagem (informação obtida na base de dados). O programa obtém o identificador da pista de decolagem utilizando a informação do rumo (coluna F).

Quando o programa identifica um evento, ele produz uma representação do evento identificado baseada na ontologia de eventos de tráfego aéreo mostrada na figura 4.5.

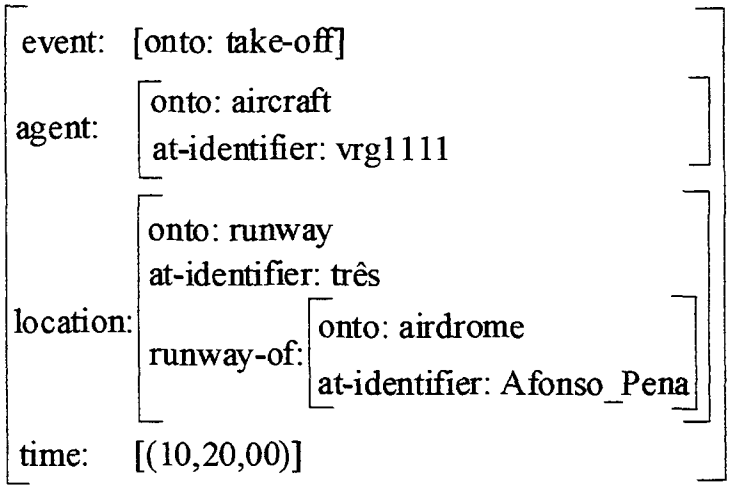


Figura 4.5 Representação do evento *decolar*

Conforme observamos na figura 4.5, o evento *decolar* tem como agente uma *aeronave* que está associada a um *plano de vôo* cujo *identificador*[24] é "vrg1111". Além disso, a localização espacial do evento é uma *pista* cujo *identificador* é "três" que, por sua vez, pertence ao *aeródromo* cujo *identificador* é "Afonso Pena"; finalmente, a localização temporal do evento é 10 horas e 20 minutos.

Perder

O evento *perder* indica que um radar que detectava uma aeronave passa a não detectá-la mais, a partir de um dado instante, por um período maior do que o seu ciclo do radar. Por exemplo, analisando as linhas 9 e 10, verificamos que houve uma perda, pois a diferença de horário entre elas foi de 30 segundos, supondo que o ciclo do radar r1 seja de 10 segundos.

Detectar

Na verdade, o evento *detectar* ocorre a todo instante. Entretanto, esse evento vale a pena ser relatado quando o radar perde a aeronave e volta a detectá-la ou quando ocorre uma mudança do radar que está detectando a aeronave. Por exemplo, na linha 13, o radar que estava detectando a aeronave era r1 e passou a ser r2. Nesse caso, o programa identificador de eventos produz a representação mostrada na figura 4.6.

---

[24] Para identificar uma aeronave em vôo é utilizado o identificador do seu plano de vôo e não o identificador da aeronave

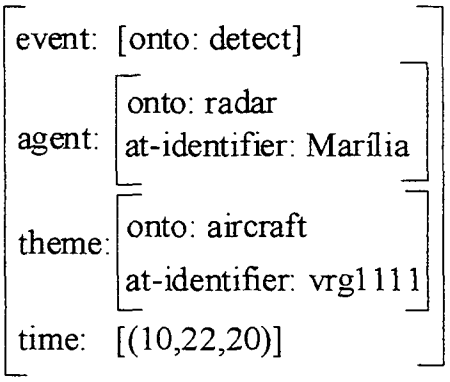


Figura 4.6 Representação do evento *detectar*

Conforme observamos na figura 4.6, o evento *detectar* tem como agente um *radar* cujo identificador é "Marília", como tema uma *aeronave* cujo *identificador* do *plano de vôo* é "vrg1111" e uma localização temporal cujo valor é 10 horas e 22 minutos e 20 segundos.

Estabilizar

Se a variação do rumo (coluna G) mudar de um valor diferente de zero para zero de uma linha para outra, isso significa que a aeronave estabilizou-se no rumo indicado pela coluna F. Por exemplo, analisando as linhas 8 e 9, verificamos que a variação do rumo mudou de –5 para zero, indicando que a aeronave estabilizou-se.

Atingir

Se a razão de subida ou descida (coluna J) mudar de um valor diferente de zero para zero de uma linha para outra, isso significa que a aeronave atingiu um nível de vôo (coluna I). Por exemplo, analisando as linhas 10 e 11, verificamos que a razão de subida ou descida variou de 6 para zero, indicando que a aeronave atingiu o nível de vôo 040.

Cruzar

Se mudar a informação sobre o último ponto autorizado (coluna M) de uma linha para outra, isso significa que a aeronave cruzou um ponto geográfico, um fixo, um auxílio à navegação, um radar ou um aeródromo identificados pelo novo valor da coluna M. Caso a aeronave tenha cruzado um ponto geográfico, um fixo ou um auxílio à navegação pertencente a um limite de espaço aéreo (informação obtida na base de dados), isso indica que a aeronave cruzou esse espaço aéreo. Por exemplo, analisando as linhas 6 e 7, verificamos que o último ponto autorizado mudou de p1 para p2. Supondo que p2 é um fixo que pertence ao limite de um espaço aéreo, isso permite constatar que a aeronave cruzou esse espaço aéreo.

Acionar

O evento *acionar* é identificado em duas situações. Na primeira, se a identificação transponder (coluna K) mudar de zero para um de uma linha para outra, isso significa que a aeronave acionou a identificação. Na segunda situação, se o código transponder (coluna H) for modificado, isso significa que a aeronave acionou o código constante nessa coluna. Por exemplo, considere a primeira situação. Analisando as linhas 5 e 6, verificamos que a identificação transponder variou de zero para um. Repare que na linha 7 a identificação transponder continuou valendo um. Apesar disso, o evento *acionar* é pontual, pois a identificação transponder de uma aeronave só pode ser acionada por até trinta segundos.

Autorizar

O evento *autorizar* é identificado em duas situações. Na primeira, se mudar o valor da coluna que representa o nível de vôo autorizado (coluna O) de uma linha para outra, isso significa que o controle autorizou esse nível de vôo para a aeronave

cujo indicativo do plano de vôo está na coluna A. Na segunda situação, caso o valor modificado seja o da coluna N, isso significa que foi autorizado o ponto geográfico, o aeródromo, o fixo ou o auxílio à navegação constantes nessa coluna na segunda linha analisada. Por exemplo, analisando as linhas 11 e 12, verificamos que foi autorizado o nível de vôo 035 para a aeronave vrg1111; analisando as linhas 6 e 7, verificamos que foi autorizado o ponto p3 (fixo Maria) para a aeronave vrg1111. Nesse caso, o programa identificador de eventos produz a representação mostrada na figura 4.7. O identificador do controle de tráfego aéreo para todas as frases que contenham esse conceito é *Curitiba*, pois supomos que o programa identificador de eventos é utilizado apenas pelo controle *Curitiba*.

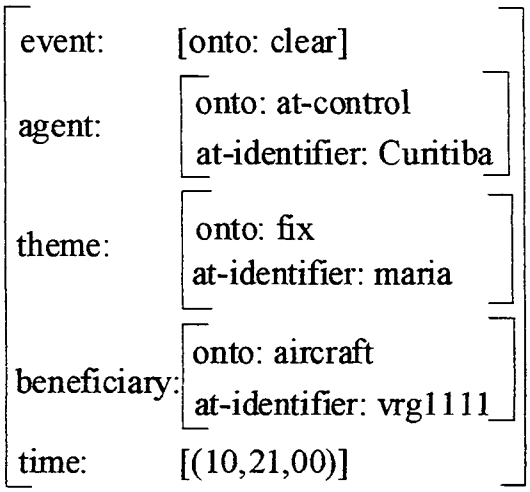


Figura 4.7 Representação do evento *autorizar*

Conforme observamos na figura 4.7, o evento *autorizar* tem como agente um *controle de tráfego aéreo* cujo identificador é "Curitiba", como tema um *fixo* cujo *identificador* é "maria", como beneficiário uma *aeronave* cujo *identificador* do *plano de vôo* é "vrg1111" e uma localização temporal cujo valor é 10 horas e 21 minutos.

Subir e descer

A maioria dos eventos identificados podem ser considerados como pontuais, pois são identificados a partir de mudanças em informações de duas linhas consecutivas do conjunto de dados. Porém, alguns eventos apresentam uma duração temporal e eles podem ser identificados considerando-se várias linhas consecutivas. Por exemplo, verificamos a subida da aeronave a partir da linha 2 até a linha 10, pois o valor da coluna I que indica o nível de vôo foi aumentando.

Entretanto, em nosso trabalho fizemos uma simplificação, pois vamos considerar apenas o momento inicial dos eventos com duração temporal. Dessa forma, a identificação do evento *subir* é feita analisando apenas duas linhas consecutivas. Então, se a razão de subida ou descida (coluna J) variar de zero para um valor positivo de uma linha para outra, isso significa que a aeronave começou a subir. Caso a informação dessa coluna variar de zero para um valor negativo, isso significa que a aeronave começou a descer. Por exemplo, analisando as linhas 2 e 3, verificamos que a razão de subida variou de zero para 6, indicando que a aeronave começou a subir; sendo que essa subida é constatada até a linha 10. Para esse evento, o programa produz a representação mostrada a seguir na figura 4.8:

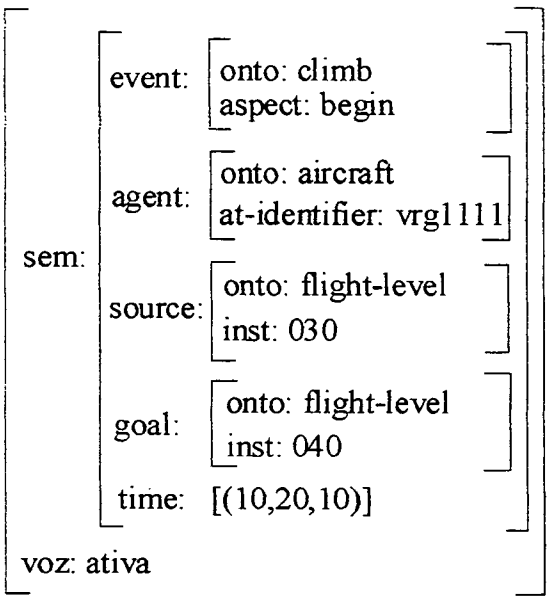


Figura 4.8 Representação do evento *subir*

Na figura 4.8, observamos que associado ao evento *subir*, temos além de seu nome, o parâmetro aspecto[25] (*aspect*). Esse parâmetro indica, para eventos duradouros, qual o momento do evento que está sendo considerado. Nesse caso, o valor do parâmetro *aspect* é *begin*, indicando que o momento considerado do evento é o momento inicial. Repare também na figura 4.8, que o evento *subir* tem como agente uma aeronave cujo *identificador do plano de vôo* é "vrg1111", como origem um *nível de vôo* cuja instância é 030, como destino um *nível de vôo* cuja instância é 040 e uma localização temporal igual a 10 horas, 20 minutos e 10 segundos.

Fazer uma curva

O processo de identificação do evento *fazer uma curva* é semelhante ao dos eventos *subir* e *descer* no que se refere à duração. Ele também é identificado no seu momento inicial. A diferença é que se houver a alteração do valor da coluna G (variação de rumo) de zero para um valor positivo, isso significa que a aeronave começou a fazer uma curva à direita. Se houver alteração dessa coluna de zero para um valor negativo, isso significa que a aeronave começou a fazer uma curva à esquerda. Por exemplo, analisando as linhas 2 e 3, verificamos que a variação de rumo variou de zero para –5, o que significa que a aeronave começou a fazer uma curva à esquerda.

Pousar

Em geral, o evento *pousar*, é identificado na última linha do conjunto de dados exemplificado na tabela 4.11. Isso é feito comparando o valor do nível de vôo

---

[25] Em lingüística, o aspecto do verbo expressa o grau de desenvolvimento ou de duração do evento expresso pelo verbo. Consideramos somente o aspecto que denota o evento no seu momento inicial

da última linha com a informação sobre a altitude do aeródromo, obtida na base de dados. O aeródromo no qual a aeronave pousou pode ser obtido pela coluna N que representa o último ponto autorizado. Por exemplo, para identificar o evento *pousar*, o programa analisa a linha 18, verifica que o ponto p4 corresponde ao aeródromo Bacacheri, compara a altitude desse aeródromo com o nível de vôo constante nessa linha e, por fim, obtém o identificador da pista pelo valor do rumo (coluna 0).

4.5 CONSIDERAÇÕES FINAIS

Apresentamos neste capítulo o conceito de papéis temáticos que são utilizados para representar os argumentos semânticos associados a um evento. Para isso definimos uma ontologia de papéis temáticos. Essa ontologia é bastante simples e genérica, podendo ser utilizada em outras aplicações e outros domínios.

Também definimos uma ontologia de eventos para o domínio de tráfego aéreo, delimitando os papéis temáticos desses eventos e o preenchimento desses papéis temáticos por conceitos definidos na ontologia de tráfego aéreo. Essa ontologia não está completa, pois não representamos todos os eventos possíveis do domínio de tráfego aéreo, mas sim apenas aqueles considerados relevantes para nossa aplicação. Entretanto, a ontologia pode ser facilmente estendida de forma que outros eventos sejam considerados. Além disso, os papéis temáticos associados aos eventos de tráfego aéreo podem ser preenchidos por conceitos que não estão presentes na ontologia de tráfego aéreo. Por exemplo, pode-se incluir outras origens e destinos para os eventos *subir* e *descer*, tais como: descer *para pouso*, subir ou descer *para uma altitude*, etc. Também o evento *autorizar* pode ter outros temas que não constam na ontologia de tráfego aéreo como, por exemplo, autorização *de alijamento de combustível* ou *para ligar os motores*.

Finalmente, mostramos como funciona o programa identificador de eventos de tráfego aéreo. A partir de dados gerados por um sistema de controle de tráfego aéreo, esse programa identifica e representa em ordem cronológica os eventos mais

significativos associados a uma ou mais aeronaves. A representação dos eventos é feita utilizando-se a ontologia de papéis temáticos, a ontologia de eventos de tráfego aéreo e a ontologia de tráfego aéreo. Essa representação será o ponto de partida para o programa gerador de frases que relata os eventos de tráfego aéreo identificados.

No próximo capítulo, apresentamos conceitos básicos sobre gramáticas para processamento de língua natural que serão necessários para entender a gramática do gerador de frases que será apresentado no capítulo 6.

5 GRAMÁTICAS PARA PROCESSAMENTO DE LÍNGUA NATURAL

O processamento de língua natural (PLN), também conhecido como processamento de linguagem natural, é um subcampo da inteligência artificial e da lingüística computacional que utiliza técnicas computacionais que processam a língua humana falada e escrita.

O que distingue as aplicações na área de processamento de língua natural de outros sistemas de processamento de dados é que as aplicações na área de PLN utilizam conhecimento sobre a língua.

O conhecimento sobre uma língua engloba o estudo de sua fonética, morfologia, sintaxe, semântica, discurso, etc. Em nosso trabalho, tratamos apenas a sintaxe e a semântica.

Para construir uma frase, há uma organização (forma) de seus componentes. Essa organização é denominada **sintaxe** que significa, em grego, "montagem". Se identificarmos as categorias das palavras (substantivo, verbo, artigo, preposição, pronome, etc.), podemos combiná-las sintaticamente para formar uma frase. A estrutura da sintaxe é uma árvore cujas palavras são as folhas e os nodos internos as categorias.

A expressão **semântica** tem origem grega e quer dizer "significado". A semântica tem por objetivo a identificação do significado das palavras e como elas se combinam para formar o significado das frases. Por exemplo, as palavras *planador, balão* e *radar* têm a propriedade sintática *gênero masculino* em comum, mas apenas *radar* tem a propriedade semântica *tipo de dispositivo eletrônico.* Por meio dessa propriedade semântica, concluímos que a frase "O radar está desligado" está correta, mas "O radar sorriu" não faz sentido, pois apenas os termos que têm a propriedade semântica *ser humano* podem estar relacionadas ao ato de sorrir.

Nas próximas seções, apresentamos a base teórica para entender a gramática do gerador de frases que será apresentado no próximo capítulo.

5.1 GRAMÁTICA LIVRE DE CONTEXTO

De acordo com [Sag99], uma **gramática** é a generalização da estrutura de sentenças bem formadas, através da definição de regras. O propósito de uma gramática é listar padrões de sentenças bem formadas baseadas em categorias gramaticais.

Uma **gramática livre de contexto** (GLC) é um formalismo que consiste em um conjunto de regras que definem quais são as cadeias de símbolos válidos em uma língua. Uma GLC possui dois tipos de símbolos: terminais e não-terminais. Os símbolos terminais correspondem a palavras da língua ("o", "controle", "autorizou", etc.) e os símbolos não-terminais são aqueles que expressam generalizações sobre como os símbolos podem ser agrupados.

Em cada regra de uma GLC, o item no lado direito da seta ("-->") é uma lista de um ou mais símbolos terminais e não-terminais separados por vírgula. À esquerda da seta, o item é um símbolo não-terminal associado à categoria gramatical da palavra. Por exemplo, observe um exemplo simples de regra:

N --> aeronave
V --> pousou
S --> N, V

Nesse exemplo, N (substantivo), V (verbo) e S (frase) são símbolos não-terminais e *aeronave* e *pousou* são símbolos terminais. Outra forma de reconhecer um símbolo terminal é olhando a árvore que representa a estrutura sintática de uma frase. Observe na figura 5.1 a estrutura sintática da frase "A aeronave pousou":

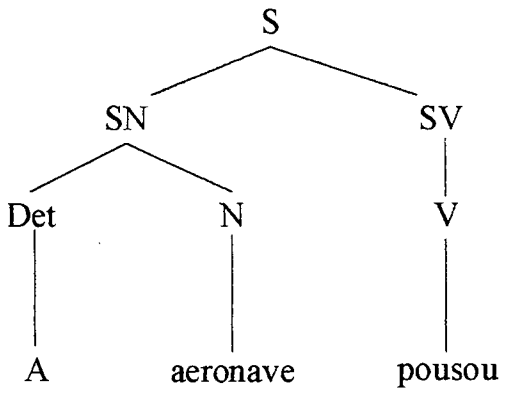


Figura 5.1 Estrutura sintática da frase "A aeronave pousou"

Na estrutura sintática da figura 5.1, os símbolos que estão no mais baixo nível (as folhas) são símbolos terminais e os demais (S, SN, SV, Det, N e V) são símbolos não-terminais.

Na próxima seção, apresentamos uma gramática básica do português feita com base na gramática livre de contexto.

5.2 GRAMÁTICA BÁSICA DO PORTUGUÊS

Nesta seção, apresentamos uma gramática básica do português. Inicialmente, apresentamos o conceito de sintagma e depois os tipos de sintagma na língua portuguesa. A explicação dos sintagmas foram baseadas em [Gag02] e [Per96].

Um **sintagma** é uma seqüência de palavras que formam uma unidade significativa com função específica. A palavra *sintagma* vem do grego e significa "coisa alinhada com outra". Por exemplo, vamos analisar as seguintes frases divididas em dois blocos:

1° bloco	**2° bloco**
O avião	sobrevoou o lago
Uma aeronave	pousou no aeroporto
Ele	ligou os motores

Essa divisão demonstra que os elementos de cada um dos blocos são equivalentes o que denota suas funções específicas. Portanto, sintaticamente podem ser substituídos um pelo outro.

Os tipos de sintagmas em português são: sintagma nominal, preposicional, adjetival e verbal e seus núcleos, respectivamente, são: um substantivo, uma preposição, um adjetivo e um verbo.

Os sintagmas podem se agrupar para formar outros sintagmas. Por exemplo, na frase "O radar detectou a aeronave" o sintagma nominal *a aeronave* agrupada com o verbo *detectou* forma o sintagma verbal *detectou a aeronave*. A frase é formada pela combinação desse sintagma verbal com *o radar* que é um sintagma nominal.

Já vimos quais são os tipos de sintagmas em português e seus núcleos. Agora, apresentamos mais detalhes sobre cada um deles.

Sintagma nominal

Um sintagma nominal é aquele que pode ser sujeito de alguma oração [Per96]. Por exemplo, na frase "Essa aeronave é um helicóptero", *essa aeronave* é o sujeito e é, portanto, um sintagma nominal e *um helicóptero* também é um sintagma nominal, pois, embora não seja o sujeito nessa frase, pode ser sujeito em outra frase.

Como já vimos, o núcleo de um sintagma nominal é um substantivo. Os substantivos normalmente estão acompanhados de palavras que especificam o seu sentido. Essas palavras são chamadas de determinantes (Det) e em português são: um artigo, um numeral, um pronome possessivo, etc. Por exemplo, são determinantes: a, o, um, uma, dois, meu, etc. Um sintagma nominal simples é formado por um determinante mais um substantivo.

Porém, além do determinante, no sintagma nominal podem aparecer os modificadores (Mod) que são palavras que modificam o significado do sintagma

nominal. Por exemplo, *vrg1111* em "A aeronave *vrg1111* decolou". Um modificador pode ser um sintagma adjetival (SA), um nome próprio (NP), um sintagma preposicional (SP) ou pode não aparecer. Por exemplo, o modificador é um SA em "aeronave *grande*", um NP em "controle *Curitiba*", um SP em "pista *do aeroporto Bacacheri*" e não aparece em "o radar".

Além desse tipo de sintagma nominal, podemos ter sintagmas nominais formados por pronomes (Pron). Por exemplo, o pronome *ele* em "*ele* partiu" é um sintagma nominal. Eis então, algumas regras que expressam estruturas possíveis para sintagmas nominais (SN) e modificadores (Mod):

SN --> Det, N, Mod, Mod
SN --> Pron
Mod --> SA
Mod --> SP
Mod --> NP
Mod --> []

Na última regra, percebemos que o modificador pode não aparecer, isto é, pode ser vazio ("[]").

Em [Per96], há um estudo bastante completo da estrutura do sintagma nominal.

Sintagma preposicional

Um sintagma preposicional é aquele formado por uma preposição (Prep) seguida de um sintagma nominal (SN) ou um sintagma verbal (SV). Por exemplo, *para o nível de vôo 150* é um sintagma preposicional formado pela preposição *para* seguida de um SN e *de voar* é um sintagma preposicional formado pela preposição *de* seguida de um SV. Portanto, as regras para o sintagma preposicional são:

SP --> Prep, SN

SP --> Prep, SV

Sintagma adjetival

Um sintagma adjetival é aquele que contém um adjetivo que pode ser intensificado por um advérbio como *bem* em "bem alto" ou complementada por um sintagma preposicional como *à saúde* em "prejudicial à saúde". Então, temos as seguintes regras:

SA --> A

SA --> Adv, A

SA --> A, SP

Sintagma verbal

Um sintagma verbal é aquele cujo núcleo é um verbo que pode ou não ter um complemento. Por exemplo, o verbo *detectar* aceita como complemento um sintagma nominal. Então, a gramática deve conter uma regra que permite combinar um verbo e um sintagma nominal para formar um sintagma verbal. Então, eis a regra:

SV --> V, SN

Nessa regra, o sintagma verbal (SV) é formado por um verbo (V) e um sintagma nominal. Para determinar a estrutura do sintagma, é preciso um dicionário para relacionar as palavras com as categorias. Observe um exemplo simples de entradas do dicionário:

V --> detectou

N --> aeronave

Det --> a

Nessas entradas, as palavras *detectou*, *aeronave* e *a* são categorizadas, respectivamente, como verbo (V), substantivo (N) e determinante (Det).

Com essa gramática e esse dicionário, podemos formar a estrutura sintática para o sintagma verbal *detectou a aeronave* apresentado na figura 5.2:

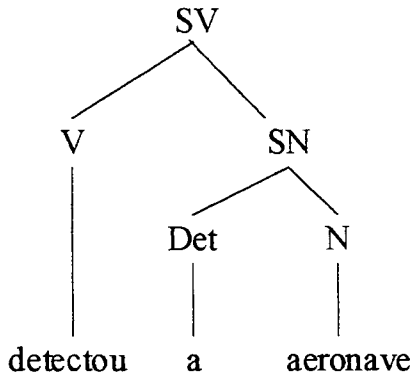


Figura 5.2 Estrutura sintática do sintagma verbal "detectou a aeronave"

Os verbos que exigem ou não um complemento são divididos nos que não aceitam complemento e os que aceitam como complemento: um SN, um SP, um pronome (Pron) seguido de um SP, um SN seguido de um SP e dois SP seguidos. Observe os exemplos da tabela 5.1:

Regra	**Exemplo**
SV --> V	$[\text{pousou}]_V$
SV --> V,SP	$[\text{pousou}]_V$ $[\text{na pista}]_{SP}$
SV --> V	$[\text{decolou}]_V$
SV --> V,SP	$[\text{decolou}]_V$ $[\text{da pista 12}]_{SP}$

SV --> V,SN	[detectou]$_V$ [uma aeronave]$_{SN}$
SV --> V,SN	[acionou]$_V$ [a identificação transponder]$_{SN}$
SV --> V, Pron, SP	[estabilizou]$_V$ [-se]$_{Pron}$ [no rumo 150]$_{SP}$
SV --> V, SN, SP	[autorizou]$_V$ [o nível de vôo 150]$_{SN}$ [para a aeronave]$_{SP}$
SV --> V, SP	[subiu]$_V$ $_{SN}$ [para o nível de vôo 100]$_{SP}$
SV --> V, SP, SP	[subiu]$_V$ [do nível de vôo 100]$_{SN}$ [para o nível de vôo 150]$_{SP}$

Tabela 5.1 Regras e exemplos de sintagmas verbais

Conforme vemos na tabela 5.1, cada tipo de verbo tem a sua regência, indicando qual o tipo de complemento exigido pelo verbo. Por exemplo, os verbos *pousar* e *decolar* não aceitam complemento ou aceitam um SP, os verbos *detectar* e *acionar* aceitam somente um SN, o verbo *estabilizar* aceita um pronome seguido de um SP, o verbo *autorizar* aceita um SN seguido de um SP e o verbo *subir* aceita um ou dois SP.

Com esses exemplos, podemos ver que existem restrições sobre o tipo de complemento do verbo aceito e sobre a ordem desses complementos. Essas restrições são conhecidas por subcategorização dos verbos. Se subcategorizarmos os verbos, então, seqüências como "decolou uma pista" e "detectou do aeronave" não são aceitas como um sintagma verbal. Na seção 5.3.2, apresentamos como é feito o tratamento da subcategorização dos verbos utilizando a gramática de unificação.

Os sintagmas verbais também podem ser formados por verbos auxiliares. Por exemplo, a regra mostrada a seguir trata frases na voz passiva:

SV --> Aux_ser, V, SP

Essa regra trata, por exemplo, a frase "A aeronave foi detectada pelo radar" cuja estrutura sintática é mostrada na figura 5.3:

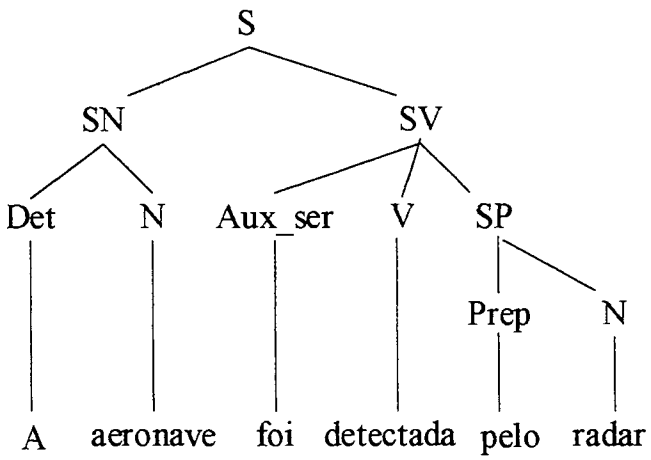


Figura 5.3 Estrutura sintática de uma frase com verbo auxiliar

Nessa estrutura, temos o sintagma verbal formado por um verbo auxiliar (ser) e um verbo principal (detectar) na forma verbal particípio.

Existem também sintagmas verbais com verbos semi-auxiliares. Um verbo semi-auxiliar é aquele que pode ser o verbo principal de uma frase e também um verbo auxiliar. Por exemplo, o verbo *começar* é principal na frase "O vôo *começou*" e auxiliar na frase "O avião *começou* a subir" cuja estrutura sintática é apresentada, a seguir, na figura 5.4:

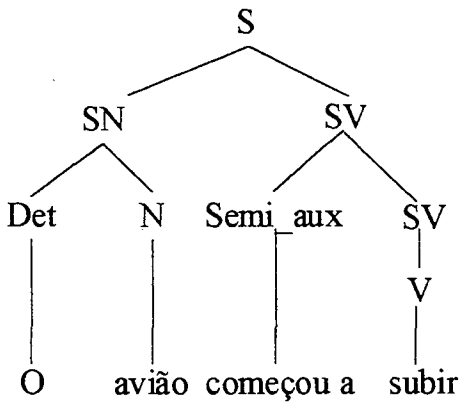


Figura 5.4 Estrutura sintática de uma frase com verbo semi-auxiliar

Repare na figura 5.4 que o sintagma verbal de mais alto nível é formado pelo verbo semi-auxiliar mais um sintagma verbal cujo núcleo é o verbo principal.

Um sintagma verbal também pode conter verbos de ligação (Lig), como: ser, estar, parecer, permanecer, etc. Nesse caso, o que vem depois do verbo é um sintagma nominal ou preposicional que qualifica ou identifica o sujeito do verbo. Considere os seguintes exemplos:

A cor do céu *é* azul.
A aeronave *está* no aeroporto de Porto Alegre.

Então, é preciso definir mais duas regras:

SV --> Lig, SN
SV --> Lig, SP

Existem outras regras para os sintagmas na língua portuguesa. Essas regras podem ser vistas em [Per96].

5.3 GRAMÁTICA DE UNIFICAÇÃO

Uma gramática de unificação (GU) é uma GLC que utiliza estruturas de traços. Esses traços permitem associar informações às categorias gramaticais, de forma a reduzir o número de regras na gramática. Nesta seção, apresentamos como podem ser expressas as regras para concordância e subcategorização utilizando a gramática de unificação. Também vamos mostrar como é feita a unificação dos traços em uma GU.

5.3.1 Concordância

Vamos agora tratar o problema da concordância em nossa gramática, pois até agora, ela aceita uma seqüência de palavras como *as aeronave*. Uma solução

simples para esse problema é multiplicar as regras. Por exemplo, considere a regra simples:

SN --> Det, N

Analisando essa regra, para que a concordância ficasse correta, seria necessário substituir a regra original pelo seguinte conjunto de regras e dicionário (onde f=feminino, m=masculino, s=singular e p=plural):

SN_ms --> Det_ms, N_ms
SN_mp --> Det_mp, N_mp
SN_fs --> Det_fs, N_fs
SN_fp --> Det_fp, N_fp

Det_fs --> a
Det_ms --> o
Det_fp --> as
Det_mp --> os
N_fs --> aeronave
N_fp --> aeronaves

Com essas novas regras e as novas entradas no dicionário não será possível interpretar como sintagma nominal a seqüência *as aeronave*, pois nenhuma das novas regras permite combinar um Det_fp com um N_fs. Essa solução não é a mais apropriada, pois há um número muito elevado de regras. Então, para tratar esse problema em nossa gramática, utilizamos as estruturas conhecidas por traços.

Os **traços** são um conjunto de atributos que permitem acrescentar informações sobre as categorias gramaticais nas regras. Por exemplo, caracterizamos a palavra *aeronave* como um substantivo (N) feminino (traço *gênero*) que está no

singular (traço *número*). Essa palavra tem a entrada do dicionário mostrada a seguir na figura 5.5:

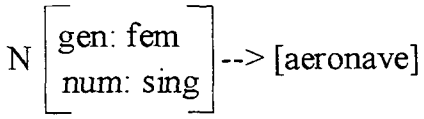


Figura 5.5 Entrada do dicionário com traços para a palavra *aeronave*

Nessa entrada, o traço *gen* representa o gênero e o traço *num* representa o número.

O tratamento da concordância com a inclusão dos traços nas regras gramaticais impede a multiplicação das regras. Por exemplo, considere novamente a regra:

SN --> Det, N

Associamos a cada item dessa regra (SN, Det e N) uma estrutura que atribui valores a esses traços. As entradas do dicionário também têm valores atribuídos aos traços. Por exemplo, a entrada do dicionário para a palavra *as* é mostrada na figura 5.6:

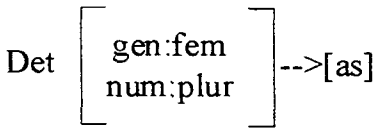


Figura 5.6 Entrada do dicionário com traços para a palavra *as*

Na figura 5.6, a palavra *as* é um determinante do gênero feminino que está no plural. A regra SN --> Det, N com a utilização de traços é mostrada na figura 5.7:

$$SN\begin{bmatrix} gen{:}G \\ num{:}N \end{bmatrix} \text{-->} Det\begin{bmatrix} gen{:}G \\ num{:}N \end{bmatrix}, N\begin{bmatrix} gen{:}G \\ num{:}N \end{bmatrix}$$

Figura 5.7 Regra gramatical com traços para sintagma nominal

Na figura 5.7, vemos que a regra contém variáveis repetidas em vários lugares. Assim que uma variável é instanciada, o seu valor é colocado em todo lugar onde ela aparece (será unificada). Por exemplo, para o sintagma nominal *as aeronave*, quando o determinante *as* é instanciado, as variáveis G e N da regra recebem os valores *fem* (feminino) e *plur* (plural). Esses valores são colocados na estrutura de traços do determinante (Det) e do substantivo (N) que são os constituintes da regra. Como a entrada do dicionário para a palavra *aeronave* tem o traço *num* com o valor *sing* (singular), a regra não se unifica, ou seja, o constituinte N não tem o mesmo valor no traço *num* que o constituinte Det. Portanto, essa seqüência não é válida de acordo com a gramática. Veremos mais sobre o processo de unificação na seção 5.3.3.

5.3.2 Subcategorização

Como vimos na seção 5.2, os verbos são subcategorizados pelo seu complemento. Por exemplo, o verbo *acionar* que admite como complemento um sintagma nominal é da subcategoria *SN*.

Para tratar esse problema também podemos utilizar os traços. Por exemplo, considere os verbos *pensar* que aceita somente um SP como complemento, *comprar* que se combina com um SN e *oferecer* que aceita um SN seguido de um SP:

SV --> [pensou] $_V$ [em uma solução] $_{SP}$

SV --> [comprou] $_V$ [uma casa] $_{SN}$

SV --> [ofereceu] $_V$ [um presente] $_{SN}$ [ao professor] $_{SP}$

Nesses exemplos, vemos que a subcategorização impõe restrições sobre o tipo de constituinte aceito e sobre a ordenação desses constituintes. Então, o mais natural para representar esse fenômeno é o uso de um traço *subcat* ao qual será associado uma lista de constituintes aceitos pelo verbo. Por exemplo, veja a parte das entradas do dicionário que tratam a subcategorização para os verbos *pensar*, *comprar* e *oferecer*:

V [subcat:<sp>] --> [pensou]
V [subcat:<sn>] --> [comprou]
V [subcat:<sn,sp>] --> [ofereceu]

Nessas regras, percebemos que o verbo *pensar* admite como complemento um SP. Então, um sintagma verbal que contenha o verbo *pensar* só estará correto se for seguido por um SP. A seguir, apresentamos a regra com a estrutura de traços para o tratamento do sintagma verbal da subcategoria *SP*:

SV --> V [subcat:<sp>], SP

Nessa regra, um sintagma verbal cujo núcleo é um verbo da subcategoria *SP* é formado pelo verbo seguido de um sintagma preposicional. Dessa forma, a regra não aceita sintagmas verbais formadas por outros tipos de complementos. Então, podemos ver facilmente que uma seqüência como *pensou uma solução* não é um sintagma verbal válido.

Os substantivos também admitem a subcategorização para seus modificadores. Por exemplo, no sintagma nominal *pista dez do aeródromo*, temos os modificadores *dez* e *do aeródromo* para o substantivo *pista*. Nesse caso, *dez* cumpre o papel de nome próprio (np) e *do aeródromo* é um sintagma preposicional. Então, a entrada do dicionário com o traço de subcategoria para a palavra *pista* é a seguinte:

N [subcat:<np,sp>] --> [pista]

Em nosso trabalho, utilizamos uma forma mais elegante para representar as entradas do dicionário do nosso gerador de frases. Então, a entrada apresentada anteriormente fica da seguinte forma:

< **pista,** N [subcat:<np,sp>] >

De acordo com essa entrada do dicionário, o primeiro modificador é um nome próprio (np) e o segundo é um sintagma preposicional (sp). Então, os sintagmas nominais que contenham a palavra *pista* terão um nome próprio e um sintagma preposicional, nessa seqüência, como modificadores. Subcategorizando os substantivos, em vez de termos as seguintes regras gramaticais para tratar um sintagma nominal com um substantivo da subcategoria <np,sp>:

SN --> Det, N, Mod, Mod
Mod --> NP
Mod --> SP

Podemos ter apenas a seguinte regra:

SN --> Det, N [subcat:<np,sp>], NP, SP

Adotamos essa simplificação nas regras gramaticais de nosso gerador de frases que será apresentado no próximo capítulo.

5.3.3 Unificação

Os valores dos traços são passados através das regras da gramática por meio da unificação. O algoritmo de unificação funciona da seguinte maneira:

- Suponhamos duas estruturas de traços E_1 e E_2;
- Se um mesmo traço aparece nas duas estruturas, os valores atribuídos a esse traço nas duas estruturas devem ser unificados;
- Se um traço t aparece com o valor v em uma estrutura e esse traço não existe na outra estrutura, o traço e seu valor serão acrescentados na estrutura que não contém esse traço.

Esse algoritmo é recursivo, pois o valor atribuído a um traço pode ser uma estrutura de traços. Por exemplo, a unificação dos traços ocorre da seguinte forma (onde t1 e t2 são traços, v1, v2 e v3 são valores e V é uma variável):

[t1: V] unificado com [t1: v1] resulta em [t1: v1]

[t1: v1] unificado com [t2: v2] resulta em $\begin{bmatrix} t1:v1 \\ t2:v2 \end{bmatrix}$

Não é possível unificar $\begin{bmatrix} t1:\begin{bmatrix} t3:V \\ t4:v1 \end{bmatrix} \\ t2:V \end{bmatrix}$ com $\begin{bmatrix} t1:[t3:v3] \\ t2:v2 \end{bmatrix}$

O último exemplo não é unificável, porque na estrutura da esquerda, a variável V não pode receber os valores a serem atribuídos aos traço t2 e t3 que são v2 e v3.

5.4 GRAMÁTICAS EM PROLOG

Em PROLOG, é fácil representar gramáticas, pois essa linguagem foi originalmente projetada para o processamento de língua natural. Por exemplo, considere a seguinte regra gramatical:

S --> SN, SV

Em PROLOG, essa regra poderia ser representada pelo seguinte programa (onde L1, L2 e L3 são listas):

```
s(L1,L3):-
  sn(L1,L2),
  sv(L2,L3).
```

Para simplificar a definição da gramática, PROLOG oferece uma maneira concisa de escrever as regras conhecida como DCG (*Definite Clause Grammar*) [Gal91]. DCG é um formalismo que permite expressar regras de gramáticas livres de contexto como declarações lógicas. Então, a regra fica na forma original:

```
S --> SN,
      SV.
```

Nessa regra, observamos que os lados direito e esquerdo da regra separados pela seta têm símbolos não-terminais. O lado direito das regras em DCG também aceita símbolos terminais. A diferença entre um símbolo terminal e um não-terminal numa DCG é que um terminal é escrito como uma lista, ou seja, é representado entre

os símbolos "[" e "]". Por exemplo, a entrada no dicionário para a palavra *controle* fica da seguinte forma:

N --> [controle]

Também temos casos nos quais o símbolo terminal é vazio. Nesse caso, a representação fica da seguinte forma:

Mod --> []

O algoritmo para o funcionamento das regras em DCG para o PROLOG é simples. Imaginemos, por exemplo, que uma regra recebe uma lista de palavra. Ela retira dessa lista as palavras que correspondem ao sintagma que ela representa e retorna uma lista das palavras que sobram. Dessa forma, uma regra que analisa um sintagma retira da lista as palavras que formam o sintagma. Por exemplo, considere a regra:

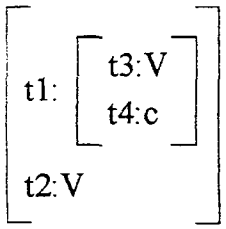
SN --> Det, N

Se essa regra receber a lista [a, mulher, dormiu], a lista [dormiu] será retornada. Para retirar as palavras [a, mulher], a regra Det será chamada recursivamente e obtemos a lista [mulher, dormiu] que será passada para o dicionário que verificará que *mulher* tem a categoria N (substantivo) e retornará a lista [dormiu]. Como isso termina a regra, a lista [dormiu] será retornada. Considere agora que tenhamos mais as seguintes regras:

S --> SN, SV
SV --> V

Se a regra S receber a lista [a, mulher, dormiu], o PROLOG retornará *yes*, pois essa lista está de acordo com as regras. Como já vimos, a lista [a, mulher] será analisada pela regra SN. Já a lista [dormiu] será analisada pela regra SV que passará *dormiu* para o dicionário. Como essa palavra tem a categoria V (verbo), a regra S é bem sucedida. Caso a regra S receba como entrada uma lista vazia ("[]"), serão retornadas todas as frases possíveis de serem geradas a partir das entradas do dicionário. Dessa forma, é possível utilizar potencialmente as regras gramaticais implementadas em PROLOG tanto para análise quanto para geração de frases.

Para demonstrar como os traços são implementados em PROLOG, veja um exemplo de representação gráfica de traços (onde, t1, t2, t3 e t4 são traços, V é variável e c constante) mostrado a seguir:



As informações constantes nessa representação gráfica podem ser implementadas em PROLOG graças a uma extensão para o tratamento dos traços gramaticais conhecida por GULP (*Graph Unification Logic Programming*) [Cov94]. Em GULP, a representação fica da seguinte forma:

```
(t1: (t3: V ..
      t4: c) ..
 t2: V)
```

Nessa representação, os traços do mesmo nível são separados por ".." e os traços estruturados estão agrupados por parênteses.

Terminamos de apresentar alguns conceitos básicos sobre gramáticas para PLN. No próximo capítulo, apresentamos a gramática do gerador de frases para o domínio de tráfego aéreo.

6 GRAMÁTICA DO GERADOR DE FRASES PARA O DOMÍNIO DE TRÁFEGO AÉREO

Neste capítulo, apresentamos a gramática do gerador de frases para o domínio de tráfego aéreo. Esse gerador tem como entrada a representação dos eventos identificados pelo programa descrito na seção 4.4 e tem como saída frases do domínio de tráfego aéreo, como mostrado na figura 6.1.

Nossa gramática utiliza as três ontologias definidas: a ontologia de tráfego aéreo (capítulo 3), a ontologia de papéis temáticos e a ontologia de eventos de tráfego aéreo (capítulo 4). A gramática foi definida em PROLOG utilizando GULP (seção 5.4).

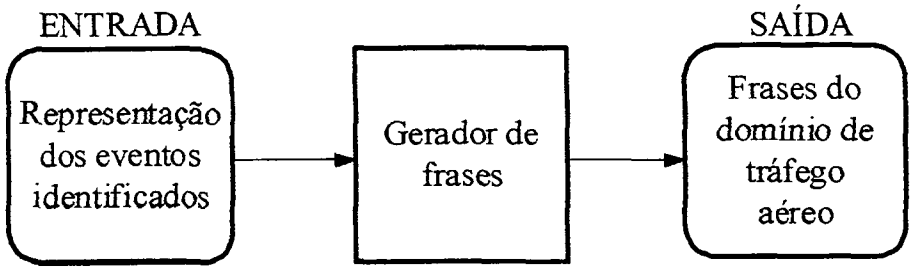


Figura 6.1 O gerador de frases

Este capítulo está organizado da seguinte forma: na próxima seção, descrevemos o formato das mensagens de entrada do gerador de frases, a partir das quais as frases serão geradas. Na seção 6.2, apresentamos a geração de uma frase a partir de uma mensagem de entrada e na última seção, fazemos algumas considerações gerais sobre o gerador.

6.1 MENSAGENS DE ENTRADA

Como mostramos no capítulo 4, o programa identificador de eventos gera uma representação dos eventos de tráfego aéreo indicando os argumentos do evento em termos dos papéis temáticos e dos conceitos e relações da ontologia de tráfego aéreo. Por exemplo, o evento *detectar* é representado conforme a figura 6.2 mostrada a seguir:

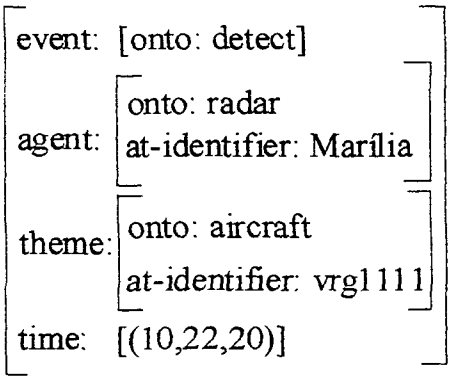


Figura 6.2 Representação para o evento *detectar*

Para servir como mensagem de entrada para o gerador de frases, essa representação foi modificada. As informações associadas ao evento foram reunidas em um conjunto que corresponde às informações semânticas desse evento. Além das informações semânticas, o gerador vai precisar da informação da voz utilizada para gerar a frase. Veja na figura 6.3 apresentada a seguir, como a representação do evento é transformada em uma mensagem de entrada para o gerador de frases:

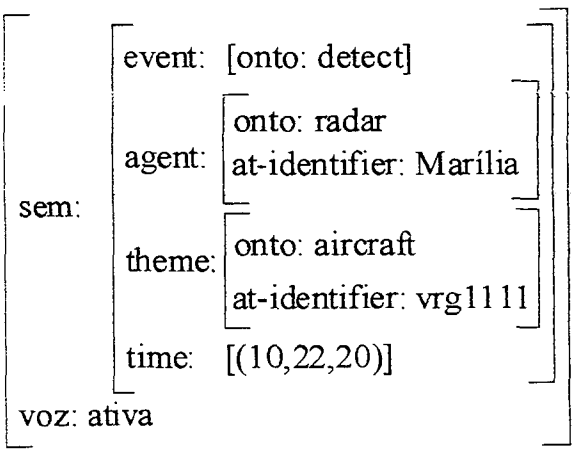


Figura 6.3 Entrada do gerador de frases para o evento *detectar*

Observe na figura 6.3 que as informações associadas ao evento foram agrupadas no traço semântico (*sem*) e a informação da voz a ser utilizada na geração da frase está no traço de voz. Repare que as informações do traço semântico estão em inglês, pois correspondem aos conceitos das ontologias e a informação do traço *voz* está em português assim como todos os outros traços relativos ao gerador de frases que serão apresentados neste capítulo.

Para alguns eventos é possível gerar uma frase na voz passiva e outra na voz ativa utilizando as mesmas informações associadas aos eventos. Por exemplo, as frases "O radar Marília detectou a aeronave vrg1111 às 10h 22min 20s" e "A aeronave vrg1111 foi detectada pelo radar Marília às 10h 22min 20s" são geradas pelas mesmas informações semânticas da figura 6.3, porém uma tem o traço *voz* preenchido com *ativa* e a outra tem o traço *voz* preenchido com *passiva*. A decisão sobre a escolha da voz não é feita automaticamente pelo gerador e deve ser feita pelo usuário.

6.2 EXEMPLO DE GERAÇÃO DE FRASE

Assim que uma mensagem de entrada é submetida ao gerador, ela é unificada com a parte esquerda de uma das regras para sentença. Por exemplo, considere a mensagem de entrada vista na figura 6.3. Essa mensagem de entrada no gerador produzirá como saída a frase "O radar Marília detectou a aeronave vrg1111 às 10h22min20s". Vamos mostrar agora detalhadamente como essa frase será produzida. Para facilitar a compreensão, ilustramos na figura 6.4, a árvore sintática que representa a estrutura dessa frase.

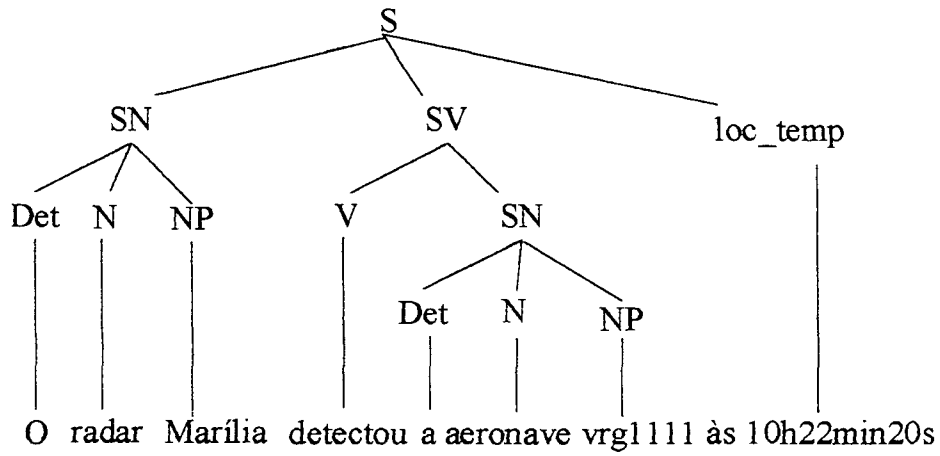
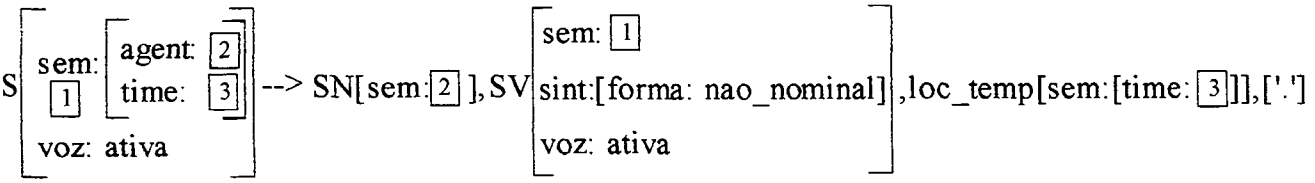


Figura 6.4 Estrutura sintática da frase "O radar Marília detectou a aeronave vrg1111 às 10h22min20s"

Pela estrutura sintática podemos ver que a primeira regra que será unificada com a mensagem de entrada é uma regra para sentença (S). Em nossa gramática, temos duas regras para sentença mostradas na figura 6.5. A regra que se unificará com a mensagem de entrada é a regra S1, pois a mensagem de entrada indica que a frase deverá ser gerada na voz ativa.

Regra S1



Regra S2

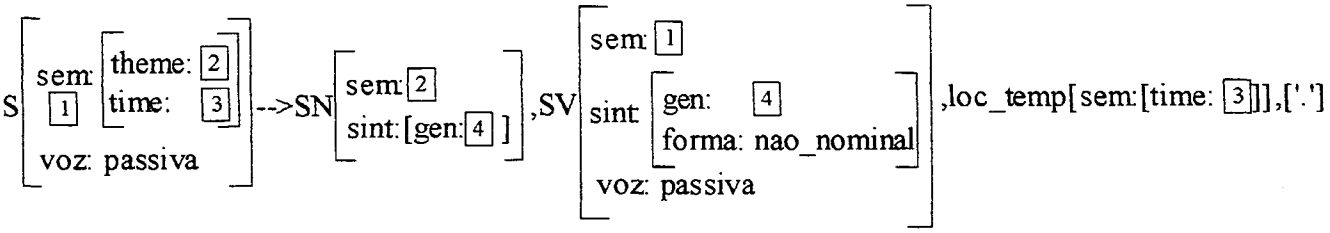


Figura 6.5 Regras para frase (regras S)

Nessas regras, as informações contidas no traço *sem* são distinguidas de forma que elas sejam passadas aos constituintes da regra separadamente. Vimos que a informação semântica da mensagem de entrada mostrada na figura 6.3 tem: o evento, o agente, o tema e a localização temporal. Nesse caso, na regra S1, a informação semântica correspondente ao agente (2) é extraída de forma que somente ela seja passada ao sintagma nominal, pois a frase está na voz ativa. Da mesma forma, a informação associada à localização temporal da frase (3) é passada ao constituinte *loc_temp* da regra que faz o tratamento da localização temporal da frase. Entretanto, toda a informação semântica (1) (evento, agente, tema e localização temporal) é passada para o sintagma verbal; isso foi feito para simplificar, de forma a não ter que distinguir todas as informações semânticas contidas na mensagem. Na figura 6.6, apresentamos a regra S1 unificada.

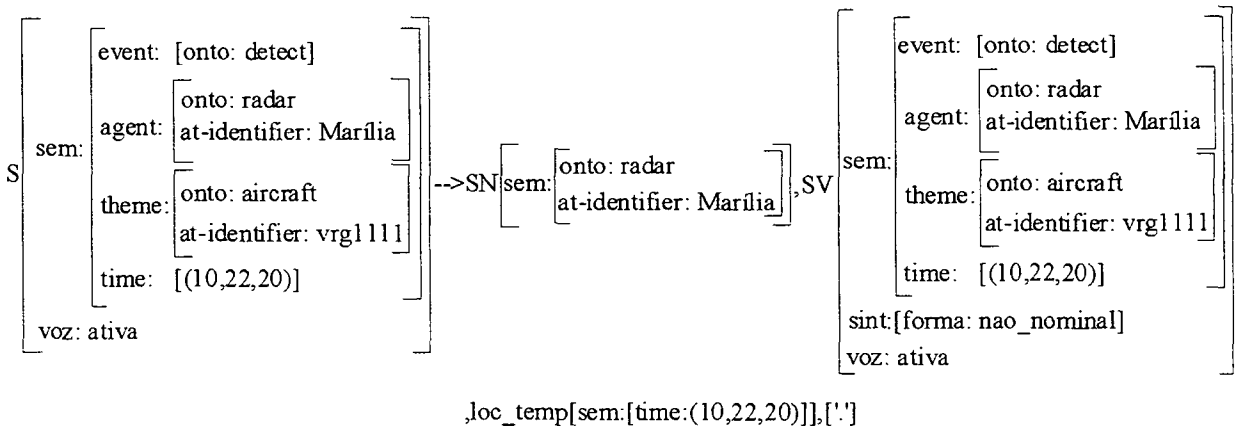
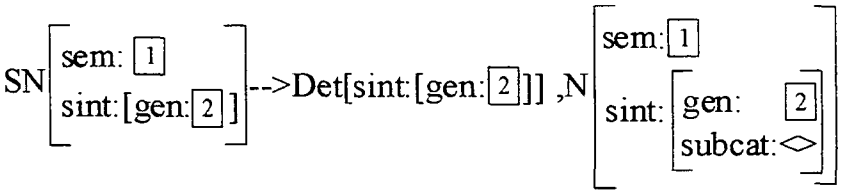


Figura 6.6 Regra S1 unificada

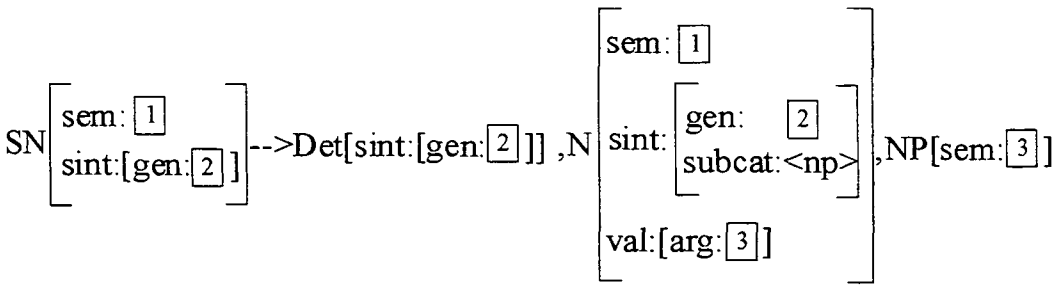
Sintagma nominal

O próximo constituinte a ser unificado é o SN (sintagma nominal) obtido pela regra S1. Na nossa gramática, temos 5 regras para o sintagma nominal, conforme mostrado na figura 6.7.

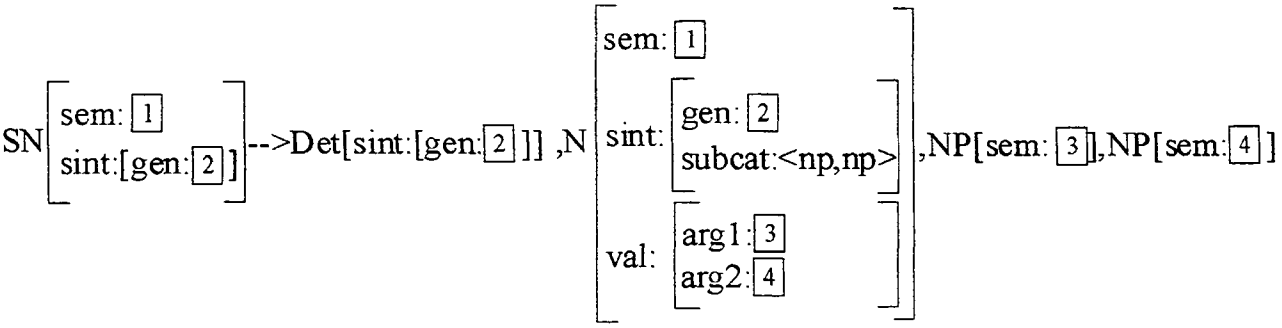
Regra SN1



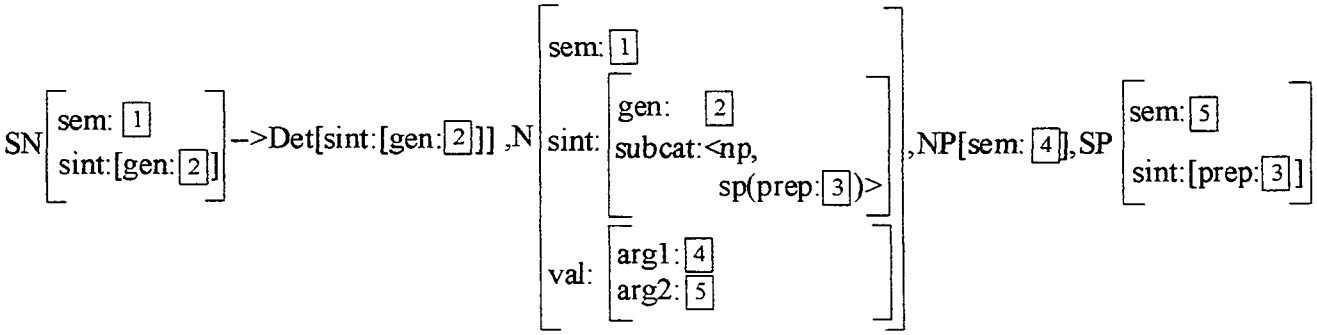
Regra SN2



Regra SN3



Regra SN4



Regra SN5

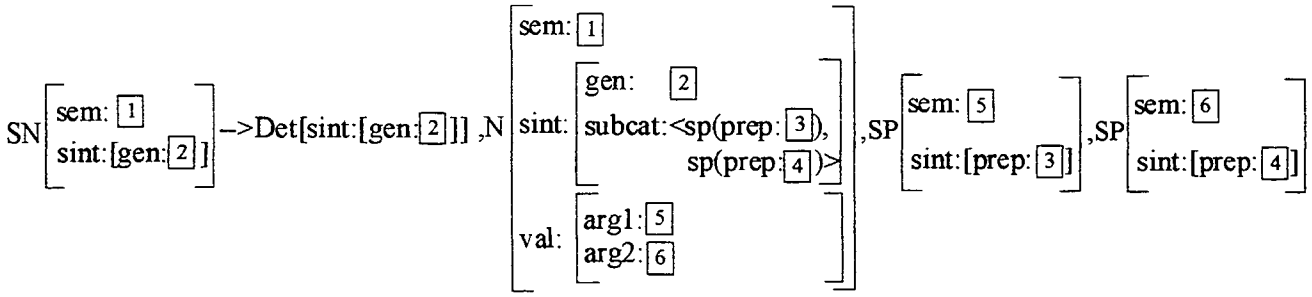


Figura 6.7 Regras para sintagma nominal (regras SN)

Observe que nessas regras, além das informações semânticas agrupadas no traço *sem*, temos um conjunto de informações sintáticas agrupadas no traço *sint*. As informações sintáticas para sintagma nominal são: gênero e subcategoria. Não incluímos as informações sintáticas número e pessoa, pois todos os sintagmas nominais de nosso gerador estão no singular e na terceira pessoa. Além dos traços *sem* e *sint*, temos o traço *val* que será explicado a seguir.

Para gerar o sintagma nominal *o radar Marília*, como pode ser visto na figura 6.8, a regra para sintagma nominal que será utilizada pelo gerador é a regra SN2 da figura 6.7 cujo substantivo tem como subcategoria um nome próprio.

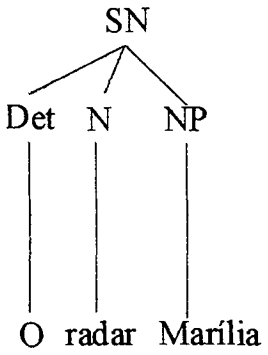


Figura 6.8 Árvore para o sintagma nominal "o radar Marília"

Na regra SN2 da figura 6.7, o valor do traço semântico (1) é igual à informação do agente que foi unificada da regra S1 para a regra SN2 conforme mostrado na figura 6.5. A regra SN2 depois de receber as informações vindas da regra S1 é mostrada na figura 6.9.

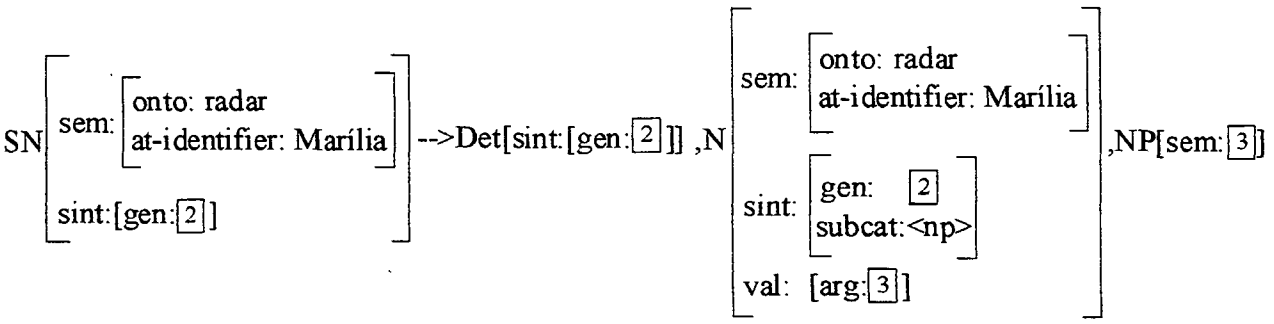


Figura 6.9 Regra SN2 depois de receber as informações da regra S1

O constituinte N da figura 6.9 é quem vai determinar os valores dos traços cujos valores são (2) e (3) ainda não unificados. Esse constituinte vai se unificar com uma entrada do dicionário que tenha a mesma informação no traço semântico, ou seja, com a entrada do dicionário mostrada na figura 6.10.

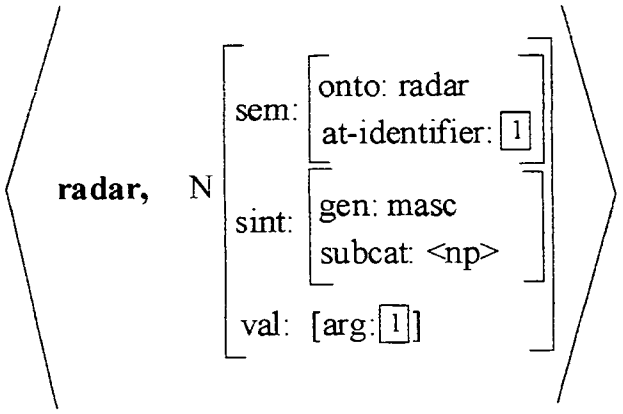


Figura 6.10 Entrada do dicionário para *radar*

Todos os substantivos (N) do dicionário têm os três traços: *sem*, *sint* e *val*. O traço *sem* associa a palavra com os conceitos e relações da ontologia de tráfego aéreo. Na figura 6.10, por exemplo, a palavra *radar* está associada ao conceito *radar* que tem um identificador indicado pelo conceito *at-identifier*.

O traço *sint*, como já vimos, explicita as informações sintáticas da palavra. Para o substantivo, essas informações são: gênero e subcategoria. No exemplo da figura 6.10, a palavra *radar* tem gênero *masculino* e subcategoria *<np>*, pois no domínio de tráfego aéreo, todo radar vai estar associado a um nome próprio que o identifica. Observe que no traço semântico, o *at-identifier* indica que um radar tem um identificador e no traço sintático esse identificador é dado pela subcategoria, pois o identificador é um nome próprio .

O traço *val* (valência[26]) vai ser o responsável pela extração da informação semântica que indica o complemento do substantivo. Isso possibilitará, como vamos mostrar a seguir, a transmissão dessa informação para o constituinte que complementa o substantivo (NP). No exemplo da figura 6.10, o valor do argumento (arg) do traço *val* tem o mesmo valor do identificador do radar presente na informação semântica (1).

---

[26] O termo valência vem da química, onde a valência de um elemento expressa o número de átomos de hidrogênio com os quais o átomo desse elemento pode se combinar

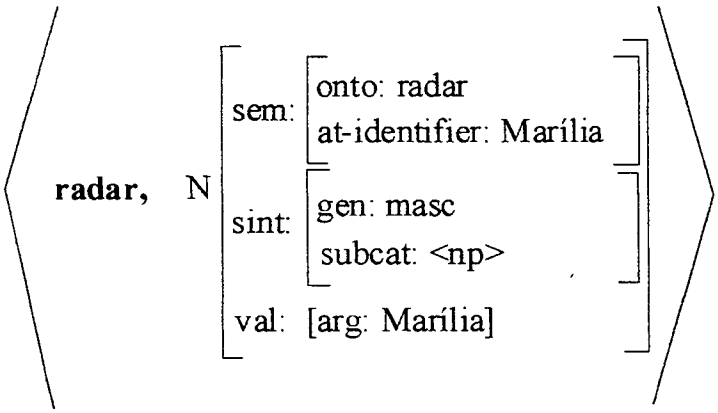


Figura 6.11 Entrada do dicionário para *radar* unificada

A entrada do dicionário para o conceito *radar* com os valores dos traços *at-identifier* e *arg* unificados é mostrada na figura 6.11. Após a unificação, os valores dos traços de N são repassados para a regra SN2 mostrada na figura 6.9 de tal forma que o traço (2), de gênero, fica com o valor *masc* (masculino) e o traço (3) que indica a informação semântica do nome próprio fica com o valor "Marília" conforme ilustra a figura 6.12.

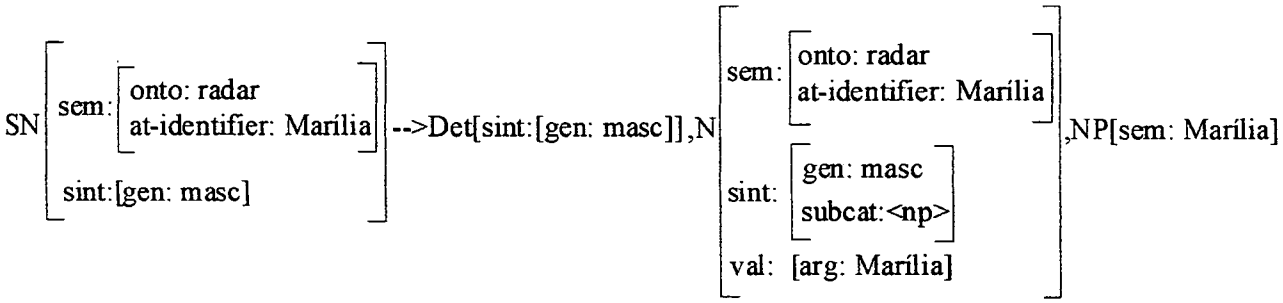


Figura 6.12 Regra SN2 unificada

O constituinte NP da regra mostrada na figura 6.9 corresponde a um nome próprio. Em nossa gramática, simplificamos o tratamento de um nome próprio de acordo com a entrada lexical para nome próprio mostrada na figura 6.13. Essa entrada lexical determina que a gramática aceita todo nome próprio que é unificado com a informação semântica do constituinte NP.

< [1] , NP[sem:[1]]>

Figura 6.13 Entrada lexical para nome próprio (NP)

Assim, o constituinte NP da regra SN2 da figura 6.12 se unifica com a entrada lexical da figura 6.13 como é ilustrado na figura 6.14.

< **Marília** , NP[sem: Marília]>

Figura 6.14 Entrada lexical para nome próprio unificada

Vamos mostrar agora como é finalmente unificado o determinante com a sua entrada correspondente no dicionário. A figura 6.15 ilustra as duas entradas lexicais para determinante na gramática. Como pode ser visto, só consideramos os artigos definidos no singular. Ora, como o substantivo *radar* é masculino, então ele se unifica com a entrada do dicionário para o determinante que também é do gênero masculino.

< **a**, Det[sint:[gen: fem]] >

< **o**, Det[sint:[gen: masc]] >

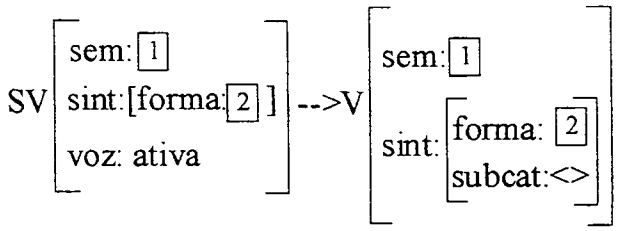
Figura 6.15 Entradas do dicionário para determinante

Temos então, finalmente, a geração do sintagma nominal *o radar Marília* de acordo com a estrutura sintática já mostrada na figura 6.8.

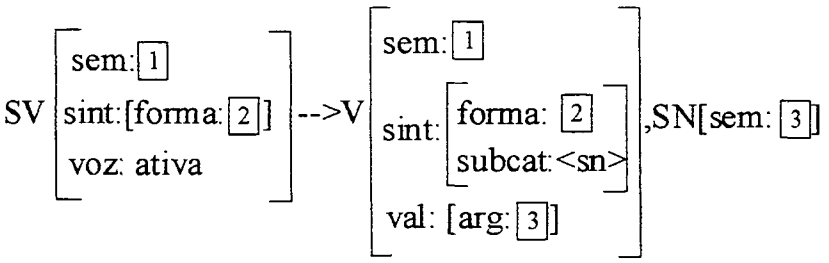
Sintagma verbal

Vamos mostrar agora como é gerado o sintagma verbal da frase da figura 6.4. As regras para sintagma verbal na voz ativa de nosso gerador são mostradas na figura 6.16.

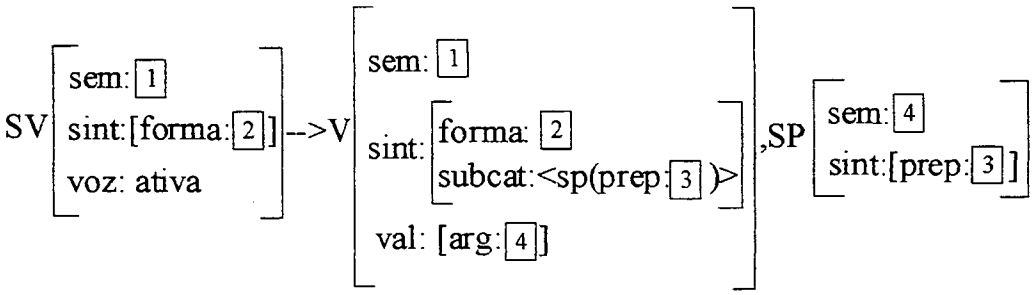
Regra SV1



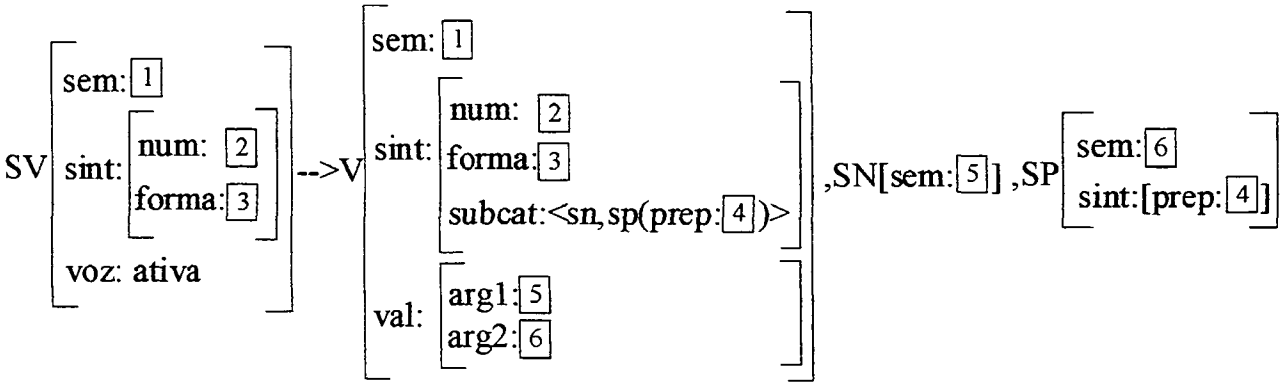
Regra SV2



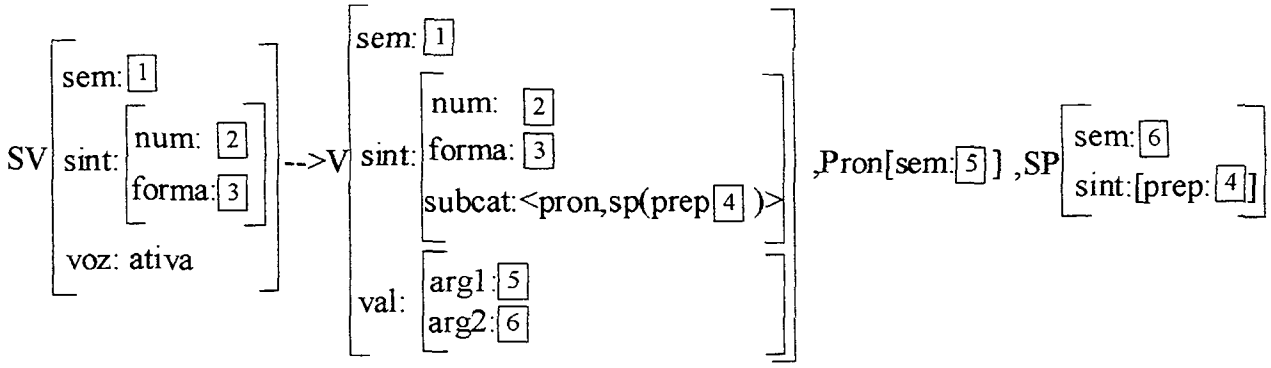
Regra SV3



Regra SV4



Regra SV5



Regra SV6

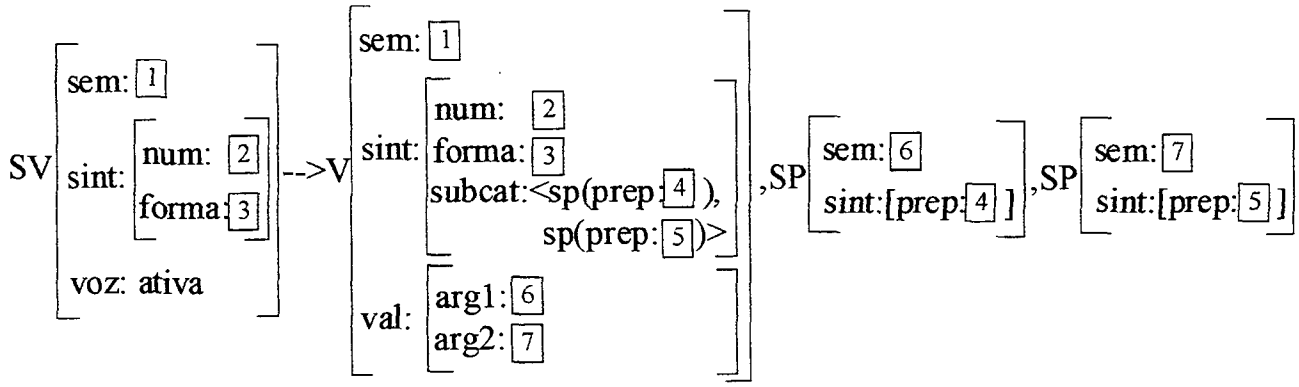


Figura 6.16 regras para sintagma verbal na voz ativa

Para gerar o sintagma verbal *detectou a aeronave vrg1111*, como pode ser visto na figura 6.17, a regra utilizada será a regra SV2 da figura 6.16, pois o verbo *detectar* tem subcategoria <*sn*>. A regra SV2 depois de receber as informações da regra S, é mostrada na figura 6.18. Observe que a informação semântica contém toda a informação inicial da mensagem de entrada. Como já observamos, isso foi feito para simplificar o processo de unificação da regra S, evitando a explicitação de toda a informação semântica nessa regra.

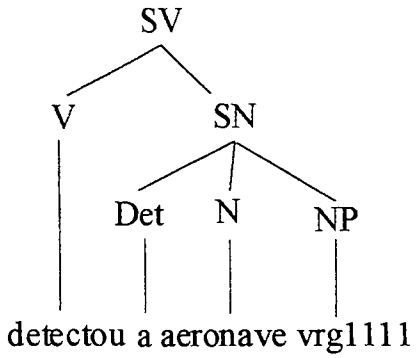


Figura 6.17 Árvore para o sintagma verbal *detectou a aeronave vrg1111*

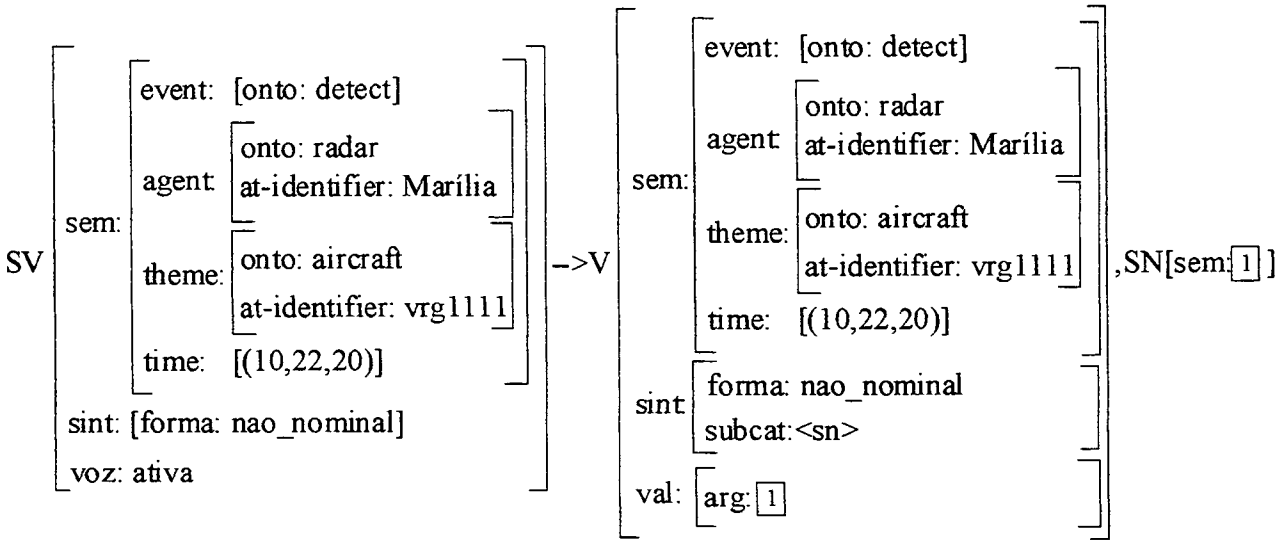


Figura 6.18 Regra SV2 depois de receber as informações da regra S1

O constituinte V da regra SV2 mostrada na figura 6.16 se unifica com a entrada do dicionário associada ao evento *detectar* mostrada na figura 6.19.

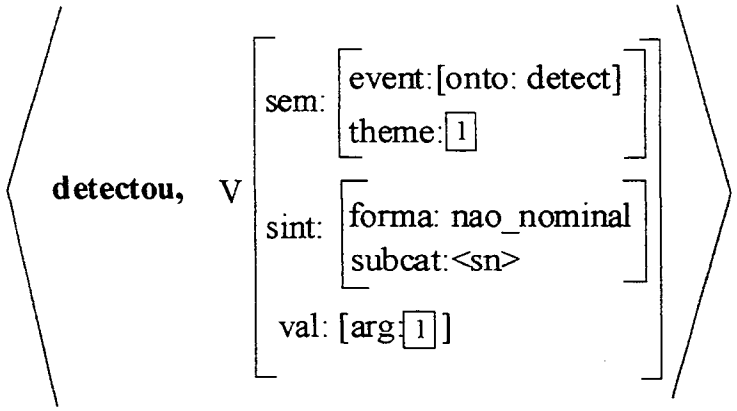


Figura 6.19 Entrada do dicionário para *detectar* na forma *não nominal*

Observe que a entrada do dicionário para o verbo contém os traços *sem*, *sint* e *val*. O traço *sem* relaciona o verbo com a ontologia de eventos e também com a ontologia de papéis temáticos. Na figura 6.19, o evento é *detectar* e esse evento tem como papel semântico um *tema*.

O traço *sint* contém informações sintáticas sobre o verbo que são: a forma verbal, a subcategoria e o gênero. Na gramática do nosso gerador, consideramos a forma não nominal (verbos no indicativo) e as formas nominais (particípio e infinitivo). Os verbos na forma não nominal estão todos na terceira pessoa do singular do indicativo, por isso não indicamos nem a pessoa, nem o número e nem o modo na entrada do dicionário. As subcategorias do verbo podem ser: <>, <np>, <sp>, <pron,sp>, <sn,sp> e <sp,sp> conforme foi explicado na seção 5.3.2. Na figura 6.19, o verbo está na forma não nominal e é da subcategoria <sn>. O gênero do verbo só é utilizado para os verbos no particípio. Por exemplo, *detectado* está no gênero masculino e *detectada* está no gênero feminino.

O traço *val* do verbo, da mesma forma que o substantivo, permite estabelecer uma relação entre a informação semântica e a sintática. Veja que a informação semântica em relação ao verbo da figura 6.19 indica que ele exige um tema. Por sua vez, a informação sintática indica que o verbo exige um complemento que corresponde a um sintagma nominal. Dessa forma, o traço *val* tem como argumento o mesmo valor que o tema. Essa informação será repassada ao sintagma nominal que complementa o verbo como veremos a seguir. Assim, a entrada lexical da figura 6.19 instanciada com a informação vinda da regra SV2 é mostrada na figura 6.20.

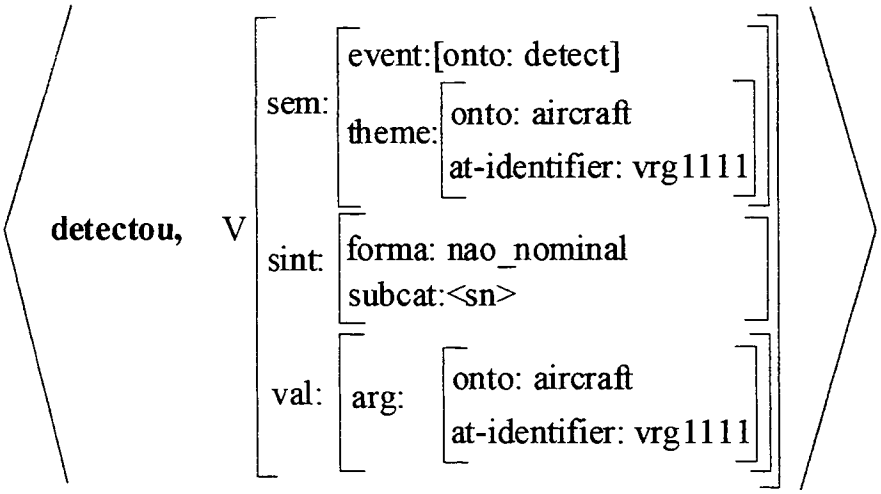


Figura 6.20 Entrada do dicionário para *detectar* instanciada

Depois do constituinte V ser unificado com a entrada lexical, a regra SV2 da figura 6.18 é instanciada, sendo mostrada na figura 6.21. Observe então que o valor do traço *val* é transmitido ao constituinte SN que complementa o verbo.

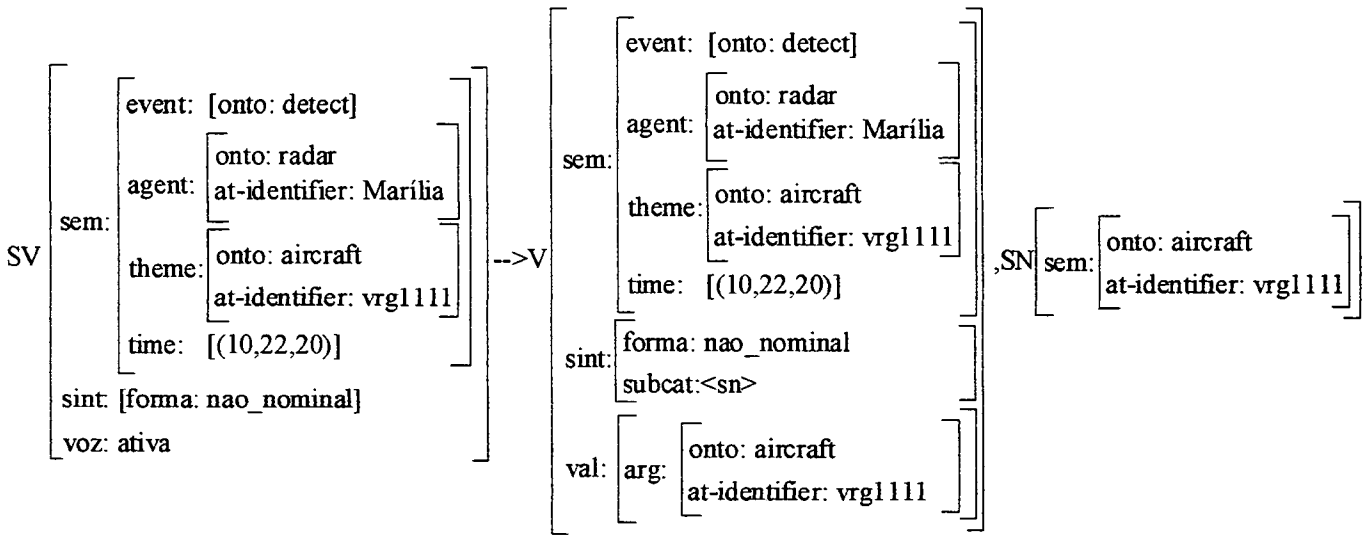


Figura 6.21 Regra SV2 instanciada

Temos então, finalmente, a geração do sintagma verbal *detectou a aeronave vrg1111*de acordo com a árvore sintática já mostrada na figura 6.17.

A geração do sintagma nominal *a aeronave vrg1111* não foi mostrada, pois é similar à geração do sintagma nominal já mostrado.

Localizador temporal

Para geração da localização temporal da frase, nossa gramática contém uma única regra mostrada na figura 6.22. Dessa forma, o localizador temporal das frases que relatam os eventos de tráfego aéreo corresponde ao horário dado em termos de horas, minutos e segundos.

< **às** [1] **h** [2] **m in** [3] **s,** loc_temp[sem:[time:([1],[2],[3])]] >

Figura 6.22 Regra para a localização temporal

6.3 CONSIDERAÇÕES SOBRE O GERADOR

Apresentamos na seção anterior, o processo de geração de uma frase de nosso gerador. Algumas mensagens de entrada com as respectivas frases produzidas pelo gerador podem ser vistas no apêndice 04. A gramática completa e também todas as entradas lexicais se encontram no apêndice 05.

Nossa gramática do gerador de frases é bastante simples. Como mostramos no exemplo da seção anterior, todas as frases estão na terceira pessoa do singular. Além disso, o gerador não faz o tratamento das contrações (em + o = no, de + o = do, por + o = pelo, etc.). Também não otimizamos as entradas lexicais de forma a ter uma única entrada para cada verbo e um conjugador que gerasse o verbo na forma específica exigida pela frase. Por exemplo, o verbo *detectar* tem uma entrada para *detectou* e uma entrada para *detectada* como pode ser visto na figura 6.23.

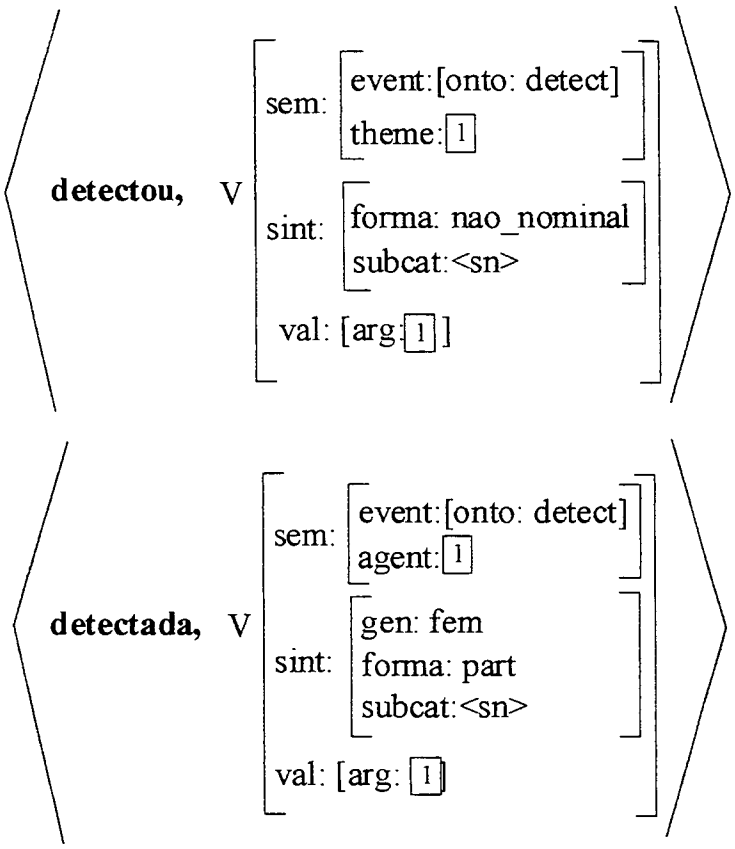


Figura 6.23 Entradas do dicionário para *detectou* e *detectada*

Observe que também não generalizamos o tratamento do gênero no particípio, pois temos que ter uma entrada para o particípio masculino e outra para o feminino. Temos também que colocar todos os papéis temáticos exigidos pelo verbo nas entradas lexicais para poder colocar uma única entrada para cada verbo e modificar a transmissão da informação semântica do complemento do verbo, o que é feito pelo traço *val*. Essa observação também vale para as entradas dos substantivos.

Essas simplificações foram feitas devido a restrições de tempo e também porque esses não eram nossos objetivos principais. Nosso objetivo principal era construir um gerador de frases que levasse em conta as ontologias construídas. Esse objetivo foi alcançado. A mensagem de entrada é representada de acordo com os papéis temáticos. Além disso, os papéis temáticos da mensagem de entrada são preenchidos de acordo com a ontologia de tráfego aéreo e com a ontologia de eventos de tráfego aéreo.

Também as entradas lexicais estão representadas de acordo com essas ontologias. Os verbos e os substantivos foram todos modelados levando em conta especificamente o domínio de tráfego aéreo.

Já as regras gramaticais não fazem referência às ontologias de tráfego aéreo e de eventos de tráfego aéreo. Nossas regras gramaticais referenciam somente a ontologia de papéis temáticos. Isso torna o nosso gerador reutilizável em outros domínios além do tráfego aéreo.

O processo de inferência do PROLOG utiliza o método de busca cuja árvore é percorrida de cima para baixo e da esquerda para direita (*depth-first search*) ou pesquisa primeiro em profundidade. Como conseqüência, o processo de geração utiliza muito o *backtracking*[27]. Por exemplo, suponha a geração do sintagma nominal *a aeronave* e que os determinantes do nosso dicionário sejam, nessa ordem: *os*, *as*, *o* e *a*. Nesse caso, o PROLOG só conseguiria fazer a concordância depois de tentar por três vezes.

Destacamos aqui a importância das ontologias para o desenvolvimento desse sistema. As ontologias tiveram um papel fundamental, permitindo a documentação do sistema, uma modelagem mais profunda e uma coerência entre os módulos do sistema: o programa identificador de eventos (seção 4.4) e o programa gerador de frases apresentado neste capítulo.

---

[27] Mecanismo da linguagem PROLOG que permite a retroação automática para o exame de novas alternativas

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste capítulo, apresentamos as principais contribuições resultantes do nosso trabalho e também algumas perspectivas para a continuação do trabalho.

7.1 CONTRIBUIÇÕES

Atingimos o objetivo principal de nosso trabalho que era implementar um sistema gerador de frases para o domínio de tráfego aéreo baseado em ontologias. As ontologias exerceram um papel fundamental no sistema, pois permitiram a documentação do sistema, uma modelagem mais profunda do domínio e uma coerência entre os módulos do sistema que implementamos: o programa identificador de eventos (seção 4.4) e o programa gerador de frases (capítulo 6).

As ontologias que definimos foram: ontologia de tráfego aéreo, ontologia de papéis temáticos e ontologia de eventos de tráfego aéreo. Definimos as ontologias na linguagem Ontolingua a fim de aumentar a clareza e potencial de reutilização.

As definições dos conceitos da ontologia de tráfego aéreo têm um engajamento ontológico mínimo, ou seja, o menor número possível de restrições semânticas. Isso foi feito visando aumentar o potencial de reutilização da ontologia. A ontologia também não está completa para o domínio de tráfego aéreo devido restrições de tempo e da complexidade do trabalho de construção de ontologias.

A ontologia de papéis temáticos é bastante simples e genérica, podendo ser utilizada em outros domínios e aplicações de processamento de língua natural devido ao seu caráter lingüístico.

A ontologia de eventos de tráfego aéreo define eventos cujos papéis temáticos devem ser preenchidos por conceitos da ontologia de tráfego aéreo. Essa ontologia é importante, pois facilita a representação dos eventos de tráfego aéreo; essa representação é utilizada pelos módulos do sistema gerador de frases. A ontologia de

eventos de tráfego aéreo não está completa, pois não definimos todos os eventos possíveis para o domínio de tráfego aéreo e nem todos os conceitos que podem preencher os papéis temáticos. Entretanto, ela pode ser facilmente estendida de forma que outros eventos e conceitos sejam considerados.

A gramática do gerador de frases que implementamos é bastante simples, pois o objetivo não era construir um gerador completo para a língua portuguesa, mas sim construir um sistema gerador de frases que levasse em conta as ontologias construídas. Esse objetivo foi alcançado.

7.2 TRABALHOS FUTUROS

Definimos ontologias do domínio de tráfego aéreo que não cobrem todo o conhecimento desse domínio, mas que podem ser facilmente estendidas para incorporar novos conceitos e relações. As ontologias de tráfego aéreo podem ser reutilizadas em outros sistemas do mesmo domínio. Por exemplo, elas podem ser reutilizadas em um sistema para detecção de anomalias técnicas[28] ou operacionais[29] que também pode ser um gerador de frases relatando tais anomalias. Esse sistema analisaria os dados do sistema de controle de tráfego aéreo e poderia concluir, por exemplo, que o piloto de uma aeronave não cumpriu uma autorização do controle, pois desceu para um nível de vôo diferente do que estava autorizado.

A gramática da língua portuguesa utilizada para geração de frases neste trabalho pode ser estendida para gerar mais tipos de frases. Por exemplo, a simples inclusão de traços para o tempo verbal, número e pessoa aumenta significativamente o número de frases possíveis geradas.

Como as regras gramaticais do gerador de frases não fazem referência às ontologias de tráfego aéreo e de eventos de tráfego aéreo e sim apenas referenciam a

---

[28] Detecção de nuvens pelo radar ou perdas de detecção em uma região, por exemplo

[29] Um piloto que desobedeceu uma ordem do controle, por exemplo

ontologia de papéis temáticos, isso torna possível a reutilização do nosso gerador em outros domínios além do tráfego aéreo. Por exemplo, pode-se adaptá-lo para gerar frases baseadas em ontologias do domínio de redes de computadores a partir dos dados de *logs* de usuários.

Outra possível experiência que pode ser feita é tornar o gerador de frases bilíngüe. A idéia seria utilizar as mesmas representações dos eventos, reaproveitar o dicionário associando duas palavras a cada uma de suas entradas e modificar ou acrescentar as regras gramaticais caso elas sejam diferentes na segunda língua.

A linguagem utilizada para implementar o gerador de frases foi PROLOG. Nos programas em PROLOG, a busca por soluções é feita pelo método de busca *depth-first search* (seção 6.3) que percorre a árvore de cima para baixo e da esquerda para direita. Esse método pode ser bastante ineficiente. Por exemplo, nas regras de sintagma nominal, vários determinantes podem ser testados antes que o determinante que concorde com o substantivo seja o escolhido. Idealmente, a busca deveria ser conduzida pelo núcleo do sintagma (o substantivo), pois é ele quem determina os valores dos traços de seu complemento [Shi89]. Assim, outra continuação possível para este trabalho seria implementar o sistema utilizando outro motor de inferência que otimizasse o processo de geração de frases.

8 REFERÊNCIAS BIBLIOGRÁFICAS

[All95] ALLEN J., **Natural language understanding**. 2. ed., Redwood City: Benjamin Cummings Publishing Company Inc., 1995.

[Bat90] BATEMAN J.A., **Upper Modeling: a general organization of knowledge for natural language processing**. In: Workshop on Standards for Knowledge Representation Systems, 1990, Santa Barbara.

[Ber01] BERNERS-LEE T., HENDLER J., LASSILA O., The Semantic Web, **Scientific American**, mai. 2001.

[Cha99] CHANDRASEKARAN B., JOSEPHSON J. R.; BENJAMINS V. R., What are Ontologies, and why do we need them? **IEEE Inteligent Systems**, v. 14, n. 1, p. 20-26, jan. / fev. 1999.

[Cov94] COVINGTON M. A., **GULP 3.1: An extension of Prolog for Unification-based Grammar**. Athens – Georgia, 1994. Artificial Inteligence Center, The University of Georgia.

[Gag02] **Processamento da linguagem natural.** Disponível em: <http://www.inf.ufpr.br/~michel/Disciplinas/Bac/IA/PLN/pln.html> Acesso em: 25 ago 02.

[Gal91] GAL A. et. al., **Prolog for natural language processing**. West Sussex: John Wiley & Sons Ltd., 1991.

[Gen87] GENESERETH M.R., NILSSON N. J., **Logical Foundations of Artificial Intelligence**. Morgan Kauffmann Publ., 1987.

[Gen92] GENESERETH M.R., FIKES R.E., **Knowledge Interchange Format Version 3.0 Reference Manual.** 1992. 67f. Computer Science Department, Stanford University, Stanford.

[Gom02] GOMÉZ-PEREZ, A., CORCHO, O., Ontology languages for the Semantic Web. **IEEE Inteligent Systems**, p. 54-59, jan. / fev. 2002.

[Gru92] GRUBER T.R., **Ontolingua: a mechanism to support portable ontologies.** Knowledge Systems Laboratory, Stanford University, Palo Alto-CA, 1992. 56f.

[Gru93] GRUBER T.R., A translation approach to portable ontology specifications. **Knowledge Acquisition**, v. 5 n. 2, p. 199-220, 1993.

[Gru94] GRUNINGER, M., FOX, M. S., **The role of competency questions in enterprise engineering**, Department of industrial engineering, University of Toronto, 1994. 17f.

[Gru95] GRUBER T.R., Towards principles for the design of ontologies used for knowledge sharing. **International Journal of Human-Computer Studies**, v. 43, n. 5/6, p. 907-928, 1995.

[Hei95] HEIJST, G. A. C. M. V., **The role of ontologies in knowledge engineering.** Universeit van Amsterdam,1995.

[Ima99] MINISTÉRIO DA AERONÁUTICA, **IMA100-12 Regras do ar e serviços de tráfego aéreo**. jun. 1999.

[Ksl02] **Stanford KSL Network Services.** Disponível em: <http://www-ksl-svc.stanford.edu:5915/> Acesso em: 10 ago 2002.

[Lop99] LÓPEZ M.F. et. al., Building a Chemical Ontology using Methontology and the Ontology Design Environment. **IEEE Inteligent Systems**, n. 14, p. 37-46, jan. / fev. 1999.

[Mah95] MAHESH K., NIRENBURG S., **A situated ontology for practical NLP**. In: Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995. p. 19-21

[Mah96] MAHESH K., **Ontology development for machine translation: ideology and methodology.** New Mexico State University, 1996. 79 f.

[Per96] PERINI M. A., **Gramática Descritiva do português**. 2. ed. São Paulo: Editora Ática, 1996.

[Rei00] REITER E., DALE R., **Building natural language generation systems**. Estados Unidos, 2000. p. 57-59, p. 159-164.

[Sag99] SAG I. A., WASOW T., **Syntactic Theory: A Formal Introduction**. Stanford CSLI Publications, 1999.

[Shi89] SHIEBER Stuart M. et. al., A semantic-head-driven generation algorithm for unification-based formalisms. **Meeting of the association for computational linguistics**, p. 7-17, 1989.

[Smi96] SMITH H. **Estabilishing the foundations for the specification of next generation (advanced) air traffic management systems**. In: EUROCONTROL EATMS ARCHITETURE WORKSHOP, 1996.

[Swa96] SWARTOUT B. et. al., **Toward distributed use of large-scale Ontologies.** USC/Information Sciences Institute, Marina del Rey - Califórnia, 1996. 19 f.

[Swa99] SWARTOUT W. Ontologies. ***IEEE Inteligent Systems***, v. 14, n. 1, p. 18-19, jan. / fev. 1999.

[Usc96] USCHOLD M., GRUNINGER M. , Ontologies: principles, methods and applications. **Knowledge Engineering Review**, v. 2, n. 2, p. 93-136, jun. 1996.

[Val99] VALENTE A. et al. Building and (re)using an ontology of air campaign planning. **IEEE Inteligent Systems**, v. 14, n. 1,p. 27-36, jan. / fev. 1999.

APÊNDICE 01 – ONTOLOGIA DE TRÁFEGO AÉREO

```
(define-ontology AIR-TRAFFIC
  (SLOT-CONSTRAINT-SUGAR FRAME-ONTOLOGY))

(define-class AT-THING (?x)
 "Classe que agrega todas as outras classes da ontologia"
 :def (thing ?x)
 :axiom-def (disjoint-decomposition
             AT-THING (setof AT-OBJECT
                             AT-ATTRIBUTE)))

(define-class AT-OBJECT (?x)
  "Classe que agrega todos os objetos da ontologia. Esses objetos podem
  ser concretos, tais como: aeronave, radar, aeródromo, pista, etc. ou
  abstratos, tais como: espaço aéreo, rota ATS, fixo, etc."
  :def (and (at-thing ?x)
            (>= (cardinality ?x has-at-identifier) 0)))
  :axiom-def (disjoint-decomposition
              AT-OBJECT
              (setof AT-CONTROL
                     FLIGHT-PLAN
                     AIRCRAFT
                     PLACE
                     ELECTRONIC-DEVICE)))

(define-class AT-ATTRIBUTE (?x)
  "Classe que agrega todos os atributos da ontologia"
  :def (at-thing ?x)
  :axiom-def (disjoint-decomposition
              AT-ATTRIBUTE
              (setof NUMERICAL-VALUE
                     CODE
                     AT-IDENTIFIER
                     DIRECTION)))

(define-class AT-IDENTIFIER (?x)
  "Identificador: nomes ou indicativos dos objetos"
```

```
  :def (at-attribute ?x))

(define-class AIRCRAFT (?x)
  "Aeronave: veículo que se desloca no espaço aéreo e é apto a
  transportar pessoas e coisas. É conhecida por um identificador, tem um
  equipamento transponder e pode ter um plano de vôo"
  :def (and (at-object ?x)
            (has-one ?x has-at-identifier)
            (has-one ?x has-transponder)
            (can-have-one ?x has-flight-plan)))

(define-class ELECTRONIC-DEVICE (?x)
  "Classe que agrega todos os dispositivos eletrônicos"
  :def (at-object ?x)
  :axiom-def (disjoint-decomposition
              ELECTRONIC-DEVICE
             (setof RADAR
                    TRANSPONDER
                    NAVAID)))

(define-class TRANSPONDER (?x)
  "Transponder: dispositivo eletrônico transmissor-receptor de código e
  identificação"
  :def (and (electronic-device ?x)
            (can-have-one ?x has-transponder-code)
            (has-one ?x has-transponder-ident)))

(define-class CODE (?x)
  "Classe que agrega todos os códigos"
  :def (at-attribute ?x)
  :axiom-def (disjoint-decomposition
              CODE
             (setof TRANSPONDER-CODE
                    TRANSPONDER-IDENT)))

(define-class TRANSPONDER-CODE (?x)
  "Código transponder: quatro algarismos em octal"
  :def (code ?x))
```

```
(define-class TRANSPONDER-IDENT (?x)
  "Identificação transponder"
  :def (and (code ?x)
            (member ?x (setof 0
                              1))))

(define-class PLACE (?x)
  "Classe que agrega todos os lugares"
  :def (at-object ?x)
  :axiom-def (disjoint-decomposition
               PLACE
              (setof RUNWAY
                     AIRSPACE
                     AIRDROME
                     POINT
                     ATS-ROUTE)))

(define-class POINT (?x)
  "Ponto geográfico: Lugar representado por coordenadas cartesianas de
  acordo com a projeção estereográfica"
  :def (and (place ?x)
            (has-one ?x has-x-coordinate)
            (has-one ?x has-y-coordinate)))

(define-class AIRDROME (?x)
  "Aeródromo: lugar destinado ao pouso e decolagem de aeronaves que tem
  pelo menos um identificador, um ponto geográfico, uma altitude e pode
  ou não ter pistas"
  :def (and (place ?x)
            (has-some ?x has-at-identifier)
            (has-one ?x has-point)
            (has-one ?x airdrome.altitude)
            (>= (cardinality ?x has-runway) 0)))
```

```
(define-class RUNWAY (?x)
  "Pista: área retangular pavimentada ou não destinada ao pouso e
  decolagem de aeronaves que tem um identificador e pertence a um
  aeródromo"
  :def (and (place ?x)
            (has-one ?x has-at-identifier)
            (has-one ?x runway-of)))

(define-class NUMERICAL-VALUE (?x)
  "Valor numérico: Classe que agrega todos os valores numéricos"
  :def (and (at-attribute ?x)
            (disjoint-decomposition
             NUMERICAL-VALUE
            (setof ALTITUDE
                   FLIGHT-LEVEL
                   COURSE)))

(define-class ALTITUDE (?x)
  "Altitude: valor numérico inteiro maior ou igual a -1000 e menor ou
  igual a 24000"
  :def (and (numerical-value ?x)
            (integer ?x)
            (>= ?x -1000)
            (<= ?x 24000)))

(define-class AIRSPACE (?x)
  "Espaço aéreo: lugar que tem pelo menos um identificador e é limitado
  por pelo menos três pontos geográficos e um nível de vôo"
  :def (and (place ?x)
            (has-some ?x has-at-identifier)
            (has-at-least ?x has-point 3)
            (has-some ?x has-flight-level)))

(define-class FLIGHT-LEVEL (?x)
  "Nível de vôo: superfície de pressão atmosférica constante representada
  por valores numéricos inteiros menores ou iguais a 3000"
  :def (and (numerical-value ?x)
            (integer ?x)
```

```
            (<= ?x 3000)))

(define-class ATS-ROUTE (?x)
  "Rota ATS: corredor por onde as aeronaves podem voar que tem um
identificador, pelo menos dois pontos geográficos e um nível de vôo"
  :def (and (place ?x)
            (has-one ?x has-at-identifier)
            (has-at-least ?x has-point 2)
            (has-some ?x has-flight-level)))

(define-class FLIGHT-PLAN (?x)
  "Plano de vôo: documento que tem um identificador e contém o
  planejamento de um vôo de uma aeronave que parte de um aeródromo com
  destino ao mesmo ou outro aeródromo e tem uma velocidade e um nível de
  vôo previstos"
   :def (and (at-object ?x)
             (has-one ?x has-at-identifier)
             (has-some ?x flight-plan-of)
             (has-one ?x has-departure-airdrome)
             (has-one ?x flight-plan.speed)
             (has-one ?x flight-plan.flight-level)
             (has-one ?x has-destination-airdrome)))

(define-class RADAR (?x)
  "Radar: dispositivo eletrônico utilizado para detectar as aeronaves que
  tem pelo menos um identificador, funciona a ciclos constantes e tem um
  ponto geográfico"
  :def (and (electronic-device ?x)
            (has-some ?x has-at-identifier)
            (has-one ?x radar.cycle)
            (has-one ?x has-point)))

(define-class FIX (?x)
  "Fixo: ponto geográfico com um identificador"
  :def (and (point ?x)
            (has-one ?x has-at-identifier)))
```

```
(define-class NAVAID (?x)
  "Auxílio à navegação: dispositivo eletrônico que tem pelo menos um
  identificador e um ponto geográfico"
  :def (and (electronic-device ?x)
            (has-some ?x has-at-identifier)
            (has-one ?x has-point)))

(define-class AT-CONTROL (?x)
  "Controle de tráfego aéreo: autoridade responsável pelo serviço de
  controle de tráfego aéreo que tem pelo menos um identificador"
  :def (and (at-object ?x)
            (has-some ?x has-at-identifier)))

(define-class COURSE (?x)
  "Rumo: valor número inteiro que define uma direção e sentido. É
  expresso em graus, variando de 0 a 359, no sentido horário"
  :def (and (numerical-value ?x)
            (integer ?x)
            (>= ?x 0)
            (<= ?x 359)))

(define-class DIRECTION (?x)
  "Direção: sentido para o qual algum objeto está se movendo"
  :def (and (at-attribute ?x)
            (member ?x (setof left
                              right))))

(define-relation HAS-AT-IDENTIFIER (?x1)  :->  ?x2
  "Mapeamento dos objetos de tráfego aéreo para seus identificadores"
  :def (and (at-object ?x1)
            (at-identifier ?x2)))

(define-relation HAS-POINT (?x1 ?x2)
  "Mapeamento de um espaço aéreo, rota ATS, aeródromo, radar ou auxílio à
  navegação para um ponto geográfico"
  :def (and (or (airspace ?x1)
                (ats-route ?x1)
```

```
                (airdrome ?x1)
                (radar ?x1)
                (navaid ?x1))
            (point ?x2)))

(define-relation HAS-RUNWAY (?x1 ?x2)
  "Mapeamento de um aeródromo para uma pista"
  :def (and (airdrome ?x1)
            (runway ?x2)))

(define-relation HAS-FLIGHT-LEVEL (?x1 ?x2)
  "Mapeamento de um espaço aéreo ou uma rota ATS para um nível de vôo"
       :def (and (or (airspace ?x1)
                     (ats-route ?x1))
            (flight-level ?x2)))

(define-function HAS-TRANSPONDER (?x1) :-> ?x2
  "Mapeamento de uma aeronave para um transponder"
  :def (and (aircraft ?x1)
            (transponder ?x2)))

(define-function HAS-FLIGHT-PLAN (?x1) :-> ?x2
  "Mapeamento de uma aeronave para um plano de vôo"
  :def (and (aircraft ?x1)
            (flight-plan ?x2)))

(define-function HAS-TRANSPONDER-CODE (?x1) :-> ?x2
  "Mapeamento de um transponder para um código transponder"
  :def (and (transponder ?x1)
            (transponder-code ?x2)))

(define-function HAS-TRANSPONDER-IDENT (?x1) :-> ?x2
  "Mapeamento de um transponder para uma identificação transponder"
  :def (and (transponder ?x1)
            (transponder-ident ?x2)))

(define-function HAS-X-COORDINATE (?x1) :-> ?x2
   "Mapeamento de um ponto geográfico para uma coordenada x"
```

```
   :def (and (point ?x1)
             (real ?x2)))

(define-function HAS-Y-COORDINATE (?x1) :-> ?x2
  "Mapeamento de um ponto geográfico para uma coordenada y"
  :def (and (point ?x1)
            (real ?x2)))

(define-function AIRDROME.ALTITUDE (?x1) :-> ?x2
  "Mapeamento de um aeródromo para uma altitude"
  :def (and (airdrome ?x1)
            (altitude ?x2)))

(define-function RUNWAY-OF (?x1 ?x2)
  "Mapeamento de uma pista para um aeródromo"
  :def (and (runway ?x1)
            (airdrome ?x2)))

(define-function FLIGHT-PLAN-OF (?x1) :-> ?x2
  "Mapeamento de um plano de vôo para uma aeronave"
  :def (and (flight-plan ?x1)
            (aircraft ?x2)))

(define-function HAS-DEPARTURE-AIRDROME (?x1) :-> ?x2
  "Mapeamento de um plano de vôo para o aeródromo de partida"
  :def (and (flight-plan ?x1)
            (airdrome ?x2)))

(define-function HAS-DESTINATION-AIRDROME (?x1) :-> ?x2
  "Mapeamento de um plano de vôo para o aeródromo de destino"
  :def (and (flight-plan ?x1)
            (airdrome ?x2)))

(define-function FLIGHT-PLAN.SPEED (?x1) :-> ?x2
  "Mapeamento de um plano de vôo para a velocidade"
  :def (and (flight-plan ?x1)
            (natural ?x2)))
```

```
(define-function FLIGHT-PLAN.FLIGHT-LEVEL (?x1) :-> ?x2
  "Mapeamento de um plano de vôo para o nível de vôo"
  :def (and (flight-plan ?x1)
            (flight-level ?x2)))

(define-function RADAR.CYCLE (?x1) :-> ?x2
  "Mapeamento de um radar para o ciclo de radar que é um número real"
  :def (and (radar ?x1)
            (real ?x2)))
```

APÊNDICE 02 – ONTOLOGIA DE PAPÉIS TEMÁTICOS

```
(define ONTOLOGY THEMATIC-ROLES
  (SLOT-CONSTRAINT-SUGAR FRAME-ONTOLOGY))

(define-class THEMATIC-ROLE (?x)
 "Classe que agrega todas as classes que compõem a ontologia de
 papéis temáticos. Pode ser: agente, evento, tema, beneficiário,
 localização espacial, origem, destino ou localização temporal"
 :def (thing ?x)
 :axiom-def (disjoint-decomposition
             THEMATIC-ROLE
            (setof AGENT
                   EVENT
                   THEME
                   BENEFICIARY
                   LOCATION
                   SOURCE
                   GOAL
                   TIME)))

(define-class AGENT (?x)
  "Papel temático agente"
  :def (thematic-role ?x))

(define-class EVENT (?x)
  "Evento associado aos papéis temáticos"
  :def (and (thematic-role ?x)
            (can-have-one ?x has-agent)
            (can-have-one ?x has-theme)
            (can-have-one ?x has-beneficiary)
            (can-have-one ?x has-location)
            (can-have-one ?x has-source)
            (can-have-one ?x has-goal)
            (can-have-one ?x has-time)))

(define-class THEME (?x)
  "Papel temático tema"
  :def (thematic-role ?x))
```

```
(define-class BENEFICIARY (?x)
  "Papel temático beneficiário"
  :def (thematic-role ?x))

(define-class LOCATION (?x)
  "Papel temático localização espacial"
  :def (thematic-role ?x))

(define-class SOURCE (?x)
  "Papel temático origem"
  :def (thematic-role ?x))

(define-class GOAL (?x)
  "Papel temático destino"
  :def (thematic-role ?x))

(define-class TIME (?x)
  "Papel temático localização temporal"
  :def (thematic-role ?x))

(define-relation HAS-AGENT (?x1 ?x2)
  "Mapeamento de um evento para um agente"
  :def (and (event ?x1)
            (agent ?x2)))

(define-relation HAS-THEME (?x1 ?x2)
  "Mapeamento de um evento para um tema"
  :def (and (event ?x1)
            (theme ?x2)))

(define-relation HAS-BENEFICIARY (?x1 ?x2)
  "Mapeamento de um evento para um beneficiário"
  :def (and (event ?x1)
            (beneficiary ?x2)))

(define-relation HAS-LOCATION (?x1 ?x2)
  "Mapeamento de um evento para uma localização espacial"
  :def (and (event ?x1)
            (location ?x2)))

(define-function HAS-SOURCE (?x1)  :->  ?x2
```

```
  "Mapeamento de um evento para uma origem"
  :def (and (event ?x1)
            (source ?x2)))

(define-function HAS-GOAL (?x1)  :->  ?x2
  "Mapeamento de um evento para um destino"
  :def (and (event ?x1)
            (goal ?x2)))

(define-function HAS-TIME (?x1)  :->  ?x2
  "Mapeamento de um evento para uma localização temporal"
  :def (and (event ?x1)
            (time ?x2)))
```

APÊNDICE 03 – ONTOLOGIA DE EVENTOS DE TRÁFEGO AÉREO

```
(define-ontology AIR-TRAFFIC-EVENT
  (THEMATIC-ROLES AIR-TRAFFIC))

(define-class AIR-TRAFFIC-EVENT (?x)
  "Evento de tráfego aéreo: classe que agrega todas as classes que
  compõem a ontologia de eventos de tráfego aéreo"
 :def (event ?x)
 :axiom-def (disjoint-decomposition
             AIR-TRAFFIC-EVENT
             (setof TAKE-OFF
                    LAND
                    DETECT
                    LOSE
                    CLIMB
                    DESCENT
                    ESTABLISH
                    REACH
                    TURN
                    CROSS
                    SQUAWK
                    CLEAR)))

(define-class TAKE-OFF (?x)
  "Decolar: evento cujo agente é uma aeronave e cuja localização é: um
  aeródromo ou uma pista"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (or (has-one-of-type ?x has-location airdrome)
                (has-one-of-type ?x has-location runway)))

(define-class LAND (?x)
  "Pousar: evento cujo agente é uma aeronave e cuja localização é: um
  aeródromo ou uma pista"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (or (has-one-of-type ?x has-location airdrome)
```

```
                (has-one-of-type ?x has-location runway)))

(define-class DETECT (?x)
  "Detectar: evento cujo agente é um radar e cujo tema é uma aeronave"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent radar)
            (has-one-of-type ?x has-theme aircraft)))

(define-class LOSE (?x)
  "Perder: evento cujo agente é um radar e cujo tema é uma aeronave"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent radar)
            (has-one-of-type ?x has-theme aircraft)))

 (define-class CLIMB (?x)
  "Subir: evento cujo agente é uma aeronave e que tem uma origem e um
   destino que são níveis de vôo"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (has-one-of-type ?x has-source flight-level)
            (has-one-of-type ?x has-goal flight-level)))

(define-class DESCENT (?x)
  "Descer: evento cujo agente é uma aeronave e que tem uma origem e um
   destino que são níveis de vôo"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (has-one-of-type ?x has-source flight-level)
            (has-one-of-type ?x has-goal flight-level)))

(define-class ESTABLISH (?x)
  "Estabilizar: evento cujo agente é uma aeronave e cujo tema é um rumo"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (has-one-of-type ?x has-theme course)))
```

```
(define-class REACH (?x)
  "Atingir: evento cujo agente é uma aeronave e cujo tema é um nível de
  vôo"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (has-one-of-type ?x has-theme flight-level)))

(define-class TURN (?x)
  "Fazer uma curva: evento cujo agente é uma aeronave e que tem um
  destino que é uma direção"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (has-one-of-type ?x has-goal direction)))

(define-class CROSS (?x)
  "Cruzar: evento cujo agente é uma aeronave e que tem um tema que pode
  ser: um ponto geográfico, um fixo, um auxílio à navegação, um radar, um
  espaço aéreo ou um aeródromo"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (or (has-one-of-type ?x has-theme point)
                (has-one-of-type ?x has-theme fix)
                (has-one-of-type ?x has-theme navaid)
                (has-one-of-type ?x has-theme radar)
                (has-one-of-type ?x has-theme airspace)
                (has-one-of-type ?x has-theme airdrome)))

(define-class SQUAWK (?x)
  "Acionar: evento cujo agente é uma aeronave e cujo tema que pode ser:
  um código transponder ou uma identificação transponder"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent aircraft)
            (or (has-one-of-type ?x has-theme transponder-code)
                (has-one-of-type ?x has-theme transponder-ident)))
```

```
(define-class CLEAR (?x)
  "Autorizar: evento cujo agente é um controle de tráfego aéreo, que tem
  um tema que pode ser: um nível de vôo, um ponto geográfico, um fixo ou
  um auxílio à navegação e um beneficiário que é uma aeronave"
  :def (and (air-traffic-event ?x)
            (has-one-of-type ?x has-agent at-control)
            (or (has-one-of-type ?x has-theme flight-level)
                (has-one-of-type ?x has-theme point)
                (has-one-of-type ?x has-theme fix)
                (has-one-of-type ?x has-theme navaid))
            (has-one-of-type ?x has-beneficiary aircraft)))
```

APÊNDICE 04 –REPRESENTAÇÃO DE EVENTOS DE TRÁFEGO AÉREO E FRASES GERADAS

Neste apêndice, apresentamos 13 exemplos de representação de eventos de tráfego aéreo com as respectivas frases geradas pelo gerador de frases.

1

```
[ sem: [ event: [onto: detect]
         agent: [onto: radar
                 at-identifier: Marília]
         theme: [onto: aircraft
                 at-identifier: vrg1111]
         time: [(10,22,10)] ]
  voz: ativa ]
```

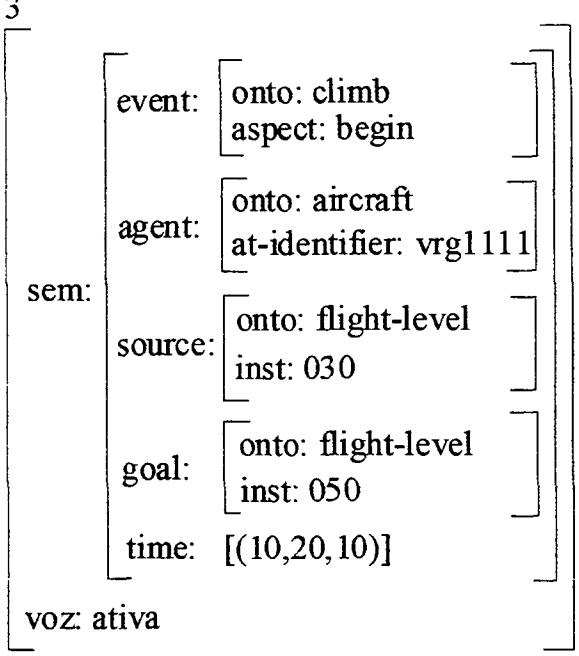
O radar Marília detectou a aeronave vrg1111 às 10h 22min 10s.

2

```
[ sem: [ event: [onto: lose]
         agent: [onto: radar
                 at-identifier: Marília]
         theme: [onto: aircraft
                 at-identifier: vrg1111]
         time: [(10,22,10)] ] ]
```

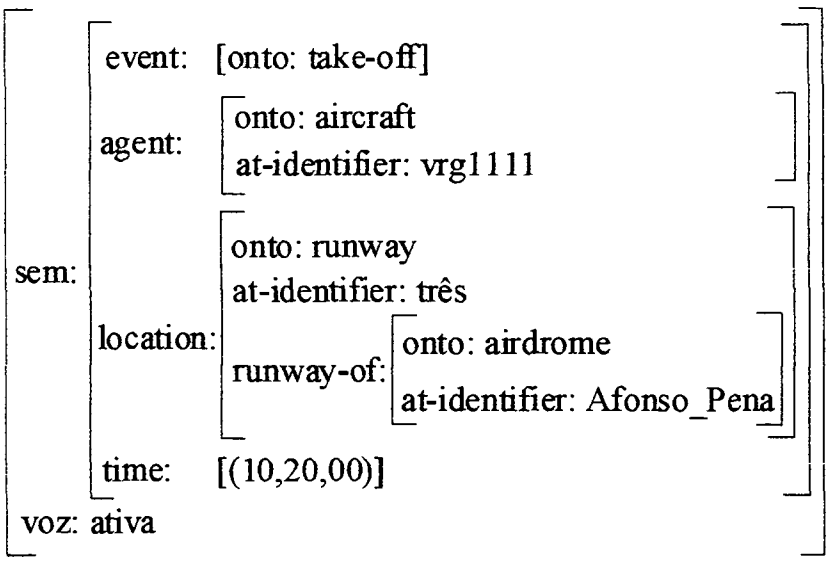
A aeronave vrg1111 foi perdida pelo radar Marília às 10h 22min 10s.

3



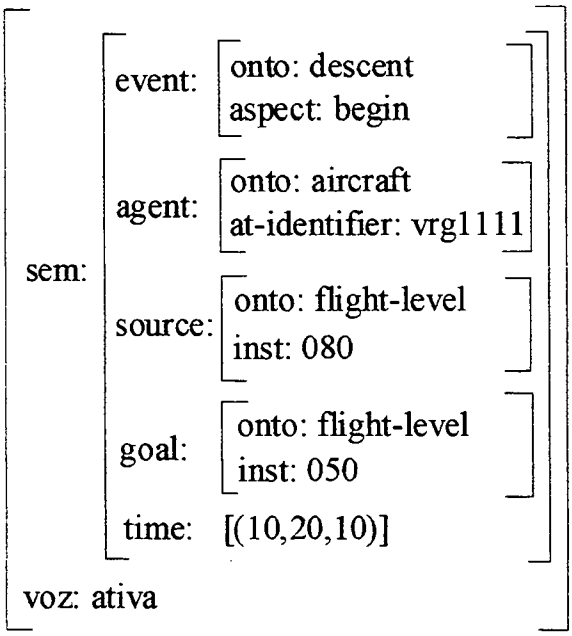
A aeronave vrg1111 começou a subir do nível de vôo 030 para o nível de vôo 050 às 10h 20 min 10s.

4



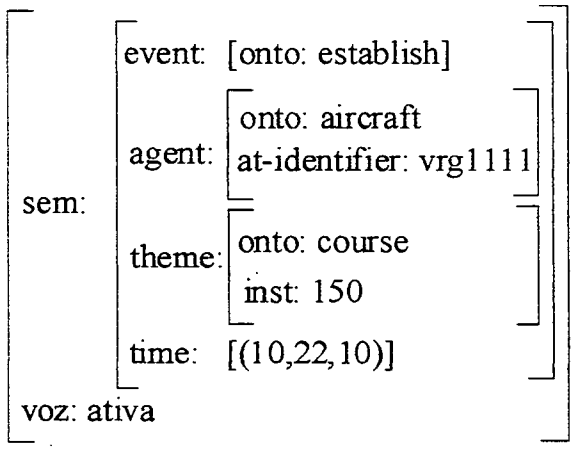
A aeronave vrg1111 decolou da pista três do aeródromo Afonso Pena às 10h 20min 00s.

5



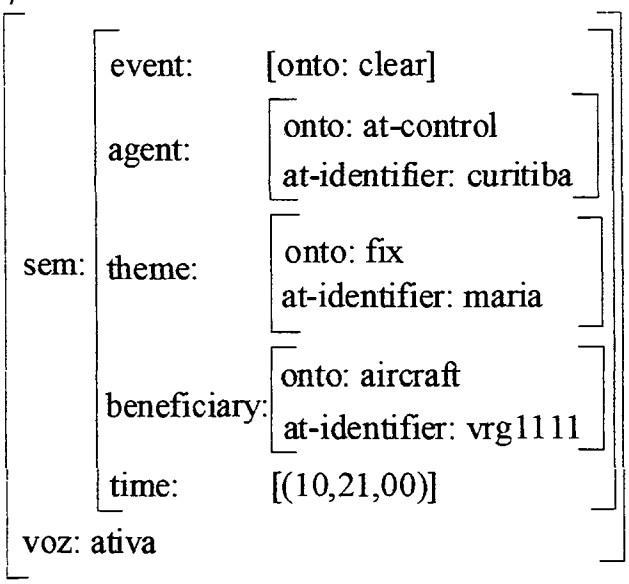
A aeronave vrg1111 começou a descer do nível de vôo 080 para o nível de vôo 050 às 10h 20min 10s.

6



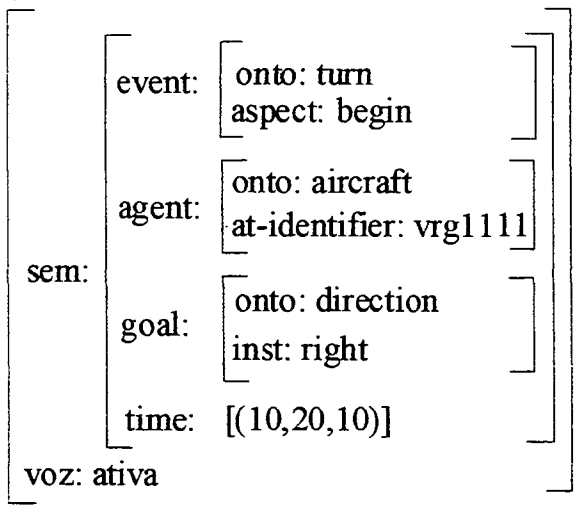
A aeronave vrg1111 estabilizou-se no rumo 150 às 10h 22min 10s.

7



O controle Curitiba autorizou o fixo maria para a aeronave vrg1111 às 10h 21min 00s.

8



A aeronave vrg1111 começou a fazer uma curva à direita às 10h 20min 10s.

9

```
[ sem: [ event: [onto: reach]
         agent: [onto: aircraft
                 at-identifier: vrg1111]
         theme: [onto: flight-level
                 inst: 150]
         time: [(10,22,10)] ]
  voz: passiva ]
```

O nível de vôo 150 foi atingido pela aeronave vrg1111 às 10h 22min 10s.

10

```
[ sem: [ event: [onto: squawk]
         agent: [onto: aircraft
                 at-identifier: vrg1111]
         theme: [onto: transponder-ident]
         time: [(10,22,10)] ]
  voz: ativa ]
```

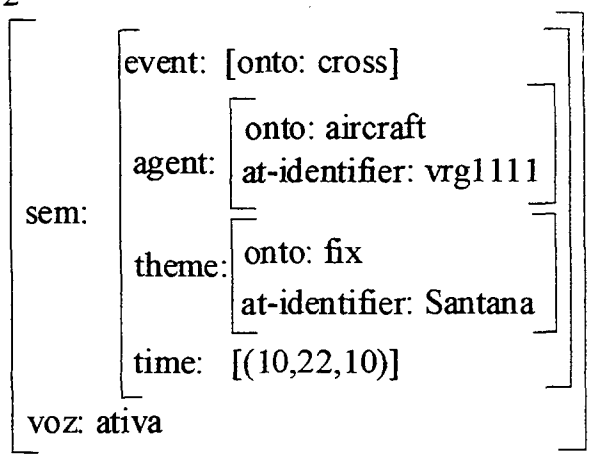
A aeronave vrg1111 acionou a identificação transponder às 10h 22min 10s.

11

```
[ sem: [ event: [onto: squawk]
         agent: [onto: aircraft
                 at-identifier: vrg1111]
         theme: [onto: transponder-code
                 inst: 4444]
         time: [(10,22,10)] ]
  voz: passiva ]
```

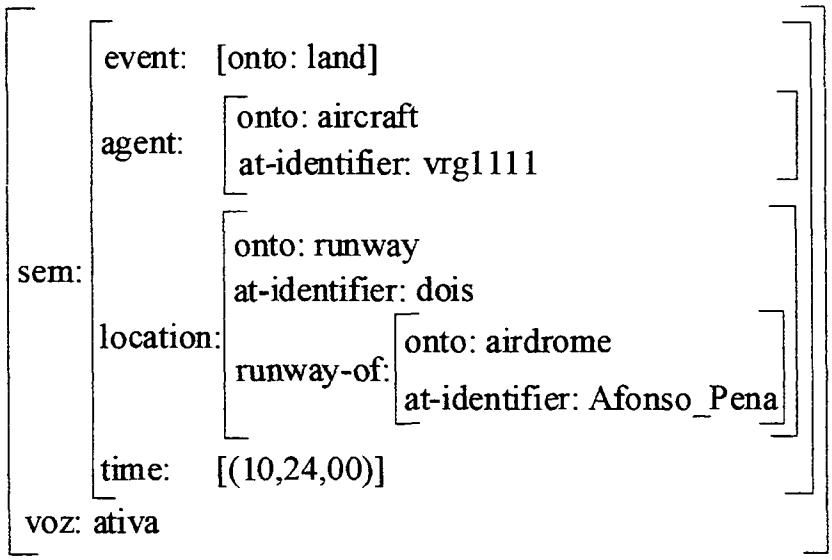
O código transponder 4444 foi acionado pela aeronave vrg1111 às 10h 22min 10s.

12



A aeronave vrg1111 cruzou o fixo Santana às 10h 22min 10s.

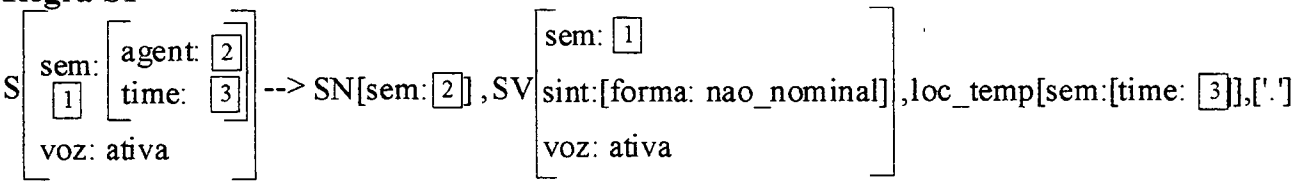
13



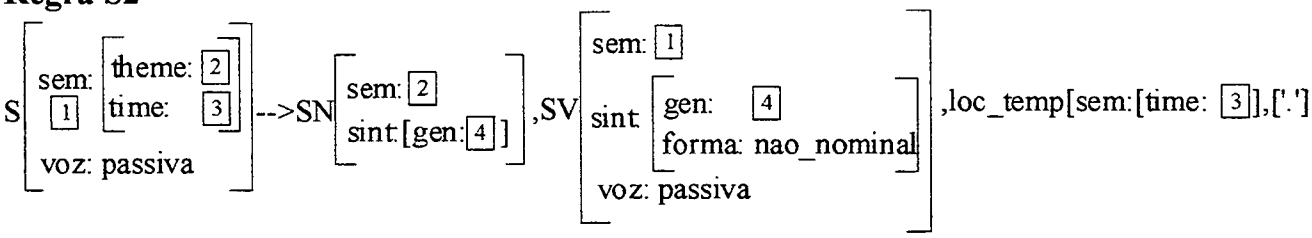
A aeronave vrg1111 pousou na pista dois do aeródromo Afonso Pena às 10h 24min 00s.

APÊNDICE 05 – REGRAS GRAMATICAIS E DICIONÁRIO DO GERADOR DE FRASES

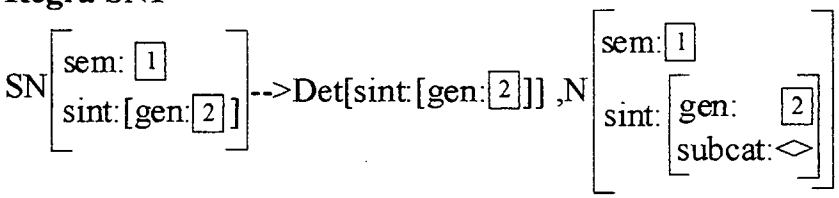
Regra S1



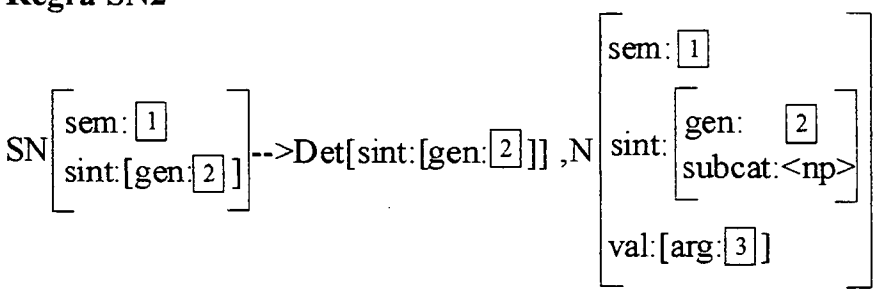
Regra S2



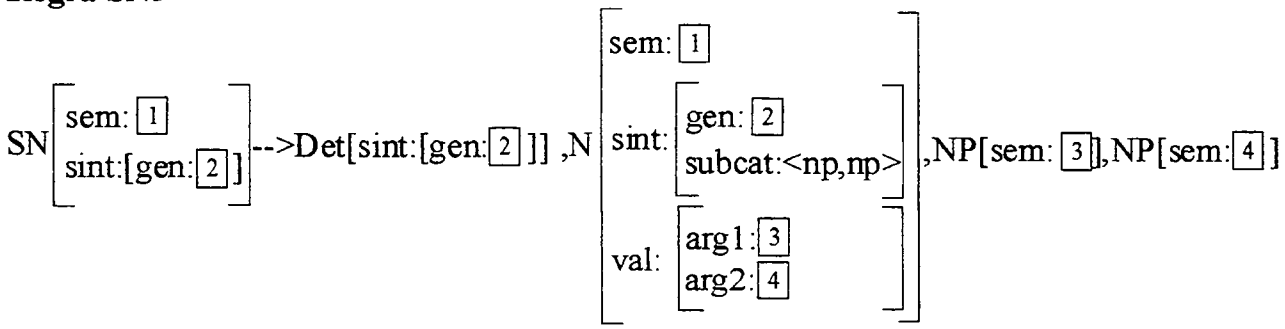
Regra SN1



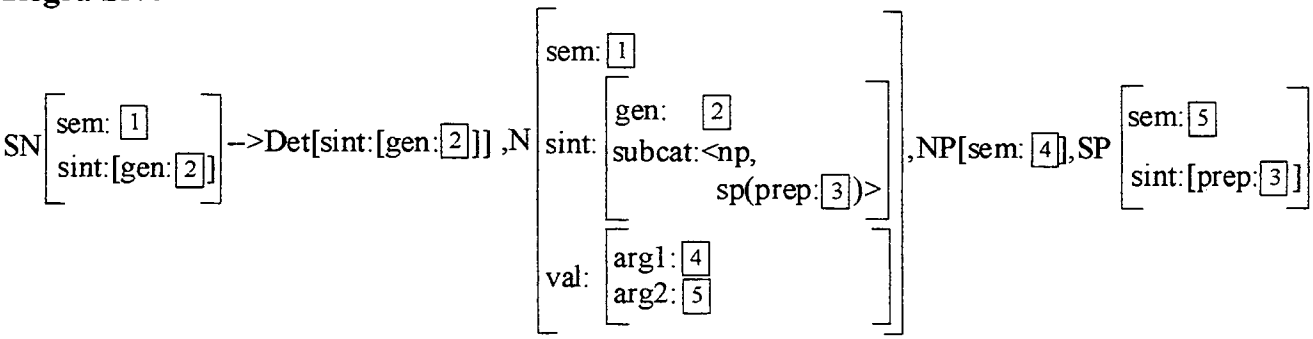
Regra SN2



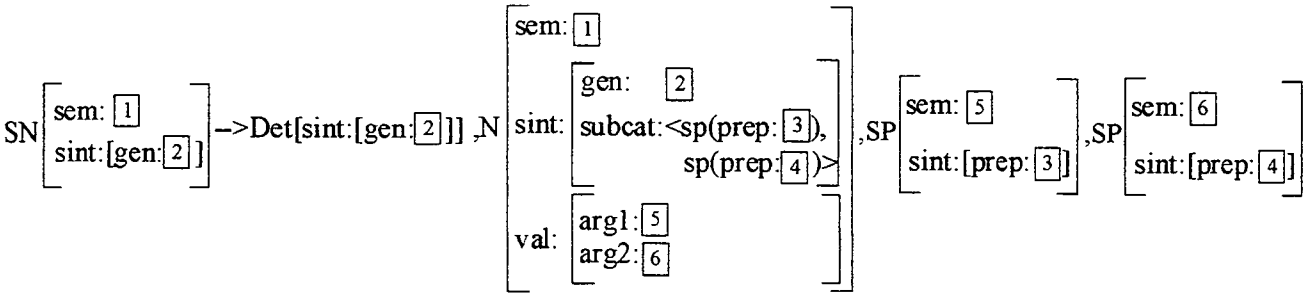
Regra SN3



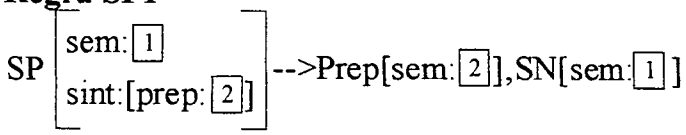
Regra SN4



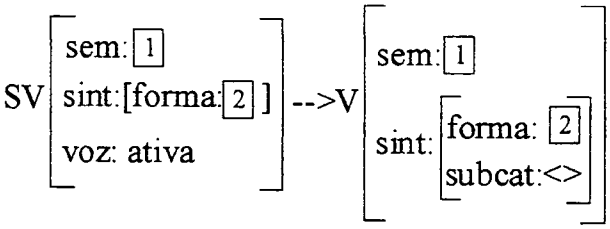
Regra SN5



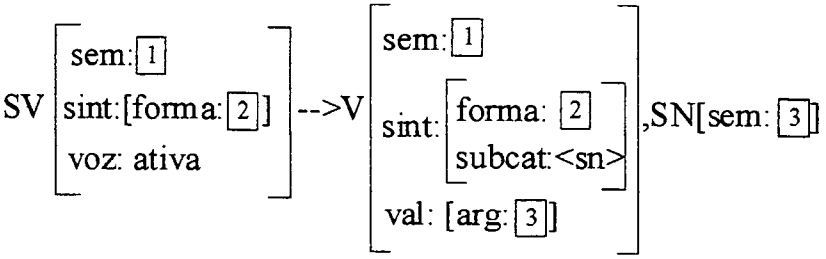
Regra SP1



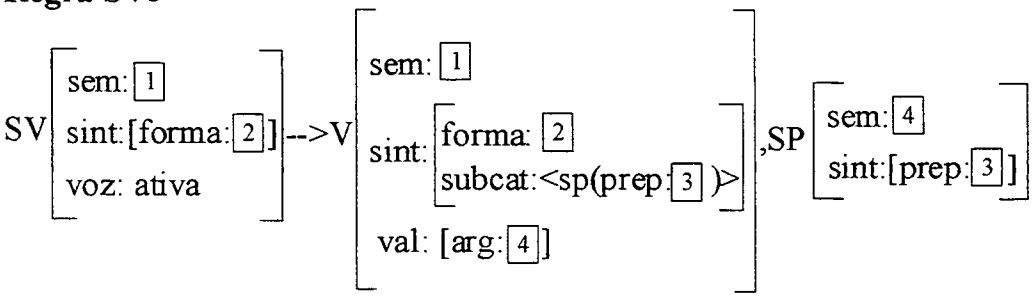
Regra SV1



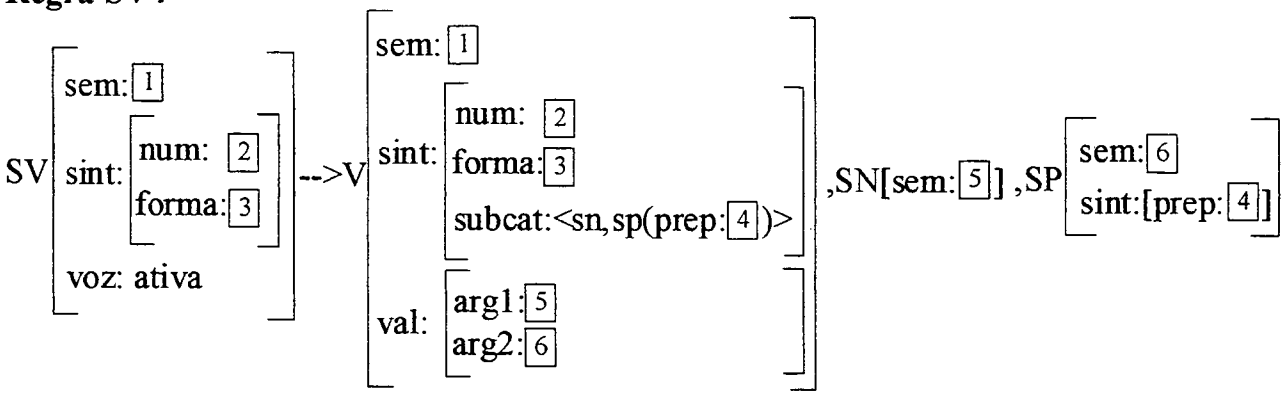
Regra SV2



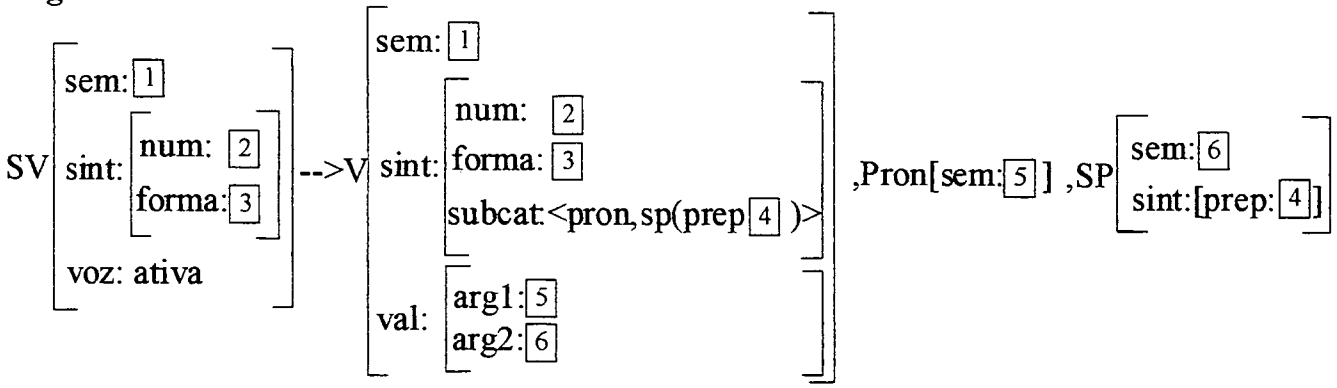
Regra SV3



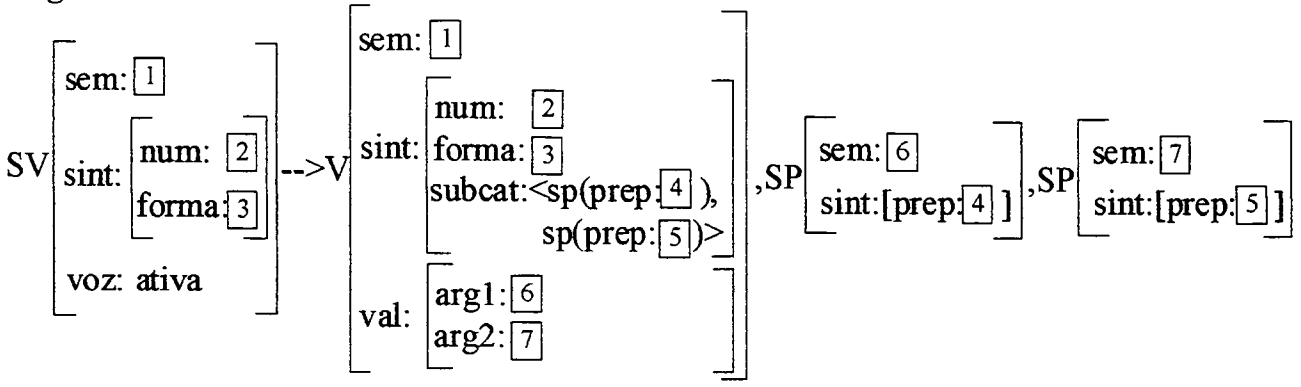
Regra SV4



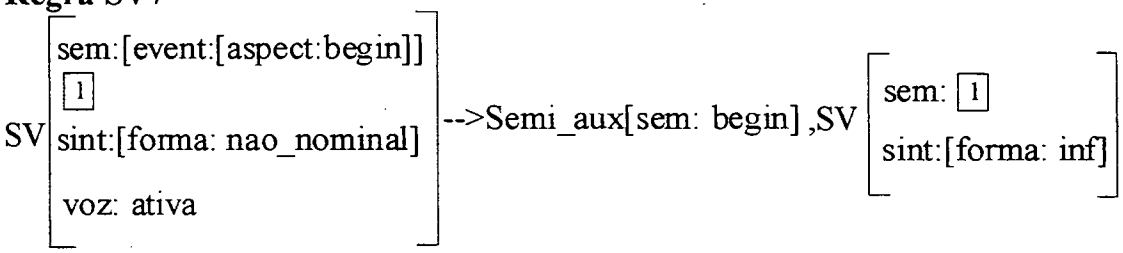
Regra SV5



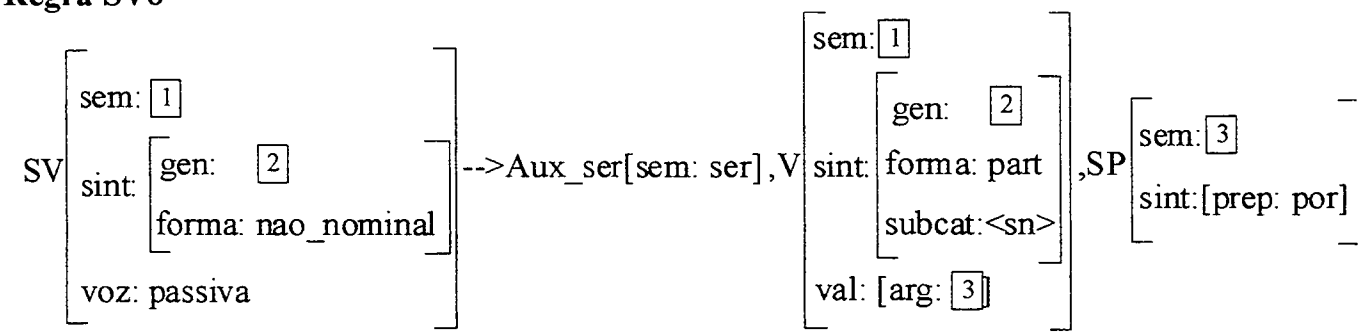
Regra SV6



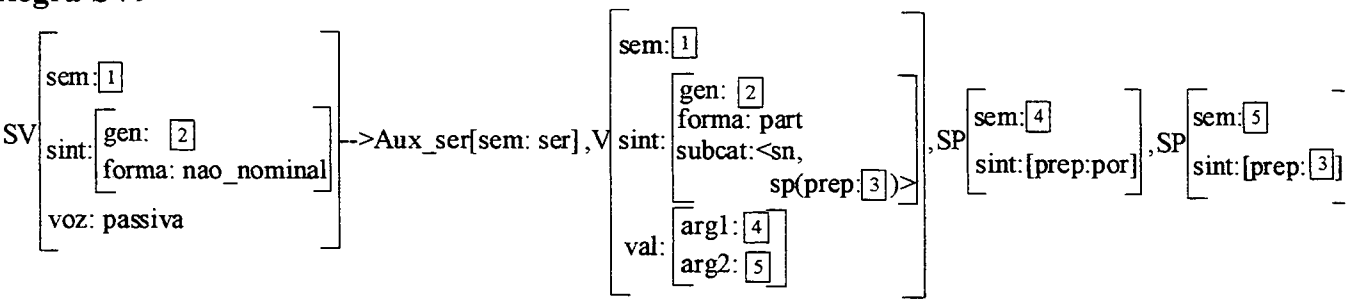
Regra SV7



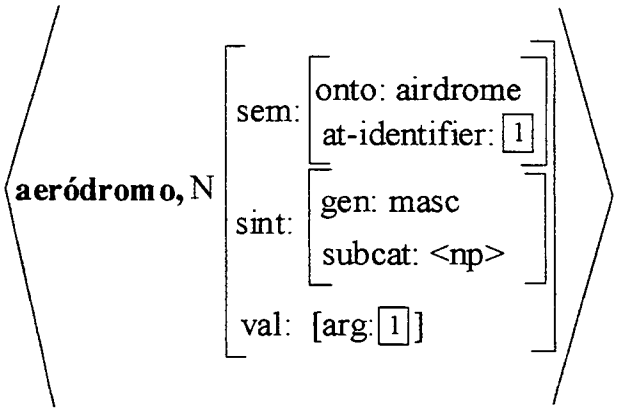
Regra SV8

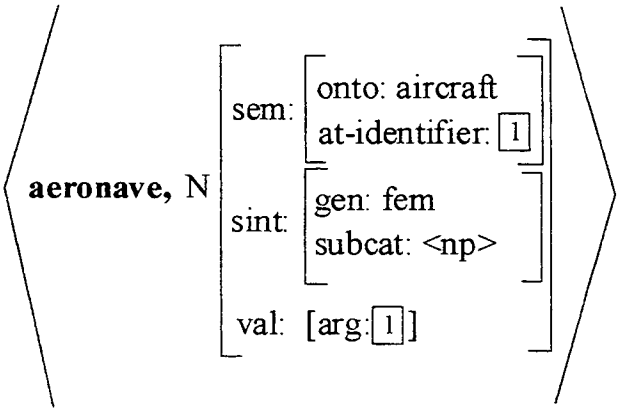


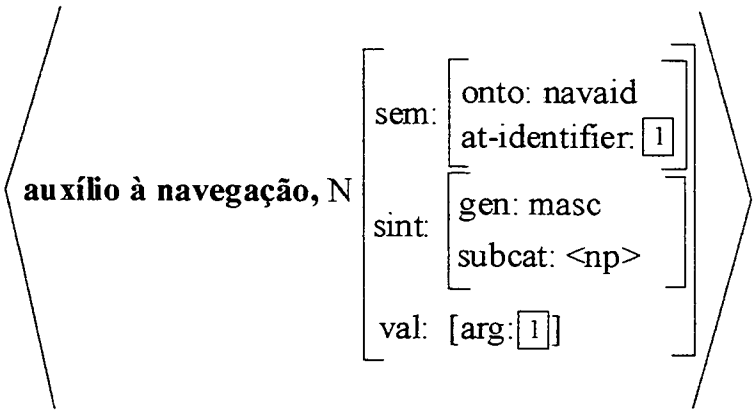
Regra SV9

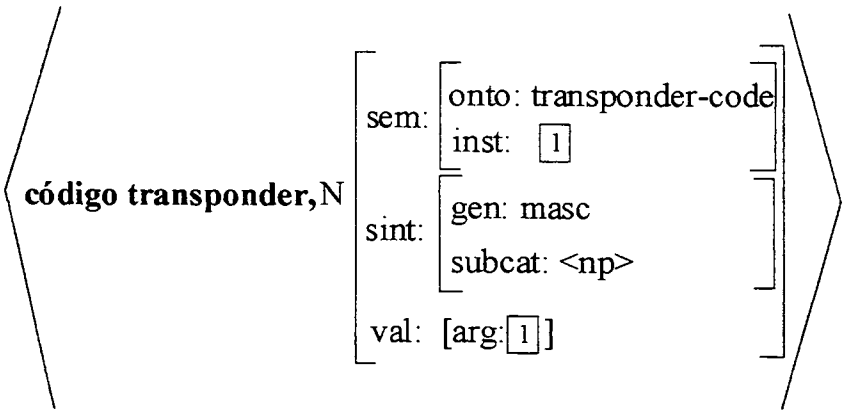


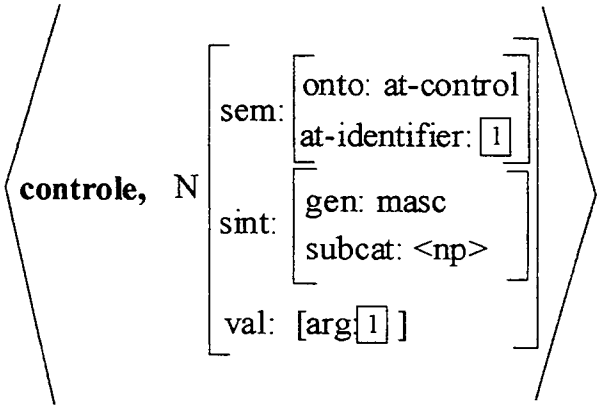
Dicionário

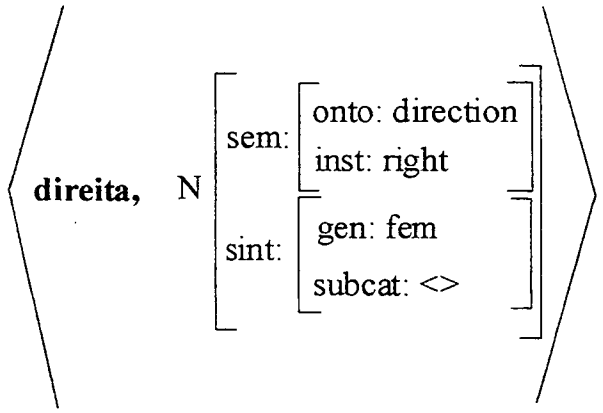


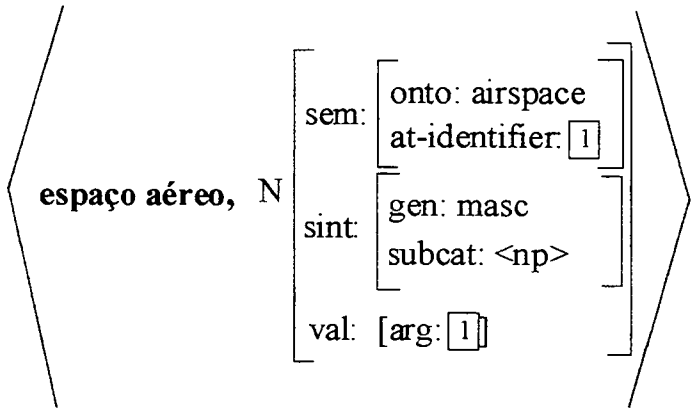

aeronave, N
sem:
onto: aircraft
at-identifier: 1
sint:
gen: fem
subcat: <np>
val: [arg:1]

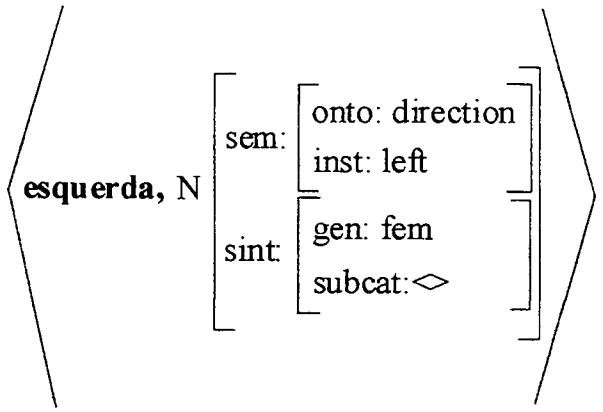

auxílio à navegação, N
sem:
onto: navaid
at-identifier: 1
sint:
gen: masc
subcat: <np>
val: [arg:1]

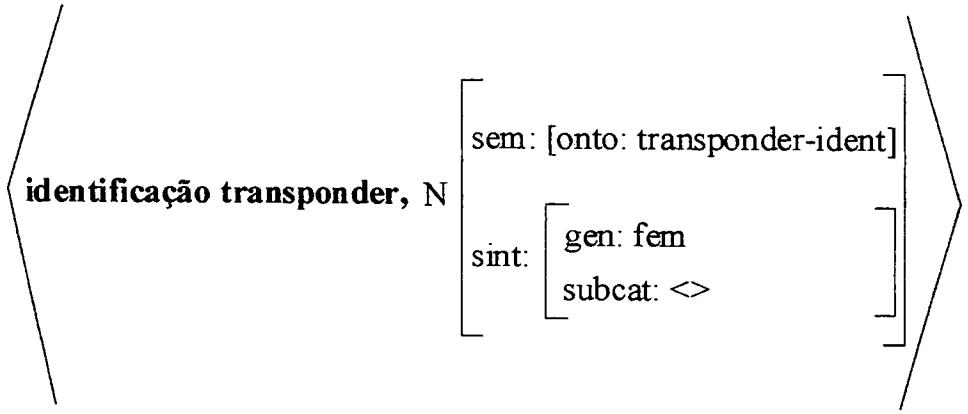

código transponder, N
sem:
onto: transponder-code
inst: 1
sint:
gen: masc
subcat: <np>
val: [arg:1]

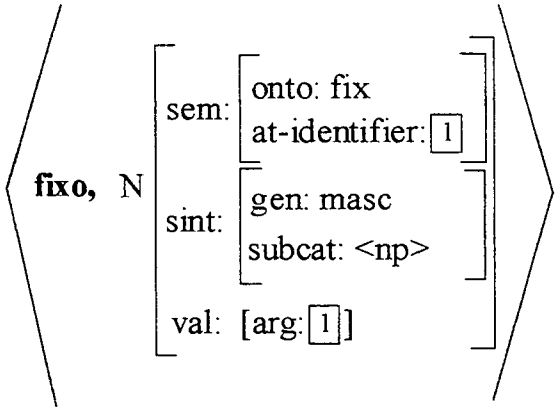

controle, N
sem:
onto: at-control
at-identifier: 1
sint:
gen: masc
subcat: <np>
val: [arg:1]

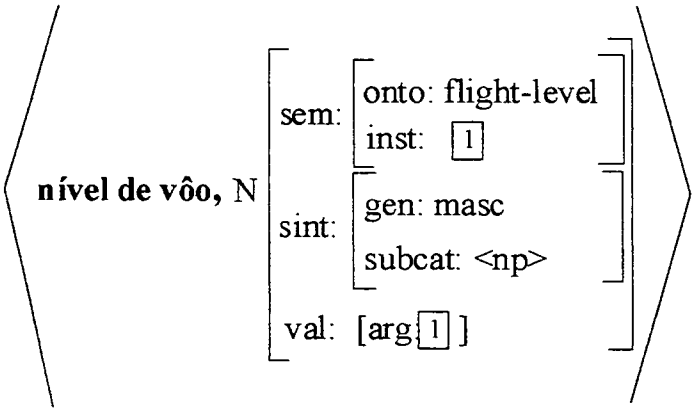

direita, N
sem:
onto: direction
inst: right
sint:
gen: fem
subcat: <>

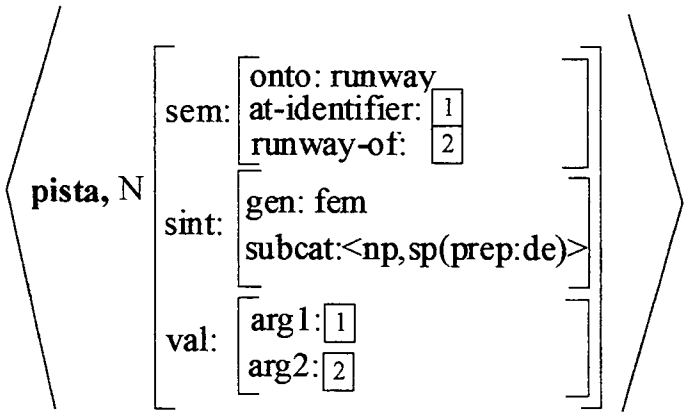

espaço aéreo, N
sem:
onto: airspace
at-identifier: 1
sint:
gen: masc
subcat: <np>
val: [arg: 1]

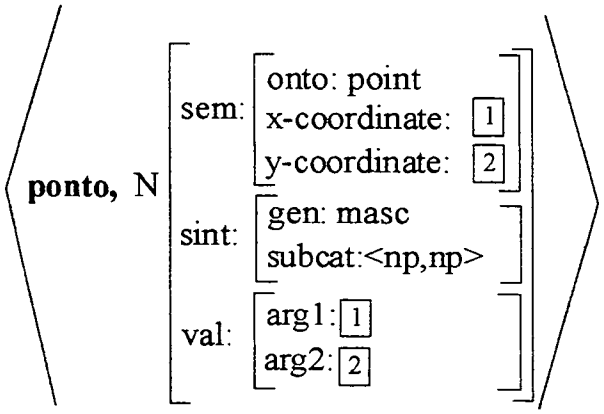

esquerda, N
sem:
onto: direction
inst: left
sint:
gen: fem
subcat:<>

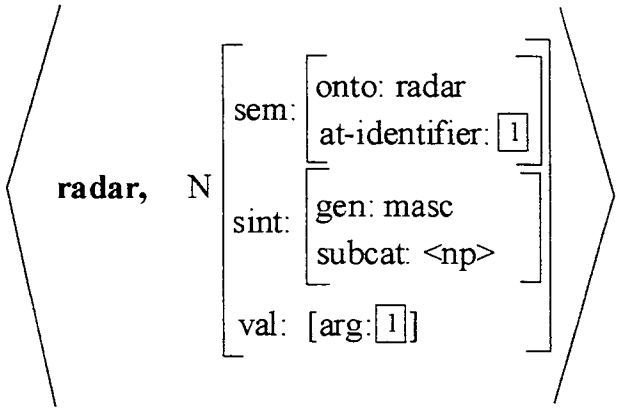

identificação transponder, N
sem: [onto: transponder-ident]
sint:
gen: fem
subcat: <>

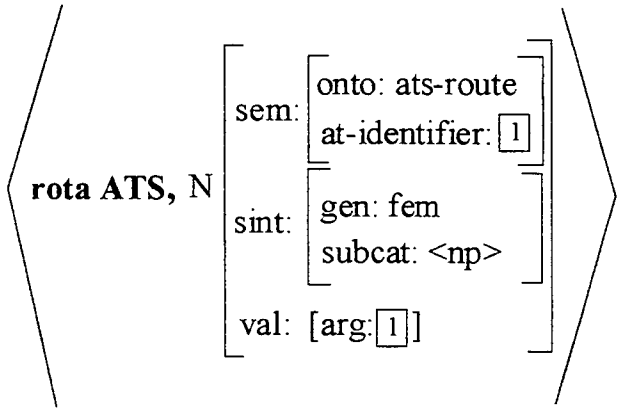

fixo, N
sem:
onto: fix
at-identifier: 1
sint:
gen: masc
subcat: <np>
val: [arg: 1]

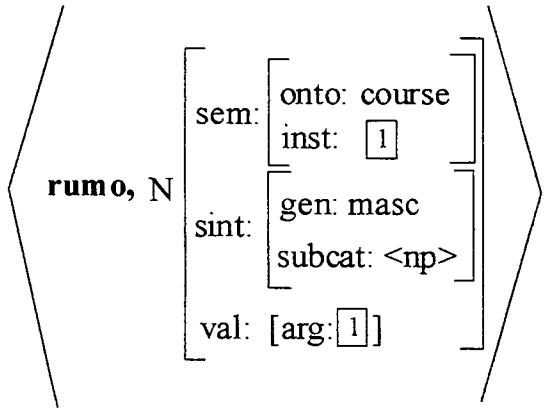

nível de vôo, N
sem:
onto: flight-level
inst: 1
sint:
gen: masc
subcat: <np>
val: [arg: 1]

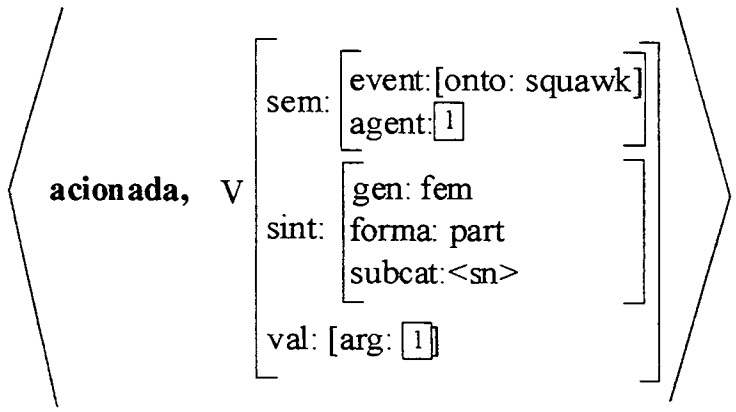

pista, N
sem:
onto: runway
at-identifier: 1
runway-of: 2
sint:
gen: fem
subcat:<np,sp(prep:de)>
val:
arg1: 1
arg2: 2

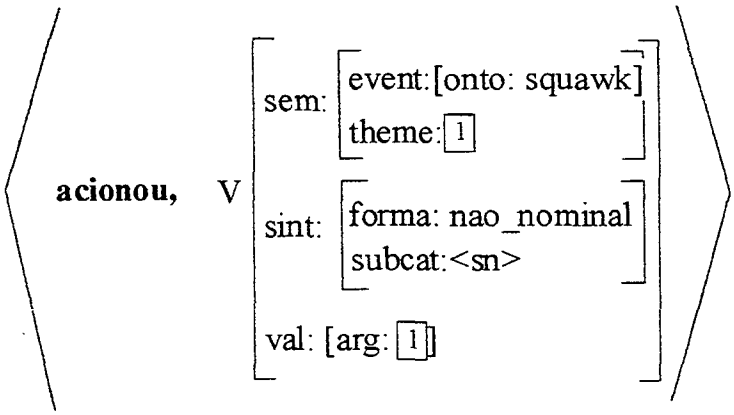

ponto, N
sem:
onto: point
x-coordinate: 1
y-coordinate: 2
sint:
gen: masc
subcat:<np,np>
val:
arg1: 1
arg2: 2

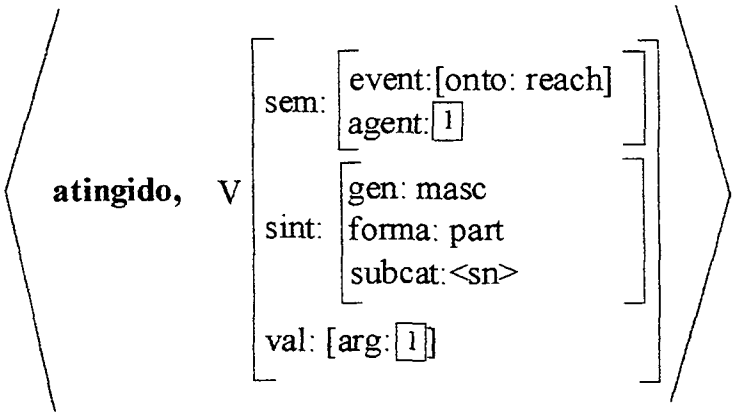

radar, N
sem: onto: radar
at-identifier: 1
sint: gen: masc
subcat: <np>
val: [arg: 1]

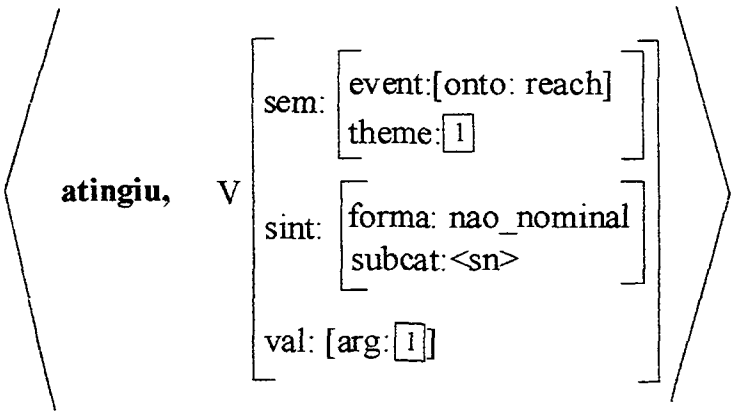

rota ATS, N
sem: onto: ats-route
at-identifier: 1
sint: gen: fem
subcat: <np>
val: [arg: 1]

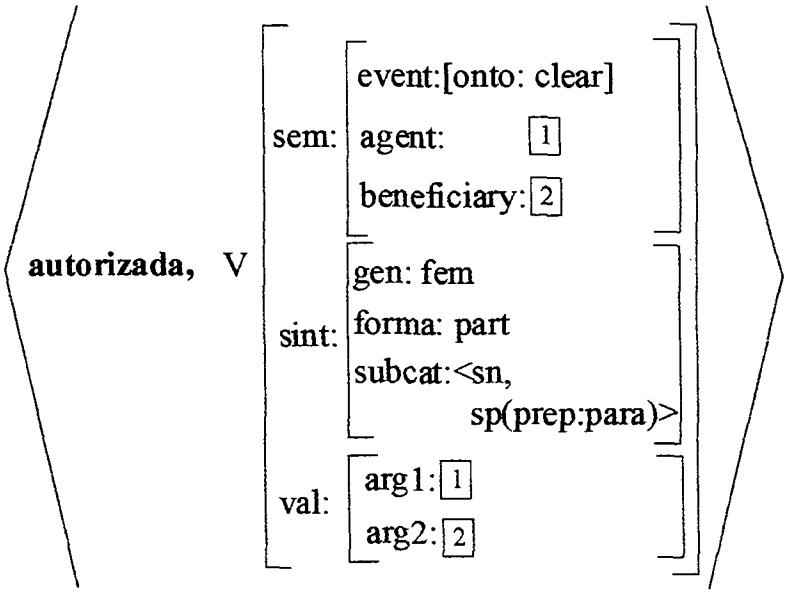

rumo, N
sem: onto: course
inst: 1
sint: gen: masc
subcat: <np>
val: [arg: 1]

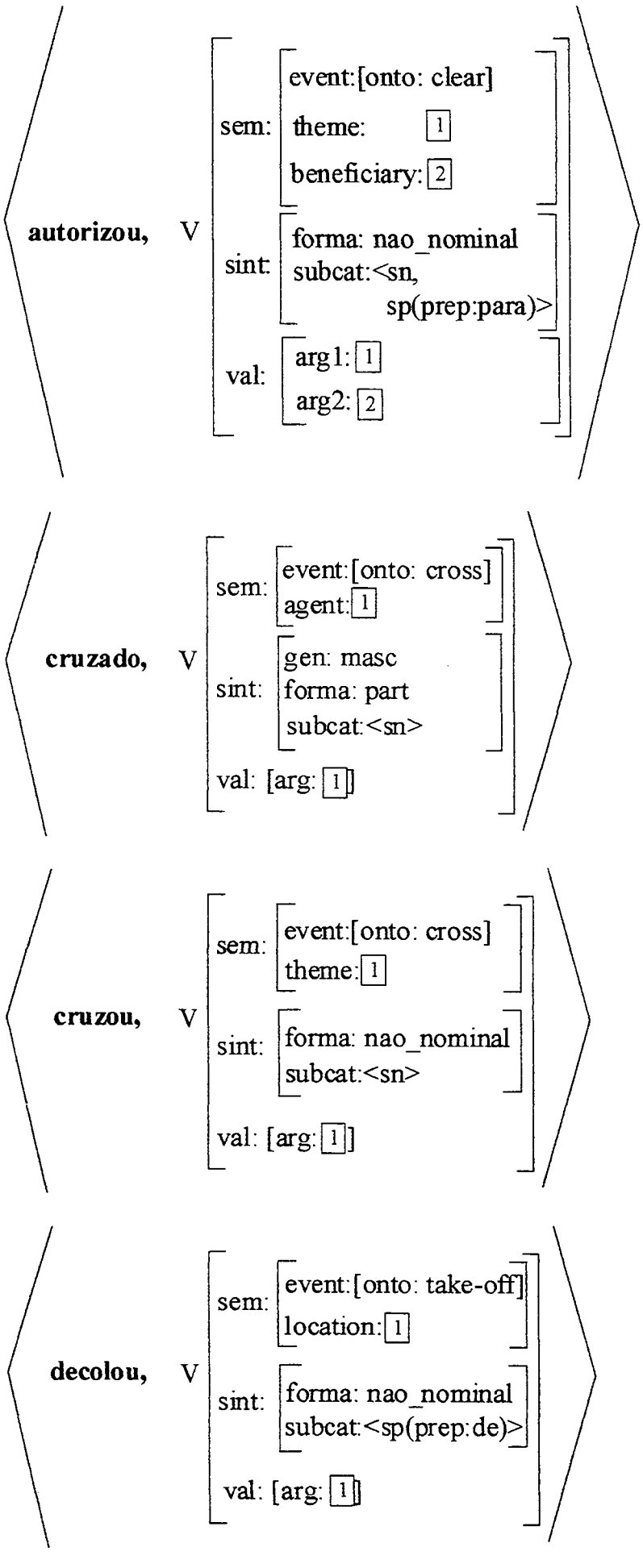

acionada, V
sem: event: [onto: squawk]
agent: 1
sint: gen: fem
forma: part
subcat: <sn>
val: [arg: 1]

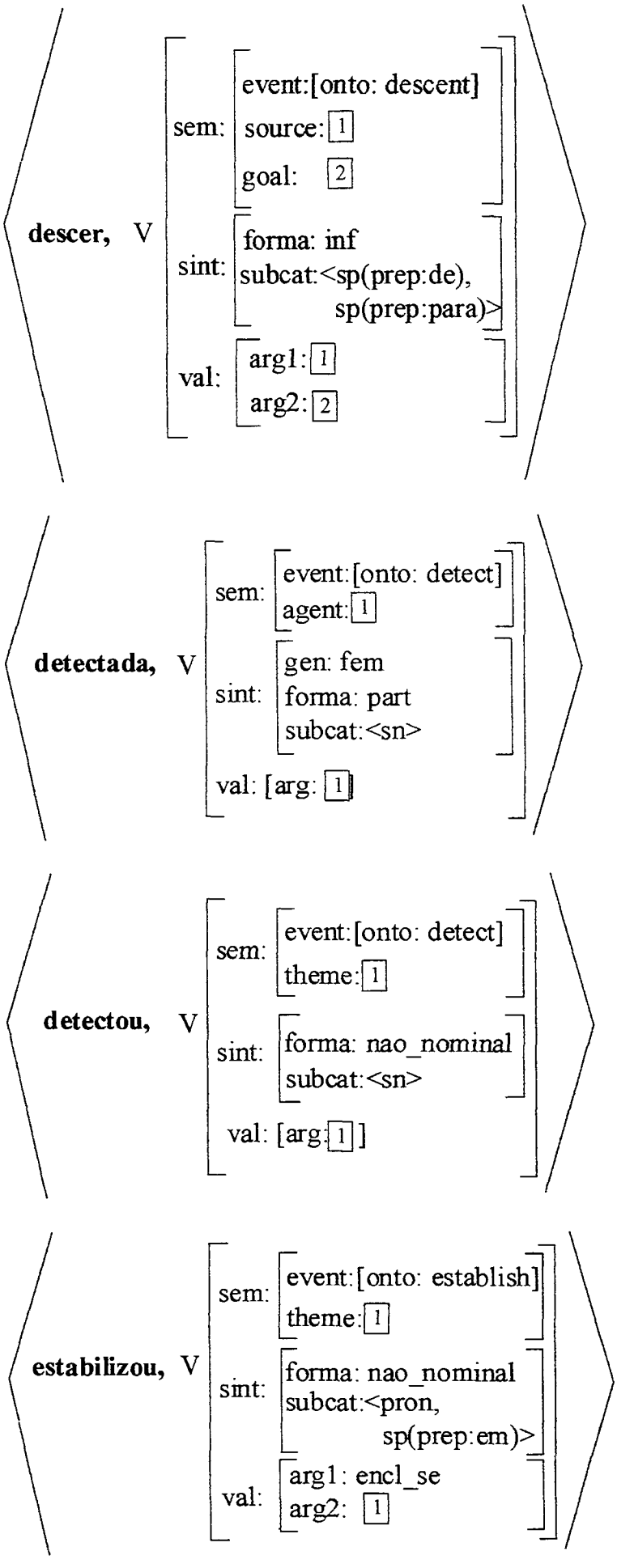

acionou, V
sem: event:[onto: squawk]
theme: 1
sint: forma: nao_nominal
subcat:<sn>
val: [arg: 1]

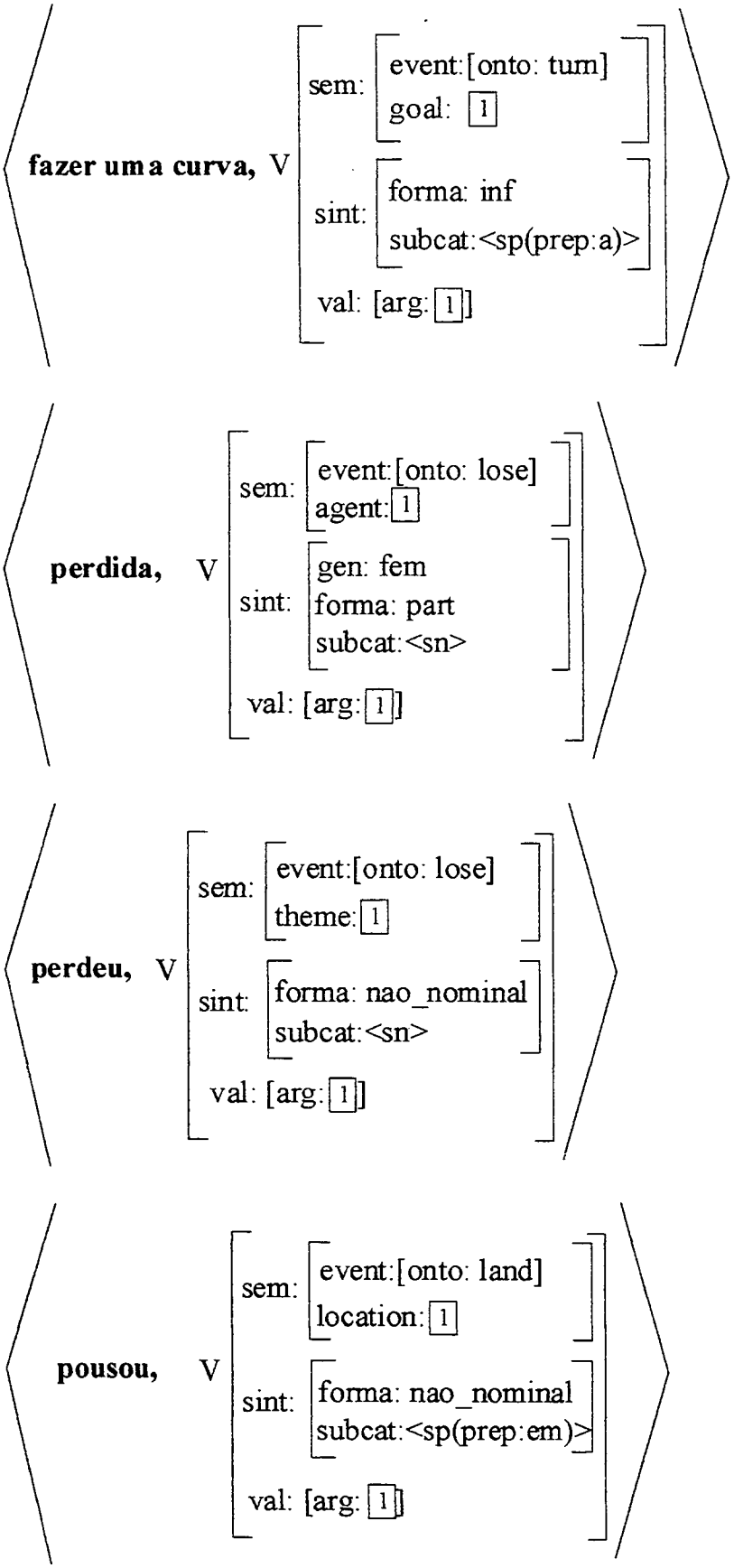

atingido, V
sem: event:[onto: reach]
agent: 1
sint: gen: masc
forma: part
subcat:<sn>
val: [arg: 1]

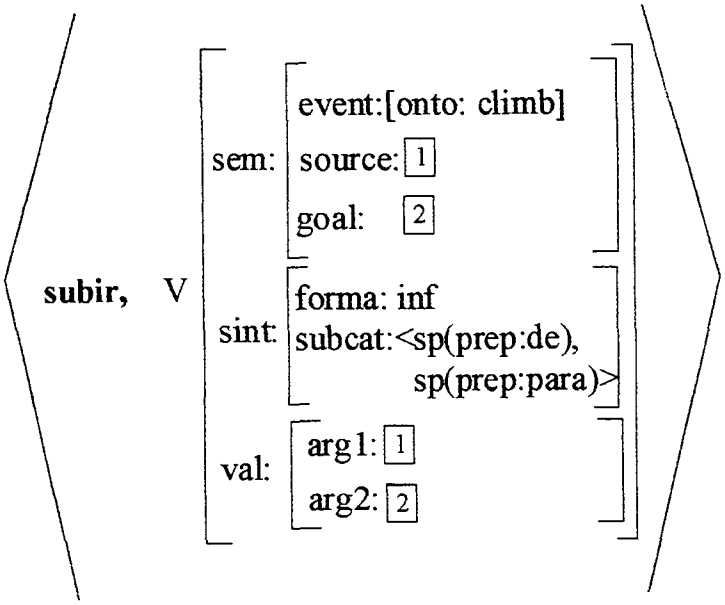

atingiu, V
sem: event:[onto: reach]
theme: 1
sint: forma: nao_nominal
subcat:<sn>
val: [arg: 1]

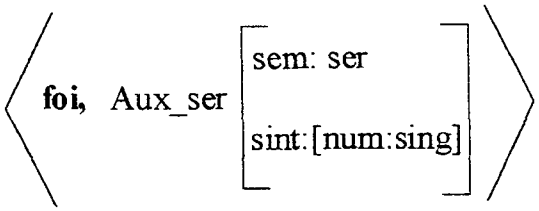

autorizada, V
sem: event:[onto: clear]
agent: 1
beneficiary: 2
sint: gen: fem
forma: part
subcat:<sn,
sp(prep:para)>
val: arg1: 1
arg2: 2

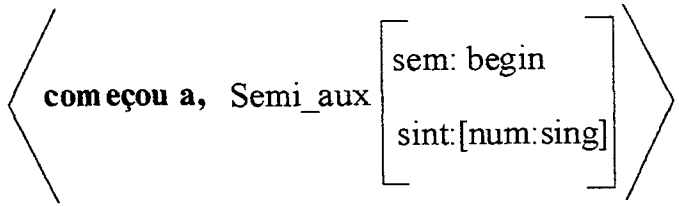

autorizou, V
sem: event:[onto: clear]
theme: 1
beneficiary: 2
sint: forma: nao_nominal
subcat:<sn, sp(prep:para)>
val: arg1: 1
arg2: 2
cruzado, V
sem: event:[onto: cross]
agent: 1
sint: gen: masc
forma: part
subcat:<sn>
val: [arg: 1]
cruzou, V
sem: event:[onto: cross]
theme: 1
sint: forma: nao_nominal
subcat:<sn>
val: [arg: 1]
decolou, V
sem: event:[onto: take-off]
location: 1
sint: forma: nao_nominal
subcat:<sp(prep:de)>
val: [arg: 1]


descer, V
sem: event:[onto: descent]
source: 1
goal: 2
sint: forma: inf
subcat:<sp(prep:de), sp(prep:para)>
val: arg1: 1
arg2: 2
detectada, V
sem: event:[onto: detect]
agent: 1
sint: gen: fem
forma: part
subcat:<sn>
val: [arg: 1]
detectou, V
sem: event:[onto: detect]
theme: 1
sint: forma: nao_nominal
subcat:<sn>
val: [arg: 1]
estabilizou, V
sem: event:[onto: establish]
theme: 1
sint: forma: nao_nominal
subcat:<pron, sp(prep:em)>
val: arg1: encl_se
arg2: 1


fazer uma curva, V
sem:
event:[onto: turn]
goal: 1
sint:
forma: inf
subcat:<sp(prep:a)>
val: [arg: 1]
perdida, V
sem:
event:[onto: lose]
agent:1
sint:
gen: fem
forma: part
subcat:<sn>
val: [arg:1]
perdeu, V
sem:
event:[onto: lose]
theme:1
sint:
forma: nao_nominal
subcat:<sn>
val: [arg:1]
pousou, V
sem:
event:[onto: land]
location:1
sint:
forma: nao_nominal
subcat:<sp(prep:em)>
val: [arg: 1]







< **a**, Det[sint:[gen: fem]] >

< **o**, Det[sint:[gen: masc]] >

< 1 , NP[sem: 1]>

< **a**, Prep[sem: a] >

< **de**, Prep[sem: de] >

< **em**, Prep[sem: em] >

< **para**, Prep[sem: para] >

< **por**, Prep[sem: por] >

< **-se,** Pron[sem: encl_se] >

< **às** [1] **h** [2] **min** [3] **s,** loc_temp[sem:[time:([1],[2],[3])]] >