Instituto de Computação
Universidade Estadual de Campinas

Um Ambiente para a Análise Superficial de Línguas Baseado em Autômatos Finitos

Mario José Cáccamo[1]

Março de 1998

Banca Examinadora:

- Prof. Dr. Tomasz Kowaltowski (Orientador)
- Profa. Dra. Maria das Graças Volpe Nunes
  Instituto de Ciências Matemáticas - USP São Carlos
- Prof. Dr. Cláudio Leonardo Lucchesi
  Instituto de Computação - UNICAMP
- Prof. Dr. Jorge Stolfi (suplente)
  Instituto de Computação - UNICAMP

[1]**Trabalho financiado pelo CNPq e pela FAPESP.**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Cáccamo, Mario José

C113a Um ambiente para a análise superficial de línguas baseado em autômatos finitos / Mario José Cáccamo -- Campinas, [S.P. :s.n.], 1998.

Orientador : Tomasz Kowaltowski

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento da linguagem natural (Computação). 2. Algoritmos. I. Kowaltowski, Tomasz. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Um Ambiente para a Análise Superficial de Línguas Baseado em Autômatos Finitos

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Mario José Cáccamo e aprovada pela Banca Examinadora.

Campinas, 30 de março de 1998.

Prof. Dr. Tomasz Kowaltowski (Orientador)

Prof. Dra. Ariadne M. B. R. Carvalho
(Co-orientadora)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Tese de Mestrado defendida e aprovada em 30 de março de 1998 pela Banca Examinadora composta pelos professores Doutores

Profa. Dra. Maria das Graças Volpe Nunes

Prof. Dr. Cláudio Leonardo Lucchesi

Tomasz Kowaltowski

Prefácio

A análise sintática é uma componente fundamental da maioria dos sistemas de processamento automático de línguas. Tradicionalmente, esta tarefa foi implementada com técnicas derivadas do formalismo das Gramáticas Livres de Contexto. A demanda por sistemas eficientes motivou a pesquisa em busca de outras alternativas para a análise sintática.

Alguns sistemas de processamento de línguas naturais não precisam de uma análise completa da estrutura profunda das sentenças (árvores de derivação), senão apenas dispor das relações superficiais entre as palavras de um texto. O objetivo de um analisador superficial é determinar quais são as sentenças de uma língua, fornecendo apenas a informação sobre alguma característica em particular, evitando entrar em detalhes correspondentes à estrutura profunda. Este tipo de sistemas é mais eficientes e pode substituir uma análise sintática completa em várias aplicações.

Nesta dissertação propõe-se um ambiente para implementar a análise superficial de línguas. A proposta consiste em representar as sentenças de uma língua usando seqüências de marcas chamadas de padrões sintáticos. Cada marca é uma categoria de palavras (adjetivos, substantivos, advérbios, etc). A hipótese é que o núcleo das sentenças usadas nos textos de uma língua pode ser capturado com um número computacionalmente tratável de padrões sintáticos.

Estruturas de dados baseadas em autômatos finitos foram utilizadas para representar de forma compacta grandes vocabulários de palavras. Os padrões sintáticos são cadeias de símbolos comparáveis, em certo sentido, às palavras de um vocabulário e autômatos mostraram-se adequadas para armazená-los. Além disso, os autômatos permitem a implementação eficiente do algoritmo de reconhecimento proposto, e outros mais complexos, como o conselheiro gramatical apresentado nesta dissertação.

Um dos problemas de muitas das propostas para a análise sintática de línguas é a falta de um método ou fonte de informação para construir um sistema que possa modelar um exemplo real. Como uma alternativa, propõe-se aqui a coleta de padrões sintáticos a partir de *corpos* de texto marcados.

Abstract

Syntactic analysis is an important component of most natural language processing systems. Typically parsers were implemented using techniques derived from Context Free Grammars. The increasing need for efficient systems was one of the reasons to search for new approaches to syntactic analysis.

Some natural language applications do not need complete parsing of the deep structure of the sentences (derivation trees). In these cases, a representation of the surface relations among words in a text is enough. The goal of a surface parser is to recognize the natural language sentences providing information only about some particular features. It is not concerned with the deep structure of the sentences. This kind of parsers are more efficient and can replace a parser implementing a complete syntactic analysis in different situations.

We propose in this dissertation an environment to implement surface parsing of natural languages. In our approach every sentence is represented by a sequence of part-of-speech tags called syntactic pattern. The hypothesis underlying our work is that the core of the sentences used in natural language texts can be captured with a computationally tractable number of syntactic patterns.

Data structures based on finite-state automata have been used in representing large word vocabularies. Syntactic patterns are strings of symbols that can be compared in some sense with words. We have shown that finite-state automata are adequate to store syntactic patterns. Furthermore they allow an efficient implementation of the recognizing algorithms, and other more complex ones, as the agreement adviser presented in this work.

One of the problems common to many approaches for syntactic parsing is the lack of a method or information source to build a system capable of mastering a real example. As an option, we propose the collection of syntactic patterns from annotated corpora of texts.

Agradecimentos

- Aos professores Tomasz Kowaltowski e Ariadne M. B. R. Carvalho pela orientação.
- Aos professores do Instituto de Computação da UNICAMP, especialmente aos professores Arnaldo Moura, Jacques Wainer e João Meidanis.
- Aos professores Tomasz Kowaltowski, Cláudio Leonardo Lucchessi e Jorge Stolfi pelas idéias que fizeram possível este trabalho.
- Aos colegas do Grupo de Estudo das "quartas": Luiz Arthur, Daniel, Freud e Ralph.
- A meus colegas e amigos do IC.
- A minha esposa Laura, a meus pais, irmãs, avós, a minha família, pelo apoio constante.
- Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução e Motivações

Os avanços da tecnologia nas comunicações, a disponibilidade de grande bancos de dados de informação, a globalização da economia e a melhora nos meios de transportes são algumas das causas do crescimento da demanda por sistemas de *Processamento de Línguas Naturais* (PLN). Aplicações de tradução automática, por exemplo, são requeridas para transpor a distância imposta pela diferenças entre as línguas. Ferramentas de consulta e de sumarização são necessárias para poder aproveitar os enormes bancos de dados disponíveis via redes de computadores. Sistemas de reconhecimento de voz, de recuperação da informação e de geração automática são algumas das tantas aplicações que o mundo atual demanda à Lingüística Computacional. A implementação da maioria dessas ferramentas de alto nível depende de uma análise da estrutura do texto, chamada análise sintática.

Embora a análise sintática seja um tópico antigo na Lingüística Computacional, as pesquisas dos últimos anos na área revelam novas abordagens e aspectos do problema. A tradição imposta pelo uso de técnicas derivadas de Gramáticas Livres de Contexto está sendo substituída por propostas caracterizadas por um maior compromisso com a viabilidade computacional. Neste contexto devem ser entendidos os objetivos do trabalho reportado nesta dissertação.

1.1 Sintaxe

Uma das habilidades básicas que uma pessoa adquire no processo de aprendizado de uma língua é a capacidade de reconhecer quais são as cadeias de palavras bem formadas, também chamadas sentenças. O conhecimento utilizado nessa tarefa envolve diferentes

níveis de informação que agem cooperativamente.

eu vai cantar. (1.1)

idéias verdes sem cor dormem furiosamente. (1.2)

As cadeias 1.1 e 1.2 seriam seguramente rejeitadas como sentenças do português por um falante nativo; porém, o conhecimento que leva a essa conclusão é diferente em ambos casos.

Em 1.1 o problema é um erro de concordância; a palavra *vai* é um verbo na terceira pessoa, enquanto o pronome *eu* pede um verbo na primeira pessoa do singular. Porém o significado pretendido por essa frase pode ser inferido facilmente.

A cadeia de palavras 1.2[1] não faz sentido e pouco pode ser inferido do significado pretendido. Porém, aplicando o raciocínio puramente sintático usado com 1.1 não existiria razão para rejeitar 1.2.

Embora o conhecimento que uma pessoa possui de uma língua não possa ser resumido apenas à capacidade de aceitar ou rejeitar uma cadeia, esse exemplo mostra como uma tarefa que parece simples envolve um esquema complexo. A separação entre o conhecimento semântico e sintático, nem sempre tão óbvio como nos exemplos 1.1 e 1.2, permitiu contornar vários problemas em PLN e motivou uma verdadeira revolução no estudo das línguas.

O estudo sintático tem como objetivo propor uma teoria para formalizar a estrutura geral por trás de uma língua e explorar os fundamentos dessa teoria. A sintaxe modela a componente básica sobre a qual são construídos os sistemas de PLN. A necessidade de uma análise da estrutura observada na implementação das ferramentas desenvolvidas em Lingüística Computacional mantém a vigência da pesquisa nesta área.

A partir dos resultados obtidos na área da análise sintática, principalmente com o movimento gerativista, a Lingüística começou um processo de "matematização". Neste contexto, a mistura dos objetivos da Lingüística e a computação foi inevitável. Assim, os primeiros anos das pesquisas em Lingüística Computacional foram caracterizados por implementações computacionais de teorias lingüísticas. Um exemplo concreto é o grande número de formalismos derivados das Gramáticas Livres de Contexto.

Nos últimos anos, os sistemas em PLN deixaram de ser produtos de laboratório das universidades para converter-se em uma linha de programação rentável. A demanda por produtos de qualidade, onde parâmetros como a eficiência e tempo de resposta são os de maior relevância, começou a evidenciar a falência das teorias sintáticas.

[1] Chomsky usa esta frase em inglês (*colorless green ideas sleep furiously*) em [Cho57] para explicar a diferença entre semântica e sintaxe; desde então tem sido um exemplo muito utilizado nos livros de teoria da sintaxe.

Um exemplo das mudanças registradas na área é o ressurgimento do processamento de *corpos*[2], técnica desconsiderada pelos gerativistas, que se perfila como uma das áreas de maior atividade nos próximos anos.

1.2 Proposta da Dissertação

Este trabalho descreve um ambiente para a análise superficial de línguas naturais. O projeto forma parte de outro mais ambicioso que pretende estender o modelo aqui apresentado para a implementação de análise sintática robusta. A hipótese de trabalho que suporta o modelo proposto é que o núcleo de sentenças usadas em uma língua pode ser capturado com um dispositivo implementado usando autômatos finitos.

A principal motivação deste trabalho é apresentar um modelo eficiente do ponto de vista computacional e que requeira relativamente pouco trabalho manual para ser configurado. Para uma primeira avaliação da viabilidade do modelo foi escolhida a concordância, um fenômeno superficial rico em exemplos e presente em várias línguas.

Um dos empecilhos apresentados por vários formalismos para a análise sintática é a dificuldade observada na hora de implementar um sistema para uma língua real.[3] A alternativa proposta para a configuração do ambiente é a coleta automática da informação a partir de *corpos* de textos marcados.

1.3 Competência e Desempenho

As diferenças entre os objetivos da Lingüística e a Lingüística Computacional podem ser caracterizadas pela dicotomia entre competência e desempenho de uma língua. Chomsky define como *competência* ao conhecimento que permite a uma pessoa aprender uma língua ([Cho57]). Quando Chomsky se refere à competência está falando do conhecimento sintático, aquele que permite rejeitar 1.3 e aceitar 1.2.

sem dormem verde cor furiosamente idéias (1.3)

O movimento gerativista apresentou uma série de teorias com o objetivo de capturar a competência das línguas naturais. Durante muitos anos esse foi o único objetivo do estudo sintático das línguas.

O *desempenho* de uma língua é a parte visível da competência. Em outras palavras, o desempenho é a realização da competência materializada nas sentenças que são escritas e

[2]Por *corpo* entende-se um grande conjunto de textos coletados de várias fontes como jornais, livros e revistas. Em inglês, esta palavra foi tomada do latim *corpus* com o plural *corpora* (também *corpuses*). Nesta dissertação utiliza-se *corpo* e o plural *corpos*.

[3]Em [Mag94], este problema é denominado *The Toy Problem Syndrome*.

faladas.

Jorge acha que Jorge acha que Jorge acha que nunca chove em ... (1.4)

Além do problema do significado (que não será tratado nesta dissertação), a cadeia 1.4 é atípica devido aos níveis de encaixamento. Claramente, essa sentença não forma parte do desempenho do português pois nunca seria enunciada por algum falante nativo. Porém, do ponto de vista da competência, 1.4 é válida pois é possível identificar a função gramatical de cada palavra e a estrutura hierárquica das componentes da sentença.

As teorias da Lingüística sempre relegaram o estudo do desempenho a um papel irrelevante, e a Lingüística Computacional dos primeiros tempos aderiu a este princípio [Win83, Gri86]. O compromisso dos sistemas de processamento de línguas naturais com a competência começou a ser quebrado com a demanda por sistemas eficientes.

1.4 Análise Sintática Superficial

O termo *análise superficial*[4] é utilizado de forma geral para caracterizar aqueles analisadores que têm uma saída menos completa do que um analisador sintático convencional. Em geral, um analisador superficial identifica alguns dos constituintes de um sintagma, ou apenas a categoria gramatical das palavras de uma frase sem especificar a estrutura da sentença.

O propósito básico de um analisador superficial é inferir um tipo de informação específico, como a relação entre as palavras estabelecida pelas regras de concordância. Os analisadores superficiais são utilizados, por exemplo, na marcação automática de texto; este tipo de programas chamados *marcadores morfo-sintáticos*[5] atribui a cada palavra de um *corpo* uma marca que representa a categoria gramatical da mesma.

A maioria dos sistemas de PLN necessita dar um tratamento à estrutura subjacente do texto, mas nem sempre precisa ser uma análise sintática completa que forneça, por exemplo, uma árvore de derivação. Esse tipo de análise mais completa é custoso em tempo e espaço, e os analisadores superficiais podem ser uma alternativa melhor para esses casos.

Em [KK95] é apresentado um resumo das principais características dos analisadores sintáticos superficiais com exemplos e referências a outros trabalhos.

1.5 Escolha do Inglês

O ambiente apresentado aqui pode ser adaptado a varias línguas naturais. Como é mostrado ao longo deste trabalho, o modelo captura as principais características superficiais

[4] Em inglês *surface parsing*, ou também *shallow parsing*.

[5] Em inglês *part-of-speech tagger*.

de várias línguas de origem européia de forma uniforme. Para um primeiro experimento foi escolhido o inglês por duas razões: simplicidade da língua e disponibilidade de recursos.

O fenômeno de concordância pode ser definido como um conjunto de regras gramaticais que rege as flexões que as palavras podem tomar em uma sentença. As línguas derivadas do latim (português, espanhol, catalão, francês, italiano e romeno, entre outras) apresentam uma grande variedade de formas verbais. Em geral cada combinação de pessoa e número requer, nestas línguas, uma flexão diferente para cada tempo verbal. O alemão, de origem germânica, apresenta uma variação menor para as formas verbais mas por outro lado incorpora declinações para artigos, adjetivos e substantivos.

O inglês é uma língua com uma variedade menor de flexões. Essa simplicidade facilitou a tarefa de preparação de dados para a coleta da informação necessária. Isto não invalida a aplicação do modelo proposto a outras línguas, pois como será mostrado ao longo da dissertação, a relação entre o tamanho do conjunto de dados tratados e a complexidade dos algoritmos propostos é mínima.

A extração de informação a partir de *corpos* marcados foi a alternativa escolhida para formar o banco de dados necessário para uma implementação que opere sobre uma língua natural. Numerosos projetos foram iniciados por entidades de diferentes países para coletar dados e formar *corpos* com o propósito de disponibilizá-los para a pesquisa. Ainda assim, a quantidade de *corpos* em inglês supera em grande número aos disponíveis em outras línguas. Este foi o fator fundamental que decidiu a escolha pelo inglês.

1.6 Organização da Dissertação

Cada capítulo da dissertação inclui no final um pequeno resumo (exceto no capítulo de conclusões) do material tratado no mesmo. A dissertação está organizada da seguinte forma:

Capítulo 2 Análise de Concordância. A definição de concordância usada neste trabalho difere da tradicional. Para evitar qualquer confusão, neste capítulo introduz-se uma definição informal de concordância ilustrada a partir de exemplos. Na segunda parte, apresenta-se uma visão pessoal da evolução das abordagens à análise sintática em Lingüística Computacional, revisando o tratamento dado à concordância.

Capítulo 3 Autômatos Finitos. O uso de autômatos finitos na Lingüística Computacional foi resistido por vários anos. Essa tendência reverteu-se, e na atualidade técnicas baseadas em autômatos são utilizadas em muitos sistemas de PLN. Este capítulo introduz formalmente o formalismo e discute os argumentos levantados contra o uso de autômatos finitos para o processamento de línguas. Por último apresentam-se as abordagens baseadas em máquinas de estados finitos.

Capítulo 4 Padrões Sintáticos. Neste capítulo descreve-se o ambiente proposto, junto com a implementação de um algoritmo de reconhecimento.

Capítulo 5 Conselheiro de Concordância. A flexibilidade do modelo de padrões é mostrada a partir da implementação de um conselheiro de concordância. Este capítulo descreve com detalhes e exemplos como são implementadas as operações do conselheiro sobre a estrutura de dados baseada em autômatos.

Capítulo 6 Coleta de Padrões Sintáticos. A viabilidade do modelo proposto está suportada também pela possibilidade da coleta de padrões sintáticos de *corpos* marcados. Na primeira parte deste capítulo apresenta-se um estudo resumido de *corpos* de texto em geral, e dos utilizados nos experimentos em particular. Na segunda parte, define-se o conjunto de marcas adequado para a coleta de padrões sintáticos. Por último, descreve-se a implementação de uma ferramenta para a coleta de padrões sintáticos.

Capítulo 7 Implementação. Os algoritmos apresentados foram implementados sobre as rotinas fornecidas por uma biblioteca projetada para manipular autômatos. Neste capítulo discutem-se as principais características da biblioteca, e as camadas desenvolvidas para implementar de forma eficiente o banco de dados sintáticos requerido pelo ambiente.

Capítulo 8 Conclusões e Extensões. Neste capítulo discutem-se as contribuições do trabalho de dissertação. Também propõe-se um conjunto de extensões ao ambiente.

Observações

- O termo *Lingüística Computacional* é utilizado para fazer referência à área de pesquisa que trata do processamento automático de línguas usando o computador. A Lingüística Computacional está caracterizada pela interdisciplinaridade, e não existe um critério consensuado que permita catalogá-la como uma área dentro do escopo da Computação ou da Lingüística.

- O *Processamento de Línguas Naturais* (PLN) envolve o conjunto de sistemas que manipulam de forma automática algum aspecto da língua. Em alguns trabalhos PLN é utilizado para substituir ao termo mais amplo Lingüística Computacional.

- O uso do termo *língua natural* deriva da tradução de *natural language* do inglês. Em português este termo pode ser considerado redundante. Ambos, língua e língua natural, são utilizados de forma intercambiável ao longo deste texto.

Capítulo 2

Análise de Concordância

O PLN evolui nos últimos anos para a implementação de sistemas mais específicos, focalizando o interesse em soluções eficientes para tratar fenômenos restritos a um domínio determinado. A tendência atual mostra que a implementação de um sistema abrangente deve considerar a integração de um número de técnicas originadas a partir de diversas teorias [Wil95].

Nesta dissertação propõe-se um ambiente para a análise superficial de línguas. Para mostrar o poder de representação do modelo e exemplificar algumas características, foi escolhido um fenômeno comum a diversas línguas: a concordância. Na primeira parte deste capítulo apresenta-se uma definição de concordância a partir de exemplos encontrados em diferentes línguas. Na segunda parte discute-se brevemente as abordagens tradicionais à análise sintática e a relevância da concordância às mesmas.

2.1 Concordância

A *concordância* numa língua natural rege a partir de um conjunto de regras:

- as formas flexionadas que podem aparecer simultaneamente em uma sentença; e
- as posições que podem ocupar as palavras (ordem).

Dada uma cadeia de palavras, o objetivo da *análise de concordância* é verificar que nenhuma destas regras seja violada. Se for detectada a presença de algum erro, um segundo objetivo é determinar o local onde foi gerado e corrigi-lo, se possível.

Esta definição de concordância não coincide totalmente com a definição utilizada em textos de gramática tradicionais. Do ponto de vista da gramática, a concordância pode incorporar regras estabelecidas sobre critérios sintáticos e semânticos.

O objetivo desta dissertação é prover ferramentas para capturar as relações superficiais entre as diferentes flexões das palavras no sentido puramente sintático. A seguir, apresentam-se exemplos de regras de concordância encontradas em diferentes línguas que podem ser representadas no modelo de padrões sintáticos proposto no capítulo 4. Estes exemplos são úteis para entender as decisões tomadas no projeto do conjunto de marcas e do conselheiro gramatical.

2.1.1 Concordância Verbal

A concordância verbal em línguas como o português, inglês e espanhol estabelece que o verbo deve concordar com o sujeito em número e pessoa. Neste caso, concordar significa que o verbo (termo subordinado) ajusta suas flexões de número e pessoa ao número e pessoa do sujeito (termo subordinante).

tres tigres salvajes comen trigo. (2.1)

o menino que bateu nela pulou o muro. (2.2)

the children are playing in the park. (2.3)

As sentenças 2.1, 2.2 e 2.3 apresentam exemplos em espanhol, português e inglês, respectivamente, da regra de concordância verbal enunciada. Nestas sentenças, a aplicação da regra de concordância é trivial para qualquer falante nativo.

a maioria das pessoas está cansada. (2.4)

a maioria das pessoas estão cansadas. (2.5)

** the majority of the miners was still on strike.* (2.6)

Segundo [Lim96], tanto 2.4 como 2.5 podem ser aceitas como sentenças da língua portuguesa; mas uma construção similar em inglês, por exemplo, rege-se com regras diferentes. Substantivos tais como *number*, *majority* e *plenty* concordam com verbos no plural [Lee89]; assim 2.6 não é aceita como uma sentença em inglês. Estes exemplos mostram como os critérios para definir as regras de concordância podem mudar de uma língua a outra. Seguindo a notação tradicional, uma cadeia de palavras que não pertence à língua natural modelada é distinguida com um símbolo * na frente.

2.1.2 Artigos e outros Determinadores

Outra regra de concordância importante é a que estabelece a relação entre o substantivo e os pré-determinadores em um sintagma nominal. Em línguas derivadas do latim, por

exemplo, os artigos concordam em gênero e número com o substantivo.

La leche estaba caliente. (2.7)

O leite estava quente. (2.8)

O gênero das palavras é determinado pela etimologia e outras questões culturais. Muitos substantivos mudam seu gênero quando são referenciados em diferentes línguas. O português e o espanhol, duas línguas próximas, têm vários exemplos de objetos denotados por palavras similares na escrita onde o gênero é oposto, como mostram 2.7 e 2.8.[1] Além de qualquer consideração etimológica ou pragmática, o fato é que o gênero dos substantivos é um atributo fixo representável como uma característica sintática.[2]

Outra restrição importante da concordância artigo-substantivo está relacionada com a natureza da matéria referenciada. Segundo esse critério, os substantivos são classificados em: contáveis e incontáveis. Novamente, questões culturais determinam este atributo, mas existe um consenso bem estabelecido que permite representar esse tipo de informação no léxico. Assim, certos determinadores estão reservados para serem usados com substantivos contáveis ou incontáveis exclusivamente. Em inglês, os substantivos incontáveis, por exemplo *milk*, não podem ser modificados pelos artigos indeterminados *a* ou *an*.

Em geral, os artigos são menos usados em inglês que em línguas de origem latina, mas em contraposição numerosas regras regem o uso de outro tipo de determinadores. Em inglês, os determinadores dividem-se em três grupos segundo a posição que podem ocupar em uma sentença: [QG80]

- pré-determinadores: artigos e modificadores como *the*, *a*, *all*, *both* e *half*;
- pós-determinadores: seguem os determinadores mas precedem os adjetivos, por exemplo numerais, cardinais e ordinais tais como *first* e *two*.
- quantificadores: agrupam determinadores tais como *many* e *several*.

2.1.3 Adjetivos

A relação entre o gênero e número do substantivo e a flexão do adjetivo é chamada concordância nominal. A regra geral estabelece que o adjetivo deve concordar em gênero

[1] *la* é o artigo feminino em espanhol.

[2] Deve-se notar entretanto que mesmo o gênero sendo um atributo fixo, a relação entre um substantivo e os pré-determinadores pode estar sujeita a outras regras que modelem problemas fonológicos (vide seção 2.1.5).

e número com o substantivo.

Essas meninas são bonitas. (2.9)

Uma grande mulher libertou esse povo. (2.10)

Uma mulher grande caiu na calçada. (2.11)

As sentenças 2.9, 2.10 e 2.11 são exemplos da concordância nominal em português. Note-se que não é relevante à definição de concordância a diferença semântica no uso do adjetivo *grande* em 2.10 e 2.11.

Os adjetivos em inglês não têm flexões para gênero e número como em línguas latinas, mas incorporam outros conceitos interessantes relevantes à concordância. Segundo a posição que podem ocupar, os adjetivos em inglês classificam-se como: [QG80]

- atributivos: entre os pré-modificadores e o(s) substantivo(s);
- pós-positivos: logo depois do substantivo;
- predicativos: no sintagma verbal modificando o substantivo no sujeito.

A maioria dos adjetivos em inglês são atributivos e predicativos. Entretanto, existem alguns adjetivos que podem ocupar apenas uma das duas posições, como por exemplo o adjetivo *alive* que é apenas predicativo. Os adjetivos devem ser pós-positivos quando modificam um pronome terminado em *-body*, *-one*, *-thing* ou *-where* como mostra 2.12.

she wants to try on something larger. (2.12)

Outra característica interessante do inglês são as flexões que apresentam alguns adjetivos, em geral monossilábicos, para formas comparativas e superlativas, como por exemplo: *small*, *smaller* e *smallest*.

2.1.4 Caso e Pronomes

A Teoria de Caso captura as diferentes funções que um sintagma nominal pode assumir em uma sentença [vRW86]. Os verbos e preposições atribuem um caso a cada sintagma nominal relacionado. Um verbo, por exemplo, pode atribuir caso nominativo ao sujeito e dativo ou acusativo aos objetos. O interesse desta seção não está na Teoria de Caso propriamente dita, mas sim nas regras de concordância que se definem quando um caso se realiza morfologicamente.[3]

O alemão é um exemplo de uma língua que apresenta declinações para vários casos de um sintagma nominal. Assim, o artigo, adjetivo e às vezes o substantivo têm uma flexão

[3] A realização de um caso é refletida na declinação de uma palavra a partir de uma outra original.

especial para cada combinação de caso, número e gênero. Embora o latim seja uma língua que apresenta uma grande variedade de declinações para a realização de caso, inclusive nos substantivos, as línguas derivadas incorporam-nas apenas para umas poucas categorias, como por exemplo os pronomes. A preposição *para* em português, por exemplo, atribui caso ao pronome *eu* flexionando-o para *mim* em 2.13.

esse presente é para mim? (2.13)

Uma característica interessante em inglês é o caso genitivo que se realiza com o apóstrofo e a transposição das palavras como em *Maria's house.*

Os pronomes reflexivos em inglês introduzem outra regra de concordância. As sentenças 2.14, 2.15 e 2.16 mostram que, enquanto há liberdade no uso dos pronomes *her* e *him*, o pronome reflexivo deve concordar com o gênero do sujeito.

she would look after herself. (2.14)

she would look after him. (2.15)

* *she would look after himself.* (2.16)

2.1.5 Fenômenos Fonológicos

As línguas são entidades em contínua evolução. O uso e abuso dos falantes incorpora novos modismos, estilos e palavras. Muitas dessas mudanças surgem como alternativas ou simplificações que facilitam a expressão oral, como é o caso das contrações comuns em várias línguas.

Um exemplo de como a fala pode influenciar a escrita é o tratamento dos artigos determinados *la* e *le* em francês. Um artigo na frente de uma palavra que começa com vogal é substituído por *l'* perdendo-se a marca do gênero [Wei89], como por exemplo *l'art* e *l'idée.*

O espanhol tem um fenômeno similar com o artigo determinado feminino *la* usado antes de um substantivo feminino começando com a letra *a* tônica. Para evitar o encontro vocálico usa-se o artigo masculino *el* [Esp73].[4] A mesma regra não é válida com adjetivos como mostra 2.17. Nessa sentença o artigo correspondente ao substantivo feminino *agua* é *el* mas no caso do adjetivo *árida* mantém-se o artigo *la.* O espanhol apresenta uma solução similar para o encontro de vogais com os conectores *y* e *o* ("e" e "ou").

el agua es escasa en la árida Patagonia. (2.17)

[4] A *Real Academia Española* define como artigos determinados femininos tanto a *la* como a *el* por uma questão de formalidade.

Em inglês surge um fenômeno similar no uso do artigo indeterminado *a*. A regra estabelece que na frente de uma palavra que começa com o som de vogal deve-se usar *an* [Lee89], como em *an actor*. Embora a regra seja simples, não é possível determinar a partir da morfologia de uma palavra a forma adequada do artigo, pois não há uma relação direta entre as letras e sua pronúncia. A palavra *university*, por exemplo, começa com uma vogal pronunciada como uma consoante e o artigo deve ser *a*. As palavras começando com *h* também podem gerar problemas, como é o caso de *an hour* e *a horse*.

2.2 Abordagens para a Análise Sintática

Os trabalhos revolucionários de Chomsky em Lingüística tiveram diferentes repercussões em outras ciências como a Computação. A formalização da sintaxe deslocou a metodologia tradicional do estudo da Lingüística a áreas afins à Matemática Discreta, Lógica e Computação.

A organização desta seção reflete uma visão pessoal da evolução das propostas em Lingüística Computacional para a análise sintática. Historicamente, os primeiros sistemas de PLN abordaram o problema da sintaxe subestimando-o e reduzindo-o a apenas um aspecto supostamente já resolvido. Esses primeiros sistemas desenvolveram-se sobre variações *ad-hoc* de produções livres de contexto. Nos últimos anos, o *software* para processamento de línguas deixou de ser um produto de laboratório universitário para começar a ser usado em aplicações caracterizadas pela interação com o usuário. Essa mudança introduz uma demanda pela eficiência que muitas vezes supera o compromisso que o sistema deve ter com a abrangência.

Nesta evolução diferenciam-se duas gerações de sistemas. A primeira geração, chamada de modelos de alto nível [Col95], consiste de extensões às gramáticas livres de contexto. A segunda geração propõe modelos, chamados de baixo nível, com menor ligação às teorias da Lingüística pondo ênfase na viabilidade computacional.

2.2.1 Gramáticas Livres de Contexto

Na teoria das linguagens formais uma *gramática* é uma especificação formal e finita de uma linguagem. Uma gramática pode tomar várias formas; por exemplo, no caso de uma linguagem finita pode consistir simplesmente de uma lista com todas as cadeias.

Uma *Gramática de Estrutura de Frase*[5] é definida como uma quádrupla

$$(T,N,P,S)$$

onde

[5] Em inglês *Phrase-Structure Grammar*.

- T é o conjunto de símbolos terminais, que determina o alfabeto sobre o qual são definidas as cadeias da linguagem.
- N é o conjunto de não terminais onde $T \cap N = \emptyset$.
- P é o conjunto de produções da forma $a \rightarrow b$, onde $a \in (T \cup N)^+$ e $b \in (T \cup N)^*$
- $S \in N$ é o símbolo inicial.

Cada produção deve ser interpretada como uma regra de reescrita que substitui o lado esquerdo de uma produção pelo lado direito. A linguagem definida por uma gramática é o conjunto de cadeias de terminais que podem ser derivadas iniciando uma seqüência de operações de reescrita com o símbolo inicial S. As *Gramáticas Livres de Contexto* (GLC) são uma restrição às Gramáticas de Estrutura de Frase onde cada produção $a \rightarrow b$ deve satisfazer:

- $a \in N$
- $b \in (T \cup N)^*$

Exemplo 1 As produções a seguir representam uma GLC que gera todas as cadeias de parênteses balanceados.

$$\mathsf{S} \rightarrow (\ \mathsf{S}\ )$$
$$\mathsf{S} \rightarrow \mathsf{S}\ \mathsf{S}$$
$$\mathsf{S} \rightarrow \epsilon$$

O símbolo S é o inicial, e ϵ representa a cadeia vazia. Uma possível derivação para a cadeia ( ) (( )) é:

$$\mathsf{S} \rightarrow \mathsf{S}\ \mathsf{S} \rightarrow (\ \mathsf{S}\ )\ \mathsf{S} \rightarrow (\ )\ \mathsf{S} \rightarrow (\ )\ (\ \mathsf{S}\ ) \rightarrow (\ )\ (\ (\ \mathsf{S}\ )\ ) \rightarrow (\ )\ (\ (\ )\ )$$

Com cada derivação pode ser associada uma estrutura de árvore que representa graficamente a aplicação hierárquica das operações de reescrita, como mostra a figura 2.1. □

Muitos dos conceitos e termos usados na área de análise sintática derivam do jargão próprio das GLC tais como: *árvore de derivação*, *análise descendente*, *análise ascendente*, *ambigüidade*, etc.

Na teoria de linguagens formais foram obtidos resultados aceitáveis para a análise de linguagens especificadas com GLC. Um exemplo é o algoritmo de reconhecimento dinâmico de Cocke, Kasami e Younger (CKY) [You67] de complexidade cúbica no comprimento da

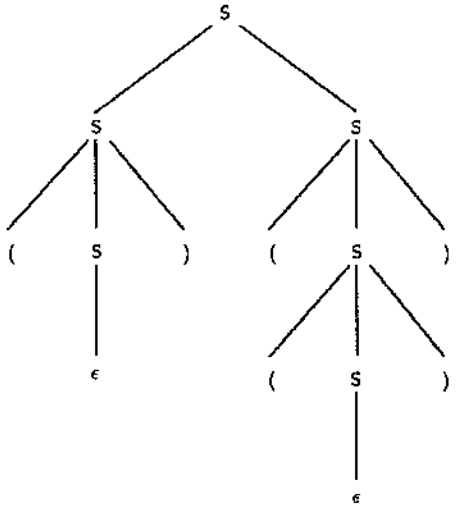


Figura 2.1: Árvore de derivação para ( ) (( )).

cadeia analisada. Na análise de complexidade associado a uma GLC existe um fator multiplicativo que depende diretamente do tamanho da gramática que se expressa em termos da quantidade de regras de produção e não terminais. Em problemas envolvendo linguagens formais esses valores são relativamente pequenos e a maioria dos algoritmos de reconhecimento executa em tempos menores que o pior caso. Um dos principais problemas da aplicação destas abordagens às línguas naturais é o grande tamanho das gramáticas, o qual degrada a eficiência dos algoritmos tradicionais [Lap96, Col95].

As GLC devem seu nome justamente à propriedade que permite construir à arvore de derivação. Uma vez que a derivação tenha escolhido um nó para expandir, a subárvore com os nós descendentes é gerada independentemente de outras subárvores, isto é, não é levada em conta a informação sobre o contexto.

Essa característica, segundo Chomsky em [Cho57] é um empecilho para representar certas relações superficiais como a concordância. Em [GM89] são frisado os problemas das GLC para representar certos fenômenos como o apresentado em um dialeto do alemão onde aparece uma estrutura análoga a $\mathsf{NP}^m\ \mathsf{NP}^n\ \mathsf{V}^m\ \mathsf{V}^n$.

João, Maria e Antônia são advogado, médica e enfermeira respectivamente. (2.18)

Em português existem também algumas sentenças que incorporam constituintes descontínuos, como em 2.18 onde a concordância deve se verificar entre pares de palavras que estão entrelaçadas no texto.

Na busca de soluções para este problema surgiram nos últimos anos formalismos considerados entre os modelos de alto nível denominadas *Gramáticas Suavemente Sensíveis*

ao Contexto.[6] Exemplos destes modelos são: *Tree-Adjoining Grammars, Head Grammars* e *Linear Indexed Grammars*[Ram89]. As linguagens que podem ser reconhecidas por estes sistemas são equivalentes entre si e o poder de representação as coloca entre as linguagens livres do contexto e as sensíveis ao contexto.

Para uma introdução à teoria das GLC ver [HU79, Har78]. Em [Win83, Gri86] é discutido o uso e alcance das GLC em Lingüística Computacional.

Gramáticas Transformacionais

As limitações observadas nas GLC para representar línguas foi uma motivação para a introdução das *Gramáticas Transformacionais* (GT) em [Cho57]. As GT são o passo inicial de um movimento revolucionário na Lingüística conhecido como o "gerativista".

Provavelmente nenhum sistema implementou o modelo das GT, mas a influência sobre as diferentes propostas nos primeiros anos da Lingüística Computacional é inegável. O modelo original, também conhecido como *Teoria Padrão*, define resumidamente duas componentes:

1. componente base: uma gramática livre de contexto que gera a denominada estrutura profunda;

2. componente transformacional: regras de reescrita que a partir da estrutura profunda geram a denominada estrutura superficial.

Exemplo 2 O objetivo das GT é modelar as sentenças aplicando transformações a um conjunto pequeno de estruturas ou a outras sentenças relacionadas. A sentença 2.19 pode definir a estrutura profunda para gerar, a partir de certas transformações, as sentenças 2.20, 2.21 e 2.22 no nível superficial.

O menino pulou o muro. (2.19)

O muro foi pulado pelo menino. (2.20)

O menino pulou o muro? (2.21)

O menino não pulou o muro. (2.22)

A figura 2.2 ilustra como age cada um dos componentes do modelo para derivar a partir da sentença afirmativa 2.19 a correspondente passiva 2.20. □

Esse exemplo apenas serve para mostrar como interage cada componente para derivar uma sentença. Uma implementação modelando línguas naturais deveria incorporar transformações mais complexas, e em geral uma estrutura profunda não corresponderá a uma

[6] Em inglês *Mildly Context-Sensitive Grammars.*

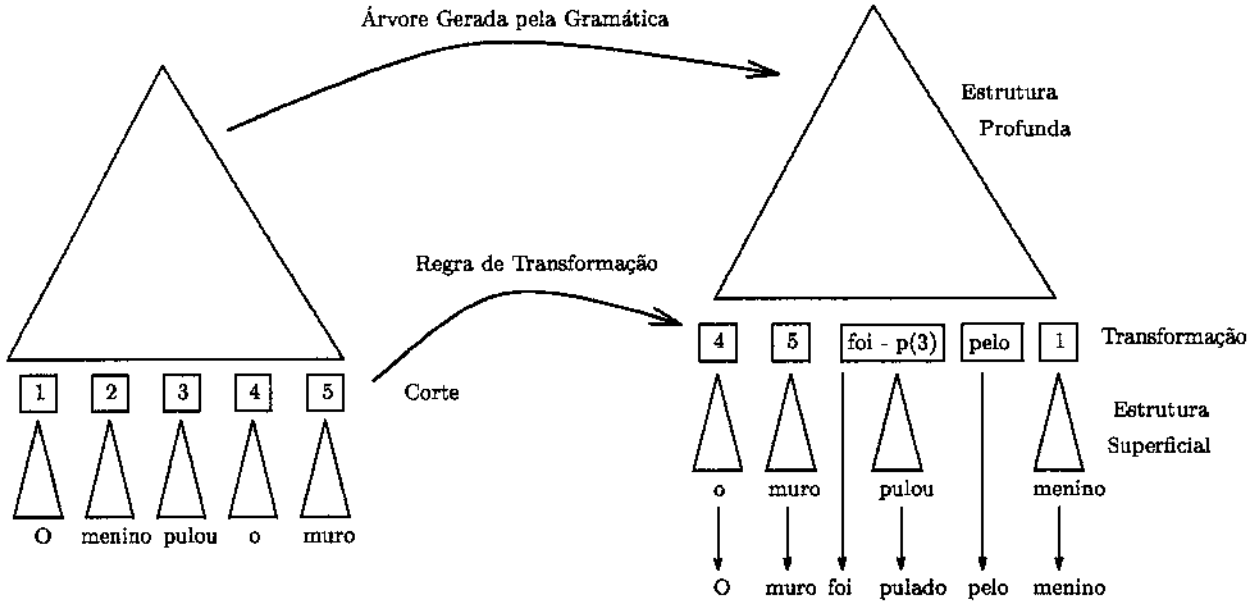


Figura 2.2: Gramáticas Transformacionais

sentença da língua. Toda a evolução do movimento gerativista foi acompanhada pelos pesquisadores da Lingüística Computacional até a década do 80. O ponto mais interessante da abordagem gerativista é a abstração que suporta o "conhecimento" sintático da língua em módulos separados que atuam de forma cooperativa [vRW86, Sel85].

Do ponto de vista da implementação de sistemas de processamento de língua natural as GT são inerentemente ineficientes; umas das razões é a falta de determinismo na aplicação das possíveis transformações. Por outro lado, como o nome do movimento indica, a própria teoria tem uma orientação à geração sem reparar na operação contrária: o reconhecimento.

Redes de Transição Ampliadas

As *Redes de Transição Ampliadas* (ATN[7]) descritas por Woods [Woo70, Woo86] estiveram entre as primeiras abordagens implementadas e talvez seja o modelo mais usado para representar GLC.

As regras de uma GLC são traduzidas para um conjunto de redes de nós ligados por arestas. Com cada produção é associada uma rede onde o nó inicial está rotulado com o símbolo não terminal do lado esquerdo, e as arestas que vinculam os nós restantes com

[7]Em inglês *Augmented Transition Networks.*

os símbolos do lado direito.

Exemplo 3 A figura 2.3 mostra uma ATN para a seguinte gramática

S → SN SV
SN → artigo substantivo
SV → verbo-transitivo SN

Nesta gramática os símbolos terminais são representados com letras minúsculas e indicam categorias de palavras. O não terminal S é o símbolo inicial. □

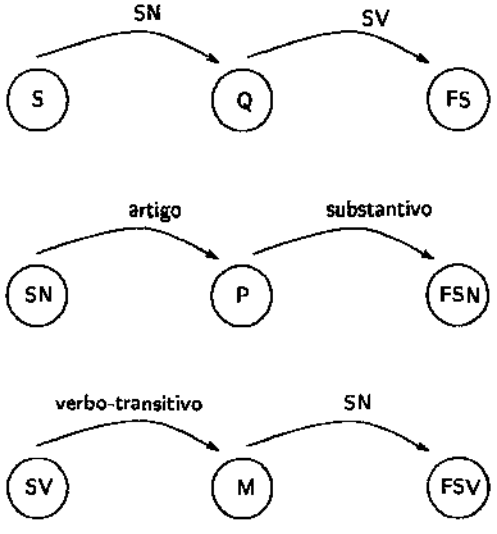


Figura 2.3: Rede de Transição Ampliada

O reconhecimento consiste no percurso da rede cujo nó inicial está rotulado com o símbolo inicial da gramática, terminando em um estado final. A transição de um nó n_i a um nó n_j é definido segundo o rótulo da aresta que os une:

- se o rótulo é o nome de um outro nó n_h, deve-se percorrer a rede cujo nó inicial é n_h;
- se o rótulo é uma categoria, verifica-se se a próxima palavra na entrada pertence a essa categoria, se for, incrementa-se o ponteiro na entrada.

Este processo é bem conhecido em análise sintática de linguagens de programação como "análise descendente recursiva com retrocesso" [AU79]. Para modelar a concordância

com este esquema precisa-se de um grande número de redes, o que é proibitivo tanto pelo espaço requerido como também pela eficiência do algoritmo de análise. Uma solução para este problema foi incorporar atributos[8] que transferem a outros níveis da estrutura hierárquica os dados relevantes processados em níveis inferiores.

Para implementar conselheiros gramaticais associa-se com cada nó rotinas que detectam a presença de erros comuns. Com esta representação os conselhos são precisos, mas só atingem um número fixo de erros e são difíceis de programar.

Programação Lógica

A evolução do paradigma da programação lógica está muito ligada à Lingüística Computacional. Como já foi dito, nos primeiros sistemas o problema da análise sintática era trivializado, e o interesse estava focalizado principalmente na representação do conhecimento. Esse interesse pelas questões semânticas abriu o caminho à programação lógica.

Um programa em uma linguagem de programação lógica, como por exemplo PROLOG, consiste de um conjunto de regras, chamadas cláusulas de Horn. Cada cláusula representa uma disjunção de predicados lógicos que pode ser interpretada como um procedimento. Todas as variáveis são locais ao procedimento onde foram definidas e a passagem de parâmetros é baseada na unificação de literais. O tipo de dado básico é a lista que consiste de uma seqüência ordenada de elementos. Representar uma GLC em PROLOG é direto como mostra o exemplo.

Exemplo 4 A seguinte GLC captura a formação de sentenças simples

$S \rightarrow NP\ VP$
$VP \rightarrow V\ NP$
$VP \rightarrow V$

Com cada não terminal é associado um predicado, e com cada produção uma cláusula de Horn, como é mostrado no seguinte programa PROLOG

```
s(Z) :- np(X),
        vp(Y),
        append(X,Y,Z).

vp(Z) :- v(X),
         np(Y),
         append(X,Y,Z).
```

[8] Mecanismo utilizado na análise semântica das linguagens de programação.

```
vp(Z) :- v(Z).
```

Uma cadeia de palavras é representada por uma lista. A cabeça de cada cláusula corresponde ao lado esquerdo de uma produção e o corpo ao lado direito. O predicado `append(X,Y,Z)` é satisfeito se a lista `Z` é o resultado de concatenar as listas `X` e `Y`. Os predicados `np(X)` e `vp(X)` devem ser implementados por um léxico. □

Embora a implementação de GLC seja simples em PROLOG, a maioria dos interpretadores dessa linguagem incorpora uma notação denominada *Gramática de Cláusulas Definidas* (DCG[9]) que facilita ainda mais a tarefa. Cada produção de uma GLC pode ser escrita diretamente usando uma regra de DCG.

A representação da informação referente à concordância poderia ser implementada utilizando não terminais específicos. A clara desvantagem, já discutida nesta seção, é o grande número de não terminais necessários.

O formalismo das DCG oferece uma outra alternativa para manipular concordância e outros fenômenos superficiais. A solução consiste em associar atributos a cada não terminal que podem ser manipulados seguindo as limitações definidas pela regra de unificação.

Exemplo 5 A seguir, é apresentada uma DCG para a GLC do exemplo 4 estendida para modelar a concordância verbal.

```
s --> np(P), vp(P).

vp(P) --> v(P), np(_).

vp(P) --> v(P).
```

O atributo `P` (variável local a cada procedimento) é utilizado para definir a informação de pessoa. Por exemplo, se o sintagma nominal resultar no pronome da primeira pessoa o sintagma verbal recebe essa informação pelo atributo e ajusta a flexão do verbo. O símbolo "_" no sintagma sintagma nominal objeto indica que nesse caso não existem restrições enquanto a flexões de pessoa o número. □

A principal vantagem das DCG é a elegância para resolver fenômenos lingüísticos, tanto profundos como superficiais. Além disso, as ferramentas fornecidas pelo paradigma de programação permitem avançar facilmente sobre a análise semântica. A principal desvantagem está relacionada aos problemas inerentes ao modelo computacional da programação lógica.

[9] Em inglês *Definite Clause Grammar*.

Os paradigmas de programação declarativos, como o lógico, estão baseados na idéia da separação entre o controle e a lógica do algoritmo. O controle deve ser implementado em um nível mais baixo de uma forma uniforme dando liberdade ao programador para especificar apenas a lógica do problema. No caso da programação lógica, o modelo computacional consiste em um motor de inferência que procura achar a interpretação que satisfaz a consulta segundo a informação no programa, fazendo uma busca recursiva no espaço definido por todas as interpretações.

Em [PW86] é descrito o processo de tradução das DCG às cláusulas de Horn. Pereira propõe em [Per81] as *Gramáticas de Extraposição* (XG), uma extensão às DCG para manipular transformações ao estilo das GT. Em [GM89] são introduzidas as noções básicas da Lingüística Computacional do ponto de vista da programação lógica.

2.2.2 Modelos Lexicais

Em [Lap96], Laporte argumenta que a falta de importância dada às palavras nas produções das GLC é artificial, e com alguns dados experimentais mostra que para uma implementação real é preciso dar relevância à informação no léxico.

Alguns anos antes da revolução gerativista, um grupo de lingüistas e matemáticos apresentam os primeiros modelos sem estrutura gramatical, onde toda a informação da língua é codificada no léxico. Um modelo *lexical* prescinde de gramáticas, transformações ou regras, pois toda a informação necessária é representada no léxico.

Gramáticas de Categorias

As *Gramáticas de Categorias*(CG[10]) agrupam um número de formalismos relacionados que têm sido propostos principalmente para a análise sintática de linguagens lógicas e matemáticas, mas com o objetivo final de tratar línguas naturais. Estas abordagens estão baseadas no conceito de conexidade sintática definida por Ajdukiewicz [Ajd35].

As CG têm de alguma maneira influenciado os fundamentos de muitas teorias sintáticas modernas como *Tree Adjunction Grammars*, *Lexical Functional Grammars* e *Generalised Phrase Structure Grammar* [Sel85]. Nestes formalismos a análise sintática é modelada como um processo de derivação formal, onde uma cadeia é reconhecida como sentença se é possível prová-la como tal.

Em uma CG todos os itens lexicais estão associados a um tipo ou categoria. Um operador é definido sobre esses tipos permitindo a composição de constituintes (em princípio palavras) em outros constituintes maiores. Uma categoria pode ser básica, como NP, ou tipo função, como NP/S. A aplicação da operação de composição segue as regras:

[10] Também Gamáticas Categoriais, em inglês *Categorial Grammar*.

1. $X/Y\ Y \rightarrow X$
2. $Y\ Y\backslash X \rightarrow X$

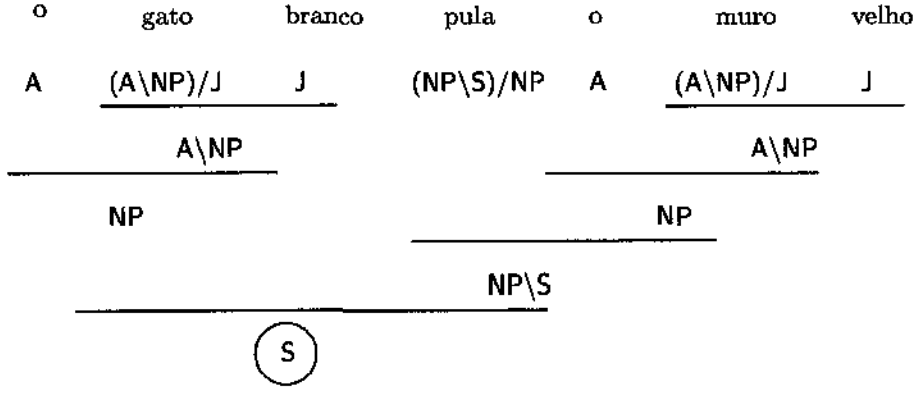
A regra 1 significa que um constituinte com a categoria X/Y pode ser composto com um constituinte imediatamente à direita com categoria Y. O resultado dessa aplicação é X. A regra 2 enuncia uma composição análoga mas na direção contrária.

O princípio de conexidade captura a idéia de que todas as palavras em um sintagma ou sentença ocupam um local que depende da sua relação com no mínimo uma outra palavra. Uma seqüência de palavras é considerada como grammatical em CG se existe um conjunto de aplicações da operação de composição que resulta em uma categoria especial, em geral denotada por S.

Exemplo 6 A tabela abaixo é um exemplo de uma CG, onde cada palavra é definida junto com um tipo.

o:	A
branco velho:	J
gato muro:	(A\ NP)/J
late pula:	(NP\S)/ NP

A seguir é mostrada uma derivação para *o gato branco pula o muro velho*



□

Em [Lam58], Lambek determina as principais motivações por trás deste formalismo. Em [Ste93] pode-se encontrar uma extensa introdução a CG com ênfase nas vantagens do ponto de vista do estudo teórico da língua.

O tratamento de fenômenos de concordância é resolvido com elegância utilizando o conceito de categoria. O formalismo não oferece uma abordagem algorítmica, porém motivou outros formalismo da Lingüística Computacional como as Gramáticas de Vínculos.

Gramáticas de Vínculos

O formalismo das *Gramáticas de Vínculos*[11], introduzido em [ST91] por Sleator e Temperley, fornece uma abordagem lexical para a análise sintática de línguas naturais com um enfoque algorítmico.

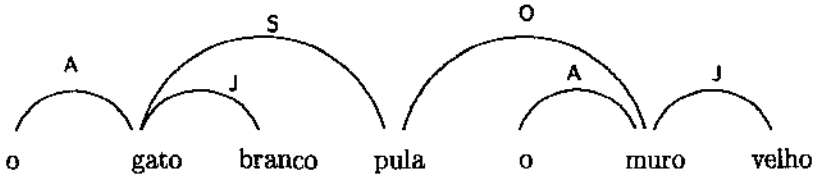
A definição das Gramáticas de Vínculos foi motivada por uma propriedade conhecida na Lingüística como *planaridade*. Informalmente, a planaridade estabelece que em uma sentença sempre é possível determinar relações entre as palavras de modo que podem ser traçadas linhas entre essas palavras sem que elas se cruzem. Esta propriedade está diretamente relacionada ao conceito de conexidade da CG.

Em uma Gramática de Vínculos, cada palavra no léxico está associada a vários conectores que definem as condições para estabelecer os grupos de palavras que podem ser relacionadas. Assim, cada palavra com seus conectores forma o que pode ser comparável a uma peça de *Lego*, onde as palavras são combinadas em blocos maiores até formar a sentença.

Exemplo 7 O dicionário representado pela tabela abaixo é um exemplo de Gramática de Vínculos. Cada conector é representado com uma letra e um sufixo + ou -. Duas palavras podem estabelecer um vínculo quando a palavra à esquerda possuir um conector "positivo" e a palavra à direita possuir o mesmo conector "negativo".

o aquele:	A+
branco velho:	J-
gato muro:	A- J+ S+ O-
late pula:	S- O+

A sentença abaixo mostra como os conectores determinam os vínculos entre as palavras, segundo a gramática definida pela tabela.



Os sinais dão uma orientação ao vínculo evitando as agramaticais "*gato o*" ou "*o gato o muro pula*". A restrição de planaridade, por seu lado, evita cadeias como "*o pula gato*". A cadeia "*o muro pula o gato*" é aceita, pois embora sem sentido é uma frase sintaticamente correta. □

[11] Em inglês *Link Grammars*.

Quando dois conectores estão vinculados dizem-se satisfeitos. Cada linha traçada entre duas palavras representa um vínculo que satisfaz um conector, sendo dai deriva o nome do formalismo. Os conectores de alguma maneira definem uma taxonomia hierárquica sobre as palavras no léxico. Por exemplo, todos os substantivos terão um subconjunto de marcas comuns e por seu lado os substantivos no singular terão outro conjunto mais específico.

Para solucionar as ambigüidades que poderiam surgir com palavras pertencendo a mais de uma categoria introduz-se o conceito de requerimento de conexão que generaliza a noção de conector. Com cada palavra do léxico, deve ser associado um requerimento de conexão que consiste de um conjunto de pares de listas de conectores. Cada par tem uma lista para os conectores com sinal positivo e outra para sinal negativo.

As estruturas das sentenças em uma língua natural têm características mais fortes que a propriedade de planaridade. Para controlar essas particularidades, o formalismo impõe algumas restrições extras. Uma seqüência de palavras é reconhecida como uma sentença da linguagem especificada por uma Gramática de Vínculos se existe uma maneira de estabelecer vínculos entre as palavras tal que todos os conectores de um par de listas do requerimento de conexão sejam satisfeitos seguindo as seguintes meta-egras:

- Planaridade
- Conectividade: o conjunto de vínculos deve formar um grafo conexo.
- Ordem: a ordem nas listas de conectores no requerimento de conexão indica como devem ser estabelecidos os vínculos.
- Exclusão: um conector não pode receber dois vínculos.

A meta-regra ordem, além de facilitar a programação do algoritmo de reconhecimento, caracteriza uma propriedade presente em muitas línguas. Por exemplo, em um sintagma nominal em português o artigo sempre deve preceder aos adjetivos ou ao próprio substantivo. A última meta-regra é óbvia mas pode incorporar restrições excessivas como é o exemplo de limitar o número de adjetivos que podem modificar um substantivo.

A concordância é modelada com os conectores. Por exemplo, para representar a concordância nominal em uma Gramática de Vínculos para o português, devem-se definir quatro pares de conectores diferentes para as combinações de gênero e número. De igual maneira pode ser representada a concordância verbal; cada combinação de pessoa e tempo deve definir um conector que poderá ser ligado ao conector correspondente em um pronome ou substantivo.

Como já foi dito as restrições que dizem respeito à posição que uma palavra pode tomar em uma sentença são determinadas pelas meta-regras. Como em todo modelo lexical, as

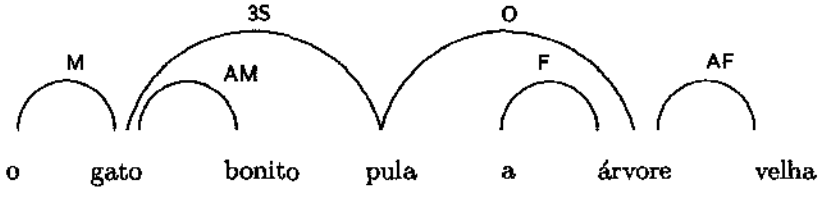
exceções são facilmente implementadas, e cada palavra têm sua própria definição. Em uma Gramática de Vínculos para o inglês a entrada para *university* terá um conector em comum com o artigo *a*, enquanto *hour* o terá com *an* solucionando o problema mencionado em 2.1.5.

Exemplo 8 A seguinte Gramática de Vínculos mostra como é modelada a concordância. A lista de palavras representa o léxico com os requerimentos de conexão.

eu:	{ ()(1S) }
o:	{ ()(M)}
a:	{ ()(F)}
gato, muro:	{(M)(*AM,3S)} { (M,O)(*AM) }
gata, arvore:	{(F)(*AF,3S)} { (M,O)(*AF) }
bonito, velho:	{(AM)()}
bonita, velha:	{(AF)()}
lato pulo:	{ (1S) (O) }
late pula:	{ (3S) (O) }

Entre chaves são representados os pares de listas de conectores que em conjunto formam os requerimentos de conexão. Um * indica um conector que aceita múltiplos vínculos; na tabela esse tipo de conector é usado para indicar que um substantivo pode ser modificado por mais de um adjetivo.

O esquema abaixo mostra como podem ser traçados os vínculos mantendo as restrições impostas pelos requerimentos de conexão e das meta-regras com a sentença *o gato bonito pula a árvore velha.*



□

Em [ST91], Sleator e Temperley também propõem um algoritmo para verificar as soluções possíveis dada uma cadeia de palavras. O algoritmo consiste de uma busca descendente exaustiva de todos os possíveis esquemas de vínculos. O algoritmo desenvolve uma estratégia de dividir para conquistar. No mesmo trabalho é demonstrada a equivalência de poder de expressão entre as Gramáticas de Vínculos e as GLC; como conseqüência não deve surpreender o fato de que a complexidade do algoritmo de reconhecimento seja quadrática no comprimento da cadeia de entrada.

A principal vantagem das Gramáticas de Vínculos é a elegância e a relativa simplicidade da implementação. Por outro lado, a distribuição da informação no léxico parece facilitar o projeto de uma gramática real, em comparação a uma GLC. Além disso, oferece uma visão algorítmica da análise sintática.

Em [GLS96], estende-se este modelo para implementar uma versão robusta do algoritmo de reconhecimento baseado na noção de vínculo nulo que permite por exemplo ligar duas palavras adjacentes sem seguir as restrições dos respectivos requerimentos de conexão. Em [LST92] propõe-se uma outra extensão que resulta em um híbrido de um modelo estocástico com uma Gramática de Vínculos.

2.2.3 Modelos Estocásticos

O uso de métodos estocásticos começou a ser utilizado na Lingüística Computacional principalmente para manipular fonte de dados com interferências, como por exemplo nos sistemas de reconhecimento de voz. O sucesso obtido nesta área levou aos pesquisadores a testar este tipo de ferramentas em outras áreas.

A motivação atual para o uso deste tipo de método reside principalmente na procura de sistemas que possam tratar fontes de dados sem restrições; um exemplo típico são os marcadores de *corpos*. A marcação consiste em atribuir a cada palavra de um *corpo* uma marca de um conjunto finito que em geral vai representar a categoria gramatical. O principal empecilho para esta tarefa são as palavras ambíguas.

Os métodos estatísticos tentam solucionar o problema da ambigüidade gerando um esquema que captura a probabilidade de que uma palavra possa receber uma marca em um contexto dado. Em geral, o contexto de palavras contempla apenas uma ou duas palavras além da analisada (*bigrama* ou *trigrama*). O esquema é configurado a partir de um "treinamento" feito com os dados em um *corpo* menor já marcado.

Os modelos estocásticos são implementados sobre autômatos finitos e são discutidos no capítulo seguinte. Para uma introdução ao uso de estatística em processamento de línguas ver [KS96]. Em [Sch94] é apresentado um método de marcação automática probabilístico baseado em árvores de decisão. Métodos estocásticos foram propostos também para a análise sintática; um exemplo desta aplicação pode ser visto em [Mag94]. Em [Vil95] é avaliado um marcador probabilístico para a língua portuguesa.

2.3 Resumo

A definição tradicional de concordância é restrita à relação de flexões de gênero e número. Nesta dissertação, a noção é estendida para dar cobertura a outros fenômenos presentes nas línguas; como por exemplo as flexões impostas pelas declinações de caso, ou os

restrições derivadas de fenômenos fonológicos (*a university, an hour*).

Neste capítulo apresenta-se, também, uma visão pessoal da evolução das propostas para a análise sintática na Lingüística Computacional. O impulso maior a esta área foi dado a partir do sucesso atingido pelo movimento gerativista. Esse fato reflete-se nos primeiros sistemas de PLN, pois a maioria deles utilizava alguma adaptação do modelo de GLC.

Quase paralelamente, e com uma repercussão menor, alguns lingüistas trabalharam em modelos caracterizados pela ausência de regras ou estruturas além do léxico. Nesses formalismos, chamados lexicais, toda a informação necessária para modelar a língua é representada no léxico junto com as palavras. Exemplos destes modelos são as Gramáticas de Categorias e as Gramáticas de Vínculos.

A demanda por sistemas eficientes e não tão comprometidos com a abrangência abriu nos últimos anos outras alternativas para a área de análise sintática, como mostra o capítulo seguinte.

Capítulo 3

Autômatos Finitos

> *"...it is impossible, not just difficult, to construct a finite-state device which will produce all and only the gramatical sentences of English."*
> — Noam Chomsky (em [Cho57]).

Técnicas baseadas em autômatos finitos foram utilizadas em ciência da computação desde suas origens. O sucesso deste formalismo reside basicamente na simplicidade, elegância, regularidade e poder de representação. Programas usados comumente, tais como processadores de textos e analisadores léxicos de compiladores são projetados sobre máquinas de estados finitos.

Nos começos da Lingüística Computacional os autômatos finitos foram desconsiderados como uma técnica viável. O argumento levantado por Chomsky em [Cho57] sobre o limitado poder de representação das máquinas de estados finitos foi, talvez, a principal razão. Neste capítulo apresenta-se uma breve introdução à teoria de autômatos finitos e discutem-se as principais abordagens baseadas neste formalismo.

3.1 Autômatos Finitos

Um autômato finito determinístico é um modelo matemático de um sistema com entrada e saída discretas e um conjunto finito de possíveis configurações. Formalmente, um autômato finito é denotado por uma quíntupla

$$(Q, \Sigma, \delta, q_0, F)$$

onde:

- Q é um conjunto finito de estados;
- Σ é o alfabeto finito dos símbolos de entrada;

- δ é a função de transição ou movimento definida sobre $Q \times \Sigma \rightarrow Q$;
- $q_0 \in Q$ é o estado inicial;
- $F \subseteq Q$ é o conjunto de estados finais.

A máquina abstrata consiste de uma unidade de controle e uma fita de entrada escrita com símbolos em Σ. Em um momento dado, a unidade de controle está configurada com um estado em Q e lendo um símbolo da fita de entrada.

Em um movimento, a máquina no estado q e lendo o símbolo a, passa ao estado definido por $\delta(q, a)$ e avança uma posição na fita. Dado um símbolo na entrada e um estado, a função δ define exatamente uma transição a um outro estado; por isso este tipo de autômato é chamado determinístico.

Dada uma cadeia de símbolos pertencentes a Σ na fita de entrada, uma computação do formalismo consiste de uma seqüência de transições começando no estado inicial q_0 e avançando sobre os estados conforme as restrições impostas pela função δ. Uma computação termina com sucesso quando todos os símbolos na fita de entrada foram analisados e a última configuração da unidade de controle é um estado $f \in F$.

Grafos orientados são utilizados para representar autômatos finitos. Cada vértice do grafo é associado com um estado em Q e as arestas, rotuladas com símbolos em Σ, representam a função de transição δ. Em geral, o vértice correspondente ao estado inicial é indicado com uma seta, e os vértices associados com os estados finais são representados com contorno duplo. A topologia do grafo associado permite definir algumas propriedades sobre os autômatos. Por exemplo, um autômato é acíclico se o grafo que o representa não contém ciclos. Este tipo de diagrama é utilizado nesta dissertação para representar autômatos finitos.

Na teoria de linguagens formais, os autômatos finitos são as máquinas reconhecedoras das *linguagens regulares*. Informalmente, as linguagens regulares são aquelas que podem ser reconhecidas com um dispositivo com "memória finita".

Exemplo 9 Um exemplo de linguagem regular é a definida sobre o alfabeto binário {0,1} formada pelas cadeias terminadas em 01. O autômato da figura 3.1 é uma máquina reconhecedora desta linguagem. A cadeia da linguagem 00101 é reconhecida seguindo a seqüência de estados A B A A B C. □

Existem numerosas variantes à máquina de estados finitos, tais como: autômatos não-determinísticos, autômatos com movimentos-ϵ e autômatos bi-direcionais. Em [HU79] é demonstrado que todas essas variantes podem ser reduzidas a um autômato conforme é definido aqui. De agora em diante será usado o termo autômato finito ou simplesmente autômato para referenciar a definição apresentada nesta seção, ou a uma instância em particular, dependendo do contexto.

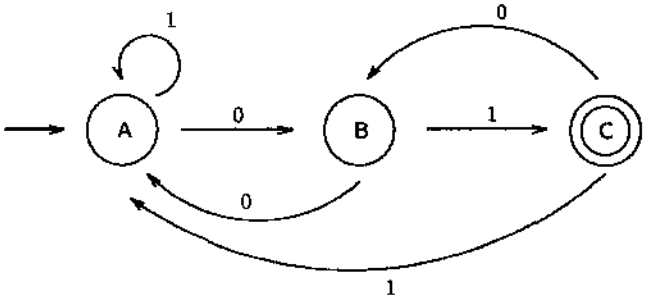


Figura 3.1: Autômato reconhecedor da linguagem (0:1)*01.

Para uma introdução à teoria de linguagens formais e autômatos o leitor pode se remeter a [HU79, Har78]. Uma aplicação de autômatos finitos para a implementação de analisadores léxicos para a construção de compiladores é descrita em [AU79]. O uso de autômatos finitos para projetar algoritmos de casamento de padrões é descrita em [CLR90].

3.2 Análise Morfológica

Estruturas de dados baseadas em autômatos permitem representar grandes vocabulários de uma forma muito compacta e mantendo o acesso eficiente à informação. Em [LK93, Rev91] é descrito o uso de autômatos finitos acíclicos para a construção de corretores e conselheiros ortográficos.

A partir dos resultados destes trabalhos foram desenvolvidas ferramentas para manipular grandes vocabulários (vide 7.1). A principal vantagem das estruturas de dados fornecidas pelos autômatos é que permitem aproveitar a grande quantidade de prefixos e sufixos compartilhados pelas palavras em vocabulários de línguas naturais. Nesta representação cada transição entre estados de um autômato é rotulada com uma letra do alfabeto e cada caminho de transições, desde o estado inicial até um estado final, armazena uma palavra.

A figura 3.2 é um exemplo de como são tratados os afixos comuns; são necessários 14 estados e 16 transições do autômato para representar as palavras: *conviver, reviver, conceber, receber, convivendo, revivendo, concebendo* e *recebendo*. Deve-se notar que os prefixos da figura seriam compartilhados em um autômato com mais informação por todas as formas dos verbos *conviver, reviver, conceber* e *receber*. Além disso, os sufixos do autômato podem ser compartilhados também pelas formas dos verbos regulares da segunda conjugação, tais como *escrever* e *viver*.

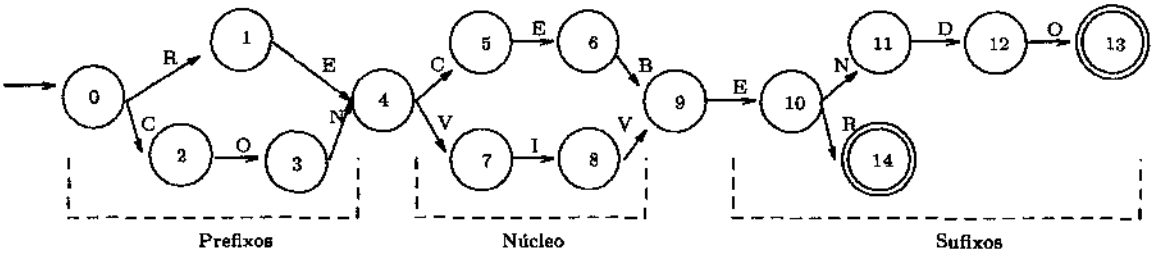


Figura 3.2: Autômato para armazenar palavras

Uma primeira utilização desta representação foi o desenvolvimento de corretores ortográficos. Um esquema para este tipo de sistema é formado por um autômato armazenando as palavras da língua, e o algoritmo de reconhecimento simplesmente consiste em procurar um caminho no autômato para uma cadeia de símbolos. Uma vantagem desta implementação para corretores ortográficos é que o procedimento de análise não precisa fazer nenhum tipo de inferência diminuindo as possibilidades de aceitar uma palavra inexistente.[1]

Em [LK93] também apresentam-se extensões para implementar aconselhamento a partir de semelhanças na pronúncia, facilidades de sinônimos e implementação de dicionários multilíngües. Em [KLS95a] utilizam-se autômatos para a depuração semi-automática de vocabulários; a partir da "forma" do grafo associado ao autômato e com a manipulação de certos parâmetros inferem-se, por exemplo, formas verbais ausentes no léxico.

Sistemas de processamento de línguas naturais requerem a implementação de funções léxicas mais sofisticadas como aquelas que retornam as flexões de verbos ou declinações de substantivos. Em [KLS98] propõe-se uma codificação eficiente baseada também em autômatos finitos que permite representar o tipo de informação necessário por esse tipo de funções.

3.3 O Argumento de Chomsky

Em [Cho57], Chomsky apresenta a famosa argumentação sobre as limitações das máquinas de estados finitos para capturar a essência das línguas naturais. Essa posição causou controvérsia na própria Lingüística e foi contestada por alguns pesquisadores (ver [Zif74, Rei69]). Porém teve uma aceitação generalizada, influenciando inclusive os primeiros trabalhos na Lingüística Computacional.

[1] O corretor SPELL do UNIX aceita palavras analisando apenas afixos e raízes. Por exemplo, a palavra em inglês *misunsubcow* é reconhecida pois é formada pelos prefixos válidos *mis*, *un* e *sub* mais a palavra *cow* presente obviamente no dicionário.

Para justificar sua proposição *"o inglês não é uma linguagem de estados finitos"*, Chomsky apresentou o exemplo do encaixamento ilimitado de sentenças. A linguagem derivada a partir da aplicação de sucessivos encaixamentos é análoga à linguagem de parênteses balanceados. Essa linguagem não pode ser reconhecida por um autômato finito, mas pode ser gerada por uma GLC (vide exemplo 1). As máquinas reconhecedoras de linguagens livres de contexto necessitam memória infinita [AU79, Har78].

Chomsky ainda insiste na limitação dos autômatos quando considera a validade de impor um limite na profundidade do encaixamento. Segundo ele, as máquinas de estados finitos ainda falhariam em capturar a estrutura inerente.

Cabe destacar que embora o objetivo de Chomsky não fosse achar um modelo computacional para o processamento eficiente de língua natural, o sucesso do modelo gerativista impôs este preconceito à Lingüística Computacional [Kap95].

A diferença de objetivos entre a Lingüística e a Lingüística Computacional, e a demanda por sistemas mas especializados e eficientes, começaram a abrir novas aplicações onde técnicas baseadas em máquinas de estados finitos oferecem soluções viáveis para o processamento automático de línguas.

3.4 Análise Sintática

O uso de autômatos finitos para o processamento sintático tem duas linhas bem diferenciadas: análise superficial e marcação automática.

3.4.1 Gramáticas de Interseção de Estados Finitos

Em [Kos90], Koskenniemi apresenta as Gramáticas de Interseção de Estados Finitos (FSIG[2]) como uma alternativa para implementar análise superficial. As FSIG foram uma das primeiras aplicações de autômatos finitos no nível sintático.

O objetivo de Koskenniemi é desenvolver um analisador sintático superficial para determinar as categorias das palavras de uma sentença eliminando as ambigüidades. Cada sentença em uma FSIG é representada como uma máquina de estados finitos que aceita todas as possíveis *leituras* da mesma. Uma leitura é uma seqüência ordenada de categorias que pode ser alinhada com a sentença. Cada categoria deve ser uma escolha válida para a palavra que forma o par. A tarefa da gramática é aceitar as leituras corretas (se houver mais de uma), excluindo as incorretas.

Para completar o modelo, é construído um autômato que armazena todas as cadeias representando leituras válidas na língua modelada. Não existe nenhuma restrição quanto à forma do autômato, que potencialmente pode representar um número infinito de leituras.

[2]Em inglês *Finite State Intersection Grammar*.

O autômato obtido como a interseção dos autômatos de leituras da cadeia analisada e o autômato de todas as leituras é o resultado da análise.

Exemplo 10 O autômato da figura 3.3 representa todas as leituras possíveis da sentença em inglês *the program runs*. Aqui o modelo está simplificado para levar em conta apenas informação referente a categorias de palavras. No trabalho original as leituras incorporam informação mais fina, como a marcação da separação entre sintagmas. A tabela abaixo apresenta uma relação entre as marcas e seu significado no exemplo.

ART	Artigo
N1	Substantivo no singular
N2	Substantivo no plural
VI	Verbo infinitivo (após *to*)
VP	Verbo no presente
V3	Verbo na terceira pessoa

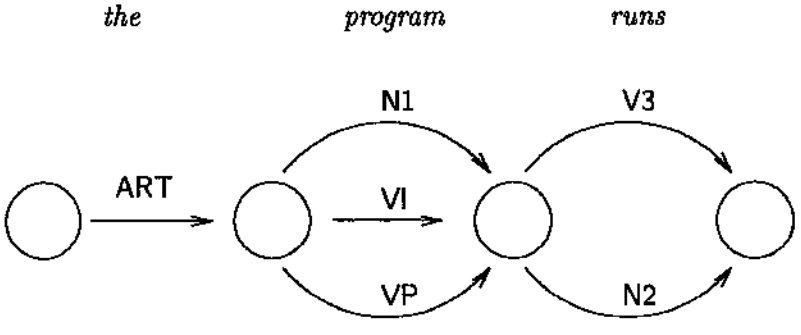


Figura 3.3: Autômato para *the program runs*

Um autômato que aproxime um número razoável das leituras válidas em inglês deve conter ART N1 V3, eliminando as outras possibilidades. □

A principal desvantagem deste formalismo reside justamente em como é definida a operação de reconhecimento. Dados dois autômatos A_1 e A_2, a interseção de ambos pode levar a um autômato com um número de estados $N_1 \times N_2$, onde N_1 e N_2 representam o número de estados de A_1 e A_2 respectivamente (ver detalhes do algoritmo de interseção de autômatos em [HU79]). Embora a representação das leituras nos autômatos possa ser compacta, a operação de interseção, além de potencialmente explodir o número de estados, pode ser muito custosa em tempo. Em [Kos92], o autor do formalismo discute as possíveis implementações do modelo, propondo algumas alternativas. Em [YJ95], propõe-se um esquema para a operação de interseção que reduz as necessidades de memória, mas ainda pode ser muito custoso em espaço e tempo.

3.4.2 Marcação Automática de *Corpos*

Os modelos estatísticos são utilizados especialmente na área de marcação automática. Este tipo de técnica é baseado em processos markovianos que podem ser implementados de forma simples usando máquinas de estados finitos.

Um modelo de Markov consiste de um processo estocástico reconhecedor ou gerador de uma linguagem específica, e pode ser de dois tipos:

- Cadeia de Markov
- Modelo de Markov Oculto (HMM[3])

Uma cadeia de Markov é um modelo matemático que pode ser diretamente implementado por um autômato finito. Cada transição do autômato é rotulada com um número que modela a probabilidade de ocorrência de um símbolo na entrada.

Exemplo 11 O autômato da figura 3.4 representa uma cadeia de Markov definida sobre o alfabeto {a,b}. A probabilidade de uma seqüência de símbolos é dada pelo produto das probabilidades associadas às transições percorridas durante o processo de análise. Por exemplo a probabilidade resultante para a cadeia aaabbbab seguindo os estados 1 1 1 2 2 1 2 é 0,0072576 (este exemplo foi extraído de [Vil95]). □

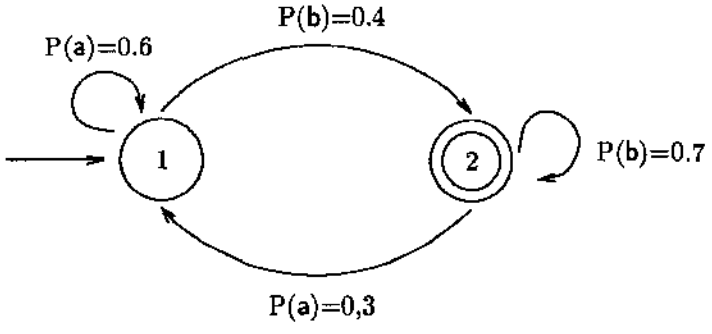


Figura 3.4: Cadeia de Markov

Um modelo HMM é uma generalização das cadeias de Markov onde o autômato subjacente pode ser não determinístico. Em outras palavras, podem existir estados de onde partem mais de uma transição rotulada com o mesmo símbolo da entrada.

Exemplo 12 O autômato da figura 3.5 representa um modelo HMM. O não determinismo está associado ao fato de que do estado 1 partem duas transições rotuladas com a probabilidade para o símbolo a (este exemplo foi extraído de [Vil95]).

[3]Em inglês *Hidden Markov Model.*

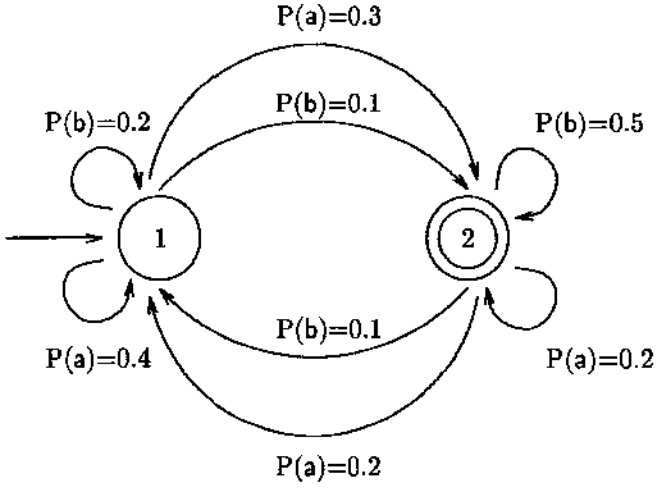


Figura 3.5: Modelo HMM

A probabilidade associada a uma cadeia é a resultante da somatória das probabilidades de cada possível caminho que a reconhece. □

Calcular todas as bifurcações possíveis para reconhecer o caminho mais provável em um modelo HMM pode ser uma tarefa complicada. Quase todas as técnicas estão baseadas no algoritmo de Viterbi descrito em [Vil95].

Sem entrar em detalhes, é fácil ver como um modelo representando bigramas ou trigramas pode ser utilizado para fazer marcação. Cabe notar que o principal objetivo desde tipo de programas é estabelecer a marca mais provável para uma palavra, embora seja incorreta. Um marcador sempre tenta atribuir a marca mais provável para uma palavra em um contexto dado. Este tipo de modelo deve ser configurado com as probabilidades para cada possível n-grama. Para esta tarefa em geral utilizam-se *corpos* de texto especialmente preparados.

Recentemente, o trabalho de Brill em [Bri93] mostrou que podem ser obtidos bons resultados usando regras ao invés de métodos estocásticos. Em [RS95], Roche e Schabes propõem uma implementação do sistema de regras de Brill utilizando autômatos.

3.5 Resumo

Embora a teoria de autômatos finitos tenha sido amplamente utilizada em ciência da computação, durante muitos anos sua aplicação a sistemas de PLN foi desconsiderada. Umas das razões desse desinteresse é o argumento de Chomsky no famoso livro *Syntax Structure* [Cho57] sobre o limitado poder de representação dos autômatos finitos.

As primeiras aplicações de autômatos finitos em Lingüística Computacional foram para armazenar grandes vocabulários. Os autômatos fatoram a informação relacionada aos prefixos e sufixos permitindo armazenar um grande número de palavras, consumindo pouco espaço e mantendo o acesso à informação eficiente.

Os sistemas baseados em modelos estocásticos para marcação morfo-sintática foram os primeiros a utilizar autômatos finitos no nível sintático. Dispositivos baseados neste formalismo permitem representar de uma forma natural modelos de Markov. Estas representações capturam n-gramas (em geral bigramas ou trigramas) em transições de um autômato finito com dados extraídos a partir de *corpos* marcados.

A demanda por sistemas eficientes motivou a procura por outras alternativas para o tratamento sintático de línguas, reconsiderando o uso de autômatos finitos. Um exemplo são as Gramáticas de Interseção de Estados Finitos propostas por Koskenniemi, que representam uma aproximação de todas as leituras válidas das sentenças de uma língua em um autômato. O processo de reconhecimento proposto é a intersecção desse autômato com outro contendo todas as leituras possíveis da palavra analisada.

No próximo capítulo apresenta-se um ambiente para a análise superficial de línguas naturais baseados em autômatos finitos. Como é mostrado no resto do trabalho esse tipo de estruturas permite a implementação de diferentes algoritmos de forma eficiente e simples.

Capítulo 4

Padrões Sintáticos

> *"A sentence is acceptable to me if my estimate of the probability of occurrence of a sentece of like construction in a natural language text is greater than zero. I exclude from natural language text sentences dreamed up by linguistics, psychologist, English teachers and poets."*
> — Peter Reich (em [Rei69])

Neste capítulo apresenta-se o modelo de padrões sintáticos para a análise superficial de línguas. O objetivo é mostrar como com um esquema simples podem ser modelados os aspectos relacionados ao desempenho de uma língua. Como é mostrado nos capítulos seguintes, essa modelagem fornece a informação suficiente para implementar uma análise mais sofisticada, como o conselheiro de concordância.

A idéia de usar padrões para a análise sintática não é nova. Em [Min68], por exemplo, é apresentado um trabalho de Bertrand Raphael sobre o sistema SIR[1], cujo objetivo é manter uma conversa inteligente com o usuário. Como outros pesquisadores da época, Raphael estava preocupado com os aspectos semânticos das línguas, mas precisava de uma análise da estrutura da entrada. A solução foi projetar um conjunto de padrões para modelar todas as entradas possíveis.

Em Lingüística, o trabalho de Harris [Har71] tentou capturar a matemática por trás das estruturas das línguas naturais a partir da modelagem de um sistema discreto de símbolos similares aos padrões sintáticos. Porém, o objetivo de Harris não era definir um modelo computacional, mas sim formalizar as línguas para facilitar o seu estudo.

Nos últimos anos surgiram propostas que usam padrões como ferramentas complementares; um exemplo dessa abordagem é a apresentada em [TJ94], onde se propõe a coleta de padrões para resolver problemas de concordância na vizinhança de palavras conflitantes.

[1] Em inglês *Semantic Information Retrieval*.

4.1 Alfabeto e Léxico

A representação de uma língua a partir da expressão escrita é modelada como um sistema discreto de símbolos e um conjunto de regras que determinam as seqüências bem formadas aceitas como sentenças. Como o objetivo é fornecer um modelo que capture características sintáticas de uma língua natural, as estruturas primitivas são as *palavras*. Cada sentença da língua natural é constituída por uma seqüência de palavras tomadas de um léxico. Uma cadeia de palavras é denotada por $w_1 \ldots w_n$ onde cada w_i é uma palavra. Para os objetivos deste trabalho, uma palavra é considerada um elemento indivisível e deve ter uma entrada associada no léxico.

A cada palavra do alfabeto é associada no mínimo uma *marca*. Uma marca representa, em geral, uma categoria de palavras e está relacionada ao conceito de marca ou rótulo morfo-sintático[2] [Vil95].

O léxico é uma estrutura de dados, onde se armazena a informação relativa a palavras e marcas. Podem existir palavras com mais de uma marca associada. Por exemplo, a palavra *dog* pode ser marcada como substantivo contável singular e verbo transitivo no infinitivo. Dada a palavra w, a função $tag(w)$ retorna o conjunto de todas as marcas associadas a essa palavra.

O léxico deve também implementar a função $corr(w)$ que retorna as palavras correlatas associadas a uma palavra w. A definição de *palavra correlata* dependerá da utilização do modelo. Para um conselheiro gramatical, por exemplo, as palavras formando a conjugação de um verbo, ou todas as formas de gênero e número de um substantivo podem formar um conjunto de palavras correlatas. Associado a esta noção define-se a palavra base ou canônica que é utilizada como representante de uma classe de palavras correlatas. Os verbos no infinitivo e substantivos masculinos no singular são possíveis exemplos de palavras canônicas.

4.2 Padrões Sintáticos

A hipótese que suporta o trabalho desta dissertação é que núcleo de sentenças que são usadas em um texto, sem que nenhuma restrição seja imposta, seguem padrões, estruturas e esquemas que podem ser limitados. Esses padrões refletem as relações superficiais entre as categorias governadas principalmente pelas regras gramaticais, como por exemplo as regras de concordância. O ambiente aqui proposto baseia-se justamente em capturar esses esquemas em uma estrutura de dados especial.

Um *padrão sintático* é definido como uma cadeia $t_1 \ldots t_n$, onde cada t_i representa uma marca para uma categoria de um conjunto pré-definido. Um padrão sintático permite

[2] Em inglês *part-of-speech tag*.

capturar as regularidades na estrutura de um conjunto de sentenças, impondo restrições sobre a ordem e posição que as palavras podem tomar.

Exemplo 13 Diferentes sentenças utilizadas em todo tipo de textos repetem a mesma estrutura básica, capturada por um único padrão sintático.

Θ= {N1,V2,PF,ART...} Σ= {*o,cachorro,late,a,menina,canta...*}

Léxico

o:	ART
a:	ART
cachorro:	N1
menina:	N1
late:	V2
canta:	V2
.:	PF

ART N1 V2 PF (4.1)

o cachorro late . (4.2)

a menina canta . (4.3)

o cachorro canta . (4.4)

Neste pequeno exemplo, Σ é um conjunto de palavras, e Θ representa o conjunto de marcas. A tabela combina essa informação em um léxico. O padrão sintático 4.1 representa as sentenças em português 4.2, 4.3 e 4.4

O ponto é considerado uma palavra com marca PF. Claramente para representar as relações de concordância é necessário refinar o conjunto de marcas para evitar que o padrão sintático 4.1 aceite a cadeia 4.5.

* *a cachorro late .* (4.5)

□

Dada uma estrutura que possa armazenar os padrões sintáticos, a operação de reconhecimento consiste na busca por um padrão que case com a cadeia na entrada. Embora um padrão sintático tenha um grande poder de representação, é preciso armazenar muitos deles para implementar um reconhecedor de uma língua natural.

Os padrões sintáticos são cadeias de símbolos que compartilham sufixos e prefixos, e um autômato é um bom candidato para fornecer a estrutura de dados necessária para solucionar o problema de armazenamento. Intuitivamente, padrões sintáticos parecem compartilhar afixos da mesma maneira que as palavras em um vocabulário.

A origem dos afixos comuns no nível léxico é etimológica; no caso da estrutura dos padrões sintáticos é uma conseqüência das regras de formação das sentenças, as quais são independentes do significado das partes componentes. Por exemplo, o subpadrão 4.6, definido a partir dos dados do exemplo 13, pode definir o sintagma nominal de uma grande

quantidade de sentenças cujos sintagmas verbais podem ser muito mais complexos, ou consistir apenas de um verbo como mostram as sentenças 4.7 e 4.8.

ART N1... (4.6)

o cachorro late . (4.7)

a menina canta no coral da universidade . (4.8)

Uma cadeia de palavras $w_1 \ldots w_n$ é reconhecida como uma sentença, se é possível achar um padrão sintático $t_1 \ldots t_n$ onde cada t_i representa uma marca para w_i, para $1 \leq i \leq n$. A operação que, dada uma cadeia, tenta achar um padrão que satisfaz essas restrições é chamada de *casamento de padrões.* Até aqui, pode-se resumir o modelo proposto como consistindo de dois dicionários: léxico e padrões sintáticos. A operação de casamento deve decidir se a cadeia de entrada é uma sentença procurando achar um padrão; para isso, precisa do léxico para obter a informação relativa às marcas das palavras.

Em [Lap96], Laporte argumenta sobre a necessidade de incorporar informação léxica em modelos gramaticais baseados em regras para manipular exceções, comuns em línguas naturais. O modelo aqui apresentado não fica isento desse problema como é mostrado a seguir. Certas exceções podem incorporar complicações que são solucionadas agregando informação literal ao possível conjunto de marcas.

Em inglês os advérbios de grau precedem a palavra modificada, sendo *enough* uma exceção. Um padrão para representar o contexto de um advérbio de grau pode ser 4.9. Esse padrão sintático representa, entre muitas outras, a sentença 4.10 mas não a igualmente válida 4.11.

ART N1 V2 ADV ADJ (4.9)

the water is very hot (4.10)

the water is hot enough (4.11)

Para solucionar este problema são definidas marcas que representam diretamente uma palavra. Este tipo de marca é chamada *literal*, e geralmente são notadas entre parênteses angulares ("<" e ">"). No caso da palavra *enough*, uma solução é usar o padrão 4.12.

ART N1 V2 ADJ <enough> (4.12)

Note-se que a incorporação de literais deve se refletir na configuração do léxico. No caso de *enough* a entrada no léxico como advérbio deve ser eliminada pois pode gerar a cadeia incorreta 4.13.

* *the water is enough hot* (4.13)

Os literais também são utilizados como marcas para categorias de palavras com apenas um elemento, com a partícula em inglês *to*. A operação de casamento de padrões deve ser estendida para manipular literais. Se a marca t_i é um literal, o reconhecedor deve verificar a igualdade com a palavra w_i na cadeia de entrada.

4.3 Casamento de Padrões

Como mostra a experiência reportada ao longo desta dissertação, a coleta de padrões sintáticos armazenados em autômatos finitos acíclicos fornece um modelo viável e simples para implementar análise sintática de línguas naturais. O uso de autômatos como estrutura subjacente não é apenas sustentada pela eficiência do armazenamento e recuperação de informação. Autômatos finitos fornecem um esquema que permite a implementação simples dos algoritmos de reconhecimento do conselheiro de concordância. Nesta seção apresenta-se um algoritmo para a operação de reconhecimento. Os algoritmos apresentados ao longo deste trabalho partem do pressuposto que o autômato que implementa a estrutura de armazenamento é determinístico e acíclico. A abordagem mais simples para a operação de casamento consiste numa busca em profundidade no autômato de padrões sintáticos. O algoritmo da figura 4.1 procura todos os padrões que podem ser casados com uma cadeia $w_1 \ldots w_n$. Em cada invocação recursiva, o algoritmo *Prefixo* avança sobre um prefixo de um conjunto de possíveis padrões sintáticos, que casa com o prefixo da cadeia de entrada de mesmo comprimento.

O primeiro argumento de *Prefixo* é o índice i da palavra w_i sendo analisada, o segundo é o estado do autômato visitado; e o último argumento armazena o prefixo atual casado na busca. A rotina *Sucessor* implementa a função de movimento δ sobre o autômato de padrões; os argumentos são o estado e a marca que determinam a transição. A operação binária & implementa a concatenação de duas cadeias.

A figura 4.2 mostra parte de um autômato armazenando padrões sintáticos definidos sobre um conjunto de marcas para sentenças em inglês. No capítulo 6 introduz-se o conjunto de marcas utilizado em uma implementação real do modelo. Porém, para entender o exemplo é suficiente uma explicação superficial do significado de cada marca:

```
Prefixo(i,estado,prefixoPad)
    se i=n+1 então
        se Final(estado) então
            Imprimir(prefixoPad)
        senão
            próximoEstado ← Sucessor(estado,<w_i>)
            se Válido(próximoEstado) então
                Prefixo(i+1,próximoEstado,prefixoPad & <w_i>)
            para cada t ∈ tag(w_i) fazer
                próximoEstado ← Sucessor(estado,t)
                se Válido(próximoEstado) então
                    Prefixo(i+1,próximoEstado,prefixoPad & t)
```

Figura 4.1: Algoritmo para o Casamento de Padrões

N1	Substantivo contável singular
NU	Substantivo não contável
NP	Substantivo próprio
A1	Adjetivo "curto"
AL	Adjetivo "longo"
V1	Verbo intransitivo infinitivo
V2	Verbo transitivo infinitivo
S1	Verbo intransitivo terceira pessoa
S2	Verbo transitivo terceira pessoa

Um adjetivo "curto" é aquele que forma o comparativo e o superlativo com uma flexão da própria palavra, por exemplo *cheap*. Um adjetivo "longo" forma o comparativo e o superlativo com as partículas *more* e *most* respectivamente, por exemplo *expensive*.

Tomando o estado 1 como inicial no autômato da figura 4.2, <the> N1 S2, <a> AL N1 S2 e <a> A1 A1 N1 S2, dentre outros, são prefixos válidos. A função *tag*() foi definida como retornando um conjunto, porém o algoritmo de casamento da figura avalia cada marca em uma ordem pré-determinada. Essa ordem, dentre outros fatores, determina a

quantidade de passos necessária para a operação de casamento.

a following red page has ... (4.14)

A cadeia 4.14 é um prefixo válido de uma sentença em inglês; a tabela a seguir esquematiza um traço dos passos executados pelo algoritmo de casamento com essa cadeia na entrada considerando o autômato na figura 4.2.

tag(following)= {N1,AL} *tag(red)*= {N1,A1} *tag(page)*= {N1,V2}

a	*following*	*red*	*page*	*has ...*
<a> (1-2)	N1 (2-6)	*		
	AL (2-4)	N1 (4-6)	*	
		A1 (4-5)	N1 (5-6)	<has> (6−)

A *i*-ésima coluna da tabela representa as marcas tentadas com a *i*-ésima palavra da cadeia. Cada entrada contém uma marca e a transição em questão representada como um par de estados. Uma entrada com * significa que nenhuma marca pode ser usada em uma transição válida a partir do estado atual. O traço da operação do algoritmo mostra como a ordem na avaliação das marcas pode afetar a eficiência gerando retrocessos na busca.

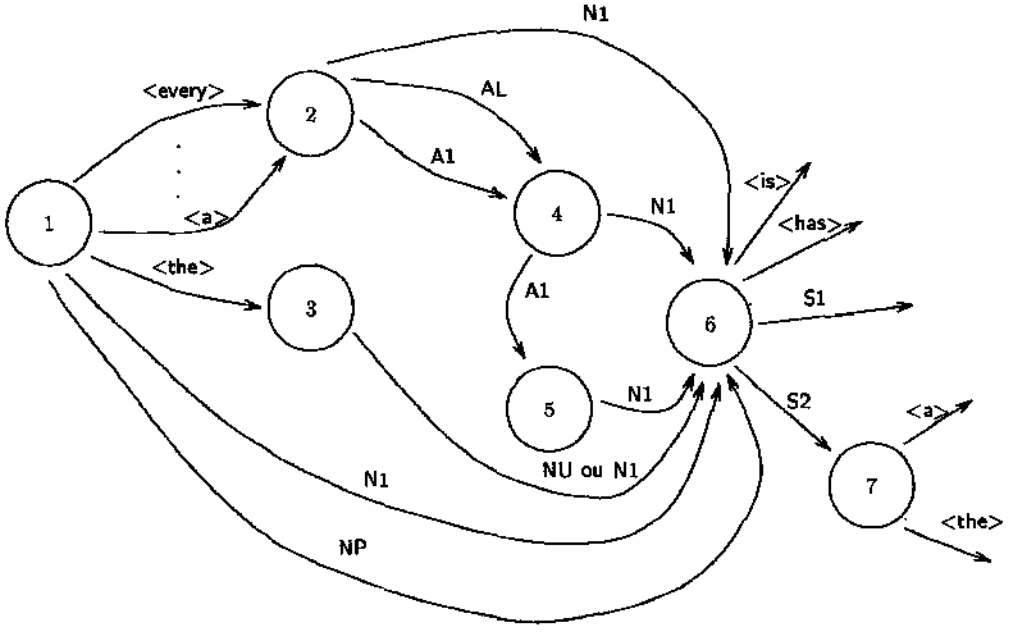


Figura 4.2: Autômato de Padrões Sintáticos

4.4 Algumas Considerações

A complexidade de uma busca em profundidade recursiva com retrocessos, como a implementada pelo algoritmo da figura 4.1, pode ser exponencial no comprimento da cadeia de entrada. Seria fácil projetar uma linguagem artificial com uma sentença que apresente esse pior tempo de execução.

Um exemplo é a análise de uma cadeia $w_1 \ldots w_n$ no autômato da figura 4.3, onde cada w_i pode ser marcada com N ou S. Qualquer autômato modelando uma árvore de decisão, como na figura 4.3, pode conduzir a um pior caso exponencial para o algoritmo de casamento. Este exemplo mostra como as palavras com múltiplas entradas no léxico são as que geram os retrocessos, porém esse tipo de situação é pouco freqüente na prática.

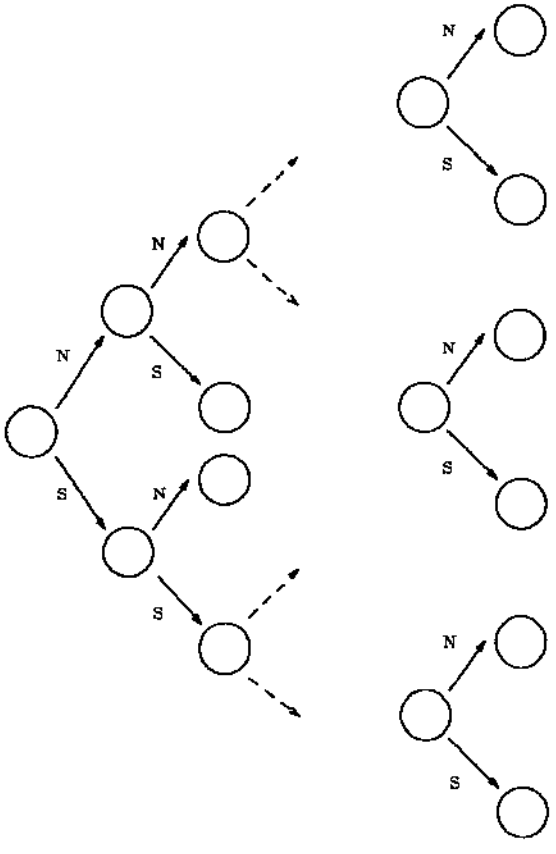


Figura 4.3: Árvore de Decisão

As línguas naturais apresentam algumas características que limitam consideravelmente o número de retrocessos. A primeira particularidade está relacionada com o número de palavras com múltiplas marcas. No léxico *OTA* da universidade de *Oxford* utilizado na

primeira experiência, 75% das 69.800 palavras têm apenas uma marca; o maior número de marcas corresponde às palavras *down, long, pop* e *round*, com quatro marcas cada uma. Esta medida pode dar uma noção um pouco enganosa; para uma conclusão mais precisa devem ser calculadas as freqüências de uso das palavras. De qualquer maneira, o número máximo de marcas está limitado por valores razoáveis. Além disso, palavras comuns como pronomes, determinantes e conectores são capturados nos padrões sintáticos como marcas literais, diminuindo assim os retrocessos.

Devem ser consideradas algumas sentenças "patológicas" como aquelas contendo listas de palavras ambíguas[3] separadas por vírgulas.

$$\ldots \textit{ square, red, dark and } \ldots \tag{4.15}$$

A subcadeia 4.15 pode ser reconhecida como, uma lista de adjetivos ou uma lista de substantivos, sendo que apenas uma destas interpretações será válida em uma sentença. Assim, se o algoritmo tomou um caminho errado, só será percebido no fim da lista.

Na prática, sentenças contendo listas de palavras ambíguas parecem não ser comuns, como mostra uma experiência realizada sobre 2.311 sentenças do *corpo SUSANNE*. Os resultados foram os seguintes:

- número total de cadeias: 2.311
- quantidade de palavras: 35.287
- comprimento médio das cadeias: 15,31
- quantidade total de passos do algoritmo: 37.820
- quantidade média de passos por cadeia: 16,36

Estes dados mostram que, em média, o algoritmo executa um passo a mais que o comprimento da cadeia analisada. Cabe destacar que cadeias agramaticais, em geral, levaram a um número menor de passos de execução.

Uma segunda particularidade das línguas naturais é a forte relação que existe entre as marcas das palavras em sentenças. Os modelos estocásticos de marcação automática fundamentam-se neste princípio [KS96]. Em geral, cada marca em um padrão é vizinha de um número limitado de outras marcas. A tabela 4.1 mostra dados extraídos do *Wall Street Journal Corpus* [Vil95] que exemplificam este fato. Uma experiência similar foi realizada com o banco de padrões gerado a partir do *corpo LOB* (ver mais detalhes no capítulo 6). A curva do gráfico da figura 4.4 representa a quantidade de n-gramas diferentes observadas. O número de n-gramas possíveis cresce quanto maior seja n, porém o gráfico mostra que a partir de $n = 7$ as n-gramas observadas diminuíram.

[3] Define-se como ambígua uma palavra com mais de uma marca no léxico.

	Bigramas	Trigramas
Marcas	48	48
Combinações Possíveis	2.304	110.592
Combinações Observadas	1.366	14.306

Tabela 4.1: *Wall Street Journal Corpus*

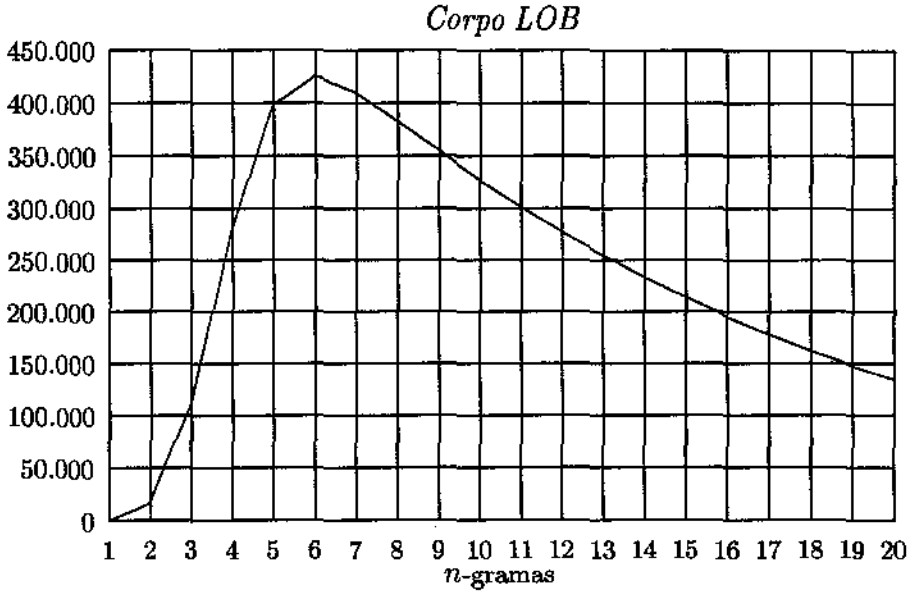


Figura 4.4: Relação dos n-gramas no *corpo LOB*

4.5 Coleta de Padrões

Para testar a viabilidade da abordagem aqui proposta é necessário capturar as sentenças com padrões coletados de uma fonte de informação confiável. Uma possibilidade é fazer uma coleta manual, ou partir para uma abordagem automática como a coleta de padrões sintáticos a partir de *corpos*.

Nos últimos anos, centros de pesquisa em Lingüística Computacional começaram a coletar textos de diferentes fontes criando *corpos* marcados. Um exemplo é o projeto *The Bank of English* [HB] que tem um total de 320 milhões de palavras. Outro projeto de envergadura é o *British National Corpus* [Oxf] que armazena 100 milhões de palavras com um esquema de marcação muito completo. Para as primeiras experiências na implemen-

tação do algoritmo de reconhecimento foi utilizado o *corpo SUSANNE*.[4] Posteriormente foi adquirida a coleção de *corpos ICAME*. Esta coleção inclui um conjunto de *corpos* de tamanho médio como o *Brown Corpus* e o *LOB*.

No capítulo 6 propõe-se um esquema de marcação para implementar a coleta de padrões sintáticos a partir de *corpos* de texto marcados.

4.6 Resumo

Neste capítulo apresenta-se o modelo gramatical de padrões sintáticos sobre o qual foi desenvolvido o trabalho desta dissertação. Um padrão sintático é uma seqüência de categorias de palavras tais como artigo, substantivo ou verbo, que permite capturar um grande número de sentenças de uma língua.

Alguns fenômenos presentes em algumas línguas requerem um poder de representação maior para serem capturadas. Um exemplo é o uso do adjetivo *enough* no inglês. Para manipular esta e outras exceções o modelo incorpora as categorias para representar palavras diretamente nos padrões, chamadas literais.

Os padrões sintáticos consistem de cadeias de símbolos que podem ser armazenados de uma forma eficiente em autômatos finitos. Assim, o modelo gramatical proposto consiste de dois dicionários: o léxico e um banco de padrões sintáticos. A operação de reconhecimento resume-se à busca no banco de padrões por um padrão que case com a cadeia na entrada. O algoritmo de casamento de padrões proposto neste capítulo implementa uma procura em profundidade sobre o autômato armazenando padrões sintáticos. Esta operação, embora seja teoricamente ineficiente, mostrou bons resultados na prática.

[4] O *corpo SUSANNE* está disponível no endereço `ftp://ota.ox.ac.uk/pub/ota/public/susanne`.

Capítulo 5

Conselheiro de Concordância

O objetivo principal desta dissertação é mostrar como padrões sintáticos armazenados em autômatos finitos podem capturar um conjunto representativo das sentenças de uma língua natural. O caminho escolhido para mostrar o poder de representação e a viabilidade do modelo gramatical proposto é a implementação de algoritmos para o tratamento de erros e fenômenos de concordância, conforme definido em 2.1.

Neste capítulo são apresentados algoritmos que, diante de um erro de concordância numa cadeia dada, propõem sentenças alternativas próximas. As operações para gerar esse tipo de sentenças são implementadas a partir do esquema de busca definido pelo algoritmo de casamento básico. Este tipo de operação de correção foi definido para sistemas de análise léxica e utilizado especialmente para implementar conselheiros ortográficos [LK93, Ofi96].

5.1 Erro de Concordância e Correção

Um erro gramatical pode ter diversas origens. Os mais comuns talvez sejam erros mecânicos introduzidos por descuido que geram pequenos desvios, nem sempre detectados e corrigidos no nível léxico. Outra fonte de erros é a falta de conhecimento da língua, como é o caso das pessoas que não são falantes nativas. Cabe frisar que o objetivo é tratar o tipo de erro puramente sintático sem entrar em questões tais como estilo, uso ou significado.

O conselheiro é implementado sobre três operações: substituição, remoção e inserção. A partir da aplicação de uma destas operações sobre a cadeia de entrada gera-se as sentenças propostas como alternativas. Essas sentenças alternativas, chamadas *conselhos*, devem casar com um dos padrões sintáticos armazenados.

Detectar o local onde o erro foi gerado não é simples, porque nem sempre é possível

determinar qual é a palavra ou palavras "incorretas".

** she see a dog.* (5.1)

we see a dog. (5.2)

she sees a dog. (5.3)

O algoritmo de casamento analisa uma cadeia avançando sobre transições válidas no autômato até atingir um estado final ou algum outro estado a partir do qual não existe uma transição válida, começando nesse ponto o retrocesso. Parece uma alternativa válida considerar o índice da palavra onde se inicia o retrocesso como uma regra heurística para detectar palavras conflitantes. Na cadeia 5.1, por exemplo, esta heurística determinaria que *she* é a palavra que causou o problema e assim 5.2 seria proposta como alternativa, em local da mais apropriada 5.3.

5.1.1 Substituição

Esta operação consiste na substituição de uma palavra x por uma em $corr(x)$ na cadeia de entrada. Este tipo de operação tenta corrigir o uso de uma flexão em um contexto errado.

Como foi discutido em 2.1.2, os gêneros das palavras nem sempre seguem regras determinadas pela morfologia, causando problemas inclusive aos falantes nativos. Em línguas latinas e algumas germânicas como o alemão, o gênero tem um papel muito importante na construção da frase; uma palavra com gênero equivocado pode gerar facilmente um erro de concordância. A substituição de uma palavra possivelmente errada pela forma correlata em outro gênero permite gerar sentenças alternativas a uma cadeia com erros de concordância deste tipo.

Em inglês o gênero tem uma importância menor; aliás, os substantivos comuns são tratados a maioria das vezes como neutros. Porém, outros fatores como a natureza contável dos substantivos impõem restrições de concordância, e a operação de substituição pode também fornecer alternativas neste caso.

** a milk is white.* (5.4)

the milk is white. (5.5)

Por exemplo, a cadeia 5.4 é agramatical pois *milk* é um substantivo incontável e não pode ser modificado por um artigo indeterminado. A sentença 5.5 é uma alternativa próxima a 5.4 obtida trocando o artigo indeterminado *a* pela forma correlata determinada *the*.

A substituição é útil também para gerar soluções para erros de concordância verbal. A substituição de uma forma verbal por outra deve ser feita mantendo-se a raiz e mudando-se

a desinência. Claramente, não faria sentido trocar uma palavra por outra com significado totalmente diferente. No caso da cadeia 5.1, substituindo a palavra *see* pela forma correlata *sees* obtém-se a sentença 5.3. Note-se que a qualidade da operação de substituição dependerá da definição dos conjuntos de formas correlatas *corr*() no léxico.

5.1.2 Remoção

A substituição de palavras não é suficiente para gerar alternativas para todos os erros de concordância, como a repetição de palavras e o uso indevido de artigos.[1]

* *the Paris is beautiful.*	(5.6)
Paris is beautiful.	(5.7)

Línguas como o espanhol e o inglês não permitem o uso de artigos com certos substantivos próprios, como nomes de cidades. Assim, 5.6 é agramatical, mas eliminando o artigo *the* obtém-se a sentença 5.7.

5.1.3 Inserção

Embora a substituição e a eliminação permitam tratar muitos erros de concordância gerando sentenças alternativas, existem exemplos de *conselhos* que podem ser gerados a partir da inserção de palavras.

* *the cat sees dog.*	(5.8)
the cat sees dogs.	(5.9)
the cat sees a dog.	(5.10)
the cat sees the dog.	(5.11)

Nem sempre fica claro quais são as melhores alternativas diante de cadeias como 5.8. Por exemplo, substituindo *dog* pela forma correlata *dogs* obtém-se a sentença 5.9. Outras alternativas parecem igualmente boas, como as obtidas a partir da inserção dos artigos *a* e *the* produzindo 5.9 e 5.11 respectivamente.

A classe de palavras que uma pessoa tentaria inserir para solucionar um problema de concordância é relativamente pequena e é restringida pelo contexto. Uma boa opção é definir como palavras plausíveis a serem inseridas àquelas representadas como literais nos próprios padrões sintáticos.

[1] O uso de artigos em inglês é mais limitado quando comparado a algumas outras línguas como as de origem latina; este fenômeno é chamado *zero article* [QG80].

Erros graves, como a falta de um verbo no predicado, não podem ser solucionados pelas operações apresentadas, mas a inserção permite gerar formas gerais de sentenças inserindo marcas extraídas dos padrões sintáticos.

* *the cat a dog.* (5.12)

the cat S2 *a dog.* (5.13)

Em 5.12 o verbo está ausente e seria impossível adivinhá-lo sem analisar o significado das palavras e o contexto. Porém, a operação de inserção poderia gerar uma forma geral como 5.13 indicando a falta de um verbo, neste caso transitivo e na terceira pessoa do singular (S2).

5.2 Algoritmos

As operações de substituição, remoção e inserção estendem a busca em profundidade gerando outras alternativas por onde continuar o casamento de padrões. Não há preocupação em identificar o local onde o erro foi gerado.

5.2.1 Substituição

O algoritmo da figura 5.1 implementa a substituição de palavras. O primeiro argumento de *Substituição* indica a palavra w_i a ser substituída que é usada como parâmetro da função $corr()$. O algoritmo substitui w_i por uma outra palavra w em $corr(w_i)$ e tenta avançar na busca com w como marca literal; nos passos seguintes o algoritmo procura transições utilizando as marcas em $tag(w)$.

Exemplo 14 Dada a cadeia 5.4 e o autômato de padrões da figura 4.2, a tabela mostra como, aplicando uma substituição, é gerado o prefixo 5.5.

$$tag(milk)=\{\text{NU}\ldots\}$$

$$corr(a)=\{the,\ an\}$$

a	*milk*	*is ...*
<a> (1-2)	*	
subs: *the*		
<the> (1-3)	NU (3-6)	<is> (6-)

□

Substituição(w_i,i,estado,prefixoPad)

para cada $w \in corr(w_i)$ **fazer**

próximoEstado ← *Sucessor(estado,<w>)*

se *Válido(próximoEstado)* **então**

Prefixo(i+1,próximoEstado,prefixoPad & <w>)

para cada $t \in tag(w)$ **fazer**

próximoEstado ← *Sucessor(estado,t)*

se *Válido(próximoEstado)* **então**

Prefixo(i+1,próximoEstado,prefixoPad & t)

Figura 5.1: Algoritmo de Substituição.

Exemplo 15 Como foi dito anteriormente, a substituição pode fornecer também *conselhos* para cadeias com problemas de concordância verbal.

** the cat see a dog.* (5.14)

the cat sees a dog. (5.15)

A tabela seguinte mostra como a substituição de *see* pela forma correlata *sees* em 5.14 resulta na sentença 5.15, a qual casa com um padrão no autômato.

$corr(see)$={*sees, seeing, seen, saw...*}

$tag(sees)$={S2...}

the	*cat*	*see*	*a*	*dog...*
<the> (1-3)	N1 (3-6)	*		
		subs: *sees*		
		S2 (6-7)	<a> (7-)	...

□

5.2.2 Remoção

A operação de remoção é implementada incrementando o índice da palavra atual mantendo o último estado válido da busca. Dado um estado onde a palavra atual não tem uma

marca que permita avançar sobre alguma das transições, a remoção consiste em continuar a busca com o mesmo estado e a próxima palavra.

Remoção(i,estado,prefixoPad)
  se *i=n* **então**
    se *Final(estado)* **então**
      Imprimir(prefixoPad)
  senão
    Prefixo(i+1,estado,prefixoPad)

Figura 5.2: Algoritmo de Remoção.

O algoritmo da figura 5.2 implementa a operação de remoção. O primeiro argumento de *Remoção* é o índice da palavra eliminada, o segundo o estado atual. Se a palavra atual é a última da cadeia analisada a operação consiste em verificar se o estado atual é final.

Exemplo 16 A remoção é uma operação simples, como mostra o traço para a cadeia 5.6 segundo o autômato de padrões da figura 4.2.

$$tag(paris)=\{\mathsf{NP}\}$$

the	*paris*	*is...*
<the> (1-3)	*	
remoção	NP (1-6)	<*is*> (6–)...

□

5.2.3 Inserção

A operação de inserção estende a busca em profundidade agregando uma marca à cadeia na entrada. O algoritmo da figura 5.3 descreve a operação. A função *Transições* retorna todas as transições partindo de um estado.

Cada transição é representada por um registro com dois campos: *rótulo* e *destino*. O campo *destino* é o estado aonde chega a transição e o campo *rótulo* é uma marca para uma categoria ou um literal. Cada rótulo possível é agregado ao prefixo do padrão atual e continua-se a busca a partir da palavra com índice *i*.

Inserção(i,estado,prefixoPad)
para cada *tran* ∈ *Transições(estado)* **fazer**
Prefixo(i,tran.destino,prefixoPad & tran.rótulo)

Figura 5.3: Algoritmo de Inserção.

O primeiro argumento de *Inserção* indica o índice onde a nova palavra ou marca será inserida. Note-se que a inserção sempre é realizada na posição anterior à palavra atual (indexada por i); inserções na posição posterior à última palavra deveriam ser consideradas como um caso especial.

Exemplo 17 A tabela seguinte mostra como o algoritmo de casamento utilizando inserção pode gerar o *conselho* 5.10 para a cadeia 5.8. No autômato da figura 4.2, os rótulos que definem transições válidas a partir do estado 7 são <a> e <the>.

the	*cat*	*sees*	*dog...*
<the> (1 – 3)	N1 (3-6)	S2 (6-7)	*
			ins: <a> (7-) N1

□

Exemplo 18 Como foi discutido, a operação de inserção nem sempre pode gerar sentenças como *conselhos*. No caso da cadeia 5.12 claramente falta um verbo; a seguir apresenta-se um traço que mostra como a partir da informação no banco de padrões do autômato 4.2 pode-se obter a forma geral 5.13. A função *Transições* para o estado 6 define as marcas: <is>, <has>, S1 e S2.

the	*cat*	*a*	*dog ...*
<the> (1-3)	N1 (3-6)	*	
		ins: S2 (6-7) <a> (7-)	...

Com o mesmo autômato também podem-se gerar a partir de 5.12 as cadeias *the cat* <is> *a dog* e *the cat* <has> *a dog*. □

5.3 Marcação

Um dos experimentos executados para testar a implementação do algoritmo de casamento da figura 4.1 consistiu em tentar reconhecer as mesmas sentenças usadas para a extração

Marcação(i,estado,prefixoPad)
se *Vazio(tag(w_i))* **então**
para cada *tran* ∈ *Transições(estado)* **fazer**
Prefixo(i,tran.destino,prefixoPad & tran.rótulo)

Figura 5.4: Algoritmo de Marcação

de padrões sintáticos. Na primeira experiência varias sentenças foram rejeitadas, principalmente por conter palavras ausentes no léxico.

Um analisador deve enfrentar este tipo de problemas, pois o uso de palavras ausentes no léxico é muito comum. Um exemplo é o uso de nomes próprios; embora uma lista possa armazenar um grande número de nomes de pessoas, cidades ou países, é muito provável o uso de um nome não contemplado. Em [WRC97] discutem-se técnicas complexas para a detecção de nomes próprios em textos que são inviáveis no modelo de padrões. Uma alternativa para solucionar este tipo de exceções é tentar marcar a palavra ausente no léxico e continuar a busca por um padrão sintático no autômato.

Uma técnica de marcação consiste em um estudo da morfologia da palavra usando regras heurísticas. Um exemplo deste tipo de regra pode estabelecer que as palavras terminadas em *-ing* tem muitas chances de ser um particípio do presente. Outra alternativa mais simples é usar a informação contida nas transições do autômato.

No momento de analisar uma palavra ausente no léxico, o algoritmo de casamento está posicionado em um estado do autômato; as marcas rotulando as transições saindo desse estado podem ser utilizadas para tentar marcar a palavra. O algoritmo da figura 5.4 mostra a implementação da operação. Se a procura após a marcação tem sucesso, a marca atribuída à palavra tem boas chances de ser a correta. A falta de uma palavra x no léxico é indicada com $tag(x)=\emptyset$.

Exemplo 19 A palavra *born-again* não aparece no léxico de palavras *OTA*. Assim, a sentença 5.16 presente no *corpo LOB* não seria reconhecida. Por outro lado, essa sentença contribui com o padrão sintático 5.17 na coleta. Esse padrão sintático permitiria reconhecer 5.16 se *born-again* estivesse no léxico com marca AA.

the pastor had to be satisfied that each was a born-again christian (5.16)
<the> N1 <had> <to> <be> PP <that> <each> <was> <a> AA AL (5.17)

O esquema de marcação proposto pode usar o padrão sintático 5.17 para inferir que a marca de *born-again* deve ser AA. □

5.4 Algumas Considerações

A ordem de aplicação das operações descritas é importante. A inserção e remoção são mais "agressivas" que a substituição. Considerando a proximidade dos *conselhos* com a sentença pretendida pelo usuário a ordem preferida de aplicação das operações deve ser: substituição, remoção e inserção.

Outro fator importante é a quantidade de operações de cada tipo permitidas. Com um número arbitrário de inserções e remoções podem ser gerados *conselhos* de todo tipo, a maioria seguramente inúteis. No entanto, a substituição opera de uma forma mais controlada e pode ser aplicada com menos restrições.

Em [Ofl96] propõe-se a transposição de letras como uma quarta operação para reconhecimento robusto no nível morfológico. Esta operação pode ser utilizada também no esquema proposto nas seções anteriores no nível gramatical. A implementação simplesmente consiste no intercâmbio de duas palavras consecutivas na cadeia de entrada.

Em línguas como o português existem situações onde a ordem das palavras é importante. A posição dos pronomes oblíquos átonos, por exemplo, apresenta muitas dificuldades até para falantes nativos.

* *não convence-nos da tua culpa.* (5.18)

não nos convence da tua culpa. (5.19)

A próclise é obrigatória em português nas orações negativas [Lim96]. Esta regra estabelece que 5.18 não é gramatical. A partir da aplicação de transposição ao par de palavras *convence* e *nos* obtém-se a sentença 5.19. A transposição pode ser aplicada também em inglês, por exemplo, para solucionar o uso incorreto de *enough*.

5.5 Resumo

Em geral, a implementação de conselheiros gramaticais é uma tarefa complexa. A preocupação deste tipo de sistemas está em detectar a fonte do erro para disparar uma mensagem, em geral pré-determinada, que permita ao usuário descobrir o problema.

Para mostrar o poder de representação do modelo de padrões sintáticos propõe-se a implementação de um conselheiro gramatical. A novidade é a alternativa utilizada para fornecer as mensagens de erro. Em local de tentar descobrir a fonte do problema, procura-se aproximar a sentença pretendida pelo usuário propondo um conjunto de sentenças ou

esquemas chamados "conselhos". Essas opções guiam o usuário à sentença correta. Esta representação está baseada na hipótese que os erros de concordância envolvem pequenos deslocamentos de uma sentença.

O conselheiro é implementado sobre três operações de cadeias: substituição, remoção e inserção. A primeira operação trata, principalmente, os casos de uso incorreto de alguma flexão (** she watch television*). A remoção trata os problemas derivados do mau uso de palavras (** I must to go*). Por último, a inserção procura uma solução para a falta de uma palavra (** I want go*).

Da experiência realizada sobre alguns textos pré-analisados, percebeu-se que a ausência de palavras, tais como nomes próprios, no léxico era a causa da rejeição de sentenças. A alternativa escolhida é a marcação automática desse tipo de palavras.

Capítulo 6

Coleta de Padrões Sintáticos

> *" A system based upon the analysis of a corpus can uncover generalizations and weigh the import of different phenomena that are indicated by large data analysis but may not be apparent to a person attempting to hand-code a grammar."*
> — Eric Brill (em [Bri93])

Neste capítulo são descritas as atividades e decisões tomadas na preparação dos dados para a coleta automática de padrões sintáticos a partir de *corpos* de texto marcados. A disponibilidade de *corpos* marcados e de ferramentas como marcadores automáticos abre uma possibilidade interessante para a coleta de padrões sintáticos. Entretanto, a variedade de esquemas de marcação e a falta de um consenso para a definição de um formato comum são empecilhos que podem complicar a operação.

Por outro lado é definido o conjunto de marcas utilizado nos algoritmos e exemplos apresentados. Os algoritmos de casamento e o conselheiro gramatical requerem a implementação de funções por parte do léxico. Neste capítulo também é descrita a codificação definida para armazenar os dados no léxico. Por último, é descrita a implementação do `toTagset`, um programa que, dado um *corpo* marcado, permite traduzir marcas e extrair os padrões sintáticos.

6.1 Processamento de *Corpos*

Um *corpo* é um conjunto de documentos, geralmente textos, coletado, organizado e classificado seguindo algum critério e para um fim definido. O avanço da tecnologia no armazenamento de dados e nas comunicações impulsionou nos últimos anos diferentes entidades a promover e investir em projetos de desenvolvimento de recursos lingüísticos como léxicos, gramáticas e *corpos* [Col95].

O movimento do estruturalismo na Lingüística no começo do século foi suportado principalmente pelo estudo empírico da língua. Os pesquisadores deste período trabalha-

vam sobre coleções de textos ou listas de palavras das quais podiam inferir a estrutura da língua. Um exemplo referenciado na literatura é o trabalho de Boas [Bri93, MW97], que estudou as línguas dos indígenas americanos observando as similaridades estruturais.

Com a irrupção do movimento gerativista, o enfoque no estudo da Lingüística mudou rapidamente descartando sem discussão os métodos empíricos. Em [Cho65], Chomsky argumenta que o uso que um falante faz da língua, o desempenho, está necessariamente subordinado à competência. Seguindo esse raciocínio, argumentou que um conjunto de textos representa uma amostra reduzida do desempenho e não tem valor para o estudo de uma língua.

Na década de 80, a Lingüística Computacional retomou alguns dos objetivos do estruturalismo motivada por diferentes razões. O sucesso dos métodos estocásticos, por exemplo, foi uma das causas do novo interesse na coleta de *corpos*. Os sistemas envolvendo estatísticas, como os baseados em modelos HMM, utilizam *corpos* marcados para extrair dados que permitam fixar seus parâmetros [KS96, Sch94, Vil95]. Para o treinamento de parâmetros são usados *corpos* marcados onde cada palavra é acompanhada por uma marca indicando a sua categoria. Os *corpos* também são utilizados na avaliação, melhoramento e comparação de sistemas de PLN em geral. A Lingüística aplicada retomou o uso de *corpos*, sobretudo para implementar técnicas de ensino de línguas baseadas em exemplos.

Os *corpos* servem também como fonte de dados para a prova e desenvolvimento de hipóteses sobre fenômenos lingüísticos. Outro exemplo de uso é a compilação de dados para a construção de dicionários; diversas editoras estão investindo na criação de *corpos* com esse fim. O projeto *The Bank of English*, por exemplo, resultou na produção de um dicionário [Har96, Jär94] onde cada termo é explicado com exemplos obtidos a partir da freqüência observada dos mesmos no *corpo*. O interesse no trabalho apresentado nesta dissertação está focalizado nos *corpos* marcados que oferecem uma excelente fonte para a extração de padrões sintáticos.

6.1.1 Taxonomia

Os *corpos* são classificados dependendo da origem e qualidade da informação armazenada. Em [KS96] propõe-se a seguinte classificação:

- Segundo o tipo de texto
  - Balanceado: consiste em textos de diferentes gêneros uniformemente distribuídos. Um exemplo é o *Brown Corpus*.
  - Piramidal: agrupa grande quantidade de textos de poucos gêneros, e uns poucos textos de uma grande variedade. O *Wall Street Journal Corpus* é um exemplo.

  - Oportunista: o princípio é tomar todos os textos disponíveis. O projeto *The Bank of English*, por exemplo, recebe textos através de correio eletrônico daqueles que quiserem colaborar na compilação de dados.

- Segundo o tipo de marcação
  - Sem Formato: consiste de texto pré-processado onde apenas foram eliminados os caracteres de controle.
  - Marcado ou Rotulado: cada palavra do *corpo* é acompanhada de uma categoria. Por exemplo o *corpo LOB*.
  - Analisado: o texto é rotulado com a estrutura sintática das sentenças, em geral usando uma notação baseada em parênteses. Um exemplo é o SUSANNE.

- Segundo o uso
  - Treinamento: usados para "sintonizar" os parâmetros de modelos estatísticos.
  - Teste: utilizados como parâmetro para a avaliação de sistemas estocásticos.
  - Avaliação: utilizados para a avaliação de modelos, não necessariamente estocásticos.

Os *corpos* marcados são classificados em *verticais* ou *horizontais* segundo o formato utilizado na organização interna dos dados. O *SUSANNE* é um exemplo de *corpo* vertical onde cada linha é reservada para uma palavra, como mostra a figura 6.1.

```
A01:0010a -YB <minbrk>-[Oh.Oh]
A01:0010b -AT The the [O[S[Nns:s.
A01:0010c -NP1s Fulton Fulton [Nns.
A01:0010d -NNL1cb County county .Nns]
A01:0010e -JJ Grand grand .
A01:0010f -NN1c Jury jury .Nns:s]
A01:0010g -VVDv said say [Vd.Vd]
A01:0010h -NPD1 Friday Friday [Nns:t.Nns:t]
```

Figura 6.1: Formato do *Corpo SUSANNE*

O *corpo LOB*, por exemplo, está disponível em duas versões; a figura 6.2 mostra um exemplo da versão horizontal.

```
A01   2 ^ *'_*' stop_VB electing_VBG life_NN peers_NNS **'_**' ._.
A01   3 ^ by_IN Trevor_NP Williams_NP ._.
A01   4    ^ a_AT move_NN to_TO stop_VB \OMr_NPT Gaitskell_NP from_IN
A01   4 nominating_VBG any_DTI more_AP labour_NN
A01   5 life_NN peers_NNS is_BEZ to_TO be_BE made_VBN at_IN a_AT meeting_NN
A01   5 of_IN labour_NN \OMPs_NPTS tomorrow_NR ._.
A01   6    ^ \OMr_NPT Michael_NP Foot_NP has_HVZ put_VBN down_RP a_AT
A01   6 resolution_NN on_IN the_ATI subject_NN and_CC
```

Figura 6.2: Formato do *Corpo LOB*

6.1.2 Conjunto de Marcas

A partir do uso maciço de recursos lingüísticos surgiram nos últimos anos certos problemas relacionados ao intercâmbio da informação e das convenções utilizadas na formatação e marcação de *corpos*. Em 1988, a ACL (*Association for Computational Linguistics*), ALLC (*Association of Literary and Linguistic Computing*) e ACH (*Association for Computing in the Humanities*) coordenaram a criação do TEI (*Text Encoding Initiative*) com o objetivo de estabelecer padrões para regulamentar a definição, coleta e intercâmbio de *corpos*. Embora o TEI já tenha definido guias [Ide94], cada projeto que envolve *corpos* e marcações ainda define seu próprio esquema que, em geral, não se adapta diretamente às novas aplicações. A razão principal deste desentendimento é que o processamento de *corpos* é uma área nova e ainda não existe uma noção clara do tipo de aplicações relacionadas.

Outros empecilhos são o alto custo dos projetos envolvidos na coleta de textos e marcações, e o reaproveitamento de *corpos* antigos. O custo dos projetos incorpora a necessidade de investidores particulares; a participação de entidades não comprometidas com a pesquisa é muitas vezes negativa para o intercâmbio de informação.[1]

O *corpo* coletado pela universidade *Brown* no final da década de 60 foi o trabalho pioneiro na área e estabeleceu um modelo padrão; outros projetos posteriores [MMM93, Sam95] definiram marcações derivadas.

Segundo a qualidade e o método usado para a marcação, distinguem-se duas gerações de *corpos* marcados. Na primeira geração, os *corpos* são da ordem de 100 mil palavras e incluem textos obtidos de jornais, livros e outras fontes de língua escrita. A marcação é desenvolvida em grande parte manualmente e está baseada em conjuntos de marcas grandes; o *SUSANNE*, por exemplo, define mais de 400 marcas.

A segunda geração se caracteriza por *corpos* que incluem uma variedade maior de

[1] Os investidores particulares do *British National Corpus*, por exemplo, impuseram a restrição de que o *corpo* seja apenas distribuído no âmbito da Comunidade Econômica Européia. Conseqüentemente, ele não pôde ser utilizado neste trabalho.

APPGf	*her*
APPGh1	*its*
APPGh2	*their*
APPGi1	*my*
APPGi2	*our*
APPGm	*his*
APPGy	*your*

Tabela 6.1: Pronomes Possessivos no *SUSANNE*

fontes incorporando como novidade a língua falada. Em geral, utilizam-se marcadores automáticos baseados em modelos estatísticos com esquemas da ordem de 50 marcas. Outra característica da segunda geração é o tamanho; o *corpo British National Corpus* contém mais de 100 milhões de palavras.

6.2 Casos de Estudo

Ao longo do trabalho foram realizados dois experimentos sobre os *corpos* marcados *SUSANNE* e *LOB*. A escolha destes *corpos* foi determinada diretamente pela disponibilidade.

6.2.1 O *Corpo SUSANNE*

O *corpo* marcado *SUSANNE* forma parte de um projeto que pretende desenvolver um esquema de marcação abrangente para a gramática do inglês. O autor propõe um método para representar todos os aspectos do inglês susceptíveis a uma notação formal. Esse objetivo constitui um empecilho para a coleta de padrões sintáticos.

O projeto do *SUSANNE* tomou um subconjunto do *Brown Corpus* estendido com uma marcação mais detalhada, com indicações da estrutura lógica das sentenças, tais como limites de um sintagma.

O *corpo* consiste de 64 arquivos, cada um contendo aproximadamente 2.000 palavras anotadas com uma categoria. Os arquivos são classificados de acordo com a fonte utilizada em: artigos de jornais, escrita formal (cartas, memórias, biografias, etc), artigos científicos e ficção.

Como foi mostrado na figura 6.1, o formato deste *corpo* é vertical. Cada linha em um arquivo tem 6 campos separados por caracteres de tabulação. Esses campos representam a palavra, uma marca e a raiz da palavra, entre outros. O *SUSANNE* define um conjunto de 352 marcas, sem contar aquelas utilizadas para representar diretivas de estilo como

tipo de letra ou espacejamento. Este critério exige das marcas muita informação, e o esquema termina sendo confuso. Um exemplo é o esquema usado para pronomes possessivos mostrado na tabela 6.1.

O mesmo esquema mostrado na tabela é utilizado para as outras categorias, como por exemplo substantivos (NN...), adjetivos (JA... e JB...) e verbos (VV...). As flexões do verbo *to be* e os auxiliares têm suas próprias marcas.

Uma característica do *SUSANNE* é o tratamento das frases idiomáticas como *as soon as possible* ou *as well as*. As marcas usadas neste caso indicam a função gramatical da frase. Cada componente da frase é assinalado com um par de números, onde o primeiro é o tamanho da frase e o segundo a posição da palavra na mesma. Por exemplo, a frase adjetiva *all rigth* é marcada:

- *all*: JA21
- *rigth*: JA22

6.2.2 O *Corpo LOB*

O *corpo Lancaster-Oslo/Bergen Corpus (LOB)* é uma coleção de um milhão de palavras de textos atuais. Este *corpo* é considerado a contrapartida britânica do *Brown*. O *LOB* contém 500 amostras de aproximadamente 2.000 palavras cada uma, distribuídas em 15 categorias de textos, incluindo: artigos de jornais, editoriais de revistas, novelas, artigos científicos e ficção.

A distribuição do *corpo* provê uma versão vertical e outra horizontal. Cada palavra é acompanhada por uma marca de um conjunto de 30 marcas. O esquema de marcação do *LOB* é mais compacto que o usado no *SUSANNE*.

As marcas utilizadas são derivadas de um conjunto básico formado por prefixos que determinam a categoria principal. Os sufixos de uma marca modelam outras características como caso e número. O *corpo LOB* também modela as frases idiomáticas utilizando a marca *ditto*. Por exemplo, a frase *each other* é marcada como:

- *each*: PPLS
- *other*: PPLS"

A marca PPLS representa um pronome reflexivo complexo, as aspas (") indicam que *other* é uma extensão de *each*. Em [JAGL86] descreve-se com detalhe a marcação do *LOB*.

6.3 Coleta de Padrões

A abordagem mais simples para implementar a extração de padrões sintáticos a partir de um *corpo* marcado consiste na implementação de um filtro que possa reter as marcas eliminando o texto. Diferentes aspectos dos conjuntos de marcas definidos na maioria dos *corpos* marcados podem dificultar esta tarefa. Os conjuntos de marcas utilizados nas marcações de *corpos* tentam associar a cada palavra a maior quantidade de informação possível. O ideal é fornecer marcas para todas as classes de palavras com diferentes comportamentos gramaticais [GLB94]. Este objetivo, como será mostrado, não é totalmente compatível com a informação necessária para representar padrões sintáticos.

Outro problema está relacionado com a definição da unidade mínima para segmentação e marcação do *corpo*. A maioria dos esquemas define as palavras e os símbolos de pontuação como as unidades mínimas sujeitas à marcação, mas muitas vezes não fica claro o que é considerado como uma palavra. Um exemplo crítico são as palavras compostas com hífen. A marcação do *SUSANNE*, por exemplo, não é homogênea com este tipo de palavras, como mostram as marcas para *well-known* e *anglo-saxon*. No caso de *well-known*, *SUSANNE* considera duas palavras: *well* como um advérbio (RR) e *known* como o particípio do verbo *know* (VVNv).[2] Por outro lado, *anglo-saxon* é marcada como uma palavra com a marca de adjetivo (JJ).[3]

No léxico *OTA* da Universidade de *Oxford* todas as palavras compostas são marcadas como uma unidade; palavras derivadas do latim como *ad hoc* ou *anno domini* têm uma entrada própria. No caso de *well-known*, a marca corresponde a um adjetivo sem flexão para as formas comparativa e superlativa (OA).

Os seguintes exemplos, tomados de *corpos* de textos, mostram algumas das diferenças entre os objetivos perseguidos pelos esquemas de marcação e a definição de padrões sintáticos para representar informação de concordância:

- O *SUSANNE* incorpora informação semântica na marcação. Por exemplo NP1c é utilizada para rotular nomes de países.
- O projeto *Penn Treebank* [MMM93] define apenas uma marca para os adjetivos, sem distinguir entre atributivos e predicativos (vide 2.1.3).
- O *British National Corpus* [GLB94] utiliza uma marca para representar artigos, incluindo a partícula negativa *no* (vide 2.1.2).
- O esquema de marcas de *The Bank of English* [HB] não distingue substantivos incontáveis dos contáveis (vide 2.1.2).

[2]Nas linhas J03:1670a-c.
[3]Na linha G01:1460h.

Outra diferença importante é a representação da informação literal; a maioria dos esquemas de marcação define alguma marca com apenas uma palavra associada. O exemplo típico é a partícula *to* que a maioria das vezes é marcada com TO.[4] As marcas para definir padrões sintáticos devem considerar unicamente aspectos relativos à relação superficial, evitando ambigüidades e a representação de informação semântica ou contextual.

6.4 Marcas para Padrões Sintáticos

Nesta seção é apresentado um conjunto de marcas utilizado para avaliar os algoritmos de reconhecimento e aconselhamento. O projeto deste conjunto põe ênfase na representação dos fenômenos de concordância definidos em 2.1. Obviamente, com este conjunto não será possível capturar todos os fenômenos sintáticos superficiais em inglês, mas a intenção é mostrar que um modelo mais abrangente só deve estender o conjunto para enriquecer o banco de padrões.

As seguintes marcas representam os substantivos:

N1 substantivo comum no singular (*dog* e *child*).

N2 substantivo comum no plural (*dogs* e *children*).

NU substantivo incontável (*water*, *milk* e *beggary*).

NP substantivo próprio (*Robert* e *Brazil*).

A distinção entre substantivos no plural e singular, assim como entre contáveis e incontáveis, representa a informação necessária para definir as flexões dos artigos e verbos relacionados (vide 2.1.2).

As seguintes marcas representam os adjetivos e advérbios:

A1 Adjetivo em geral (não considerado pelas outras marcas).

AA adjetivo atributivo (*present-day* e *aged*).

AP adjetivo predicativo (*alive* e *broke*).

AL adjetivo que utiliza as partículas *more* e *most* para as formas comparativa e superlativa respectivamente (*expensive* e *interesting*).

AC adjetivo na forma comparativa (*smaller* e *cheaper*).

AS adjetivo na forma superlativa (*smallest* e *cheapest*).

BB advérbio.

[4]The *Bank of English* e *Penn Treebank* usam a marca TO para marcar a partícula *to*.

A constituição das formas comparativa e superlativa é levada em conta, para evitar as frases agramaticais do estilo *more smaller* e *most smallest*.

As seguintes marcas são utilizadas para representar os verbos:

V1 verbo intransitivo infinitivo (*appear* e *arise*).

V2 verbo transitivo infinitivo (*mail* e *foresee*).

S1 verbo intransitivo na terceira pessoa do singular (*appears* e *arises*).

S2 verbo transitivo na terceira pessoa do singular(*mails* e *foresees*).

G1 verbo intransitivo no particípio presente (*appearing* e *arising*).

G2 verbo transitivo no particípio presente (*mailing* e *foreseeing*).

H1 verbo intransitivo conjugado no passado (*appeared* e *foresaw*).

H2 verbo transitivo conjugado no passado (*mailed* e *arose*).

P1 verbo intransitivo no particípio passado (*appeared* e *arisen*).

P2 verbo transitivo no particípio passado (*mailed* e *foreseen*).

As cinco formas que um verbo pode tomar em inglês são subdivididas em duas categorias segundo a transitividade. Cada marca restringe a relação do verbo com os substantivos, auxiliares (*do, have* ou verbos modais) e objetos.

Para símbolos e números são utilizadas as seguintes marcas:

NO Número exceto *one* (considerado como literal).

NF Número fracionário.

SS Símbolo em geral, tal como moedas e medidas.

Os símbolos de pontuação são considerados palavras, e a maioria deles não oferece problemas de ambigüidades, exceto o ponto e o hífen. O ponto pode ser utilizado em vários contextos, além do final da sentença, e muitas vezes é difícil reconhecer sua verdadeira função.

A citation from Conservation Commissioner Salvatore A. Bontempo credits... (6.1)

Em 6.1 o ponto depois da abreviatura *A* do nome pode ser facilmente confundido com um ponto final, pois a palavra seguinte começa com letra maiúscula. Um problema similar surge com o uso do hífen, que é utilizado como indicador de começo de diálogos e parte de palavras compostas. A razão de definir marcas para os símbolos de pontuação foi prática: facilitar a programação dos algoritmos e generalizar o modelo. As marcas a seguir correspondem aos símbolos de pontuação:

YF ponto final de uma sentença.

YC vírgula.

YX símbolo de exclamação.

YQ símbolo de interrogação.

YE reticências.

YS ponto e vírgula.

YN dois pontos.

Numerosas palavras do inglês podem ser definidas com mais de uma marca deste conjunto básico. Por exemplo, *burst* pode corresponder a um verbo com a marca H1, como em 6.2, ou a um substantivo com a marca N1, como em 6.3. A maioria dos *corpos* marcados diferencia estas duas possibilidades eliminando a necessidade de detectar a ambigüidade. Porém, a palavra *burst* pode também aparecer marcada com H2 como em 6.4.

One of my tyres **burst**. (6.2)

It is like a **burst** *of flame.* (6.3)

The river **burst** *its banks.* (6.4)

Este último caso não se diferencia facilmente da marca H1, pois trata-se de duas subcategorias gramaticais[5] que não são registradas por todos os *corpos* marcados. Os *corpos SUSANNE* e *LOB*, por exemplo, não fazem essa diferença, definindo uma marca para aqueles verbos que podem ser transitivos e intransitivos.

Do ponto de vista da concordância, a melhor solução para este tipo de palavras é definir uma entrada no léxico para cada par palavra-marca, no exemplo $tag(burst)$= {N1,H1,H2}. Porém, em *corpos* que não identificam esta informação, seria preciso inferi-la, o que é difícil e custoso.

Uma outra opção mais simples e prática para o problema é definir marcas especiais que contemplem essas situações. As marcas especiais a seguir são acrescentadas ao conjunto básico definido acima para solucionar problemas de ambigüidade:

NN substantivo contável ou incontável (*water*).

VV verbo intransitivo ou transitivo em infinitivo (*burn*).

GG verbo intransitivo ou transitivo no particípio presente (*burning*).

SS verbo intransitivo ou transitivo na terceira pessoa do singular (*burns*).

HH verbo intransitivo ou transitivo conjugado no passado (*burned* ou *burnt*).

[5]Chama-se de categoria gramatical aos grandes conjuntos tais como: verbos, substantivos e pré-determinadores; divisões internas destes conjuntos são chamadas de subcategorias.

PP verbo intransitivo ou transitivo no particípio passado (*burned* ou *burnt*).

Estas marcas solucionam o problema da ambigüidade mas diminuem o poder de expressão dos padrões sintáticos.

<the> NN <is> A1 YF (6.5)

the milk is cold . (6.6)

Por exemplo, a marca NN no padrão 6.5 permite reconhecer cadeias onde o substantivo necessariamente possa assumir o papel de contável e incontável, embora na sentença particular de onde o padrão foi extraído seja válido utilizar um substantivo que é somente incontável ou vice-versa. Assim, 6.6 não é reconhecida com esse padrão pois $tag(milk)=$ {NU}.

Por último, a marca LL é utilizada para aquelas palavras que são consideradas como literais. Os pronomes e os artigos, por exemplo, caem nessa categoria. Esta marca não é necessária no processo de análise ou aconselhamento, nem forma parte de padrão sintático nenhum, mas é útil no processo de coleta.

Deve-se notar que símbolos como aqueles usados em fórmulas matemáticas e outros fenômenos superficiais não são capturados com este conjunto de marcas; a representação deste tipo de informação requer um estudo mais profundo da língua. O conjunto de marcas definido nesta seção atinge um subconjunto representativo do inglês que permite verificar a viabilidade do modelo.

6.5 Preparação do Léxico

Os padrões sintáticos combinam informação léxica e gramatical em um mesmo esquema. Porém, os algoritmos de casamento e aqueles que implementam as diferentes operações do conselheiro precisam consultar a informação no léxico. A operação mais sofisticada é a consulta por palavras correlatas requerida pela operação de substituição do conselheiro.

O esquema utilizado para codificar o tipo de informação necessária para implementar essas consultas é descrito em [KLS98]. A definição do léxico, por outro lado, deve modelar o conjunto de palavras correlatas usado pelo conselheiro. Dada uma palavra, o conjunto da suas correlatas captura a noção de flexão restrita e guiada pelos objetivos do conselheiro.

As regras morfológicas que permitem manipular sufixos para obter as diferentes flexões de uma palavra são codificadas com cada palavra de uma forma eficiente e elegante, fácil de atualizar e que pode ser armazenada de forma compacta usando um autômato finito. Este esquema foi originalmente projetado para o português, mas mostrou ser adequado para outras línguas, inclusive o inglês. Definem-se dois tipos de entradas:[6] sintéticas e

[6]Esta implementação de analise morfológica é chamada em inglês de ***cut-and-paste***.

analíticas.

As entradas *sintéticas* correspondem às palavras canônicas juntamente com regras para obter o conjunto de correlatas. Por exemplo, as entradas sintéticas para *university* são:

university>N1:*0* (6.7)

university>N2:*1ies* (6.8)

O sufixo após o separador ">" codifica uma regra que indica como obter uma flexão do grupo de palavras correlatas. A entrada sintética 6.7 indica que para obter a forma de *university* marcada com N1 não é preciso fazer nenhuma operação. A entrada 6.8 significa que para obter a flexão N2 (*universities*) deve-se tirar a última letra e concatenar o sufixo *ies*.

As entradas *analíticas* correspondem a palavras em geral junto com regras para obter a forma canônica correspondente. As entradas analíticas para *university* e *universities* são:

university:N1:*0* (6.9)

universities:N2:*3y* (6.10)

As regras para obter as formas canônicas operam da mesma forma que com as entradas sintéticas. A entrada 6.9 indica que para obter a palavra canônica do conjunto de palavras correlatas ao qual pertence *university* marcada com N1 não é necessário realizar nenhuma operação. A entrada 6.10 codifica a regra que indica que para obter a forma canônica associada com *universities* com a marca N2 devem-se tirar as três últimas letras da palavra e concatenar a letra *y*.

As cadeias codificando as entradas sintéticas e analíticas podem ser armazenadas eficientemente em autômatos finitos. As regras de derivação são codificadas como sufixos que podem ser compartilhados por um número grande de palavras. Por exemplo, a maioria dos substantivos terminados em *y* compartilhará os sufixos tanto para as entradas sintéticas como analíticas, pois a formação das diferentes flexões segue o mesmo padrão.

Cabe notar que o esquema está baseado na regularidade morfológica de uma língua; palavras irregulares geraram codificações peculiares. As entradas 6.11, 6.12 e 6.13 correspondem às entradas sintéticas para o adjetivo *good*, considerando como formas correlatas as flexões para comparativo e superlativo.

good>A1:*0* (6.11)

good>AC:*4better* (6.12)

good>AS:*4best* (6.13)

Esses exemplos são pouco comuns, e a vantagem obtida com a maioria das palavras justifica o esquema. Este exemplo mostra também que as irregularidades são tratadas de maneira uniforme (embora signifique um aumento no tamanho do autômato).

Esta codificação tem a vantagem de manter relacionadas as formas derivadas de uma palavra, o que permite uma implementação simples de operações tais como listar todas as formas correlatas de um verbo, necessária para implementar a função *corr*().

O processo de construção do léxico foi iniciado com o vocabulário *OTA*, coletado pela Universidade de *Oxford*, que contém 69.800 palavras diferentes distribuídas em 96.480 entradas.

O primeiro passo consistiu na tradução do conjunto de marcas do *OTA* ao conjunto definido na seção anterior, uma tarefa relativamente fácil. Os passos seguintes consistiram na criação das entradas sintéticas e analíticas. Esta tarefa foi realizada considerando a informação das marcas e conhecimento mínimo da morfologia do inglês.

O processamento foi realizado em várias etapas eliminando as pequenas irregularidades, seguindo uma técnica de prova e erro. Por exemplo, para obter a entrada sintética correspondente ao verbo *love* para a terceira pessoa no singular tentou-se com a palavra *loves*; como a mesma aparece no léxico rotulada com SS foi criada a entrada *love*>SS:*0s*. Porém, se a mesma opção é tentada com o verbo *cry* obtém-se *crys*, que não corresponde a uma entrada rotulada com S1. Outras irregularidades foram tratadas dessa forma, como por exemplo as repetições e inserções de letras (*submit, submitted, picnic, picnicking*), até obter um pequeno conjunto de palavras irregulares que foram tratadas manualmente.

As seguintes são as principais estatísticas relacionadas com a construção do léxico e do autômato para representá-lo:

- 220.225 entradas (entre sintéticas e analíticas).
- 3.129.188 letras (considerando os separadores e regras)
- 38.167 estados
- 85.901 transições

6.6 Extração de Padrões

Para a primeira experiência de coleta de padrões sintáticos utilizou-se o *corpo* marcado *SUSANNE*. O tamanho relativamente pequeno do *corpo* permitiu repetir diferentes experimentos e assim ganhar experiência. A extração final de padrões foi realizada sobre o *corpo LOB*.

Como já foi discutido, a falta de uma marcação consensuada e a representação de informação irrelevante à coleta de padrões sintáticos podem atrapalhar a tarefa. A complexidade do processo de extração motivou o desenvolvimento do programa `toTagset`. Para implementar este programa foi utilizada a linguagem de programação AWK [Rob96], que oferece várias facilidades para a manipulação de cadeias.

O programa de tradução de marcas e extração de padrões sintáticos deve resolver alguns dos problemas enunciados em 6.3. Outro problema é a tradução de certas marcas ou símbolos que não são contemplados pelo conjunto de marcas definido. O desenvolvimento de `toTagset` visou superar essas diferenças para automatizar a extração de padrões sintáticos a partir de um *corpo* marcado.

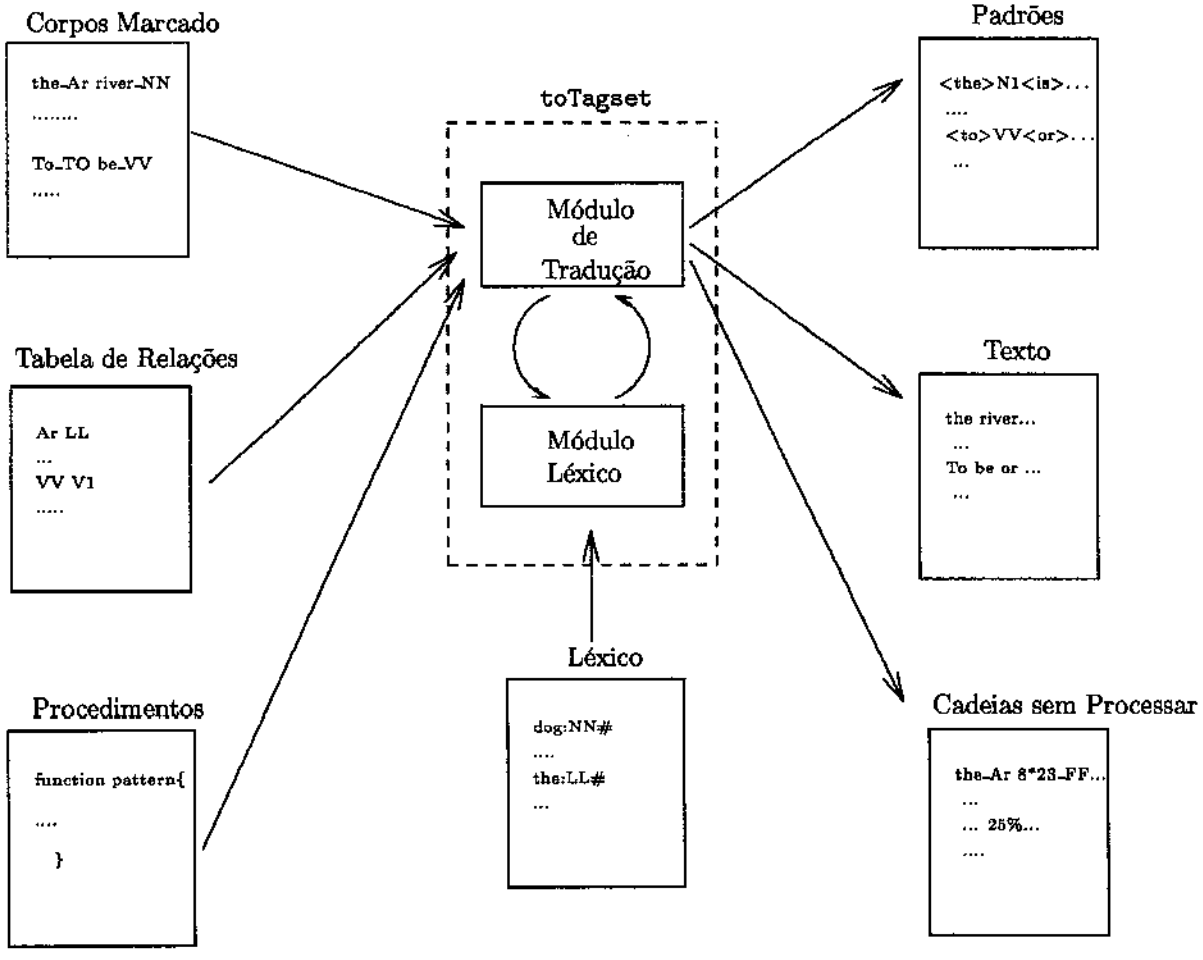


Figura 6.3: Arquitetura do `toTagset`.

Como mostra a figura 6.3, a arquitetura do `toTagset` é definida sobre dois módulos diferenciados: tradução e léxico. Para implementar o módulo léxico utilizou-se a biblioteca *Reduced*, especialmente programada para a manipulação de autômatos finitos (para uma

descrição ver seção 7.1). A entrada do programa consiste de:

1. um *corpo* marcado;
2. uma tabela de relações de marcas; e
3. procedimentos que definem o formato do *corpo* e a saída.

O usuário pode personalizar o processo de extração e de tradução a partir da definição da tabela de relações e dos procedimentos. O formato da tabela de relações é de duas colunas obrigatórias; as restantes podem ser utilizadas para comentários. A primeira coluna contém as marcas originais do *corpo*. A segunda coluna é uma *instrução* que indica como deve ser traduzida a marca da primeira coluna na mesma linha. A entrada 6.14 exemplifica a instrução mais simples.

NN1c N1 Substantivo no singular (6.14)

A instrução em 6.14 estabelece que qualquer ocorrência da marca NN1c deve ser substituída por N1.

Os *corpos* em geral incorporam cadeias de palavras das quais não devem ser extraídos padrões, como por exemplo as manchetes de jornais. Por outro lado, também podem existir palavras ou símbolos irrelevantes que podem ser eliminados de uma sentença sem alterar a gramaticalidade, como por exemplo aspas ou indicadores da fonte da letra. Para manipular este e outro tipo de exceções, `toTagset` introduz variáveis que descrevem marcas fora do conjunto definido para representar operações especiais.

Literal marca cujas palavras associadas devem ser incluídas no padrão como literais (<the>).

Invalida marca que invalida a extração do padrão sintático.

SemMarca marca sem tradução no padrão extraído, mas que não invalida o padrão (aspas).

MarcaFinal marca para o final do padrão.

MarcaHífen marca especial para processar hífens ou itens semelhantes.

Para solucionar os problemas resultantes da diferença de granularidade entre o conjunto de marcas do *corpo* e o conjunto destino da tradução, `toTagset` define instruções mais sofisticadas.

JB ** AA AP A1 AL ** Adjetivo (6.15)

NN @@ N1 NU NN @@ Substantivo comum (6.16)

Por exemplo, a instrução 6.15 foi utilizada na tabela para implementar a coleta de padrões do *SUSANNE*. Um dos problemas neste *corpo* é que a marca para os adjetivos é muito geral. A instrução especifica uma consulta ao léxico para procurar uma das marcas da lista; se nenhuma ou mais de uma forem válidas, o padrão sintático correspondente é descartado e continua-se com o próximo padrão.

Um dos empecilhos na marcação do *corpo LOB* é a falta de detalhe na marcação de substantivos; apenas a marca NN é usada para rotular a maioria dos substantivos. Existe uma diferença entre este problema e o anterior com o *corpo SUSANNE*. Para o caso dos adjetivos procura-se uma marca única da lista, mas no caso específico dos substantivos existem alguns que podem ser considerados tanto contáveis como incontáveis.

A instrução 6.16 utiliza a última marca na lista se a palavra for marcada com todas anteriores. A palavra *milk*, por exemplo, apenas é marcada como incontável e é traduzida com a marca NU. Porém a palavra *water* pode adotar as duas possibilidades e é traduzida como NN. Se nenhuma marca na lista é válida, o padrão sintático é rejeitado e continua-se com a próxima cadeia no *corpo*.

6.7 Resumo

A alternativa proposta para a configuração do ambiente de padrões é a coleta a partir de *corpos* marcados. Em um primeiro momento parece uma tarefa simples, porém a experiência com *corpos* de tamanho pequeno e mediano mostrou a complexidade por trás dessa tarefa.

A maioria dos problemas tem relação direta com o tipo de marca utilizada pelo *corpo*. Embora numerosas entidades tenham financiado nos últimos anos diferentes projetos para estabelecer esquemas de marcação padrão, as diferenças ainda persistem.

O autor do esquema de marcação do *corpo SUSANNE*, por exemplo, apresenta uma marcação que pretende copiar os esquemas usados para a classificação de amostras em ciências como a biologia. Essa idéia resulta em um conjunto de mais de 400 marcas, com uma notação confusa. Visando superar esses problemas, primeiro foi definido um conjunto de marcas próprio seguindo as restrições impostas pelo fenômeno de concordância.

A partir do conjunto de marcas foi construído o léxico, implementado também sobre uma estrutura baseada em autômatos. Na codificação utilizada, com cada palavra é armazenada a informação necessária para determinar as flexões e palavras canônicas. Essas operações são necessárias para a implementação da operação de substituição no conselheiro gramatical. Por último é descrito o programa `toTagset` que permite traduzir as marcas de um *corpo* marcado para um conjunto próprio e extrair os padrões sintáticos seguindo as instruções confeccionadas pelo usuário.

Capítulo 7

Implementação

Neste capítulo apresenta-se uma proposta para a implementação eficiente e compacta do modelo de padrões sintáticos. O trabalho de implementação foi desenvolvido a partir de um conjunto de rotinas programadas para manipular autômatos finitos acíclicos projetados especialmente para armazenar palavras de um vocabulário. As extensões feitas sobre o esquema básico são apresentadas de forma incremental.

A primeira alternativa consiste de um autômato onde cada padrão sintático é representado por uma seqüência de transições. Essa representação simples é estendida para solucionar diferentes problemas até chegar a um esquema mais sofisticado com três níveis de autômatos.

7.1 Biblioteca *Reduced*

A biblioteca *Reduced* foi desenvolvida como parte do projeto *Dicio* no Instituto de Computação da UNICAMP, e consiste de um conjunto de rotinas para criar e atualizar grandes autômatos finitos acíclicos minimais determinísticos, com o objetivo primário de armazenar cadeias de letras.

A biblioteca foi implementada usando a linguagem de programação MODULA-3 [Har92, Nel91] desenvolvida no *Systems Research Center* (SRC) da *Digital Equipment Corporation*. MODULA-3 continua o paradigma de programação iniciado pelas linguagens PASCAL e MODULA-2. A especificação desta linguagem inclui a maioria das características desejadas em uma linguagem de programação moderna, tais como orientação a objetos, programação modular, procedimentos genéricos, *threads*, coleta de lixo, exceções e definição de subtipos. MODULA-3 implementa esses conceitos com um sistema de verificação de tipos forte, assegurando uma programação robusta e elegante. Os compiladores implementados pelo SRC são de distribuição livre e estão disponíveis para várias plataformas.

A biblioteca *Reduced* provê um tipo de dado abstrato cuja representação está baseada

na teoria de autômatos finitos binários descrita em [KLS95b]. Esta teoria é definida para a implementação de autômatos com uma grande quantidade de estados e relativamente poucas transições válidas. Na representação interna proposta, cada estado é implementado como uma lista (pequena) de *subestados*. De cada subestado apenas partem duas transições, uma correspondente ao autômato original e outra ligando o subestado ao próximo. Esta representação evita a necessidade de representar transições inválidas e oferece um modelo adequado para implementar a operação de minimização de autômatos.

Para o nível do usuário, a biblioteca fornece uma classe que representa um tipo de dado abstrato com todas as componentes e funções necessárias para manipular um autômato finito. Os estados do autômato são numerados; o índice maior corresponde ao estado inicial, também chamado raiz. O número de um estado deve ser manipulado pelo usuário da biblioteca apenas como uma referência. Existem dois estados pré-definidos:

- estado nulo
- estado raiz

O estado nulo é definido para reforçar a propriedade de determinismo; todas as transições que não formam parte de nenhuma cadeia da linguagem reconhecida têm como destino esse estado. Cabe destacar que as transições partindo do estado nulo são laços. O estado nulo é o único que viola a propriedade de aciclicidade.

Um *sufixo* de um estado é uma cadeia formada pelos rótulos de um caminho de transições começando nesse estado e terminando em um estado final. Segundo o definido, o conjunto de sufixos do estado nulo é vazio pois não é possível alcançar nenhum outro estado a partir dele. No caso de um estado final sem transições partindo dele, o conjunto de sufixos contém apenas a cadeia vazia.[1] A implementação assegura que o autômato é mínimo, no sentido que dois estados diferentes não têm o mesmo conjunto de sufixos. Um *prefixo* de um estado é uma cadeia obtida seguindo um caminho de transições que conduz do estado inicial até esse estado.

Cada transição é implementada como um registro com dois campos: *rótulo* e *destino*. Por razões de implementação interna, o *rótulo* consiste de apenas um *byte*. O campo *destino* é um número indexando o estado destino da transição.

Um conjunto básico de métodos permite criar um autômato e atualizar as cadeias da linguagem reconhecida. Outro conjunto de métodos permite consultar, por exemplo, qual é o estado raiz, quantas transições partem de um estado, etc.

A biblioteca *Reduced* fornece métodos para implementar enumerações sobre as componentes de um autômato. Por exemplo, o método *EnumStrings* enumera todos os sufixos de

[1] A cadeia vazia é denotada na literatura pelas letras gregas λ ou ϵ. Um autômato finito determinístico reconhece a cadeia vazia se e somente se o estado inicial pertence ao conjunto de estados finais.

um estado. O usuário pode definir procedimentos que são executados durante o processo de enumeração em pontos pré-definidos. Os procedimentos fornecidos pelo usuário podem também levantar exceções para limitar o espaço da enumeração ou terminar a operação.

Em uma primeira experiência tentou-se utilizar a estrutura de dados fornecida por *Reduced* com poucas mudanças para armazenar padrões sintáticos. Simplesmente, definiu-se uma subclasse que estendeu as rotinas necessárias para manipular padrões sintáticos. A figura 7.1 exemplifica esta idéia.

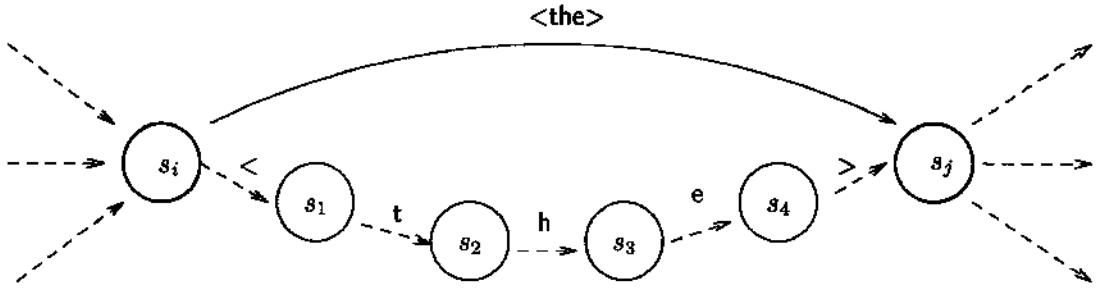


Figura 7.1: Autômato de padrões: Versão 1

Na figura os estados i e j devem ser os únicos visíveis ao usuário do banco de padrões sintáticos. A transição rotulada com o literal <the> é representada no nível da biblioteca *Reduced* por uma cadeia de estados com uma transição para cada letra.

A vantagem desta representação é a simplicidade da implementação uma vez que se conta com a biblioteca *Reduced*. O autômato para armazenar padrões sintáticos consiste de uma "maquiagem" feita sobre um outro autômato de baixo nível. A desvantagem é o desperdício de espaço causado pelas cadeias estado-transição. Além disso, as marcas são replicadas no autômato agravando ainda mais o problema.

Esta implementação exige que certas operações muito usadas, tal como a função que avança sobre uma transição, devam levar em consideração a informação sobre o comprimento de uma marca, se for fixo, ou dos separadores utilizados. Esta informação deveria ser representada em outro nível para dar maior liberdade à escolha da notação utilizada para o conjunto de marcas.

7.2 Representação em Dois Níveis

Uma alternativa é representar os padrões sintáticos e as marcas em duas estruturas de dados diferentes, para quebrar assim o problema ocasionado pela replicação de marcas e a mistura dos dois níveis de informação. Como é mostrado a seguir, essas duas estruturas podem ser implementadas de forma adequada usando autômatos finitos.

```
Chave(i,cadeia[1...n],estado)
    se i=n então
        se Final(estado) então
            retornar 1
        senão
            retornar Indefinida!
    senão
        k ← 0
        trans[1...m] ← VetorTrans(estado)
        para j ← 1 até m fazer
            se trans[j].rótulo = cadeia[i] então
                retornar Chave(i+1,cadeia,trans[j].destino) + k
            senão
                k ← k + NSufs(trans[j].destino)
        retornar Indefinida!
```

Figura 7.2: Função de Espalhamento

Em uma representação em dois níveis as marcas são armazenadas em uma tabela de espalhamento, onde cada marca está associada com uma chave de acesso. Neste esquema, os padrões sintáticos são definidos como cadeias de chaves, onde cada chave identifica uma única marca na tabela de espalhamento. Um autômato pode implementar esse tipo de tabela com poucas restrições.

Em [LK93] apresenta-se um esquema de numeração dos estados de um autômato finito acíclico e minimal que permite definir uma função de espalhamento perfeita para cada cadeia armazenada. A numeração consiste em associar a cada estado a cardinalidade do conjunto de sufixos correspondente. Os estados finais cujo conjunto de sufixos apenas contém a cadeia vazia são numerados com 1 e o estado nulo é rotulado com 0.

A biblioteca *Reduced* define o método *NSufs* que retorna o número de sufixos de um estado. Essa rotina é implementada por meio de uma tabela preenchida sob demanda que introduz o custo da operação apenas na primeira invocação.

Os algoritmos nas figuras 7.2 e 7.3 são uma versão modificada dos apresentados

```
Cadeia(chave,prefixo,estado)
    trans[1...m] ← VetorTrans(estado)
    para j ← 1 até m fazer
        se NSufs(trans[j].destino) ≤ chave então
            chave ← chave - NSufs(trans[j].destino)
        senão
            se Final(trans[j].destino) então
                chave ← chave - 1
            se chave = 0 então
                retornar prefixo & trans[j].rótulo
            senão
                Cadeia(chave,prefixo & trans[j].rótulo,trans[j].destino)
    retornar Indefinida!
```

Figura 7.3: Função de Acesso

em [LK93]. O algoritmo na figura 7.2 retorna a chave de acesso correspondente à cadeia na entrada. A especificação desta função pressupõe que a avaliação das transições partindo de um estado é realizada sempre na mesma ordem. A função *VetorTrans* retorna um vetor cujos elementos correspondem a todas as transições saindo do estado passado como parâmetro. A ordem dos elementos do vetor retornado por *VetorTrans* é fixa, sempre que o autômato não seja atualizado com novas cadeias. O algoritmo da figura 7.3 implementa a função inversa: dada uma chave, a função *Cadeia* retorna a cadeia associada.

A implementação de uma tabela de espalhamento utilizando um autômato tem a vantagem de representar de forma compacta as marcas. A atualização do conjunto de marcas pode alterar as chaves; porém é viável supor que o conjunto de marcas é fixo (vide 7.3). A estrutura resultante consiste de dois autômatos:

- Autômato de Padrões Sintáticos (AP)
- Autômato de Marcas (AM)

A figura 7.4 esquematiza a arquitetura, a qual pode ser encapsulada em uma classe que forneça os métodos requeridos pelos algoritmos de reconhecimento e o conselheiro de

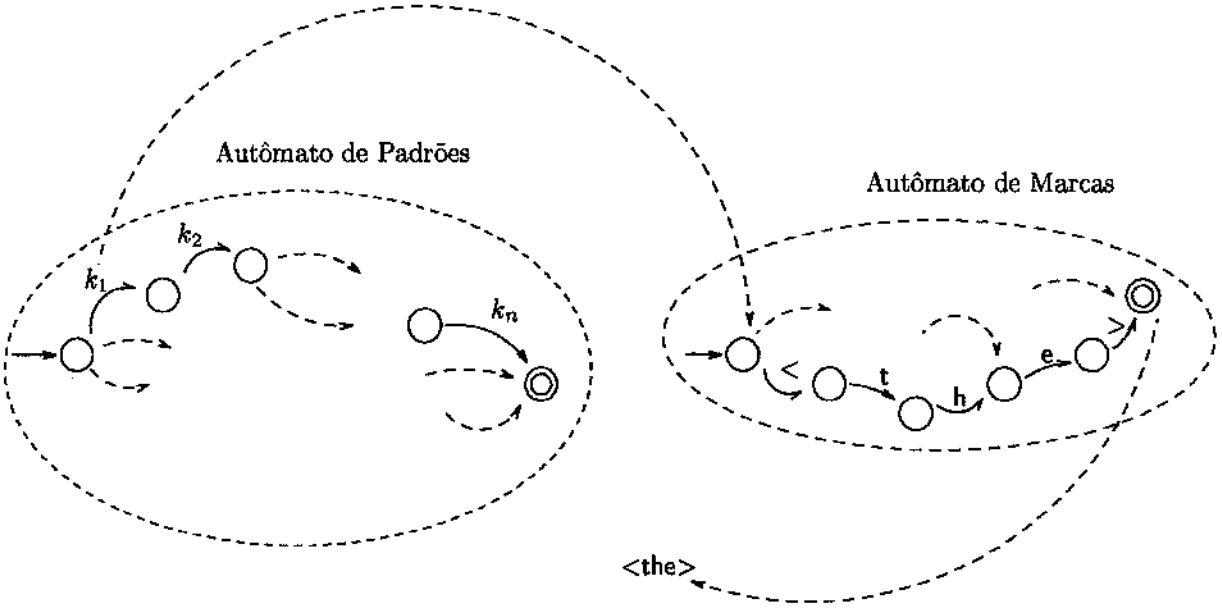


Figura 7.4: Autômato de Padrões: Versão 2

forma transparente. Na figura, cada k_i no autômato de padrões representa uma chave no autômato de marcas. A chave k_1, por exemplo, permite recuperar a marca literal <the>.

Como mostram os dados nas tabelas 7.1 e 7.2, a representação em dois níveis permitiu poupar 50% dos estados necessários para a versão com um nível. Além disso, a programação do banco de padrões sintáticos é mais clara, pois não é mais necessário definir um formato fixo para as marcas, nem especificar separadores especiais. Note-se que as melhorias obtidas com os *corpos SUSANNE* e *LOB* são similares em termos relativos. Outro fator a destacar é que o tamanho dos autômatos de marcas é insignificante em comparação aos autômatos de padrões sintáticos. O aproveitamento é expresso como o número de transições por estado.

7.3 Atualização do Conjunto de Marcas

No modelo de dois níveis cada padrão sintático é implementado como uma cadeia de chaves, onde cada chave permite recuperar sem ambigüidades a marca do autômato de marcas. A função de espalhamento depende diretamente da ordem em que as marcas estão armazenadas. A modificação do autômato de marcas pode significar um reordenamento de todas cadeias armazenadas, mudando a chave de acesso de algumas delas. A atualização

	LOB	*SUSANNE*
Sentenças	48.664	6.823
Padrões Sintáticos	35.080	4.675
Estados	1.432.887	196.065
Transições	1.467.030	200.717
Aproveitamento	1,0238	1,0237

Tabela 7.1: Estatísticas: Versão 1

	LOB	*SUSANNE*
Estados AP	782.555	100.627
Transições AP	816.704	105.279
Aproveitamento	1,043	1,046
Estados AM	1.113	245
Transições AM	1.932	511

Tabela 7.2: Estatísticas: Versão 2

das ocorrências de uma chave de acesso que mudou no autômato de padrões é uma tarefa muito custosa.

Uma solução para este problema é forçar um esquema onde toda marca nova possa ser inserida no último local do autômato de marcas, evitando alterar a ordem. Existe outra opção menos drástica que faz uso de uma técnica muito conhecida em ciência da computação: tabela de tradução.

A figura 7.5 ilustra a idéia do uso de uma tabela de tradução como passo intermediário entre o autômato de padrões e o autômato de marcas. Cada número em um padrão sintático é um índice invariável de acesso à tabela de tradução. O i-ésimo elemento armazena a chave atual para a marca que aparece representada com o índice i nos padrões sintáticos.

Se uma nova marca, por exemplo N0, deve ser inserida no autômato de marcas no local ocupado pela marca N1, todas as chaves maiores ou iguais à chave de N1 são incrementadas em um. A alteração das chaves pode ser facilmente implementada percorrendo a tabela de tradução, sem necessidade de atualizar os padrões sintáticos.

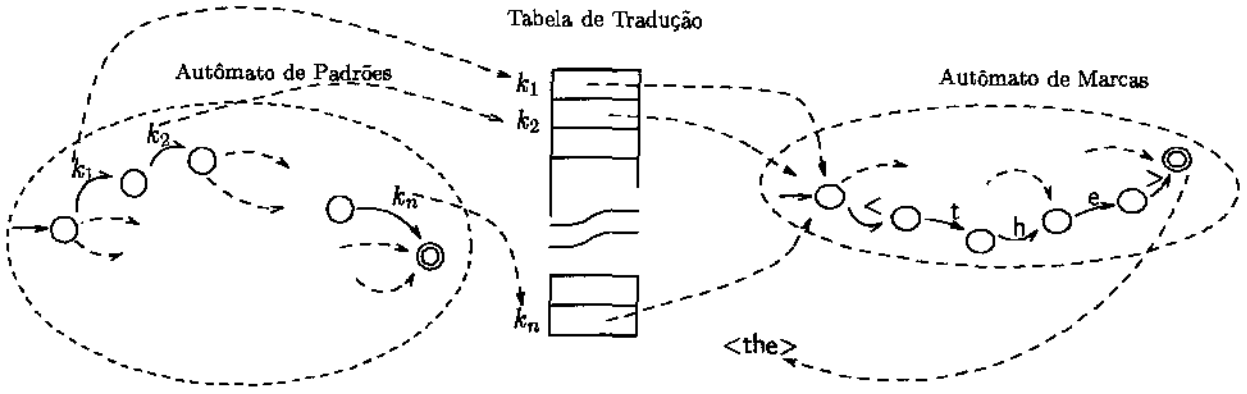


Figura 7.5: Tabela de Tradução

7.4 Subcadeias

Um dos argumentos usados no começo desta dissertação para explicar as motivações por trás do ambiente proposto é a analogia entre as palavras em um vocabulário e os padrões sintáticos. Sem contradizer essa hipótese, existem algumas diferenças que devem ser levadas em conta na implementação. Um exemplo é a possibilidade de encaixamento ou concatenação de sentenças, fenômeno não observado no nível morfológico. Com o objetivo de contornar essas diferenças projetou-se uma arquitetura mais sofisticada para a implementação do autômato de padrões sintáticos baseada em três autômatos.

O encaixamento e a concatenação de sentenças, embora limitados, introduzem subcadeias na coleta de padrões que não podem ser compartilhadas como prefixos ou sufixos de outros padrões. Padrões sintáticos representando este tipo de sentenças podem sobrecarregar a estrutura de autômatos penalizando a viabilidade do modelo.

Uma solução para este problema é estender o modelo de dois níveis com mais um autômato para armazenar subcadeias replicadas no banco de padrões. Esta representação permite fatorar as subcadeias espalhadas e repetidas no autômato de padrões. A arquitetura consiste em três autômatos:

1. Autômato de Padrões Sintáticos (AP)
2. Autômato de Subcadeias (AS)
3. Autômato de Marcas (AM)

Cada transição em AP representa uma chave para uma subcadeia em AS ou uma marca em AM. Todas as subcadeias em AS representam seqüências de chaves de acesso a

marcas em AM, como era feito no modelo de dois níveis. Resta definir como individualizar e extrair as subcadeias.

O problema de definir a extração das subcadeias replicadas para formar o modelo de três níveis, pode ser abordado de duas formas diferentes. Uma abordagem consiste em tentar extrair qualquer subcadeia repetida mais do que um número pré-determinado de vezes. A segunda abordagem dá um tratamento lingüístico ao problema melhorando o algoritmo de extração.

7.4.1 Subseqüências

Uma *subseqüência* é uma subcadeia de marcas contida em um padrão sintático que se repete mais do que um número pré-determinado de vezes no banco de padrões. A primeira experiência de identificação, extração e substituição de subcadeias foi realizada sem definir objetivos de eficiência.

O primeiro passo foi extrair todas as subcadeias de tamanho 2 até 10. Subcadeias de tamanho maior que 10 se repetem com pouca freqüência. O passo seguinte consistiu em ordenar todas as subcadeias separadas do banco de padrões e contar as repetições. Aquelas que não se repetiram mais do que um número prefixado foram eliminadas.

A experiência foi realizada sobre os padrões sintáticos coletados do *corpo LOB*, fixando em 10 o número mínimo de repetições requeridas. Alguns dados relevantes são:

- 20.051 subseqüências repetem-se apenas 10 vezes
- <the> NN registrou o maior número de ocorrências: 10.394
- não se registraram repetições de subseqüências de tamanho maior que 10

A observação da forma apresentada pelas subseqüências permitiu concluir que muitas delas eram quase padrões, como <i> <shall> BB, <i> <will> <not> e <i> <would> <have>. Isto levou a pensar na possibilidade de restringir a definição de subseqüência na procura de um algoritmo mais simples para a extração de subcadeias.

7.4.2 Subpadrões

Um *subpadrão* é um padrão sintático que ocorre como uma subcadeia de um outro padrão sintático maior.[2] Claramente, um subpadrão é um caso especial de subseqüência. Esta restrição permite definir uma alternativa mais simples para a extração de subcadeias.

O algoritmo na figura 7.6 imprime todos os subpadrões achados em um autômato de padrões sintáticos. Este algoritmo enumera todas as cadeias de um autômato seguindo um percurso em profundidade.

[2] O ponto final é eliminado por um pré-processador e não aparece no banco de padrões.

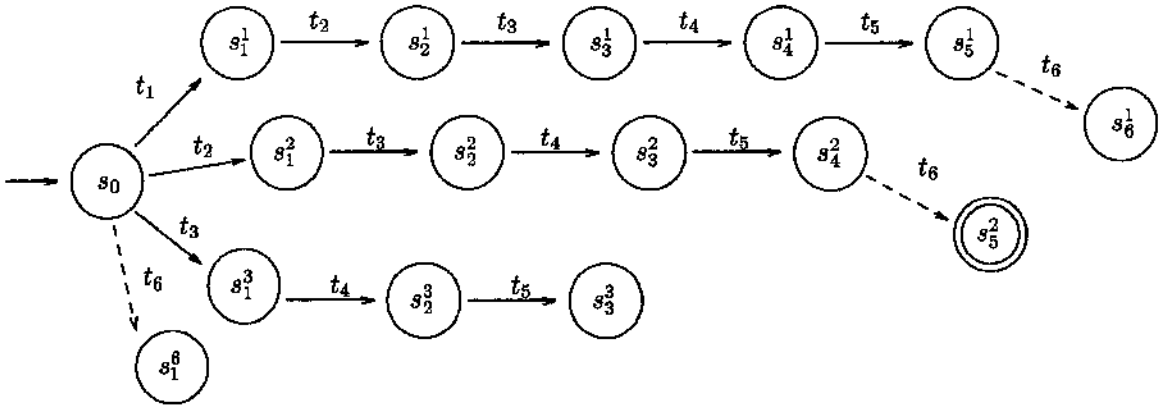
```
Subpadrões(estado, prefixo, subpads[2...n])
    para cada tran ∈ Transições(estado) fazer
        para i ← 2 até n fazer
            se Válido(subpads[i]) então
                e ← Sucessor(subpads[i],tran.rótulo)
            senão
                e ← X (*Nulo *)
            se Válido(e) então
                novoSubpads[i] ← e
            senão
                novoSubpads[i] ← X (*Nulo *)
            se Final(novoSubpads[i]) então Imprimir(prefixo[i..n])
        novoSubpads[n+1] ← Sucessor(bancoPadrões.raiz, tran.rótulo)
        Subpadrões(tran.destino, prefixo & tran.rótulo, novoSubpads)
```

Figura 7.6: Algoritmo de Extração de Subpadrões

A figura 7.7 esquematiza um passo do algoritmo *Subpadrões*. Na situação ilustrada, o algoritmo encontra-se visitando o estado s_5^1 do autômato, e está pronto a avançar sobre a transição rotulada com a marca t_6. O estado s_5^1 foi alcançado seguindo a seqüência de estados $s_0, s_1^1, \ldots, s_4^1$ sobre a cadeia de transições $t_1 \ldots t_5$. Esta cadeia é chamada prefixo atual.

O vetor *subpads* registra a seguinte informação:

- A subcadeia $t_2 \ldots t_5$ é um prefixo válido do autômato; *subpads[2]* contém uma referência ao último estado desse prefixo: s_4^2.
- A subcadeia $t_3 \ldots t_5$ também é um prefixo válido do autômato de padrões e o estado s_3^3 é o último da seqüência.
- Não existe prefixo $t_4 t_5$ no autômato; isto é marcado em *subpads[4]* com X indicando o estado nulo.
- Não existe prefixo t_5; em outras palavras, não existe uma transição válida saindo do estado raiz com rótulo t_5.



	2	3	4	5
subpads	s_4^2	s_3^3	X	X

	2	3	4	5	6
novoSubpads	s_5^2	X	X	X	s_1^6

Figura 7.7: Exemplo de um Passo do Algoritmo *Subpadrões*

A propriedade de determinismo do autômato estabelece que qualquer prefixo que começa com uma transição com rótulo t_1 é compartilhada pelo prefixo atual. Por essa razão os elementos do vetor *subpads* são indexados a partir de 2.

O laço externo do algoritmo *Subpadrões* visita todos os estados alcançáveis a partir de transições válidas saindo do estado atual. Para cada transição atualiza-se o vetor *subpads* com a informação que mantém a pista dos possíveis subpadrões para o prefixo atual.

Voltando ao exemplo, o próximo passo é avançar sobre a transição t_6 com destino no estado s_6^1. O vetor *novoSubpads* atualiza *subpads*. Esta tarefa é realizada pelo laço interno do algoritmo, o qual percorre todos os estados em *subpads* procurando estender com mais um rótulo algum dos prefixos válidos até o estado anterior. O estado *bancoPadrões.raiz* representa uma referência ao estado inicial do autômato com padrões. No exemplo, *novoSubpads* registra o seguinte:

- Existe uma transição válida partindo do estado s_4^2 com rótulo t_6 com destino no estado s_5^2; como esse estado é final imprime-se o subpadrão $t_2 \ldots t_6$.
- Não existe uma transição válida saindo do estado s_3^3 com rótulo t_6 (indicado com um X).

- Como já não existia subpadrão começando com as marcas t_4 e t_5, os elementos com índice 4 e 5 de *subpads* também correspondem ao estado nulo.
- Existe uma transição saindo do estado inicial s_0 com rótulo t_6 e destino no estado s_1^6 que pode ser a primeira marca de um padrão sintático.

O vetor *subpads* mantém a pista dos possíveis subpadrões para o prefixo atual. O *i*-ésimo elemento desse vetor é uma referência ao último estado correspondente a um prefixo que casa com as *n-i+1* marcas do prefixo atual de tamanho *n*.

Pela estrutura do autômato, se o prefixo atual é de comprimento n, existem n prefixos possíveis de comprimento *1,2, ... ,n* respectivamente. Por isso, em cada invocação recursiva o tamanho de *subpads* é incrementado em um elemento. Como já foi dito, o padrão de comprimento n corresponde ao prefixo atual e não é considerado.

O algoritmo percorre todos os padrões sintáticos aproveitando algumas propriedades do autômato. Uma vez calculado o conjunto de subpadrões para uma cadeia $t_1 \ldots t_n$, esse conjunto é aproveitado pelo algoritmo para qualquer subpadrão começando com esse prefixo. A chamada inicial do algoritmo é *Subpadões(bancoPadrões.raiz," ", [])*, onde *""* representa a cadeia vazia.

7.4.3 Cobertura de Padrões

Uma vez que tenham sido identificadas as subcadeias seguindo qualquer uma das duas abordagens, devem ser construídos os autômatos correspondentes. Primeiro é gerado o autômatos de subcadeias (AS) e a partir dele o autômato de padrões (AP).

Antes de agregar um novo padrão sintático a AP verifica-se se uma subcadeia em AS forma parte do mesmo, para substitui-la pela chave correspondente. Pode acontecer que mais de uma subcadeia em AS forme parte de um padrão sintático.

Assim, dado um conjunto de subcadeias podem existir diferentes seqüências de subcadeias que dão cobertura a um padrão sintático. Esta seção propõe um algoritmo genérico de programação dinâmica para determinar uma cobertura de um padrão sintático a partir de um conjunto de subcadeias pré-determinadas.

Ante várias possíveis coberturas, é preciso determinar uma que seja a "melhor". Porém, não está claro qual pode ser um critério para determinar a cobertura ótima. O algoritmo da figura 7.8 implementa a cobertura de um padrão a partir de um conjunto de subcadeias. Nesse algoritmo é atribuído um peso a cada solução possível; cada peso é calculado pela função *PesoSC* a qual deve ser definida pelo usuário.

Um critério pode ser dar preferência à cobertura envolvendo a menor quantidade de subcadeias. Esse critério é implementado pela versão da função *PesoSC* da figura 7.9.

O exemplo da figura 7.10 ilustra um passo do algoritmo *Cobertura* utilizando o critério de menor quantidade de subcadeias. O algoritmo começa percorrendo o padrão sintático

```
Cobertura(padrão[1...n],subcadeias)
    melhorPeso ← 0
    para i ← n até 1 fazer
        melhorPeso ← melhorPeso + PesoSC(padrão[i],cob[i...n])
        melhorSC ← padrão[i]
        para cada s ∈ subcadeias tal que s[1...m] = padrão[i...i+m-1] fazer
            peso ← PesoSC(s,cob[i...n])
            se peso ≤ melhorPeso então
                melhorPeso ← peso
                melhorSC ← s
        cob[i].s ← melhorSC
        cob[i].peso ← melhorPeso
```

Figura 7.8: Algoritmo para a Cobertura de Padrões Sintáticos

$t_1 \ldots t_{10}$ a partir da última posição até a primeira. Para cada possível sufixo do padrão, o algoritmo calcula a melhor cobertura. Essa informação é registrada pelo vetor *cob*. O i-ésimo elemento desse vetor é um registro com dois elementos: s e *peso*. O campo *peso* registra o peso da melhor cobertura para o sufixo $t_i \ldots t_n$. O campo s indica a subcadeia que deve ser usada na cobertura para a posição i.

Na figura, o algoritmo percorreu todos os elementos do padrão desde t_{10} até t_3. O conjunto de marcas deve ser considerado dentro das possíveis subcadeias, pois em geral não serão cobertas todas as marcas e nesses casos a marca é a única escolha.

O vetor *cob* da figura registra a seguinte informação:

- *cob[10]* indica que a melhor solução é cobrir o sufixo *padrão[10]* com a marca t_{10}, nesse caso a única opção. O número 1 representa o peso segundo o critério de menor quantidade de subcadeias.

- *cob[9]* indica que não existe nenhuma subcadeia que case com *padrão[9...10]*; então a única opção é a própria marca como indica o par $(s_{t_9},2)$. O número *2* significa que para dar cobertura ao *padrão[9...10]* são necessárias duas subcadeias.

```
PesoSC(s[1...m],cob[1...n])
    se m = n então
        retornar 1
    senão
        retornar cob[m+1].peso + 1
```

Figura 7.9: Algoritmo *PesoSC*

- *cob[8]* indica que a subcadeia s_1 casa com *padrão[8...10]* e a melhor cobertura (usando essa subcadeia) leva apenas uma subcadeia.
- *cob[7]*, *cob[6]* e *cob[5]* indicam que não existem subcadeias começando com as marcas t_7, t_6 e t_5 que melhorem a cobertura realizada pelas próprias marcas. O campo *peso* reflete o incremento de 1 por cada marca.
- *cob[4]* indica que a subcadeia s_2 casa com *padrão[4...6]* (prefixo de *padrão[4...10]*), e a melhor cobertura (usando essa subcadeia) leva três subcadeias.
- *cob[3]* indica que a subcadeia s_3 casa com *padrão[3...7]* (prefixo de *padrão[3...10]*) e a melhor cobertura (usando essa subcadeia) leva duas subcadeias.

Seguindo esse esquema, quando o algoritmo atingir a marca t_1, a melhor cobertura pode ser determinada a partir da informação em *cob*. O uso deste vetor permite diminuir o custo do algoritmo pois a informação de um sufixo é utilizada para determinar a melhor cobertura de outro sufixo maior.

A operação de procura por uma subcadeia que case com um prefixo do sufixo atual pode ser implementada facilmente se o conjunto de subcadeias for armazenado em um autômato. A complexidade do pior caso do algoritmo de cobertura é quadrática se for levado em conta o custo desta operação de casamento, mas pode também depender da implementação da função *PesoSC*.

7.4.4 Representação em Três Níveis

No modelo com dois autômatos, a referência de um estado corresponde diretamente ao número do mesmo no autômato de padrões. No modelo de três autômatos proposto nesta

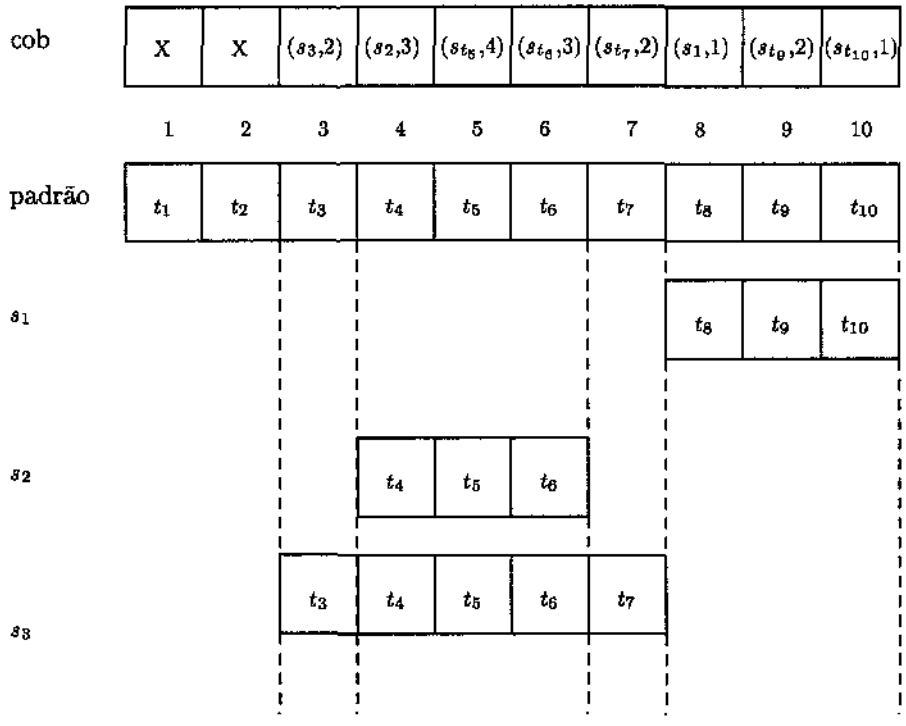


Figura 7.10: Exemplo de um Passo do Algoritmo *Cobertura*

seção a situação é diferente.

O algoritmo de casamento, que implementa um percurso exaustivo das transições do autômato, pode em um momento dado estar visitando estados no autômato de padrões sintáticos (AP) ou do autômato de subcadeias (AS). Algumas modificações devem ser realizadas para manter a transparência da representação interna usada para os algoritmos de casamento de padrões e do conselheiro de concordância. Para evitar confusões na notação, o estado que é manipulado como uma referência ao banco de padrões é chamado de *superestado*; a palavra *estado* é reservada para denotar um estado em AP, AS ou eventualmente AM.

Em uma representação em três níveis uma transição em AP pode estar rotulada com uma chave para recuperar uma marca no autômato de marcas (AM) ou uma subcadeia em AS. O primeiro caso pode ser tratado como no modelo de dois níveis; as complicações surgem na segunda situação. O rótulo de uma transição em AP deve reservar um *bit* para indicar se a chave corresponde a AS ou AM.

Para manter a transparência da representação interna, um superestado é definido como uma tripla

$$(S_{AP},S_{AS},n)$$

onde

- S_{AP} é um estado em AP;
- S_{AS} é um estado em AS; e
- n é a chave de um sufixo do estado S_{AS}.

Uma referência a um estado S_{AP} em AP é denotada simplesmente pela tripla (S_{AP},S^0_{AS},0) onde S^0_{AS} é o estado raiz de AS.

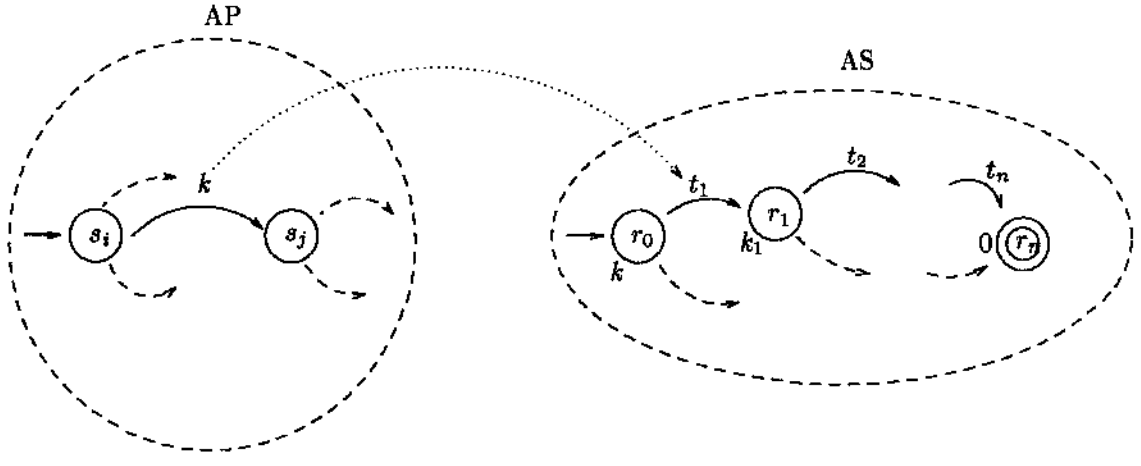


Figura 7.11: Autômato de Padrões: Versão 3

Na figura 7.11 mostra-se o caso de uma transição em AP rotulada com uma chave k de uma subcadeia em AS. Essa chave determina sem ambigüidades a cadeia $t_1 \ldots t_n$ seguindo as transições na seqüência de estados $r_0,\ldots,r_n$ em AS, onde r_0 é o estado raiz. O usuário manipulando uma referência a um estado em AS não tem (nem deveria) a informação necessária que lhe permita inferir qual é o próximo superestado em uma transição desse tipo.

Existe uma diferença clara entre a situação de um superestado referenciando um estado em AP ou em AS. No primeiro caso, todas as transições partindo do estado podem ser escolhidas, por exemplo, no próximo passo de um percurso em profundidade. Quando foi alcançado um estado em AS a partir de uma chave em AP, existe uma única transição possível determinada pela cadeia indexada pela chave, embora do estado partam outras transições.

Um superestado referenciando um estado em AS é configurado com a tripla (S_{AP},S_{AS},n) onde S_{AP} é o estado destino em AP a ser atingido após percorrer o caminho em AS; S_{AS}

é o estado referenciado em AS e n é a chave do sufixo que determina o resto do caminho em AS.

No exemplo da figura 7.11, a transição rotulada com k partindo do estado s_i em AP referenciado pelo superestado $(s_i,r_0,0)$ tem como destino o estado r_1 em AS referenciado pelo superestado (s_j,r_1,k_1), onde k_1 é a chave do sufixo $t_2 \ldots t_n$ em AS com origem em r_2. Assim k_1 determina univocamente a próxima transição com rótulo t_2. A transição partindo de $(s_i,r_0,0)$ com destino em (s_j,r_1,k_1) tem como rótulo a marca associada à chave t_1 em AM. Cabe notar que não existe a configuração $(s_j,r_n,0)$ pois o fim do caminho em AS corresponde ao próximo estado em AP, o estado s_j referenciado pelo superestado $(s_j,r_0,0)$.

	Subseqüências	Subpadrões
Número	20.051	1.238
Estados AP	465.863	736.173
Transições AP	500.379	770.229
Aproveitamento	1,074	1,046
Estados AS	3.879	1.418
Transições AS	16.681	2.474
Aproveitamento	4,300	1,744

Tabela 7.3: Estatísticas sobre o *LOB*: Versão 3

O modelo de três níveis foi testado seguindo as duas abordagens para a extração de subcadeias com o banco de padrões sintáticos coletados do *corpo LOB*. A tabela 7.3 resume as principais estatísticas. A implementação com três autômatos oferece uma representação mais compacta de uma forma totalmente transparente aos algoritmos propostos. A tabela 7.4 resume os dados obtidos do processamento do *corpo LOB*; nessa tabela SP e SC representam os experimentos feitos com a abordagem de extração de subpadrões e subcadeias respectivamente.

	Estados AP	Transições AP	Aproveitamento
Versão 1	1.432.887	1.467.030	1,023
Versão 2	782.555	816.704	1,043
Versão 3 (SP)	736.173	770.229	1,046
Versão 3 (SC)	465.863	500.379	1,074

Tabela 7.4: Estatísticas do Processamento do *corpo LOB*.

7.5 Resumo

A biblioteca *Reduced* foi desenvolvida no Instituto de Computação da UNICAMP como parte do projeto *Dicio* com o objetivo de fornecer rotinas para a manipulação de autômatos finitos para armazenar grandes vocabulários.

Em uma primeira experiência tentou-se implementar um autômato para armazenar padrões sintáticos diretamente sobre as estruturas de dados oferecidas por *Reduced*. O problema observado foi a replicação das marcas no autômato, o qual incorria em desperdício de espaço.

As cadeias em um autômato são ordenadas de tal forma que é possível associar a cada uma uma chave de acesso. Assim, dada uma cadeia em um autômato existe uma função de espalhamento perfeita que retorna a chave de acesso. Do mesmo modo, pode-se definir a função contrária que permite, dada uma chave, recuperar a cadeia associada.

Visando resolver os problemas da primeira experiência, projetou-se uma representação em dois níveis de autômatos: um autômato para as marcas e outro para os padrões sintáticos. Neste esquema cada padrão sintático é representado internamente como uma seqüência de chaves de acesso ao autômato de marcas. Assim, foi possível poupar 50% do espaço necessária para o modelo com um autômato.

Um problema similar ao da primeira implementação foi observado no modelo de dois níveis. Subcadeias de chaves de marcas, tais como <the> AL N1, são replicadas no autômato de padrões. Para fatorar a informação relativa a subcadeias repetidas agrega-se mais um autômato. Neste capítulo propõe-se duas alternativas para identificar esse tipo de subcadeia.

Capítulo 8

Conclusões e Extensões

> *"It will be bags of tricks and not theory that will advance computational linguistics in the future."*
> — Yehoshua Bar-Hillel.

A análise sintática de línguas naturais ainda se apresenta como um desafio de singulares características à comunidade de Lingüística Computacional. Embora tenham sido apresentadas numerosas propostas para tratar o problema, nenhuma mostrou-se como a definitiva. A ambigüidade estrutural, uma das motivações para a análise sintática, é também o principal empecilho para achar um modelo abrangente e eficiente do ponto de vista computacional.

A tendência atual está voltada às abordagens empíricas, as quais procuram uma integração de ferramentas. Nesta direção deve ser entendido o ambiente proposto nesta dissertação. Como é mostrado ao longo do trabalho, o modelo de padrões sintáticos é viável, eficiente e flexível, permitindo capturar fenômenos abrangendo uma grande quantidade de casos.

Neste capítulo descrevem-se em primeiro lugar as principais vantagens do ambiente. Finalmente propõem-se algumas extensões ao modelo.

8.1 Vantagens do Modelo Proposto

As principais vantagens do modelo de padrões sintáticos são:

- O modelo é simples. Construir uma GLC ou uma Gramática de Vínculos para modelar uma língua natural é uma tarefa difícil de automatizar e pode requerer muito tempo e esforço. A simplicidade do ambiente aqui proposto permite implementar a coleta de informação de forma automática a partir de *corpos* marcados facilitando a construção da gramática.

- O processo de análise é eficiente. Nas abordagens derivadas de GLC, a eficiência está diretamente ligada ao tamanho da gramática, expresso geralmente pela quantidade de regras. Os padrões sintáticos tampouco oferecem um modelo eficiente do ponto de vista teórico, mas segundo os resultados mostrados no capítulo 4, na prática a operação de reconhecimento é eficiente. O tempo que pode levar executar o algoritmo de reconhecimento é praticamente independente do tamanho dos dicionários léxico e de padrões sintáticos.

- A informação na especificação é positiva e não há necessidade de inferências que podem incorporar erros. Uma técnica comum é a implementação de corretores gramaticais a partir de listas de erros comuns. Nesse tipo de sistemas cada possível erro é codificado utilizando regras. A vantagem dessas abordagens está na precisão das mensagens e conselhos ante um erro. A clara desvantagem é a falta de flexibilidade, e o trabalho de formar a lista de erros a serem considerados.

  O modelo de padrões permite implementar uma abordagem mais simples, flexível e fácil de atualizar para a correção gramatical. Como é mostrado no capítulo 5, a impossibilidade de fornecer mensagens para cada erro é superada oferecendo "conselhos" que guiam o usuário.

- A especificação é escalável. A incorporação de novos padrões pode ser cara em tempo e demandar uma atualização das estruturas de dados (ver 7.3), mas esse tipo de modificação não afeta à informação já armazenada.

- Facilidade para manipular exceções. Tipicamente cada regra gramatical tem uma exceção. Em um modelo para manipular línguas sempre deve-se prever a representação para manipular exceções. O ambiente de padrões sintáticos oferece uma alternativa a partir do uso das marcas representando literais.

8.2 Resultados e Considerações

Embora o conjunto de padrões coletados nos primeiros experimentos não seja suficiente para implementar o ambiente sobre textos sem restrições, o número de sentenças que podem ser reconhecidas é muito grande. O padrão sintático 8.1, por exemplo, permite reconhecer 2×10^{22} sentenças diferentes com a informação no léxico *OTA*.[1]

<he> <has> <now> P2 <his> A1 N2 <to> <the> N2 <and> N2 GG (8.1)

[1] Esse cáiculo foi obtido em base da quantidade de entradas no léxico com marcas N2, A1, P2 e GG. A marca N2, por exemplo, repete-se 17.285 vezes.

Porém, em um teste sobre um texto com 4.000 sentenças, apenas 4% das mesmas estava representada por algum dos padrões sintáticos coletados. Isto evidencia que o problema não é a quantidade de sentenças capturadas, mas sim a qualidade dos padrões coletados.

Por outro lado, apenas um teste não é uma medida significativa. O principal empecilho na tarefa de testar o conjunto de marcas foi não contar com um pré-processador adequado. Para uma avaliação com um conjunto maior de padrões é preciso contar com um pré-processador do texto a ser analisado que permita "limpar" de marcas estranhas o texto a ser analisado.

Para implementar um sistema baseado neste ambiente é preciso coletar um número muito maior de padrões sintáticos. O *corpo LOB* consiste de uma coleção de um milhão de palavras; em comparação o *British Nacional Corpora* é 100 vezes maior, enquanto que o *corpo The Bank of English* é 340 vezes maior. Sem dúvida uma coleta implementada sobre um desses *corpos* resultará em um aumento na quantidade de padrões. Deve-se notar que isto vai refletir em um crescimento do autômato.

8.3 Coleta de Padrões

Da coleta de padrões sintáticos depende o sucesso de uma implementação do ambiente. Nesta dissertação é proposta a coleta a partir de *corpos* marcados. A vantagem clara desta abordagem reside na possibilidade de automatizar o procedimento de configuração do ambiente.

Uma desvantagem é que *corpos* podem incorporar cadeias de palavras inválidas ou incompletas, como por exemplo sintagmas nominais de uma manchete de jornal ou títulos de capítulos em um livro. Note-se que identificar esse tipo de cadeias pode ter a mesma complexidade que uma análise sintática.

Na coleta feita a partir dos *corpos SUSANNE* e *LOB*, o problema de cadeias incompletas foi resolvido com a implementação de um pré-processamento e com a ajuda de algumas marcas pré-definidas nos *corpos*. O princípio utilizado foi descartar todas aquelas cadeias sem verbo, contendo fórmulas matemáticas ou algum outro símbolo não considerado pelo conjunto de marcas definido. Para uma implementação real é preciso fazer um estudo mais detalhado com o objetivo de definir um conjunto de marcas adequado. Numerosas entidades tem financiado diferentes projetos com o objetivo de estabelecer um esquema padrão para a marcação de *corpos*. Esse tipo de experiência deve ajudar na tarefa da coleta de padrões e definição de conjuntos de marcas.

Outro problema que pode complicar a coleta de padrões são os erros infiltrados nos *corpos*. Os grandes *corpos* atualmente são marcados utilizando marcadores automáticos que embora confiáveis podem introduzir erros. Uma revisão manual de um *corpo* é inviável;

a única opção é desenvolver regras heurísticas que permitam no mínimo detectar erros graves.

Uma extensão à coleta de padrões sintáticos consiste em utilizar métodos estatísticos para detectar cadeias de marcas pouco prováveis em um padrão. Isto pode reduzir o número de possíveis padrões mal formados a serem revisados de forma manual. Note-se que não é preciso implementar um método muito sofisticado. Uma possibilidade é criar uma tabela para registrar o número de ocorrências de certos bigramas ou trigramas em uma primeira passada na marcação. Em uma segunda passada aqueles padrões que registrem uma seqüência com um número de ocorrências menor do que um número pré-definido seriam candidatas a cadeias contendo erros.

No processamento do *corpo LOB* notou-se que muitos padrões sintáticos básicos não formavam parte do conjunto extraído, mas apareciam como subseqüências de outros padrões maiores. Nesses casos é necessário investir em heurísticas que permitam extrair mais informação oculta entre as sentenças dos *corpos*. Por exemplo, o *corpo LOB* não contêm o padrão 8.2 que permitiria reconhecer dentre outras, a sentença *I am alive*. Mas por outro lado contém o padrão sintático maior 8.3 cujo prefixo é justamente 8.2.

<i> <am> AP (8.2)

<i> <am> AP <to> V2 <this> NN <in> <the> NN (8.3)

8.4 Modelo Dinâmico

O ambiente apresentado aqui é estático, no sentido que uma vez configurado o banco de padrões sintáticos essa informação é usada sem mudanças. Porém, com algumas poucas variações, o modelo poderia ser estendido com características que permitam aproveitar com maior intensidade a informação capturada pelos padrões sintáticos.

Em 7.4.2 foi introduzida a noção de subpadrão como uma solução a um problema de implementação. Nesta seção retoma-se o conceito para estender o modelo básico a um modelo dinâmico, como mostra o seguinte exemplo.

Exemplo 20 Os padrões 8.5 e 8.6 formam parte do conjunto de padrões coletado do *corpo LOB*.

<she> <will> V2 <a> N1 <which> YC ... YC <she> <would> <have> PP AL (8.4)

Além disso ambos padrões são subcadeias do padrão maior 8.4 obtido do processamento da sentença

she will undertake a journey which , an hour before , she would have declared impossible

Tanto 8.5 como 8.6 representam um conjunto de sentenças independentes, mas existe uma relação de concordância entre esses subpadrões que permite a sua combinação para formar 8.4.

<she> <would> <have> PP AL (8.5)

<she> <will> V2 <a> N1 (8.6)

<i> <would> <have> PP AL (8.7)

A questão é se um desses subpadrões sintáticos pode ser substituído por um padrão "similar". O subpadrão 8.7, por exemplo, tem a mesma marca para o verbo que 8.5 e poderia substituí-lo em 8.4 permitindo reconhecer a sentença

she will undertake a journey which , an hour before , I would have declared impossible

□

Para implementar esse tipo de substituição é necessário descobrir a informação relativa a subpadrões usando o algoritmo apresentado na seção 7.4.2. Além disso, deve ser marcado nos padrões sintáticos o lugar e o tipo de subpadrão que pode ser substituído. A partir do padrão sintático 8.4 do exemplo anterior pode-se gerar 8.8.

<she> <will> V2 <a> N1 <which> YC <an> N1 <before> YC SP_{wh} (8.8)

A marca especial SP_{wh} significa que pode ser substituída por qualquer subpadrão com o verbo *would have*.

Para avançar nesta representação deve-se definir o que significa que dois padrões são "similares". No exemplo foi tomado o tempo do verbo, mas devem ser consideradas outras heurísticas que levem em conta outros parâmetros. A partir da definição da relação de "similaridade" o conjunto de padrões poderia ser particionado em classes.

A partição do conjunto de padrões pode ser representada com a mesma estrutura de autômatos finitos. A alternativa mais simples consiste em acrescentar uma marca artificial no começo de cada padrão para distinguir a classe à qual pertence, como mostram 8.9 e 8.10.

SP_{wh} <i> <would> <have> PP AL (8.9)

SP_{wh} <she> <would> <have> PP AL (8.10)

A marca representando cada classe de padrões é fatorada na estrutura de autômatos subjacentes, incorrendo em uma sobrecarga de espaço mínima. Por último deve-se considerar o fato de que podem ocorrer marcas especiais aninhadas, o qual sem controle pode derivar em uma GLC.

8.5 Outros Fenômenos Superficiais

Em 2.1.5 apresentam-se alguns exemplos de regras de concordância derivadas de fenômenos fonológicos, como o uso de contrações obrigatórias ou a flexão de artigos.

Um exemplo em inglês é o uso dos artigos indeterminados *a* e *an*. A alternativa mais simples para manipular este fenômeno, usando padrões sintáticos, consiste em criar categorias especiais para cada combinação artigo-substantivo ou artigo-adjetivo. Por exemplo a categoria N1 deveria ser divida em duas:

- $\mathsf{N1}_a$ para aqueles substantivos que começam com som vocálico, tais como *hour* e *exam*
- $\mathsf{N1}_{an}$ para aqueles que começam com som consoante, tais como *horse* e *university*

Um problema similar é o tratamento da regência verbal e frases verbais. A *regência verbal* determina a relação entre verbos e preposições no caso de verbos transitivos indiretos. As *frases verbais* são combinações de verbo-preposição ou verbo-advérbio com um significado associado diferente ao significado das palavras componentes. Verbos como *get*, *look* and *go* formam um grande número de frases verbais.

A alternativa utilizada com os artigos indeterminados conduziria a um grande número de marcas sobrecarregando a estrutura de armazenamento e diminuindo o poder do conjunto de padrões sintáticos. Outro problema é que a relação entre o verbo e a preposição ou advérbio nem sempre se dá entre duas palavras consecutivas como acontece com os artigos indeterminados e os substantivos. Por exemplo, entre um verbo e uma preposição pode aparecer um advérbio como mostra 8.11.

The beauty of Venice consists largely in the style of its ancient buildings. (8.11)

Nesta sentença, o advérbio *largely* foi inserido entre o verbo e a preposição da frase verbal *consist in*.

Para evitar a explosão no número de marcas, este tipo de fenômenos pode ser tratado com regras codificadas por fora da estrutura de padrões sintáticos. Uma alternativa é associar com cada verbo um procedimento que estabelece que tipo de preposição ou advérbio pode ser ligado a ele. Regras heurísticas devem ser implementadas para indicar até onde pode chegar a procura pela preposição ou advérbio. Embora esta seja uma

solução forçada e fora do esquema proposto, oferece uma saída viável e relativamente simples de implementar.

A regência e a informação acerca de frases verbais podem ser codificadas facilmente no léxico junto às palavras, utilizando um esquema similar ao usado para as flexões (ver [KLS98]).

8.6 A Caminho de um Analisador Robusto

Segundo Karlsson em [Kar94], alguns dos requisitos que um analisador sintático robusto deve satisfazer são:

- não rejeitar o tratamento de nenhuma entrada;
- atingir uma taxa de 90% de exatidão;
- executar em um tempo uniforme para todas as cadeias com a mesma complexidade;
- analisar, em média, 10 palavras por segundo.

A primeira condição é a principal característica de um analisador robusto. Em [Mag94] discute-se o problema de definir teorias para domínios de aplicação limitados, sobretudo quando o objetivo final é a abrangência. Do ponto de vista do desempenho de uma língua, o modelo de padrões sintáticos satisfaz esta condição. Como é mostrado ao longo do trabalho, não são estabelecidas hipóteses sobre a complexidade das sentenças. O limite na profundidade do encaixamento está fora de discussão pois, como ja foi mencionado, apenas é relevante para o estudo da competência.

A terceira condição tenta estabelecer um padrão que permita uma comparação entre sistemas de PLN. O modelo de padrões representa em um mesmo esquema toda a informação; as sentenças consideradas patológicas são pouco comuns e não necessariamente implicam em uma degradação do tempo total de análise. A última condição de Karlsson não faz muito sentido uma vez que considera um valor absoluto.

A segunda condição por diferentes razões enunciadas neste capítulo, não pôde ser corretamente avaliada a partir dos experimentos realizadas, principalmente por não ter contado com uma fonte de dados suficientemente grande. A extensão do trabalho apresentado nesta dissertação deveria focalizar-se na coleta de um banco de padrões sintáticos grande.

Este capítulo inicia-se com uma frase atribuída ao lingüista Yehoshua Bar-Hillel que resume a situação atual na Lingüística Computacional. O processamento automático de línguas é um problema complexo, difícil de caracterizar ou contornar por uma teoria geral. O projeto de um sistema que pretenda manipular a língua como um todo deverá

considerar a utilização de várias técnicas e ferramentas integradas. Fundamentalmente, este trabalho tenta contribuir com um passo nessa direção.

Referências Bibliográficas

[Ajd35] Kazimierz Ajdukiewicz. A conexidade sintática. Tradução ao português: Lígia Negri e José Borges Neto. Original "Die syntaktische Konnexität" em "Studia Philosophica",1,1935,pp.1-27., 1935.

O autor tenta mostrar como é possível modelar a partir da lógica a sintaxe das línguas. A principal contribuição é a definição de conexidade sintática. Este trabalho fixou as bases sobre as quais foram desenvolvidos os formalismos lexicais.

[AU79] Alfred V. Aho e Jeffrey D. Ullman. *Principles of Compiler Desing.* Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley Publishing Co, 1979.

Este livro dá cobertura à maioria dos aspectos da construção de compiladores para linguagens de programação do estilo PASCAL. Entre outros temas, descrevem-se técnicas para a implementação de analisadores léxicos, analisadores sintáticos, processamento de erros, construção de tabelas de símbolos, verificação de tipos, geração de código e alguns tópicos sobre otimização.

[Bri93] Eric Brill. *A Corpus-Based Approach to Language Learning.* Tese de Doutorado, Department of Computer and Information Science - University of Pennsylvania, 1993.

Nesta tese, o autor apresenta uma abordagem nova para a implementação de marcação automática. A proposta consiste em utilizar regras apreendidas de um *corpo* marcado. O método de aprendizado é de prova e erro. Segundo os resultados mostrados pelo autor, com essa metodologia podem-se obter resultados comparáveis aos atingidos pelos métodos estocásticos.

[Cho57] Noam Chomsky. *Syntactic Structures.* The Haugue Mounton, 1957.

Este livro é considerado o primeiro trabalho do movimento gerativista. Chomsky propõe a sua teoria de transformações (Teoria Padrão) como alternativa aos problemas das GLC para representar línguas naturais. Neste livro, apresenta-se também o famoso argumento sobre as limitações de máquinas de estado finito para tratar fenômenos das línguas.

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax.* MIT press, Cambridge, 1965.
Neste trabalho Chomsky estende a Teoria Padrão.

[CLR90] Thomas H. Cormen, Charles Leisserson, e Ronald D. Rivest. *Introduction to Algorithms.* The MIT Electrical Engineering and Computer Science. The MIT Press and McGraw-Hill Book Co., 1990.
Introdução ao estudo de algoritmos do ponto de vista da análise de complexidade. Inclui, entre outros tópicos, algoritmos de ordenação, algoritmos sobre grafos, casamento de padrões e um capítulo dedicado à teoria *NP*.

[Col95] Survey of the State of the Art in Human Language Technology. `http://www.cse.ogi.edu/CSLU/HLTsurvey`, novembro de 1995.
Survey que resume as pesquisas dos últimos anos em Lingüística Computacional. Embora os temas não sejam tratados com detalhe, contém muitas referências a outros trabalhos.

[Esp73] Real Academia Española. *Esbozo de una Nueva Gramática de la Lengua Española.* Espasa-Calpe S.A., Madrid, 1 edição, outubro de 1973.

[GJW86] Barbara J. Grosz, Karen Sparck Jones, e Bonnie Lynn Webber. *Readings in Natural Language Processing.* Morgan Kaufmann Publishers, Inc, Los Altos, CA, 1986.
Compêndio de artigos que resume o estado da pesquisa em Lingüística Computacional na da década de 80. Entre outros, apresentam-se artigos sobre DCG, ATN, análise do discurso e geração.

[GLB94] Roger Garside Geoffrey Leech e Michael Bryant. CLAWS4: The tagging of The British National Corpus. Em Makoto Nagao, editor, *COLING 94. The 15th International Conference on Computational Linguistics*, volume 1, pp. 575–578, Kyoto, Japan, agosto de 1994.
Este artigo apresenta os resultados do trabalho de marcação do *British National Corpus*. Inclui um apêndice com uma lista detalhada do conjunto de marcas utilizado.

[GLS96] Dennis Grinberg, John Lafferty, e Daniel Sleator. A robust parsing algorithm for Link Grammars. Relatório Técnico CMU-CS-95-125, Carnegie Mellon University, School of Computer Science, Pittsburgh,PA, agosto de 1996.
Este artigo propõe um algoritmo robusto para a análise sintática baseado no formalismo das Gramáticas de Vínculos. Este modelo é basicamente estendido para incluir vínculos nulos que permitem continuar a análise de uma cadeia contendo algum erro.

[GM89] Gerald Gazdar e Chris Mellish. *Natural Language Processing in PROLOG: an introduction to computational linguistics.* Addison-Wesley Publishing Company, 1989.
Os autores apresentam a evolução da Língüística Computacional sob o ponto de vista da Inteligência Artificial. O livro inclui um grande número de exemplos e exercícios.

[Gri86] Ralph Grishman. *Computational Linguistics: An Introduction.* Studies in Natural Language Processing. Cambridge University Press, 1986.
Este livro apresenta a visão tradicional dos sistemas de PLN. O trabalho está dividido em cinco partes, cada uma dedicada a uma breve introdução das técnicas utilizadas nas áreas de análise sintática, análise semântica, análise do discurso e geração.

[Har71] Zellig Harris. *Structures mathématiques du langage.* Monographies de Linguistique Mathématique. Dunod, Paris, 1971.
Neste livro, Harris propõe a formalização das línguas naturais utilizando a teoria de conjuntos. A estrutura matemática utilizada para capturar uma sentença pode ser comparada aos padrões sintáticos.

[Har78] Michael Harrison. *Introduction to Formal Language Theory.* Addison Wesley Publishing Co, 1978.

[Har92] Samuel P. Harbison. *Modula-3.* Prentice-Hall, Englewood Cliffs, NJ, 1992.
Livro de introdução à programação em MODULA-3. As ferramentas da linguagem são introduzidas a partir da tradição em programação imposta pelas linguagens PASCAL e MODULA-2.

[Har96] HarperCollins. *Collins Cobuild English Language Dictionary New Edition,* 1996.
Este dicionário foi construído a partir dos dados coletados do *corpo The Bank of English.* A definição de cada termo é modelada a partir do uso dado às palavras nas sentenças do *corpo.*

[HB] Publishers HarperCollins e University of Birmingham. The Bank of English. `http://titania.cobuild.collins.co.uk/boe_info.html`.

[HU79] John Hopcroft e Jeffrey Ullman. *Introduction to Automata Theory, Languages and Computation.* Series in Computer Science. Addison Wesley, 1979.

[Ide94] Nancy Ide. Encoding standards for large text resources: The Text Encoding Initiative. Em Makoto Nagao, editor, *COLING 94. The 15th International Conference on Computational Linguistics,* volume 1, pp. 575–578, Kyoto, Japan, agosto de 1994.
Neste artigo explicam-se os objetivos do TEI.

[JAGL86] Stig Johansson, Eric Atwell, Roger Garside, e Geoffrey Leech. *The Tagged LOB Corpus: Users' Manual.* Norwegian Computing Centre for the Humanities, Bergen, 1986.

Completa descrição do processo de marcação do *corpo LOB*. Discute-se com profundidade a escolha de cada marca com exemplos.

[Jär94] Timo Järvinen. Annotating 200 million words: The Bank of English Project. Em *Proceedings of COLING-94*, 1994.

Este artigo explica o procedimento utilizado para a marcação do *corpo The Bank of English* [HB]. Para esta atividade foi utilizada a ferramenta ENGTWOL no nível morfológico. A marcação foi executada pelo programa ENGCC (*Constraint Grammars*).

[Kap95] Robert Kaplan. Finite state technology. Em [Col95] pags: 419–422 (capítulo 11 seção 6), novembro de 1995.

Survey sobre o uso de técnicas baseadas em autômatos finitos em sistemas PLN. Também discute-se a influência na área do argumento de Chomsky sobre a limitação no poder de representação das máquinas de estados para a análise sintática de línguas.

[Kar94] Fred Karlsson. Robust parsing of unconstrained text. Em [OdH94] pags. 121-141 (capítulo 8), 1994.

Neste trabalho, o autor propõe as *Constraint Grammas* como um formalismo para implementar análise sintática robusta. Na introdução, apresenta-se uma definição informal de analisador robusto.

[KK95] Fred Karlsson e Lauri Karttunen. Sub-sentencial processing. Em [Col95] pags. 111-115 (capítulo 3 seção 2)., 1995.

Survey que resume brevemente as principais abordagens utilizadas para implementar análise sintática superficial.

[KLS95a] Tomasz Kowaltowski, Cláudio L. Lucchesi, e Jorge Stolfi. Applications of finite automata in debugging natural language vocabularies. *Journal of Brazilian Computer Society*, 1(3), abril de 1995.

Este artigo reporta a implementação de depuradores automáticos de dicionários armazenados em autômatos finitos. O esquema consiste em detectar possíveis erros ou omissões no vocabulário a partir da "forma" do autômato correspondente.

[KLS95b] Tomasz Kowaltowski, Cláudio L. Lucchesi, e Jorge Stolfi. Minimization of binary automata. *Journal of the Brazilian Computer Society*, 1(3):36–42, abril de 1995.

Este artigo apresenta o uso de autômatos binários para a implementação de grandes autômatos com relativamente poucas transições.

[KLS98] Tomasz Kowaltowski, Cláudio Lucchesi, e Jorge Stolfi. Finite automata and efficient lexicon implementation. Relatório Técnico IC-98-02, Instituto de Computação - UNICAMP, Campinas - SP - Brasil, janeiro de 1998.
Este trabalho apresenta uma codificação para a implementação eficiente de léxicos. O esquema permite codificar a informação necessária para responder a consultas sofisticadas, como por exemplo, as possíveis flexões das palavras no léxico.

[Kos90] Kimmo Koskenniemi. Finite-state parsing and disambiguation. Em H. Karlgren, editor, *COLING-90. 13th International Conference on Computational Linguistics*, volume 2, pp. 229–232, Helsinki, Finland, 1990.
Este artigo apresenta as Gramáticas de Interseção de Estados Finitos. O autor apenas oferece um esboço das possibilidades do formalismo sem entrar em detalhes de implementação.

[Kos92] Kimmo Koskenniemi. Compiling and using finite-state syntantic rules. Em *COLING-92*, pp. 156–162, Nantes, agosto de 1992. International Committee on Computational Linguistics.
Neste artigo propõe-se algumas alternativas para uma implementação das Gramáticas de Interseção de Estados Finitos. A principal contribuição é a definição de um esquema de marcação baseado em *Constraint Grammars*.

[KS96] Brigitte Krenn e Christer Samuelsson. The linguist's guide to statistics. `http://coli.uni-sb.de`, 1996.
Notas de curso que introduzem o uso de métodos estocásticos em PLN. A primeira parte trata da teoria das probabilidades. Os capítulos restantes descrevem as aplicações clássicas para este tipo de métodos, como por exemplo, a marcação automática de *corpos*.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65:154–1970, março de 1958.
Neste artigo o autor define uma notação especial para denotar Gramáticas de Categorias, conhecida como notação *Lambek*. Este é considerado um dos primeiros trabalhos sobre este formalismo.

[Lap96] Éric Laporte. Context-free parsing with finite-state transducers. Universidade de Marne-la-Vallé - Gaspard Monge Institut, 1996.
Este trabalho estuda a implementação de GLC usando transdutores. Na introdução, o autor argumenta sobre o problema ocasionado pela falta de representação de informação lexical nas GLC.

[Lee89] Geoffrey Leech. *An A-Z of English Grammar and Usage*. Edward Arnold, London - Melbourne - Auckland, 1 edição, 1989.

[Lim96] Rocha Lima. *Gramática Normativa da Língua Portuguesa.* José Olympio, Rio de Janeiro, 33 edição, 1996.

[LK93] Cláudio L. Lucchesi e Tomasz Kowaltowski. Applications of finite automata representing large vocabularies. *Software-Practice and Experience*, 23(1):15–30, janeiro de 1993.
Neste artigo são apresentados os resultados obtidos na implementação de grandes vocabulários usando autômatos finitos determinísticos acíclicos. Como é mostrado no trabalho, a estrutura básica pode ser estendida para implementar dicionários de sinônimos e multilíngües.

[LST92] John Lafferty, Daniel Sleator, e Davy Temperley. Grammatical trigrams: A probabilistic model of link grammar. Em *AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, 1992.
Este artigo propõe a extensão das Gramáticas de Vínculos para incorporar técnicas estatísticas. O esquema tenta modelar o conceito de vínculo incorporando a probabilidade de que duas palavras possam ser ligadas.

[Mag94] David M. Magerman. *Natural Language Parsing as Statistical Pattern Recognition.* Tese de Doutorado, Department of Computer Science - Stanford University, fevereiro de 1994.
Nesta tese, o autor apresenta uma visão radical das possibilidades dos métodos estatísticos em PLN. A principal contribuição é o projeto do analisador SPATTER, o primeiro sistema a utilizar técnicas próprias dos reconhecedores de voz para a análise sintática.

[Min68] Marvin Minsky, editor. *Semantic Information Processing.* MIT Press, Cambridge, Massachusetts, 1968.
Este livro descreve as primeiras aplicações em PLN. A maioria das propostas relegam a análise sintática a um segundo plano, uma característica comum nos primeiros sistemas da Lingüística Computacional.

[MMM93] Beatrice Santorini Mitchell Marcus e Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, junho de 1993.
Este trabalho discute a tarefa de marcação feita sobre o *corpo The Penn Treebank.* Inclui um apêndice com uma descrição detalhada do conjunto de marcas utilizado.

[MW97] Tony McEnery e Andrew Wilson. *Corpus Linguistics.* Edinburgh University Press, 1997.
Este livro apresenta o estado atual da pesquisa sobre o processamento de *corpos.* Começa discutindo as críticas utilizadas pelo movimento gerativista contra a tipo de técnica.

[Nel91] Greg Nelson, editor. *Systems Programming with Modula-3.* Prentice Hall, 1991.
Introdução à programação de sistemas em MODULA-3 escrita pelos próprios autores da especificação da linguagem.

[OdH94] Nelleke Oostdijk e Pieter de Haan, editores. *Corpus-Based Research into Language.* Número 12 in Language and Computers. Rodopi, Amsterdam/Atlanta, 1994.

[Ofl96] Kemal Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–91, março de 1996.
Este artigo apresenta um esquema para a análise ortográfica de línguas baseado em autômatos finitos. A correção ortográfica é baseada em operações sobre as letras de uma palavra: substituição, remoção, inserção e transposição.

[Oxf] University Computing Services Oxford. British National Corpus. `http://info.ox.ac.uk:80/bnc/index.html`.

[Per81] Fernando Pereira. Extraposition grammars. *American Journal of Computational Linguistics*, 7(4):243–256, outubro de 1981.
Uma das limitações das DCG é a falta de poder para expressar transformações. Neste artigo, o autor propõe uma extensão ao formalismo para escrever regras que permitem especificar transformações seguindo o estilo gerativista.

[PW86] Fernando Pereira e David H. D. Warren. Definite Clause Grammars for language analysis - a survey of the formalism and a comparison with Augmented Transition Networks, 1986. Em [GJW86] pags. 101–124.
Ampla introdução ao modelo de DCG dando cobertura aos aspectos teóricos e práticos do formalismo. Os autores comparam o poder das DCG com o formalismo mais utilizado para implementar GLC: Redes de Transição Ampliadas.

[QG80] Randolph Quirk e Sidney Greenbaum. *A University Grammar of English.* Longman, Hong Kong, 10 edição, 1980.

[Ram89] Allan Ramsay. Computer-aided syntactic description. Em *Computational Linguistics Handbook*, capítulo 18, pp. 204–219. Walter de Gruyter, 1989.
Survey muito completo do estado da pesquisa na Lingüística Computacional na década de 80.

[Rap68] Bertram Raphael. SIR: A computer program for semantic information retrieval. Em Marvin Minsky, editor, *Semantic Information Processing*, pp. 33–145,

Cambridge, Massachusetts, 1968. MIT Press.
O objetivo do trabalho reportado neste artigo é basicamente implementar uma conversa inteligente com o usuário. Para a análise da estrutura é utilizado um esquema de padrões similar ao apresentado nesta dissertação.

[Rei69] Peter Reich. The finiteness of natural language. *Language*, 5(4):831–844, dezembro de 1969.
Este trabalho contesta o argumento de Chomsky sobre a limitação dos autômatos finitos para representar línguas.

[Rev91] Dominique Revuz. *Dictionnaires et Lexiques, Méthodes et Algorithmes.* Tese de Doutorado, Université Paris 7, janeiro de 1991.
Um dos primeiros trabalhos de implementação de léxicos utilizando autômatos finitos. Esta tese contém uma revisão completa das diferentes estruturas de dados propostas para a implementação de dicionários.

[Rob96] Arnold Robbins. *The GNU Awk User's Guide: Effective AWK Programming.* Free Software Foundation, Boston, MA, 1.0 edição, janeiro de 1996.

[RS95] Emmanuel Roche e Yves Schabes. Determinig part-of-speech tagging with finite state transducers. *Computational Linguistics*, 2(21):227–253, junho de 1995.
Este trabalho propõe uma implementação usando autômatos finitos do método de marcação proposto por Brill em [Bri93].

[Sam95] Geoffrey Sampson. *English for the Computer: The Susanne Corpus and Analytic Scheme.* Clarendon Pr, março de 1995.
Este livro propõe um conjunto de marcas baseado no método utilizado na biologia para a classificação de amostras. Esse esquema foi utilizado na marcação do *corpo SUSANNE.*

[Sch94] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. Em *International Conference on New Methods in Language Processing*, 1994.
Apresenta um método para a marcação probabilística de *corpos* que evita os problemas gerados por probabilidades nulas. O marcador TREETAGGER, de distribuição livre, está implementado usando esta tecnologia.

[Sel85] Peter Sells. *Lectures on Contemporary Syntactic Theories.* Número 3 in CSLI Lecture Notes. Center for the Study of Language and Information, Leland Stanford Junior University, 1985.
Este livro apresenta os aspectos básicos de três teorias sintáticas modernas: *Government-Binding, Generalized Phrase Structure* e *Lexical-Functional Grammar.* A abordagem ao problema de análise sintática é feita do ponto de vista da Lingüística.

[ST91] Daniel D. K. Sleator e Davy Temperley. Parsing English with a link grammar. Relatório Técnico CMU-CS-91-196, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA 15213, 1991.
Apresentação das Gramáticas de Vínculos. Neste trabalho os autores justificam as motivações e propõem um algoritmo de reconhecimento. Inclui uma definição completa dos conectores para uma gramática para o inglês.

[Ste93] Mark Steedman. Categorial grammar. *Lingua*, 3(90):221–258, 1993.
Extenso *survey* sobre os aspectos teóricos das Gramáticas de Categorias. Neste trabalho, apresentam-se extensões à função de composição de categorias para, por exemplo, manipular coordenação, referências anafóricas e outros aspectos das línguas naturais.

[TJ94] Pasi Tapanainen e Timo Järvinen. Syntactic analysis of natural language using linguistic rules and corpus-based patterns. Em *Fifteenth International Conference on Computational Linguistics (COLING '94)*, pp. 629–634, Kyoto Japan, agosto de 1994.
Este trabalho discute a implementação das *Constraint Grammars* de Karlsson utilizando padrões em dois níveis. Os padrões representam pequenos contextos e são coletados a partir de *corpos* de dados.

[Vil95] Aline Villavicencio. Avaliando um rotulador estatístico de categorias morfo-sintáticas para a língua portuguesa. Tese de Mestrado, Instituto de Informática - Universidade Federal do Rio Grande do Sul, 1995.
Esta dissertação contém uma introdução completa à utilização de métodos estocásticos para marcação automática de *corpos*.

[vRW86] Henk van Riemsdijk e Edwin Williams. *Introduction to the Theory of Grammar*. Current Studies in Linguistics Series. The MIT Press, 1986.
Este livro é uma introdução à teoria lingüística segundo o modelo gerativista. O texto está baseado na evolução da Teoria Padrão e outras extensões, como a de Princípios e Parâmetros.

[Wei89] Harald Weinrich. *Grammaire Textuelle du Français*. Didier Hatier, Paris, 1989.

[Wil95] Yorik Wilks. Natural language processing. *Communications of the ACM*, 39(1):60–111, janeiro de 1995.
Survey que resume o estado das pesquisas americanas na área de sistemas de PLN. Os autores mostram-se pesimistas do sucesso da aplicação de teorias lingüísticas, e propõem a utilização de técnicas empíricas.

[Win83] Terry Winograd. *Language as a Congnitive Process: Syntax*. Addison-Wesley, 1983.

Neste livro são cobertas as técnicas tradicionais para implementar análise sintática. Um dos sistemas discutidos é o SIR de Raphael [Rap68]. Este é o único volume de uma coleção incompleta.

[Woo70] W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, outubro de 1970.
Este foi o primeiro trabalho sobre ATN. Deve-se notar que o artigo é antigo, porém continua sendo hoje um dos formalismos mais utilizados na implementação de GLC.

[Woo86] W. A. Woods. Transition network grammars for natural language analysis, 1986. Em [GJW86], pags. 71–87.
Introdução às ATN escrita pelo autor do formalismo. Neste artigo são cobertos brevemente os aspectos de projeto e implementação de uma ATN.

[WRC97] Nina Wacholder, Yael Ravin, e Misook Choi. Disambiguation of proper names in text. Relatório Técnico RC20735(91860), IBM, fevereiro de 1997.
Este trabalho apresenta uma série de regras heurísticas para identificar nomes próprios em textos. As regras estão baseadas em um estudo morfológico, de estilo e da estrutura do texto.

[YJ95] Anssi Yli-Jyrä. Schematic finite-state intersection parsing. Em Kimmo Koskenniemi, editor, *The Tenth Nordic Conference on Computational Linguistics (NORDALIA '95)*, Helsinki, maio de 1995. Department of General Linguistics, University of Helsinki.
Este artigo propõe uma alternativa para implementar a operação de reconhecimento usando Gramáticas de Interseção de Estados Finitos. O objetivo é diminuir os requerimentos de espaço da operação de interseção.

[You67] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

[Zif74] Paul Ziff. The number of English sentences. *Foundations of Language*, 11(4), julho de 1974.
Este artigo contesta a hipótese de Chomsky sobre a infinitude das línguas. O autor tenta mostrar a importância do estudo do desempenho das línguas.