MIRYAM DE MORAES

ALGUNS ASPECTOS DE TRATAMENTO DE DEPENDÊNCIAS DE CONTEXTO EM LINGUAGEM NATURAL EMPREGANDO TECNOLOGIA ADAPTATIVA

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

Área de Concentração: Sistemas Digitais

Orientador: Prof. Dr. João José Neto

São Paulo

2006

Moraes, Miryam de
Alguns aspectos de tratamento sintático de dependência de contexto em linguagem natural empregando tecnologia adaptativa / M. de Moraes. -- São Paulo, 2006.
150 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Linguagem natural 2.Sintaxe e semântica 3.Teoria dos autômatos 4.Gramáticas formais por computador I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

FOLHA DE APROVAÇÃO

Miryam de Moraes

Alguns Aspectos de Tratamento de Dependências de Contexto em Linguagem Natural empregando Tecnologia Adaptativa

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Engenharia

Área de Concentração: Sistemas Digitais

Aprovada em:

Banca Examinadora

Prof. Dr._____

Instituição:_____Assinatura:_____

Prof. Dr._____

Instituição:_____Assinatura:_____

Prof. Dr._____

Instituição:_____Assinatura:_____

Prof. Dr._____

Instituição:_____Assinatura:_____

Prof. Dr._____

Instituição:_____Assinatura:_____

Prof. Dr._____

Instituição:_____Assinatura:_____

DEDICATÓRIA

Eu dedido este trabalho a Monsenhor Luigi Valentini e aos meus sobrinhos, Lucas e Gabriel. Revelam o olhar, a voz, o sorriso e sobretudo, o precioso Rosto do Senhor na minha vida.

AGRADECIMENTOS

Ao Bom Deus, à Imaculada e sempre Virgem Maria, a São José e São Padre Pio, agradeço a minha vida. Eu Lhes ofereço as circunstâncias e resultados deste trabalho.

Ao professor Dr. João José Neto, agradeço sua dedicada orientação e solícita amizade.

Não posso deixar expressar minha gratidão ao professor Paulo Lopes e ao professor Arthur Bataglia, pelo apoio dado ao longo da execução simultânea à elaboração deste trabalho, de minhas tarefas profissionais.

Aos meus pais Isachar e Sylvia expresso minha gratidão por acolherem e suportarem, em idade avançada, as venturas e dissabores de sua filha.

A minha irmã Maria Ângela, querida amiga e mãe dos amores da minha vida: Lucas e Gabriel.

A Monsenhor Luigi Valentini e àqueles que encontrei ao longo destes anos. São rostos, alguns sem bem me conhecerem, que intercedem por mim junto ao Bom Deus: religiosas da Toca de Assis e monjas beneditinas do Mosteiro Nossa Senhora da Paz, irmã Matilde, ocd e Rev. Padre Peter Mary Rookey, osm, Monsenhor Julio Lancelotti e Monsenhor Joaquim Stein, Marcia Moysés.

São Paulo, 2006

Miryam de Moraes

RESUMO

MORAES, M. **Alguns aspectos de tratamento de dependências de contexto em Linguagem Natural empregando tecnologia adaptativa.** 2006. 180f. Tese (Doutorado) – Escola Politéncica, Universidade de São Paulo, 2006.

O tratamento de Linguagens Naturais requer o emprego de formalismos mais complexos que aqueles normalmente empregados para Linguagens Livre de Contexto. A maioria de tais formalismos são difíceis de serem utilizados, não práticos e sobretudo, associados a um desempenho de elevado custo. Autômatos de pilha estruturados são excelentes para se representar linguagens regulares e aspectos livre de contexto encontrados em Linguagem Natural, uma vez que é possível decompo-los em uma camada reguar (implementada com máquina de estados finitos) e uma livre de contexto (representada por uma pilha). Tais dispositivos aceitam linguagens determinísticas e livre de contexto em tempo linear. Dessa forma, trata-se de um dispositivo adequado para ser empregado como mecanismo subjacente para os autômatos adaptativos, que permitem o tratamento –sem perda de simplicidade e eficiência – de linguagens mais complexas que aquelas livres de contexo

Nesta tese, dependências de contexto são tratadas com tecnologia adaptativa. Este trabalho mostra como uma regra de Linguagem Natural descrita com uma metalinguagem pode ser convertida em um autômato de pilha adaptativo.

Foi possível verificar que problemas complexos em análise de Linguagem Natural, tais como os não-determinismos e ambigüidades presentes em situações de concordância, subcategorização, coordenação podem ser resolvidos com eficiência. De fato, todos os mecanismos adaptativos para solucionar estes problemas apresentam desempenho O(n).

Uma arquitetura para processamento em Linguagem Natural é apresentada.

Palavras-chave: dependências de contexto, tecnologia adaptativa, Linguagem Natural

ABSTRACT

MORAES, M. **Some aspects of Natural Language context dependencies handling using adaptive technology** 2006. 180f. Thesis (Doctoral) – Escola Politéncica, Universidade de São Paulo, 2006.

Since low-complexity language formalisms are too weak to handle NL, stronger formalisms are required, most of them resource demanding, hard to use or unpractical. Structured pushdown automata are excellent to represent regular and context-free aspects on NLs by allowing them to be split into regular layer (implemented as finite-state machines) and a context-free one (represented by a pushdown store). Such devices accepts deterministic context-free languages in linear time, and is suitable as un underlying mechanism for adaptive automata, allowing handling – without loss of simplicity and efficiency – languages more complex than context-free ones.

In this thesis context dependency is handled with adaptive technology. This work shows as a Natural Language rule described with a metalanguage can be converted into adaptive structured pushdown automata.

It was possible to verify that complex problems in Natural Language parsing e.g., non-determinisms and ambiguities present in agreement, subcategorization, coordination can be solved with efficiency. In fact, all adaptive mechanisms attached to these problems have O(n) performance.

An adaptive architecture for NL Language processing is presented.

Keywords: Context-dependencies – adaptive technology – Natural Language

SUMÁRIO

1 INTRODUÇÃO

Neste primeiro capítulo apresentam-se as principais motivações e objetivos do trabalho realizado e relata-se a seqüência em que foi desenvolvido. Ao final, indica-se a organização do texto desta tese.

1.1 Apresentação

O processamento de linguagens naturais requer o desenvolvimento de programas que sejam capazes de determinar e interpretar a estrutura das sentenças em muitos níveis de detalhe.

As linguagens naturais exibem um intricado comportamento estrutural visto que são profusos os casos particulares a serem considerados. Uma vez que as linguagens naturais nunca são formalmente projetadas, suas regras sintáticas não são nem simples nem óbvias e tornam, portanto, complexo o seu processamento computacional.

A formalização completa de linguagens naturais exige o emprego de modelos computacionais mais elaborados que expressões regulares e autômatos finitos, ou mesmo que gramáticas livres de contexto e autômatos de pilha.

De fato, um dos problemas mais importantes, dentre os usualmente encontrados no processamento de linguagens naturais, corresponde à dificuldade de se expressar, através de um formalismo legível e expressivo, as complexas nuanças estruturais, sempre presentes nas linguagens naturais.

O maior desafio consiste no estabelecimento de uma forma simples e clara para definir as construções lingüísticas de maneira fácil e inteligível, sem que para isso seja necessário lançar mão de mecanismos meta-lingüísticos de difícil interpretação.

A teoria das linguagens regulares e dos autômatos finitos permite lidar confortavelmente com atividades relacionadas à morfologia, dicionários, categorização, etiquetação morfológica, inflexão de palavras, etc. (KARTTUNEN, 2001).

O formalismo que melhor permite representar os aspectos livres de contexto da sintaxe de uma linguagem natural, tais como a estrutura superficial das sentenças, múltiplas árvores de derivação e ambigüidades sintáticas e não-determinismos, é o autômato de pilha.

Esta classe de problemas é muito freqüente no processamento computacional de linguagens de programação, dispondo-se de um grande repertório de algoritmos e modelos que permitem resolvê-los (AHO; ULLMAN, 1972).

O tratamento adequado de não-determinismos e ambigüidades, o tratamento de categorias vazias, de coordenações e de anáforas são exemplos de problemas mais complexos, que requerem formalismos dependentes de contexto, visto que são fenômenos lingüísticos que denotam interdependências, não apenas entre sentenças como também entre elementos de uma mesma sentença.

Linguagens dependentes de contexto geralmente se definem através de regras de substituição, eventualmente condicionais, as quais restringem as situações de contexto nas quais podem ser permitidas as substituições que definem.

Muitos métodos são empregados em sistemas de processamento de linguagem natural, adotando diferentes paradigmas. Assim, para isso dispõe-se de métodos exatos, aproximados, pré-definidos ou interativos, inteligentes ou algoritmicos, etc. (JURAFSKY; MARTIN 2000).

Os métodos estatísticos exploram probabilidades, taxas de minimização de erros, aprendizagem dinâmica. Freqüentemente não requerem gramáticas formais ou qualquer outra

representação exata da linguagem. Uma vez que empregam técnicas aproximadas proporcionam implementações mais simples, as quais infelizmente, tendem a exibir taxas de erro residuais devido à inevitável presença de casos pouco prováveis.

Métodos não-estatísticos empregam conceitos advindos da Teoria dos Autômatos e das Linguagens Formais a fim de representarem linguagens naturais e outras linguagens sensíveis ao contexto. Apesar de apresentarem resultados mais precisos, o desempenho que com eles pode ser obtido é geralmente baixo, devido às explosões combinatórias que freqüentemente devem enfrentar.

Métodos híbridos permitem a fusão e a interação complementar entre as técnicas estatísticas e não-estatísticas, proporcionando ao usuário os benefícios das soluções mais econômicas de cada técnica (MANNING, 2005). Naturalmente o desempenho pode ser diferenciado, de acordo com a abrangência de cada método e a particular aplicação a que se destina. (CHIANG, 2004).

O emprego de técnicas de inferência na construção de uma especificação de linguagem (gramática ou autômato) é atrativo sempre que não houver disponibilidade de uma descrição formal da linguagem, e sempre que houver um conjunto significativo de amostras ou de padrões corretos da linguagem desejada.

Técnicas de inferência são freqüentemente empregadas de forma relativamente adequada nas tarefas de formalização de uma linguagem a partir de uma amostra da mesma.

Nesse caso, geralmente é necessária uma interação com o ser humano, tanto para a inserção de dados complementares, para a posterior extração automática de informação, como para o fornecimento de casos irregulares.

O uso de corpora, textos alinhados, textos anotados e outros, é uma técnica fundamentada na captura informal (manual ou automática) e validação do conhecimento sobre a linguagem, através da análise de um conjunto representativo de dados experimentais.

Geralmente, uma grande quantidade de amostras rigorosamente revistas de textos pertencentes à linguagem e admitidos como significativos a partir de critérios oriundos da Lingüística é empregada como base para o treinamento de algoritmos que efetuarão o processamento da linguagem natural.

Tal prática pode ser realizada através de algoritmos de inferência e de dispositivos de reconhecimento capazes de se automodificarem. Ao final, o dispositivo, devidamente treinado, é utilizado para analisar textos desconhecidos, norteado pelo conhecimento acumulado durante a fase de treinamento. Este método tem sido largamente aplicado em atividades de processamento de linguagem natural tais como: tradução automática, extração semântica e mineração de dados.

Exemplos de Corpora para a Língua Inglesa são passíveis de serem acessados em (PENN PARSED Corpora of Historical English.), (PENN-HELSINKI Parsed Corpus of Early Modern English). Corpora para a Língua Portuguesa podem ser acessados em (LINGUATECA), um centro distribuído de recursos para o processamento computacional da Língua Portuguesa .

N-gramas (geralmente N = 2 ou 3) são uma outra aplicação de heurística ao processamento de linguagem natural que também pode eliminar o requisito de descrições formais da linguagem.

Com esta técnica, textos são localmente inspecionados através de uma janela de N palavras. As decisões são, em sua maioria, efetuadas a partir do conhecimento previamente adquirido e da informação presente no conjunto de palavras em inspeção.

Trata-se de um método aproximativo simples, com bom desempenho no tempo e que pode ser eficientemente empregado para a tarefa de desambigüização morfológica de palavras. (KINOSHITA, 1998)

O processamento de linguagens naturais com o auxílio de autômatos e gramáticas exige que se disponha de alguma definição formal da linguagem. Em sua forma original, proporciona uma técnica exata para a análise da linguagem, que usa como base a matemática discreta, autômatos e teoria das linguagens formais.

Muitas ferramentas de processamento de linguagem natural existem, muitas das quais implementadas com linguagens funcionais ou lógicas. Em particular, encontram-se implementações de DCG (*definite clause grammars*) e ATN (*augmented transition networks*) desenvolvidas em Prolog (MATTHEWS, 1998), (DOUGHERTY, 1994) e Lisp (GAZDAR; MELLISH, 1989), respectivamente.

A Gramática de Adjunção de Árvore (TAG - "*Tree Adjoining Grammar*") introduzida em (JOSHI, 1985) é um formalismo cuja estrutura de dados fundamental é a árvore. As árvores podem ser do tipo inicial ou do tipo auxiliar. As árvores iniciais são combinadas por duas operações denominadas substituição e adjunção.

Robert Frank explora em (FRANK, 2002) o papel deste formalismo na teoria da sintaxe, motivado principalmente pelo trabalho Programa Minimalista de Chomsky (CHOMSKY, 1993). Trata-se de um formalismo com o auxílio do qual é possível encontrar um conjunto de árvores elementares para cada uma das sentenças em $L_1 = \{a^n b^n c^n \mid n \in N\}$. Por outro lado, nas linguagens $L_2 = \{a^n b^n c^n d^n e^n \mid n \in N\}$ e $L_3 = \{www \mid w \in \Sigma^*\}$ não são geradas por nenhuma TAG

A alternativa para o *processamento* e *representação* de linguagem natural empregada nesta pesquisa é a utilização do formalismo adaptativo (NETO, 1993), que traz como vantagem, a possibilidade de inclusão automática de mecanismos de reconhecimento dependentes de contexto, a possibilidade de exploração das características de aprendizado e o potencial desempenho diferenciado para problemas de alta complexidade.

Nesse trabalho, o autor descreve um método poderoso para a construção automática de reconhecedores sintáticos com operação *hierarquizável*, fundamentado no uso do autômatos de pilha estruturados. Tal método possibilita a obtenção de reconhecedores diretamente a partir de descrições convencionais, livres de contexto, de uma linguagem, e a imposição de limitantes para certos parâmetros do reconhecedor que efetua o tratamento da linguagem. O método explora o fato de que os textos a serem analisados são sempre finitos na prática, e que, dada uma aplicação existe sempre um limite, além do qual os recursos dos reconhecedores gerais raramente seriam utilizados.

Este modelo permite explicitamente representar os elementos que podem tornar não-regular uma linguagem livre de contexto, armazenando-os em uma memória organizada como pilha, e permitindo que os demais símbolos da cadeia, que não participam da parte não-regular da sentença, sejam tratados como se constituíssem sentenças de uma sub-linguagem regular, utilizando para isso mecanismos similares aos autômatos finitos. O formalismo resultante é capaz de aceitar qualquer linguagem livre de contexto determinística em tempo linear. (NETO, 1987), (NETO, 1993), (NETO; PARIENTE, 1999).

O autor, em seguida, introduz a conceituação do autômato e do transdutor adaptativo, modalidades de dispositivos de reconhecimento e transdução sintática, os quais incorporam a possibilidade de auto-modificação ao longo de sua operação, estendendo-se e adaptando-se às necessidades de cada particular texto de entrada que lhes seja submetido.

Os dois grandes resultados relatados projetam-se na área de linguagens extensíveis, bem como na área do tratamento de linguagens que exibem *dependências de contexto.*

O dispositivo aí proposto é o autômato adaptativo, dispositivo reconhecedor com poder de máquina de Turing, portanto um modelo abrangente de computação, o qual pode ser empregado como mecanismo de definição e de representação para linguagens naturais.

Os autômatos adaptativos incorporam ainda, na forma de transições adaptativas, um *mecanismo básico de aprendizagem*, com o qual tais máquinas se tornam capazes de memorizar informações acerca do histórico da sua operação, bem como de alterar seu próprio comportamento em função desse aprendizado.

Em (NETO, 1998) está apresentada uma coleção de soluções para alguns problemas de alta complexidade, cujas alternativas por métodos clássicos podem se mostrar inadequadas ou excessivamente ineficientes.

Essa publicação mostra que, empregando-se o formalismo adaptativo, a solução dos problemas apresentados pode alcançar o desempenho O(n).

Os problemas em questão são: reconhecimento eficiente de palíndromes, *a elaboração de aceitadores para linguagens dependentes de contexto*, aceitadores de anagramas constituídos por um conjunto finito de símbolos, aceitadores para seqüências incorporadas assincronamente. Também discute o problema dos aceitadores para linguagens ambíguas e do aceitador simultâneo para um conjunto de linguagens.

Em (IWAI, 2000), é proposto um formalismo dual ao dos autômatos adaptativos, visando facilitar o desenvolvimento de linguagens complexas ou outras aplicações que necessitem especificar linguagens dependentes de contexto na forma de gramáticas.

Trata-se das Gramáticas Adaptativas. É apresentada uma notação para esse formalismo gramatical, que pode ser empregada como metalinguagem de entrada para uma ferramenta que permite transformar uma gramática no autômato equivalente e vice-versa. Iwai também formula teoremas que provam a equivalência entre os autômatos adaptativos e as gramáticas adaptativas.

Os aspectos de implementação prática de uma ferramenta dessa natureza foram posteriormente explorados em (RICCHETTI, 2005)

Muitas são as vantagens proporcionadas pelos autômatos adaptativos na área da Computação (NETO, 2003), destacando-se:

- Resgate do caráter sintático das dependências de contexto;
- Inclusão de recursos conceituais suficientes para dar ao seu usuário a opção de dispensar estruturas explícitas de dados, geralmente empregadas nos métodos convencionais de análise e de compilação de linguagens de programação usuais;
- Eliminação da distinção e separação entre os tratamentos léxico, sintático e semântico de uma linguagem, proporcionando descrições mais uniformes e completas da mesma;
- Tratamento sintático dos escopos dos nomes – incorporado à análise léxico-sintática, o tratamento de escopos para os nomes pode passar a ser efetuado pela manipulação da estrutura topológica do autômato, criando-se ou bloqueando-se acessos às partes do mesmo que representam elementos da linguagem sujeitos a regras de escopo, de acordo com a visibilidade contextual de tais elementos ao longo da análise;
- Tratamento sintático dos atributos associados aos nomes;
- Tratamento sintático da declaração e da chamada de procedimentos em linguagens de programação;

No que diz respeito ao tratamento de Linguagens Naturais, (TANIWAKI, 2001) demonstra a capacidade que têm os autômatos adaptativos de representarem o conhecimento de forma análoga à encontrada em formalismos consagrados dessa área, tais como ATN, DCG, Frames, XG, RTN. Para verificar a viabilidade da utilização dos formalismos adaptativos, no procedimento de análise sintática de linguagem natural, a autora apresenta propostas de algoritmos de mapeamento do formalismo ATN e da Gramática Baseada em Restrição para o Formalismo Adaptativo equivalente.

Em (MENEZES, 2000), foram discutidos diversos aspectos da aplicação prática da tecnologia adaptativa, tanto no projeto como na implementação de etiquetadores morfológicos para linguagens naturais.

Dos seus primórdios, em 1985, aos dias de hoje, marcado significativamente pela introdução do formalismo dos autômatos adaptativos, inúmeras dissertações, teses, ferramentas (PEREIRA; NETO, 1997), (PEREIRA, 1999), (PISTORI, 2003), (PISTORI; NETO, 2003a), (PISTORI; NETO, 2003b), e aplicações em software se desenvolveram no Laboratório de Tecnologia Adaptativa do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

Atualmente os principais projetos do Laboratório ligados ao tema desta tese incluem: a representação e manipulação do conhecimento, a representação e processamento de linguagens naturais, a tradução automática de idiomas, a inferência gramatical, os sistemas inteligentes de tomada de decisão, e a busca semântica de informação.

Em todos esses assuntos, o uso de tecnologia adaptativa, em particular o emprego de autômatos adaptativos, tem se mostrado muito adequado, visto que as técnicas adaptativas se mostram bastante adequadas na representação e manipulação de fenômenos como esses, de natureza inerentemente dinâmica.

1.2 Objetivos e método utilizado

A pesquisa relatada nesta tese tem como principal objetivo, identificar alguns caminhos para a aplicação da potencialidade dos formalismos adaptativos – em particular os transdutores adaptativos e a gramática adaptativa – no tratamento dos aspectos dependentes de contexto da linguagem natural, e propor uma arquitetura adaptativa para processamento de linguagem natural

Esta arquitetura adota para os aspectos livres de contexto, pertinentes à estrutura superficial da sentença, o formalismo fundamentado em autômatos de pilha estruturados, e para os aspectos mais complexos das linguagens, mecanismos adaptativos, que permitem explorar ao máximo as potencialidades de desempenho do mecanismo não adaptativo subjacente (NETO, 1993), (NETO, 1998)

1.3 Histórico desta Pesquisa

Em (NETO; MORAES, 2002), relatou-se uma visão preliminar do tratamento de não-determinismos e ambigüidades

Em um exemplo ilustrativo indicou-se a construção de um desses autômatos a partir de uma gramática que define uma aproximação livre de contexto de um pequeno, porém significativo, subconjunto da língua portuguesa.

Em (NETO; MORAES 2003), empregou-se o formalismo gramatical adaptativo no tratamento computacional do problema da concordância nominal. Na ocasião, identificou-se também a aplicabilidade do mesmo método ao tratamento do problema de coordenação, o que levou à determinação de uma solução particular para o tratamento destes problemas.

Procurou-se, a partir de então, investigar o processamento das dependências referenciais (anáforas, pronomes e expressões-R), conjugado àquele referente aos não-determinismos e ambigüidades; partindo-se do pressuposto que o inventário das categorias vazias é paralelo ao das categorias fonéticas(RAPOSO, 1998)., vislumbrava-se que estes problemas tambem seriam resolvidos Constatou-se que o processamento adequado do c-comando (FRANK, 2001), (FRANK, 2004) é crucial.

Tal fato suscitou a busca por uma solução adaptativa para o processamento do c-comando e por um mecanismo adaptativo de cópia identificado em (NETO; MORAES, 2002) como fundamental no tratamento de não-determinismos e ambigüidades.

A partir de então, procurou-se em (NETO, 1993), e (NETO, 1998), onde se apresenta a resolução de problemas que ocorrem no processamento de linguagens de programação, analisar as características inerentes ao formalismo, no que tange à representação e processamento de dependências de contexto.

De fato, o exame detalhado dos mecanismos apresentados nessas referências, permitiu a proposição de uma arquitetura adaptativa para processamento em Linguagem Natural. Como desdobramento, vislumbrou-se uma representação da Linguagem Natural para o usuário desta arquitetura.

1.4 Estrutura da Tese

O primeiro capítulo desta tese faz uma apresentação da pesquisa, aduzindo as suas motivações e finalidades e a descrição da organização do texto que a compõe.

O segundo capítulo contém a exposição de aspectos dos formalismos gramática e autômatos de pilha, cujos desdobramentos foram identificados como primordiais no tratamento de linguagens dependentes de contexto.

Também relatam-se aqui as conclusões obtidas do estudo detalhado de alguns exemplos apresentados em (NETO, 1993): sua aplicação em tarefas como a análise de concordâncias, a delimitação de estruturas coordenadas e subordinadas.

No terceiro capítulo propõe-se uma arquitetura para Processamento em Linguagem Natural, que se fundamenta na existência de um transdutor adaptativo cuja configuração inicial é tal que permite a sua expansão na ocasião em que regras da gramática de uma

Linguagem Natural são fornecidas ao sistema. Como resultado desta expansão, o transdutor atinge uma configuração dotada de uma camada capaz de realizar as tarefas de análise léxica e sintática de sentenças desta Língua.

No quarto capítulo, detalha-se o processamento da especificação de linguagem natural empregando os mesmos mecanismos adaptativos descritos em (NETO, 1993). Justifica-se a adoção desta estratégia pelo desempenho de cada uma destas técnicas.:é: proporcional ao número de símbolos presentes na sentença a ser processada, ao número de ocorrências de determinados símbolos na fita de entrada e eventualmente ao número de símbolos do alfabeto de entrada da Linguagem tratada. Ainda, tais técnicas incluem o tratamento dinâmico e incremental de não-determinismos e ambigüidades.

No quinto capítulo, são apresentadas a descrição e o processamento de algumas dependências sintáticas em linguagem natural. As aplicações, perspectivas e conclusões desta pesquisa são também aduzidas.

Ao final, são apresentadas as referências bibliográficas utilizadas ao longo da realização deste trabalho.

2 FORMALISMO ADAPTATIVO E SEU USO EM LINGUAGEM NATURAL

Este capítulo tem por objetivo apresentar, dentre os diversos aspectos do formalismo adaptativo, em particular da gramática e dos autômatos de pilha, apenas aqueles cujos desdobramentos foram identificados como primordiais no tratamento de linguagens dependentes de contexto.

Também relatam-se aqui as conclusões obtidas do estudo detalhado de alguns exemplos apresentados em (NETO, 1993). Tais exemplos referem-se a problemas encontrados em Linguagens de Programação, mas o que se verificou é que podem ser empregados também no processamento de aspectos dependentes de contexto das Linguagens Naturais, por exemplo, em tarefas como a análise de concordâncias e a delimitação de estruturas coordenadas e subordinadas.

2.1 Autômatos de Pilha Adaptativos

Autômatos de pilha adaptativos são dispositivos reconhecedores que exibem a estrutura básica dos autômatos de pilha, mas que incorporam em adição mecanismos capazes de efetuar alterações dinâmicas em sua própria estrutura, em resposta a cada particular texto de entrada analisado. Tais aceitadores podem ser empregados no reconhecimento de linguagens dependentes de contexto.

Este modelo incorpora três mecanismos de reconhecimento, a saber:

- Máquinas de estados finitos, para o tratamento de formas sintáticas regulares. São representadas pelas sub-máquinas dos autômatos de pilha estruturados que servem como formalismo subjacente do autômato adaptativo;
- Pilhas de controle, que usadas pelas sub-máquinas no controle da análise das formas sintáticas mutuamente recursivas, são fundamentais para o tratamento de formas sintáticas livres de contexto;
- Mecanismos que promovem a auto-modificação dinâmica do autômato adaptativo, destinados primordialmente ao tratamento de dependências de contexto.

Assim, no caso geral todos os recursos do autômato são utilizados no reconhecimento; no caso livre de contexto não-regular o autômato somente aciona a utilização dos recurso de sub-máquinas e pilha e naturalmente no caso de linguagens regulares, o autômato pode também dispensar o uso da pilha.

Um autômato de pilha adaptativo pode ser representado por um conjunto de regras de transição da forma $(\gamma\ \mathbf{g}\ ,\ \mathbf{e}\ ,\ \mathbf{s}\ \alpha)\ \mathbf{:}\ A\ ,\ \rightarrow\ (\gamma\ \mathbf{g'}\ ,\ \mathbf{e'}\ ,\ \mathbf{s'}\ \alpha)\ ,\ B\ ,$ onde:

- $(\gamma\ \mathbf{g}\ ,\ \mathbf{e}\ ,\ \mathbf{s}\ \alpha)$ representa a situação do autômato antes da aplicação da regra:
  - **g** (opcional) indica o conteúdo exigido do topo da pilha antes da execução da regra de transição. Este elemento é desempilhado pela aplicação da regra de transição. No caso de omissão de **g** , a aplicação da regra de transição não será dependente do conteúdo do topo da pilha**.**
  - **e** representa o estado-origem da transição descrita pela regra em questão.
  - **s** (opcional) representa o símbolo da cadeia de entrada a ser consumido pela aplicação da regra. Este elemento pode ser um dos símbolos básicos de que se compõe o texto de entrada, ou então qualquer dos símbolos nele empilhados como conseqüência da execução das regras de transição do autômato adaptativo.

- (γ **g'** , **e'** , **s'** α) representa a situação do autômato depois da aplicação da regra de transição:
  - **g'** (opcional) representa o elemento a ser empilhado como resultado da aplicação da regra de transição. Caso seja omitido, nenhum empilhamento será efetuado como decorrência da execução dessa regra.
  - **e'** representa o estado-destino da transição que a regra de transição descreve.
  - **s'** (opcional) corresponde a um símbolo, a ser incluido na cadeia de entrada como resultado da execução da transição descrita pela regra de transição:

$$(\gamma\, \mathbf{g}\, ,\, \mathbf{e}\, ,\, \mathbf{s}\, \alpha) : \mathbf{A}\, , \rightarrow (\gamma\, \mathbf{g'}\, ,\, \mathbf{e'}\, ,\, \mathbf{s'}\, \alpha)\, ,\, \mathbf{B}$$

Considere-se a situação em que **g = g'**. O mapeamento destas regras de transição para a gramática adaptativa correspondente, emprega o meta-símbolo "$\leftarrow$" introduzido por Iwai, conforme será apresentado adiante.

Cumpre observar que caso **s = s'**, tais elementos deverão aparecer explicitamente na regra de transição, o que então representa uma transição que não efetua alterações sobre a cadeia de entrada, mas que exige, para ser aplicada, que o próximo símbolo de entrada a ser consumido seja **s = s'** , implementando assim uma operação de "look-ahead".

Os elementos **g , g', s** e **s'** explicitam apenas a parte da pilha ou da cadeia de entrada que são relevantes à aplicação da regra de transição.

- **A** (opcional) representa uma ação adaptativa a ser executada antes de ser efetuada a transição de estado.
- **B** (opcional) simboliza uma ação adaptativa a ser executada após a realização da transição de estado.

Há alguns casos a considerar a respeito da interpretação do símbolo **s** nas regras de transição, a saber:

- **s** representa uma cadeia vazia: neste caso, obviamente uma transição não é condicionada ao conteúdo da cadeia de entrada;
- **s** corresponde a um símbolo ASCII denotado entre aspas: isto indica que tal símbolo deve ser utilizado diretamente pela transição. (NETO, 1993)] observa que:

> *"[..].a bem da clareza, convém confinar o uso deste tipo de regras de transição apenas à parte do autômato que executa a função de análise léxica, onde é extensivamente utilizado para a extração de átomos a partir do texto-fonte[...]*"

- **s** pode corresponder a qualquer símbolo que não seja um dos símbolos ASCII, e neste caso, deve-se subentender que este símbolo esteja representando um não-terminal, ao qual deverá estar associada uma sub-máquina que, a partir de algum dos seus estados finais, empilhe na cadeia de entrada o símbolo **s** correspondente. Esclarece o autor a respeito do consumo do símbolo **s** e observa que nas partes de autômato encarregadas da análise sintática, não devem figurar regras de transição que representem transições diretas com caracteres ASCII:

> *[...]Supondo que **I** seja o estado inicial de tal sub-máquina, a regra de transição acima ((e, s) : A → e', B)) deverá ser interpretada como sendo a abreviatura do seguinte par de regras de transição:*
>
> $$(\gamma, \mathbf{e}, \alpha) : A, \rightarrow (\gamma \mathbf{e}, \mathbf{I}, \alpha)$$
> $$(\gamma, \mathbf{e}, \mathbf{s}\ \alpha) : \rightarrow (\gamma, \mathbf{e'}, \alpha), B$$
>
> *desde que se convencione que todos os retornos de sub-máquinas empilhem na cadeia de entrada uma informação, associada ao estado final por onde se deu o retorno, correspondente ao símbolo não-terminal **s** esperado. Note-se que A e B são opcionais, e que a sub-máquina mencionada efetua normalmente a função de um analisador léxico.*

Assim, uma regra de transição da forma (e, s) : A, → e', B representa uma transição de um autômato encarregado da análise sintática. Esta regra supõe que seja realizada uma transição para uma máquina M, analisador léxico, cujo estado inicial é I. O analisador léxico é um transdutor, que poderá mediante o consumo de símbolos presentes na cadeia de entrada,

inserir subseqüentemente um símbolo ***s***. Quando o estado ***e*** for desempilhado, e uma vez que se apresente na cadeia de entrada o símbolo ***s***, inserido pelo analisador léxico, o analisador sintático poderá prosseguir com o reconhecimento a partir do estado *e'*.

- Um outro caso pertinente ao símbolo ***s*** diz respeito às transições envolvendo símbolos formados de caracteres ASCII isolados na parte sintática do autômato. Neste caso, o símbolo ***s*** representa um símbolo ASCII σ, e este não está denotado entre aspas ou, se apresenta denotado na forma ∇ "σ", ou seja, as regras de transição são:

$$(e, \sigma) : A \rightarrow e', B$$

$$\text{ou } (e, \nabla \text{ "}\sigma\text{"}) : A \rightarrow e', B$$

Deve-se entender que tal regra de transição abrevia o seguinte par:

$$(\gamma, e, \text{"}\sigma\text{" } \alpha) : A \rightarrow (\gamma e, I, \text{"}\sigma\text{"}\alpha)$$

$$e\ (\gamma, e, s\alpha) \rightarrow (\gamma, e', \alpha), B$$

com ***s*** representando, portanto, para efeito de análise sintática, um meta-símbolo que simboliza o código ASCII correspondente.

O uso de transições envolvendo símbolos formados de caracteres ASCII isolados na parte sintática do autômato pode ser indicada utilizando-se este tipo de regras de transição em lugar de transições diretas com caracteres ASCII.

Enquanto no caso anterior construções sintáticas da linguagem supõem a ocorrência de uma seqüência de símbolos na cadeia de entrada, neste caso o formalismo prediz a ocorrência de um símbolo terminal isolado. Adianta-se portanto, que um reconhecedor adaptativo poderá apresentar dois modos de operação: aquele em que se consome uma seqüência de símbolos da cadeia de entrada e a subseqüente inserção de um símbolo não-terminal na cadeia de entrada, e aquele em que se consome um único símbolo terminal "σ" e a subseqüente inserção do símbolo σ na cadeia de entrada.

2.2 Gramática adaptativa

A gramática adaptativa é um formalismo generativo capaz de representar linguagens sensíveis ao contexto através de um conjunto dinâmica e espontaneamente modificável de regras de substituição. O que distingue este formalismo das gramáticas livres de contexto convencionais é a capacidade de se auto-modificar à medida que uma sentença da linguagem vai sendo derivada.

As regras da gramática adaptativa podem, opcionalmente, estar associadas a ações adaptativas e assim, quando presentes e ativadas, promovem uma sucessão de gramáticas intermediárias. Se uma sentença ω, pertence à linguagem representada pela gramática adaptativa, sua derivação será finalizada em alguma versão G_f da gramática adaptativa G.

Uma gramática adaptativa G pode ser definida como uma tripla ordenada (G_0, T, R_0), onde:

- T é um conjunto finito, possivelmente vazio, de funções adaptativas
- $G_0 = (VN^0, VT, VC, PL^0, PD^0, S)$ é uma *gramática inicial*, onde:
    - VN^0 é um conjunto finito e não vazio de símbolos não terminais;
    - VT é um conjunto finito e não vazio de símbolos terminais;
    - $VN^0 \cap VT = \varnothing$.
    - VC é um conjunto finito de *símbolos de contexto*.
    - $V^0 = VN^0 \cup VT \cup VC$
    - VN^0,VT e VC são conjuntos disjuntos dois a dois.
    - $S \in VN^0$ é o símbolo inicial da gramática.
    - PL^0 é o conjunto de regras de produção aplicáveis às situações livres de contexto.
    - PD^0 é o conjunto de regras de produção aplicáveis às situações dependentes de contexto.

As regras de produção da Gramática adaptativa consistem de expressões com os seguintes formatos, sendo i um indicador do número de alterações adaptativas já sofridas pela gramática inicial:

Tipo 1: pertencentes ao conjunto PL^i, com $i \in \mathbf{N}$:

$N \rightarrow \{A\}\ \alpha$

onde $\alpha \in (VT \cup VN)^*$, $N \in VN^i$ e A é uma ação adaptativa opcional associada à regra de produção.

Tipo 2: pertencentes ao conjunto PL^i, com $i \in \mathbf{N}$:

$N \rightarrow \varnothing$

onde $\varnothing$ é um meta-símbolo que indica o conjunto vazio. Esta regra de transição indica que, embora o símbolo não-terminal N esteja definido, deriva um conjunto vazio, ou seja, não há qualquer substituição prevista para esse não-terminal. Isto significa que, se esta regra for aplicada em alguma derivação, a gramática não gerará qualquer sentença. Esta regra é utilizada para o caso em que na gramática existirem regras que referenciem não-terminais que deverão ser dinamicamente definidos, como resultado da aplicação de alguma ação adaptativa.

Tipo 3: pertencentes ao conjunto PD^i, com $i \in \mathbf{N}$:

$\alpha\ N \leftarrow \{A\}\ \beta\ M$, onde $\alpha \in VC \cup \{\varepsilon\}$ e $\beta \in VC$, ou

$\alpha\ N \rightarrow \{A\}\ \beta\ M$ onde $\alpha \in VC$, $\beta \in VT \cup \{\varepsilon\}$, N e $M \in VN^i$, com A opcional.

As regras de produção da primeira forma, que apresentam o meta-símbolo "$\leftarrow$", indicam que β está sendo injetado na cadeia de entrada. Esta regra substitui αN por βM, inserindo assim na cadeia de entrada a informação de contexto representada por β.

As regras de produção do segundo formato exibem uma seta "$\rightarrow$" no sentido convencional, mas possuem no seu lado esquerdo um símbolo de contexto seguido por um

símbolo não-terminal. Isto indica que αN é substituído por βM, sendo a informação de contexto β inserida na cadeia de entrada.

$R^0 \subseteq P^0 \times (T \cup \{\varepsilon\})$ é uma relação do tipo (r, A), onde $r \in (PL^0 \cup PD^0)$ e $A \in T$.

Para cada regra de produção r, a relação R^0 associa uma ação adaptativa A correspondente.

Uma regra de produção à qual está associada uma ação adaptativa é denominada *regra de transição adaptativa*.

A expressão que define uma regra de produção do tipo 1 da gramática adaptativa é do tipo livre de contexto a menos da ação adaptativa. Esta, em conjunto com as regras de produção em PD^0, responde pela representação das dependências de contexto nesta gramática.

A equivalência entre autômatos adaptativos e o das gramáticas adaptativas é demonstrado em (IWAI, 2000) através dos seguintes teoremas:

- *O resultado da execução de qualquer ação adaptativa é equivalente a uma seqüência não-vazia de ações adaptativas elementares*
- *Se uma linguagem é gerada por uma gramática adaptativa, então ela é aceita por algum autômato adaptativo.*
- *Se uma linguagem gerada por um não-terminal corresponde a uma submáquina, então essa linguagem pode ser definida por uma gramática cuja raiz é o não-terminal em questão.*
- *Se uma linguagem é aceita por um autômato adaptativo, então ela pode ser gerada por alguma gramática adaptativa.*

A autora também apresenta um roteiro para o mapeamento dos conjuntos de regras de produção, normais e de contexto, da gramática adaptativa para um conjunto equivalente de regras de transição do autômato adaptativo que reconhece a mesma linguagem, e descreve os seguintes algoritmos:

- conversão canônica da gramática adaptativa para o autômato adaptativo;
- conversão canônica do autômato adaptativo para a forma da gramática adaptativa;
- um algoritmo eficiente para a obtenção de um analisador sintático a partir de gramáticas adaptativas;
- um algoritmo para a construção do autômato.

Foram relevantes ao estudo do formalismo adaptativo as seguintes observações:

- Os símbolos de contexto de uma gramática adaptativa estão associados a um particular símbolo não-terminal da gramática;
- Um único terminal da gramática pode estar associado a um símbolo de contexto e portanto a um particular símbolo não-terminal da gramática;
- Uma sentença inicialmente constituída de símbolos terminais é *modificada* ao longo de sua derivação através da inserção de símbolos de contexto, passíveis de serem utilizados, ao longo da derivação, da mesma forma que os símbolos terminais.

Na próxima seção são apresentados mecanismos adaptativos para o tratamento adequado de linguagens dependentes de contexto. Tais exemplos refletem o uso intenso do que foi exposto até o momento.

2.3 Mecanismos adaptativos para o tratamento de linguagens dependentes de contexto

Este trabalho procurou investigar a potencialidade do formalismo adaptativo para a representação e tratamento de dependências sintáticas em Linguagem Natural. Do estudo dos formalismos mencionados anteriormente, constatou-se serem apropriados para o desenvolvimento de diversos dos elementos necessários ao processamento de dependências de contexto, características das linguagens naturais.

Com o intuito de determinar precisamente uma forma adequada de utilização das técnicas anteriormente desenvolvidas, para o processamento de linguagens naturais, procedeu-se a um estudo aprofundado dos exemplos aduzidos em (NETO, 1993), originalmente projetados contemplando a análise de Linguagens de Programação, e do estudo dos mecanismos empregados nesses exemplos, identificaram-se procedimentos, que foram adotados como modelos para o tratamento de Linguagens Dependentes de Contexto quaisquer, e no particular caso desta tese, de Linguagens Naturais.

No texto que se segue tecem-se considerações a respeito destes procedimentos e sua origem.

2.3.1 Simulador de uma Pilha

Os autômatos adaptativos podem apresentar transições que geram uma cadeia de elementos *de um alfabeto de saída pré-determinado.* Trata-se do conceito de transdutor adaptativo. Tais transições podem ser aquelas já existentes na configuração inicial do autômato ou podem ser aquelas resultantes das auto-modificações dos utômatos ocorridas ao longo do reconhecimento de uma linguagem.

Diferentemente dos formalismos Máquina de Mealy ou Máquina de Moore, os elementos pertencentes *ao alfabeto de saída* e gerados pelo transdutor adaptativo podem ser armazenados na cadeia de entrada.

A gramática adaptativa como formalismo dual ao dos autômatos adaptativos, exibe em sua definição um conjunto de símbolos de contexto e tipos de regras que permitem a inserção ou extração destes símbolos ao longo da derivação de uma sentença de entrada.

O exemplo que se segue corresponde àquele com o mesmo sub-título apresentado em (NETO, 1993), mapeado para o formalismo gramatical. Acrescentaram-se ainda ao exemplo dois símbolos de contexto, a saber: *beta* e *∇_beta*.

Este exemplo, apesar de simples, mostra a derivação de uma sentença pertencente a uma linguagem *livre de contexto*. O seguinte conjunto de regras gera, sobre o alfabeto V_T = {"(", ")", "β"}, a linguagem L(w) = { w | w = (^{n} β)^n, n ≥ 0 }:

N0 → *beta* N1
N1 → "β" N4
N1 →{A(N2,N3,N1)} "(" N2
N2 → β N3
N3 → ")" N4
N4 ← *∇_beta* N5
N5 → ε

A declaração da ação adaptativa A é apresentada a seguir:

$A(I_N, J_N, N_n)$ = { K_n*, M_n*:
+[K_n → "β" M_n]
+[M_N → ")" J_N]
+[I_N → {$A(K_N, M_N, I_N)$} "(" K_N]
-[N_N → {A (I_N, J_N, N_N)} "(" I_N]
+[N_N → "(" I_N] }

Observe-se que o símbolo "(" é um terminal da linguagem que delimita um não-determinismo. As regras de produção são especificadas de tal forma que, quando da ocorrência de n símbolos "(", a gramática se auto-modifica permitindo que a derivação prossiga normalmente, tanto em uma subseqüente ocorrência do símbolo β, como em uma

subseqüente ocorrência de outro símbolo "(". Isto é efetuado graças à execução da ação adaptativa A, que por sua vez cria dinamicamente uma regra adaptativa para o eventual consumo para o símbolo "(" e uma regra não-adaptativa para o consumo do símbolo β. A essas regras estão associados dois novos não-terminais, contornando-se assim, incremental e dinamicamente, o não-determinismo original, segundo as necessidades decorrentes da ocorrência dos símbolos da sentença em análise, presentes na cadeia de entrada.

Assim, a presença do símbolo de contexto *beta* na cadeia de entrada dá início à derivação da parte da cadeia de entrada que lhe sucede. Ao final da derivação, o analisador sintático deverá substituir o símbolo de contexto *beta* e a sentença w ∈ L(w) pelo mesmo símbolo de contexto, porém denotado como ∇*_beta.*, indicando que um trecho da cadeia de entrada foi analisada sintaticamente e classificada como do tipo ∇*_beta.*

Avaliação de complexidade:

Sendo n o número de ocorrências do símbolo "(" em uma sentença original em análise, a sua derivação, segundo a gramática apresentada, exige a execução de 2*n + 3 regras gramaticais elementares e implica na execução de 5*n operações de criação de regras, ficando assim a gramática original ampliada em 3*n regras.

Este exemplo ilustra o desempenho com que o formalismo adaptativo pode reconhecer linguagens livre de contexto, sem aumentar a sua complexidade no tempo e no espaço, quando comparado com o formalismo do autômato de pilha.

Como será discutido adiante, o mesmo princípio identificado neste exemplo norteia a criação de novas instâncias da gramática, uma vez que β pode representar, em vez de um único símbolo, toda uma sub-cadeia.

Tem-se também que *beta* é um símbolo de contexto associado à gramática original. Uma vez presente na cadeia de entrada, o seu consumo poderá promover o empilhamento do

estado do reconhecedor que o ativou. Por outro lado, a substituição na cadeia de entrada por *beta* e a palavra w ∈ L(w) pelo *token* ∇_*beta* pode ser realizada com a recuperação do estado do reconhecimento anterior á ocorrência do símbolo *beta*, pela remoção da pilha explícita onde fora interrompido o reconhecimento.

2.3.2 Alteração dinâmica dos atributos de símbolos não-terminais de uma gramática.

Um dos mecanismos adaptativos a serem empregados na descrição e tratamento das linguagens dependentes de contexto é a alteração dinâmica dos atributos associados aos símbolos não-terminais da gramática.

Para executar essa tarefa, ao longo do processo de derivação de uma sentença em que se apresenta um símbolo de contexto representando um determinado atributo, esse símbolo pode figurar isoladamente na cadeia de entrada e ser memorizado, por exemplo, como argumento de uma ação adaptativa, o que permitirá que, posteriormente, essa informação seja recuperada e utilizada.

Considere-se abaixo o trecho de gramática que descreve simplificadamente a categoria gramatical NP (grupo nominal), cujo núcleo é um substantivo.

Admita-se que o não-terminal Y1 se refira a um transdutor que seja capaz de extrair da sentença os símbolos terminais da gramática, e, portanto, de substituí-los por símbolos de contexto, tais como σ_sa se se tratar de um sintagma adjetivo que antecede um substantivo, ou α_subst_pm, se se tratar de um substantivo plural masculino, α_subst_pf se for plural feminino, e assim sucessivamente.

Iguamente, suponha-se que o não-terminal Y2 corresponda a um transdutor capaz de extrair da sentença de entrada os símbolos terminais da linguagem e de substituí-los por

símbolos de contexto, tais como σ_sa, ou seja, um sintagma adjetivo que sucede a ocorrência de um substantivo.

NP0 → { C(NP_Void) } Y1 NP1
σ_sa NP1 → NP0
α_subst_pm NP1 → { C(NP_PM) } NP2
α_subst_pf NP2 → { C(NP_PF) } NP2
α_subst_sm NP1 → { C(NP_SM) } NP2
α_subst_sf NP1 → { C(NP_SF) } NP2
α_x NP1 ← { C(NP_Vazio) } α_x NP2
NP2 → Y2 NP3
σ_sa NP3 → NP2

Uma vez que, ao longo da derivação, são memorizados os atributos do substantivo, no argumento da ação adaptativa C, esta é executada para alterar a regra que define o tipo da classe gramatical NP. Para tanto, a definição de C supõe a utilização de um não-terminal, denominado, por exemplo, 50_TipoAtual, que indica o tipo da construção sintática que está sendo derivada – no caso, o grupo nominal NP.

C(Tipo) = {X:
-[50_TipoAtual → X]
+[50_TipoAtual → Tipo]}

O símbolo de contexto α_x indica qualquer símbolo na cadeia de entrada, que não seja um substantivo, e sua utilização permite interpretar a ausência de um substantivo como a ocorrência, pela omissão associada a essa ausência, de uma construção sintática da ***categoria vazia***.

Este exemplo ilustra uma forma como se pode transferir os atributos de um símbolo de uma determinada hierarquia na gramática para outro de hierarquia superior (neste caso, a

transferência de um atributo associado a um símbolo pertencente a uma categoria lexical para a categoria gramatical da qual ele é o núcleo) e também uma possível forma de se detectar situações em que se deva interpretar como sendo a ocorrência sintática legitimamente classificável como sendo da categoria vazia.

Note-se que esta técnica é bastante geral, e independe da hierarquia dos símbolos envolvidos na derivação, de sua natureza, bem como dos atributos a eles associados.

Avaliação de complexidade:

A atribuição do atributo de gênero e número da classe lexical ao grupo nominal, depende apenas de sua detecção na cadeia de entrada, visto que sua determinação a partir do texto da sentença em análise é da competência do analisador léxico, encarregado de determinar a categoria das palavras e de verificar suas flexões, destas extraindo atributos tais como o gênero e o número a elas associados. Assim, o número de regras da gramática não é alterado pela execução da ação adaptativa C.

2.3.3 Concordância de Atributos entre elementos imediatos

A solução aqui aduzida implementa a verificação da concordância entre os atributos de dois símbolos imediatos presentes na sentença em análise, e é inspirada no exemplo 4.4.6 de (NETO, 1993).

Para ilustrar o mecanismo, apresenta-se um conjunto de regras que verifica a concordância entre a categoria gramatical DP do sujeito de uma oração e a categoria gramatical VP da mesma oração, em gênero, número e pessoa.

Pelas convenções anteriormente discutidas, a ocorrência de uma sub-cadeia, classificada como pertencente à categoria gramatical DP, está associada à presença de um símbolo de contexto na cadeia de entrada.

O mesmo se dá com uma sub-cadeia, marcada como sendo da categoria VP. Assim, uma vez que ocorra na sentença de entrada o correspondente símbolo de contexto, os atributos associados podem ser memorizados como argumentos de uma função adaptativa, neste caso a função FCV.

A execução da função FCV promoverá a alteração do conjunto de regras de produção, de forma que sejam passíveis de derivação apenas o sintagma verbal ou o grupo determinante que apresentem atributos concordantes com a categoria primeiramente presente na oração.

Após a execução da ação adaptativa FCV, é executada outra ação adaptativa, que efetua a restauração do conjunto inicial das regras, preparando a gramática para receber uma nova construção sintática similar.

Uma concordância verbal dessa natureza pode ser gramaticalmente especificada, por exemplo, conforme o conjunto de regras abaixo, as quais contemplam uma eventual inversão do sujeito.

Os símbolos de contexto que se apresentarem na cadeia de entrada referem-se ao grupo funcional Determinante (σ_dp_pessoa_número) e ao grupo gramatical Verbal (σ_vp_pessoa_número). Como as regras impõem que haja concordância entre pessoa e o número das construções sintáticas já classificadas como tais, o símbolo de contexto σ_? , presente em algumas regras, age como uma chave, que permite que a derivação prossiga ou não.

CV1 → { FCV (σ_vp_1s) } σ_dp_1s CV2
CV1 → { FCV (σ_vp_2s) } σ_dp_2s CV2
CV1 → { FCV (σ_vp_3s) } σ_dp_3s CV2

$CV1 \rightarrow \{ FCV\ (\sigma\_vp\_1p) \}\ \sigma\_dp\_1p\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_vp\_2p) \}\ \sigma\_dp\_2p\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_vp\_3p) \}\ \sigma\_dp\_3p\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_dp\_1s) \}\ \sigma\_vp\_1s\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_dp\_2s) \}\ \sigma\_vp\_2s\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_dp\_3s) \}\ \sigma\_vp\_3s\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_dp\_1p) \}\ \sigma\_vp\_1p\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_dp\_2p) \}\ \sigma\_vp\_2p\ CV2$

$CV1 \rightarrow \{ FCV\ (\sigma\_dp\_3p) \}\ \sigma\_vp\_3p\ CV2$

$CV2 \rightarrow \sigma\_?\ CV3$

$\theta\ CV3 \leftarrow \theta\ CV4 \quad \forall\ \theta \in \Sigma$

$CV4 \leftarrow \sigma\_CV\ CV5$

$CV5 \rightarrow \varepsilon$

A execução da Função Adaptativa FCV – função Concordância Verbal – habilita a inserção do símbolo σ_CV, indicador de concordância verbal na cadeia de entrada, mediante a alteração das regras $CV2 \rightarrow \sigma\_?\ CV3$ e $\theta\ CV3 \leftarrow \theta\ CV4$, conforme a descrição seguinte:

$FCV\ (x) =$ { GCV_aux*:

-[$CV2 \rightarrow \sigma\_?\ CV3$]

+[$CV2 \rightarrow x\ CV3$]

-{[$\theta\ CV3 \leftarrow \theta\ CV4$] $\forall\theta \in \Sigma$}

+{[$\theta\ CV3 \leftarrow$ {Restaura_CV(x)} GCV_Aux] $\forall\theta \in \Sigma$}

+[$GCV\_Aux \leftarrow CV4$] }

A função adaptativa Restaura_CV promove a restauração da gramática para a sua configuração inicial, no que diz respeito à concordância verbal, mantendo a gramática receptiva a novas instâncias de construções sintáticas da mesma natureza, nos escopos onde isso se faça necessário.

Restaura_CV (x) = {

-[CV2 → x CV3]

+[CV2 → σ_? CV3]

-{[θ CV3 ← θ CV4] ∀θ ∈ Σ}

-{[θ CV3 ← {Restaura_CV (x)} GCV_Aux] ∀θ ∈ Σ}

+{[θ CV3 ← θ CV4] ∀θ ∈ Σ} }

Observe-se que, após a execução da função FCV, seguida da execução de Restaura_CV, o número de regras inicialmente presentes na gramática fica inalterado.

Por tratar-se de meras manipulações simbólicas, as alterações e restaurações do conjunto de produções presentes na gramática, em cada um desses passos, não serão aqui desenvolvidas por extenso.

2.3.4 Mecanismo Adaptativo para tratamento de escopos aninhados

Por analogia com o procedimento inspirado no exemplo 4.4.7 de (NETO, 1993), intitulado "Analisador Léxico para uma linguagem estruturada em blocos", pode-se, através de uma série de interpretações e adaptações à realidade das linguagens naturais, inferir um método de análise para a delimitação dinâmica de escopos aninhados.

Também com base na mesma fonte pode-se conceituar, para o analisador sintático, dois modos de operação, a saber: o modo de "treinamento" e o modo de "uso".Um escopo é iniciado em modo "treinamento", e a partir de então os elementos da cadeia de entrada vão sendo extraídos e memorizados.

Na ocorrência de um símbolo, presente na cadeia de entrada, mas que não esteja previsto na sub-gramática da linguagem associada àquele escopo, o modo de operação do analisador sintático é alterado para o modo "uso".

Neste modo, a gramaticalidade de construções sintáticas constituídas dos símbolos recém-memorizados, podem ser verificadas, analisadas e eventualmente um escopo subordinado ao primeiro pode ser identificado e sua análise, iniciada.

O modo "uso" é finalizado quando todas as estruturas subordinadas ao escopo corrente tiverem sido tratadas, e a gramaticalidade da construção, analisada. Nesta situação, o escopo é finalizado. Em caso contrário, detecta-se um erro de sintaxe..

A seguir, detalham-se alguns aspectos desse mecanismo.

2.3.4.1 Delimitação dinâmica de escopos aninhados.

Admita-se a sub-gramática inicial indicada em (1).

(1) $I0 \rightarrow V0$

$SP \rightarrow V$

$SP \rightarrow \{Z(SP)\}\ SP$

A Fig.1 apresenta a configuração inicial de um autômato adaptativo correspondente às regras elementares em (1).

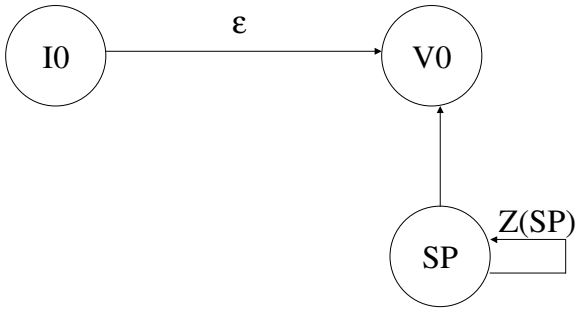


Fig. 1 – Máquina Inicial de uma estrutura auxiliar para armazenar informações em escopos aninhados

I0 deve apontar sucessivamente para os estados, I_1, I_2, ... I_n, estados iniciais para a memorização e análise da informação E_1, E_2, …, E_n.

A i-ésima informação E_i é armazenada entre os estados I_i e N_i. A ação adaptativa Z., memoriza o topo da pilha assim simulada.

O estado direito da informação N_i deverá apontar dinamicamente para o estado I_{i-1}.

O ponteiro da pilha SP, aponta sempre para o estado final N_i e é apontado por uma lista de estados N_{i-1}.

Tal lista, dessa forma, mantém a informação de quais escopos estão ainda sendo analisados pelo autômato.

Assim, na ocasião em que for detectado o início de um novo escopo sintático na sentença em análise, a seguinte função adaptativa de inserção deverá ser executada:

Inserção (I0, N0, SP) = {i_anterior, n_anterior, m_anterior, I_n*, N_n*, M_n*,

-[(I0 → i_anterior)]

+[(I0 → I_n)]

+[(N_n → i_anterior)]

-[(SP → n_anterior)]

+[(SP → N_n)]

-[m_anterior→ {Z(n_anterior)} SP]

+[m_anterior → {Z(n_anterior)} M_n]

+[M_n → {Z(n_anterior)} SP] }

Avaliação de complexidade:

Observe-se que para cada informação E_n a ser armazenada,, são criadas cinco novas regras e simultaneamente, são removidas três regras elementares, uma vez que a ação adaptativa Z(ultimo) nada realiza além de memorizar o último não-terminal do escopo ancestral.

Em outras palavras, para tratar q informações E_n, o número de regras na gramática cresce em 2*q regras, e ainda incorpora 3*q novos não-terminais.

A Fig. 2 apresenta a configuração do autômato para a inserção da informação E_1:

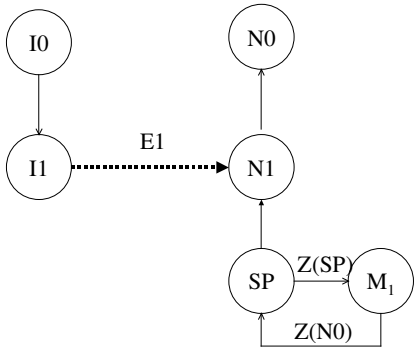


Fig. 2 - Configuração do autômato para inserção da informação E_1

Analogamente, para se remover uma informação E_n, deve-se executar a função adaptativa Remoção, descrita a seguir:

Remoção (I0, SP) = {i_atual, n_atual, i_anterior, m_atual, m_anterior:

-[(I0 → i_atual]

-[(SP → n_atual)]

-[(n_atual → i_anterior)]

+[(I0 → i_anterior)]

-[(m_atual →{ Z(n_anterior) SP]

+[(SP → n_anterior)]

-[(m_anterior → { Z(n_anterior)} m_atual]

+[(m_anterior → Z(n_anterior) SP] }

Avaliação de complexidade:

Para cada informação que seja desconsiderada pelo analisador em algum instante da derivação, duas regras da gramática são removidas. O tempo gasto para isso é portanto constante, e igual ao tempo de execução de oito ações elementares.

2.3.4.2 Extração e Memorização de uma seqüência de símbolos

Neste item, comentam-se os mecanismos adaptativos que podem ser empregados na construção de um analisador léxico para linguagens dependentes de contexto, em particular, para linguagens naturais.

A partir de uma sentença de entrada, originalmente constituída de terminais da gramática ,os mesmos são extraídos, memorizados, classificados e simultaneamente, uma gramática inicial com um conjunto de regras P^0 é alterada através da execução de particulares e adequadas ações adaptativas.

O conjunto de regras assim obtido deverá ser usado para a posterior e efetiva análise da gramaticalidade da sentença.

De imediato percebe-se a necessidade de que a gramática dinamicamente modificada apresente duas regras mutuamente exclusivas, as quais permitem que ora a mesma seja empregada em modo “treinamento”, com a extração e memorização de símbolos, ora no modo “uso”. Trata-se das seguintes regras:

97_Modo → 98_Treino

97_Modo $\rightarrow$ 99_Uso

Um conjunto inicial de regras de produção que se propõe é o seguinte:

$$P^0 = \{ \quad 1^0.\ I \rightarrow \{A_0\ (I, J, K, \theta, N)\}\ \theta\ J \quad \forall\ \theta \in V_T$$
$$2^0.\ J \rightarrow \{A_1\ (I, J, K, \theta, N)\}\ K$$
$$3^0.\ N \rightarrow \varnothing \quad \}$$

V_T ó conjunto de todas as entradas lexicais pertencentes a uma determinada linguagem;

A_0 e A_1 (descritas mais adiante) são as ações adaptativas que efetivamente possibilitam a memorização das entradas lexicais e a alteração do conjunto de regras, respectivamente.

A ação adaptativa A_0 memoriza em um de seus argumentos o símbolo presente na sentença em processo de derivação, e modifica os argumentos da ação adaptativa A_1 presente na regra seguinteCabe aqui salientar a notação "$\forall\ \theta \in V_T$" , presente na regra 1^0, pela definição original (NETO, 1993) deve representar um conjunto de regras elementares, uma para cada elemento $\theta \in V_T$.

A ação adaptativa A_1 é executada apenas no modo "treinamento". Esta ação adaptativa permite que sejam criadas regras de produção específicas a partir de cada terminal presente na sentença original em análise.

Uma vez que se encontre no texto de entrada um símbolo que não pertença ao alfabeto da Linguagem, o autômato se auto-modifica de forma que possa ser empregado no modo "uso".: a transição associada à ação adaptativa A_1 é substituída por uma transição associada à ação adaptativa A_{39}. O autômato assim modificado é disponiblizaso no modo "uso" Neste modo de operação, a execução da ação adaptativa A_0 alterará os argumentos da ação adaptativa A_{39}, de forma análoga ao que ocorre no modo "treinamento".

Sendo ϕ um símbolo que não pertence ao conjunto de terminais da linguagem, para um fragmento de sentença de entrada, tal como: *a bela praia* ϕ, as seguintes regras de produção podem ser criadas, como resultado da execução da ação adaptativa A_1:

$I \rightarrow$ "a" GJ1
GJ1 $\rightarrow$ "bela" GJ2
GJ2 $\rightarrow$ "praia" GJ3

Naturalmente, para cada novo símbolo terminal extraído, um novo não-terminal também vai sendo criado. A execução da ação adaptativa A_1 gera um conjunto de regras de produção adaptativas tais que:

- seja possível que as palavras recém-identificadas na cadeia de entrada possam ser classificadas e, por conseguinte, inseridos na cadeia de entrada os símbolos de contexto associados;
- seja memorizado o não-terminal recém-criado, em correspondência ao terminal lido da sentença de entrada;
- seja viável a transição do modo "treinamento" para o modo "uso";
- um determinado símbolo seja analisado em diversos segmentos da gramática;

Seguem-se as descrições das funções adaptativas que implementam o mecanismo adaptativo proposto:

Função Adaptativa A_0

A_0 (I, J, K, σ, N) = {M, N, P, Q, R, S, T, U, V, W, X, Y:
/* insere argumentos na chamada de A_1, se existir */
- [J $\rightarrow$ {A_1(M, N, P, Q, R)} Y]

$+[J \rightarrow \{A_1(I, J, K, \sigma, N)\}\ Y]$

/* insere argumentos na chamada de A_{39}, se existir */

$- [\ X \rightarrow \{A_{39}(S, T, U, V, W)\}\ K]$

$+[\ X \rightarrow \{A_{39}(I, J, K, \sigma, N)\}\ K]$ }

Avaliação de complexidade:

A execução da ação adaptativa A_0 não altera o número de regras presentes na gramática, e seu tempo de execução é constante, e corresponde ao de quatro ações adaptativas elementares.

Função Adaptativa A_1

$A_1(I, J, K, \sigma, N) =$ {L, M, P, Q, R, GJ*, GN*, GS*, T*:

/* substitui a regra associada à ação adaptativa A_0, se existir*/

$-[I \rightarrow \{A_0\ (L,M, P, Q, R)\ \}\ \sigma\ J]$

/* cria uma regra particular para o símbolo terminal recém-encontrado */

$+[I \rightarrow \sigma\ GJ]$

/*cria regras para que novos terminais sejam identificados na sentença de entrada*/

$\{+[GJ \rightarrow \{A_0(GJ, J, K, \theta, GN)\}\ \theta\ J]\ \forall\ \theta \in \Sigma_T\}$

/* A derivação deverá portanto prosseguir a partir da regra recém-criada. No entanto, o processo de derivação se encontra no nó K. Assim sendo, o processo de derivação deverá ser conduzido do nó K para o nó GJ. Esta regra se torna desnecessária tão logo for aplicada, por isso a ela é associada a uma ação de auto-remoção: */

$+[K \rightarrow \{AutoRemove\_1(K, GJ)\}\ GJ]$

/* cria regra que permitirá a posterior inserção na cadeia de entrada do símbolo terminal recém-encontrado e memorizado */

$+[GN \leftarrow \sigma N]$

/* cria regrasque permitem a derivação de símbolos $\theta \notin V_T$ */

$\{+[\theta\ GJ \leftarrow \theta\ GS\ ]\ \forall\ \theta \notin V_T\}$

/* As seguintes ações elementares completam o tratamento, pela ação adaptativa A_1, dos símbolos não pertencentes ao alfabeto da linguagem, fazendo uso das ações adaptativas A_{42} e A_{43}, adiante descritas */

+[GS → {A_{42} (GS, GT, GN)} G_Aux]

+[G_Aux → GT]

+[GT → {A_{43}(GS, GT, GN, GJ)} VOID_8] }

Avaliação de complexidade:

Para n símbolos terminais presentes na sentença, a execução da ação adaptativa A_1 acrescenta n*(5+l+m) regras na gramática, onde l é o número de símbolos do alfabeto da linguagem em reconhecimento e m é o número de símbolos não pertencentes ao alfabeto, porém passíveis de ocorreren na cadeia de entrada.

A seguir, comentam-se as funções adaptativas A_{42}, A_{43} e A_{39}, referenciadas nas produções das ações adaptativas A_1 e A_0.

Função Adaptativa A_{42}

A função adaptativa A_{42} deve ser ativada apenas quando a gramática está sendo usada para a análise de símbolos presentes na sentença de entrada. Quando executada, permite que uma ou mais palavras sejam buscadas na árvore de derivação, para análise em outros contextos.

Segue-se a descrição da função adaptativa A_{42}.

A_{42} (GS, GT, GN) = {y:

/* verifica se a gramática está sendo empregada em modo uso */

?[(97_Modo → y 99_Uso]

/* Se a gramática estiver em modo uso, elimina as seguintes regras da gramática: */

-[GS → {A_{42} (GS, GT, GN)} y G_Aux]

-[G_Aux → yGT]

-[GT → { A_{43}(GS, GT, GN, GJ) } y VOID_8]

/* Criação de regra, condicionada ao modo de classificação */

+[GS → y GN] }

Avaliação de complexidade:

Para cada símbolo da sentença, a execução da ação adaptativa A_{42} elimina três regras, substituindo-as por uma única regra, portanto sua execução promove uma redução de duas regras no conjunto de produções da gramática. Seu tempo de processamento de um símbolo é constante, e corresponde à execução de cinco ações adaptativas elementares.

Função Adaptativa A_{43}

A função adaptativa A_{43} deve ser ativada apenas em tempo de extração e memorização de símbolos. Sua execução permite que símbolos terminais na cadeia de entrada, sejam substituídos por símbolos de contexto, desde que haja uma consulta à regra 50_TipoAtual → Y, onde Y é o símbolo não-terminal associado a um determinado tipo do terminal recém-identificado na cadeia de entrada. Assume-se aqui que a palavra atributo, abrange também a classificação do terminal .

A_{43} (GS, GT, GN, GJ) = {X, Y, G_AUX, Y_AUX:

/* verifica se se trata de modo de extração de símbolos*/

?[97_Modo→ X 98_Treino]

/* verifica qual o não-terminal que deve ser associado à cadeia de entrada */

?[50_TipoAtual → Y]

?[Y_AUX → Y]

/* remove a regra que ativa A_{43} */

-[GT → {A_{43}(GS, GT, GN, GJ)} VOID_8]

/* remove as regras associadas à ação adaptativa A_{42} */

-[GS → {A_{42} (GS, GT, GN)} G_Aux]

-[G_Aux → GT]

/* insere regras para classificar cadeia de entrada como sendo do tipo Y */

+[GS → {A_SbAt(GN)} Y_AUX]

+[Y_AUX → Y]

/* marca o não-terminal criado */

A_SbAt(GN) }

Avaliação de complexidade:

Para cada símbolo presente na sentença de entrada, três regras da gramática são removidas e outras duas são acrescentadas, resultando em uma redução de uma regra no conjunto de produções da gramática. .A execução da ação adaptativa A_SbAt não altera o número de regras na gramática. O tempo de execução desta função adaptativa é constante, e corresponde ao tempo de execução de oito ações adaptativas elementares, acrescido do tempo de execução de A_SbAt, ou seja, do tempo de execução de duas ações adaptativas elementares.

Função Adaptativa A_{39}

A única regra presente em A_{39} implica que o primeiro símbolo que foi detectado na cadeia de entrada, e que ativou o modo de "uso" das regras, é inserido na cadeia de entrada, no escopo cujo início é demarcado pelo não-terminal K. Uma vez empregada esta regra , é imediatamente removida.

A_{39} (I, J, K, σ, N) = {+[K ← {AutoRemove_39(K,N) } σ N]}

AutoRemove_39 (X, Y) = {-[X ← {AutoRemove_39(X,Y)} σ Y]}

Avaliação de complexidade:

A execução da ação adaptativa A_{39}, por estar associada a uma ação adaptativa de auto-remoção, não altera globalmente o número de regras da gramática. Naturalmente, sua execução depende apenas do número de alterações do modo de memorização para o modo de uso das regras.

Função Adaptativa A_SbAt

Esta função adaptativa recebe como parâmetro o não-terminal N associado à regra criada a partir do terminal θ:

A_SbAt (N_θ) = {X:

-[N90 → X]

+[N90 → N_θ] }

Para ilustrar a execução dos mecanismos descritos, A Fig. 3 apresenta um autômato criado para a memorização de uma seqüência de três símbolos na cadeia de entrada.

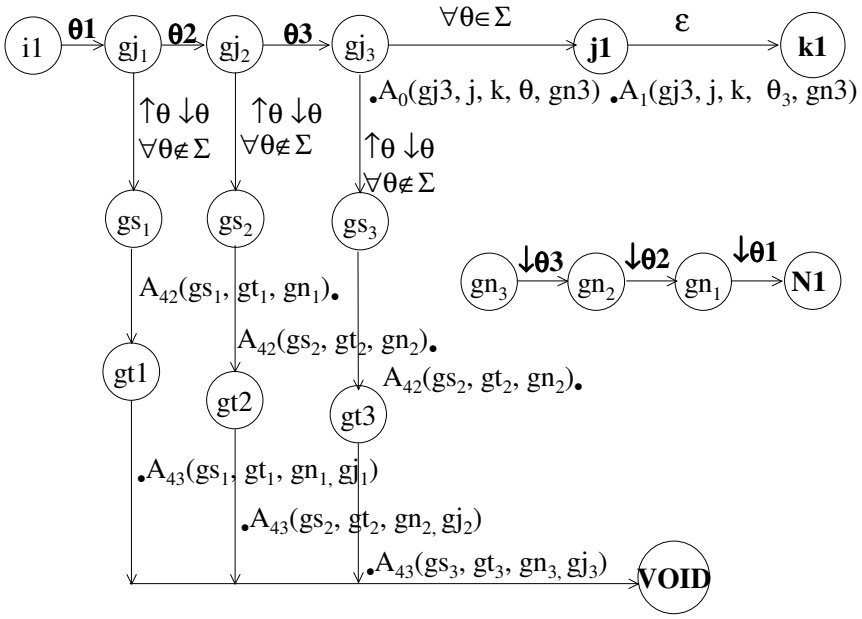


Fig. 3 - Memorização da seqüência $\theta_1\theta_2\theta_3$

Note-se que, as listas parciais são incrementalmente construídas, de tal forma que não-determinismos, eventualmente presentes em uma cadeia de símbolos, possam ser adequadamente tratados.

Por outro lado, considere-se que, no modo "treinamento" após a ocorrência dos símbolos $\theta_1\theta_2\theta_3$ seja encontrado na cadeia de entrada um símbolo não pertencente ao alfabeto, indicando o final da cadeia de entrada que está sendo armazenada. Tal símbolo é devolvido para a cadeia de entrada e a ação adaptativa A_{43} é executada. O autômato resultante é apresentado na Fig. 4.

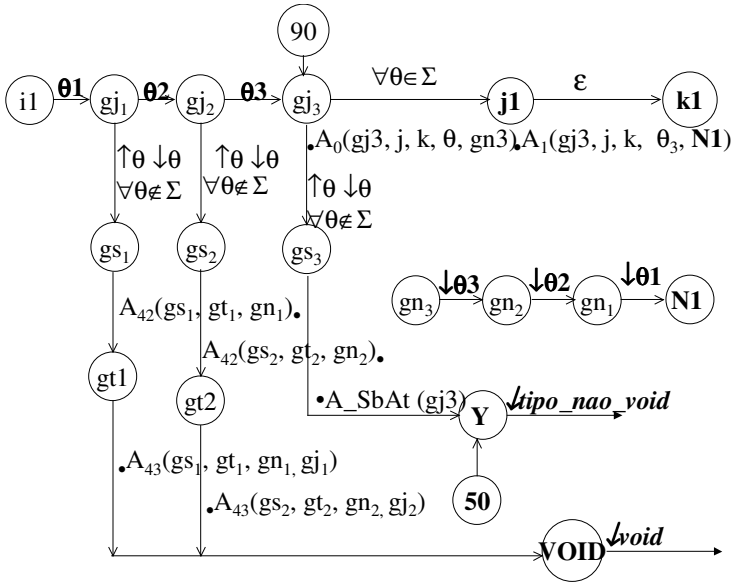


Fig. 4 – Resultado da Execução da ação adaptativa A_{43}. (modo treinamento).

No modo "uso", a ocorrência da cadeia $\theta_1\theta_2\theta_3$, promoverá a inserção do tipo que lhe representa, no caso tipo_não_void.

A ação adaptativa A_{43} ativa a execução da função adaptativa A_SbAt., a qual atualiza o ponteiro 90 para o estado final do último átomo (seqüência de símbolos) extraído (na figura acima, trata-se de gj_3).

Tal técnica não depende da natureza dos símbolos: é válida tanto para símbolos terminais como para símbolos de contexto, ou seja, símbolos correspondentes a não-terminais que são inseridos, extraídos e memorizados pelo transdutor adaptativo.

2.3.4.3 Comutação entre os modos de operação "treinamento" e "uso"

O formalismo adaptativo proposto permite que dinamicamente o modo de operação do analisador sintático seja controladamente alterado do modo "treinamento" para o modo "uso".

Empregam-se três meta-símbolos associados respectivamente a três ações adaptativas, para o tratamento de escopos aninhados, conforme a tabela seguinte:

Meta-símbolo	Função Adaptativa Associada	Tarefa associada
{	H	Tratamento da detecção do início de um novo escopo.
:	I	Alteração do modo de extração de símbolos (ou treinamento) para o modo uso. Tal meta-símbolo pode estar associado à detecção na cadeia de entrada, de um símbolo que não pertença ao conjunto Σ de símbolos admitidos nesse escopo.
}	J	Tratamento da detecção de término do escopo presente.

Tabela 1 – Meta-símbolos associados a ações adaptativas para delimitação de escopo e alteração do modo de operação

Função adaptativa H:

A função adaptativa H executa as seguintes tarefas:

- inicia um escopo;
- inicia as condições para a ativação do modo "uso", acertando os parâmetros das ações adaptativas associadas àquele escopo;
- ativa a regra 97_Modo → 98_Treino e desativa a regra 97_Modo → 99_Uso
- marca tipo dos símbolos como Não_Void.

Apresenta-se a seguir a descrição da função adaptativa H:

H ( ) = {w, v, x, y, temp, n_anterior, R, S, I_n*, N_n*, E_n*, J_N*, K_N*:

-[(97_Modo → w]
+[(97_Modo → 98_Uso]
-[50_TipoAtual → v]
+[50_TipoAtual → Não_Void]

/* Inicializa um novo escopo - ação adaptativa Inserção ( ) com I0 = 3*/

-[(3 → x)]
+[(3 → I_n)]
+[(N_n → x)]

$-[(SP\_96 \rightarrow y)]$

$+[(SP\_96 \rightarrow N_n)]$

$-[(temp \rightarrow \{ Z(n\_anterior)\} SP\_96)]$

$+[(temp \rightarrow \{ Z(n\_anterior) \} E_n)]$

$+[(E_n \rightarrow \{Z(y) SP\_96 \} Z(y)]$

/* altera os parâmetros da ação adaptativa I, associada ao meta-símbolo **:** */

$-[B3 \rightarrow \{I(R, S)\}$ ":" $B4]$

$+[B3 \rightarrow \{I(J_N, K_N)\}$ ":" $B4]$

/* a ação adaptativa A_0, responsável por extrair os símbolos da cadeia de entrada, é associada aos não-terminais I_N e J_N. */

$\{+[(I_N \rightarrow \{A_0 (I_N, J_N, K_N, \theta, N_n)\} \theta\ GJ] \forall \theta \in \Sigma\}$

/* a ação adaptativa A_1 responsável por efetivamente memorizar os símbolos da cadeia de entrada, é associada aos não-terminais J_N e K_N. Em outras palavras, o modo "uso" refere-se exatamente aos símbolos anteriormente memorizados */

$+[(J_N \rightarrow \{A_1 (I_N, J_N, K_N, \varepsilon, N_N)\} K_N]$ }

A Fig. 5 ilustra o resultado da execução sucessiva de 3 aberturas de escopos. Antes da abertura de qualquer escopo, na gramática inicial, está presente apenas uma a produção :

$$N3 \rightarrow N0$$

Após a abertura de cada escopo, a regra acima é substituída pela regra $N3 \rightarrow I$, onde I é a raiz do escopo mais recente.

Avaliação de complexidade:

Observe-se que a execução da ação adaptativa H implica na remoção de 6 regras e na criação de 9+l regras, onde l é o número de símbolos terminais da gramática. Dessa forma, a gramática resultante se apresenta com um número acrescido de 3+l produções, 5 novos não-

terminais, para cada abertura de escopo. Ainda, sua execução consome o tempo da ativação de (15 + l) ações elementares, para cada abertura de escopo.

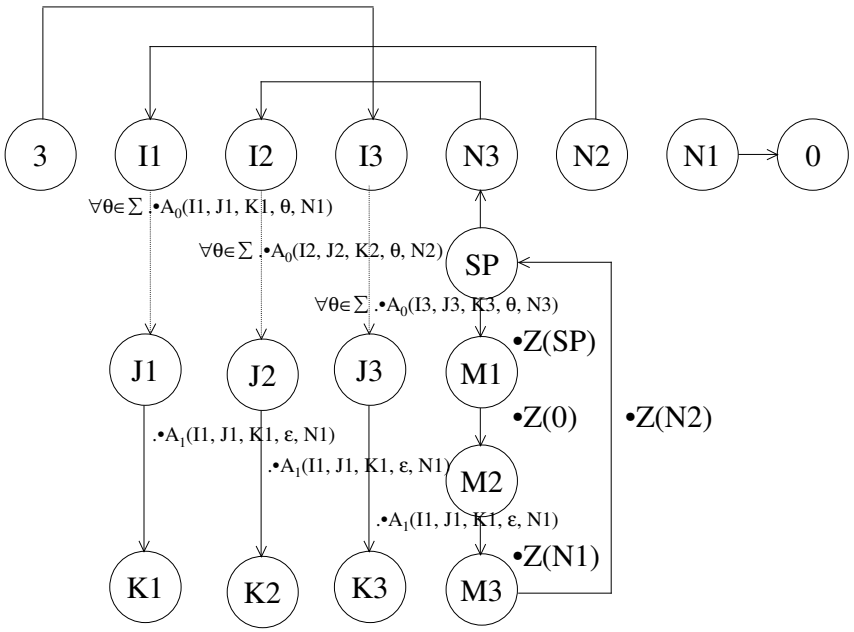


Fig. 5 – Resultado da ativação da função adaptativa H( ) para três escopos aninhados

Função adaptativa l:

A função adaptativa $l(J_N, K_n)$ executa as seguintes tarefas:

- Ativa a regra 97_Modo → 98_Uso e desativa a regra 97_Modo → 99_Treino
- Marca tipo dos símbolos como Void_8.
- Desativa a execução da ação adaptativa A_1 (I_N, J_N, K_N, ε, N_N) responsável pela memorização dos símbolos da cadeia de entrada e ativa a execução da ação adaptativa A_{39} (I_N, J_N, K_N, ε, N_N), que permite a restauração do símbolo na cadeia de entrada, para que seja buscado nos blocos ancestrais K_N

Segue-se a descrição da função adaptativa l:

$l\ (J_N, K_N) = \quad \{I_N, \sigma, N_N, x, y :$

/* Ativa a regra 97_Modo → 98_Uso e desativa a regra 97_Modo → 99_Treino e marca o tipo dos símbolos como Void. */

-[(97_Modo → y]
+[(97_Modo → 99_Uso]
-[50_TipoAtual → x]
+[50_TipoAtual → Void]

/* Desativa a execução da função adaptativa $A_1(I_N, J_N, K_N, \varepsilon, N_N)$ responsável pela memorização dos símbolos da cadeia de entrada, e ativa a execução da ação adaptativa A_{39} $(I_N, J_N, K_N, \varepsilon, N_N)$, a qual permite a restauração do símbolo na cadeia de entrada, para que seja buscado nos escopos ancestrais K_N*/

$-[J_N \rightarrow \{ A_1 (I_N, J_N, K_N, \sigma, N_n) \} K_N]$
$+[J_N \rightarrow \{ A_{39} (I_N, J_N, K_N, \sigma, N_n) \} K_N]$ }

Avaliação de complexidade:

Observe-se que a execução da ação adaptativa I não altera o número de regras da gramática e nem o número de não-terminais. Implica, para cada alteração do modo de operação, na execução de seis ações elementares.

A Fig. 6 mostra a substituição da ação adaptativa A_1 pela ação adaptativa A_{39}, na comutação do modo de operação "treinamento" pelo modo "uso", em um escopo onde foram anteriormente armazenados três simbolos terminais.A Fig. 7 ilustra o resultado da execução da função adaptativa A_{39}, que é ativada quando uma seqüência de símbolos não é encontrada em um determinado escopo. Cria uma transição efêmera, a partir da qual, a cadeia de símbolos é restaurada na cadeia de entrada para ser buscada em outros escopos

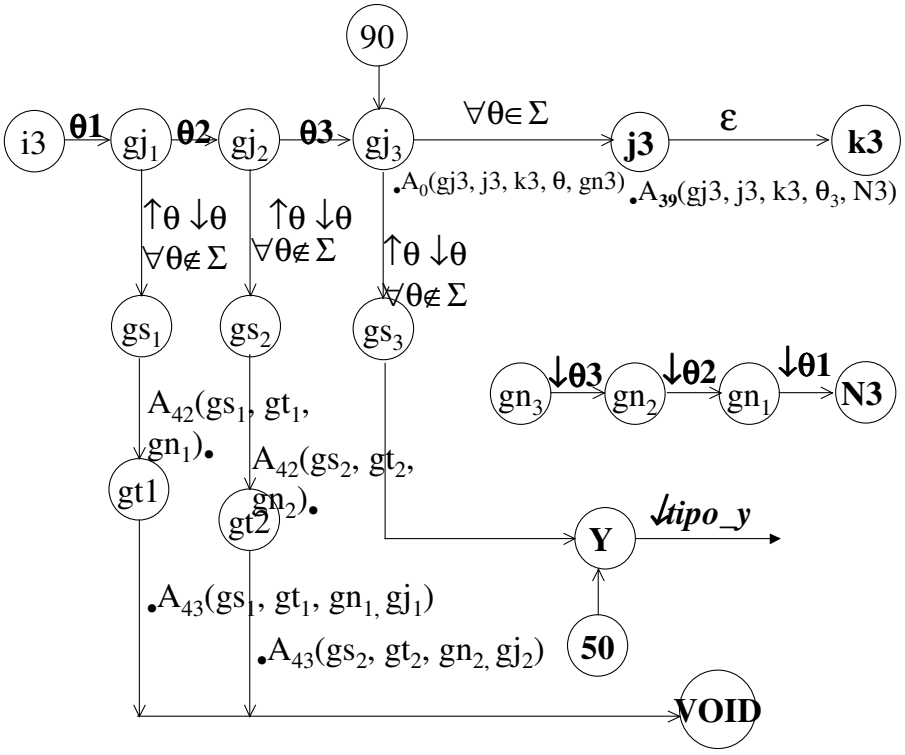


Fig. 6 – Substituição de transição associada à ação adaptativa A_1 por transição associada à ação adaptativa A_{39}

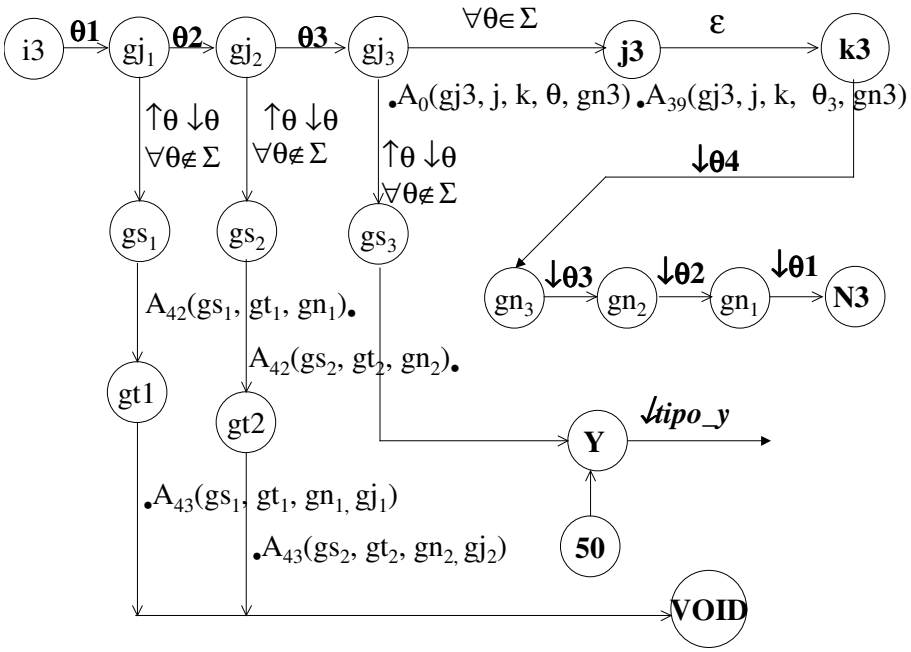


Fig. 7 – Modo "Uso":Execução da ação adaptativa A_{39}:no reconhecimento da cadeia $\theta_1\theta_2\theta_3\theta_4$ -

Na ilustração acima é possível constatar que, após a execução de A_{39}, são efetuadas as transições de inserção de símbolos recém-consumidos na cadeia de entrada para serem buscados no escopo ancestral, cujo estado inicial, neste exemplo é I_2. Confira-se transição em vazio entre os estados N_3 e I_2 na Fig. 5. Trata-se de uma situação onde a seqüência de símbolos apresenta prefixo comum àquela memorizada no escopo, porém é de comprimento maior.

Avaliação de Complexidade:

A *busca* mal sucedida de um átomo com p símbolos terminais apresenta um custo de p operações de leitura da cadeia de entrada e p operações de escrita do átomo na cadeia de entrada, acrescentado do custo de 2 operações elementares de execução das ações adaptativas A_{39} e AutoRemove_39

Se a informação assim constiuída de p símbolos estiver no q-ésimo escopo mais externo, com relação ao escopo onde se detecta a necessidade de sua busca, esta é efetuada portanto em um tempo proporcional a q*(2p+3) unidades de tempo.

Por outro lado, se a cadeia de símbolos, se apresentar com prefixo comum de com comprimento menor, a função adaptativa A_{42} é efetuada, que restaura a seqüência na cadeia de entrada. Observe-se a Fig. 8.

Avaliação de Complexidade:

A *busca* mal sucedida de um átomo com p símbolos terminais apresenta um custo de p operações de leitura da cadeia de entrada e p operações de escrita do átomo na cadeia de entrada, acrescentado do custo de 4 operações elementares de execução da execução da ação adaptativa A_4. A gramática é reduzida de uma regra em cada escopo percorrido.

Se a informação assim constiuída de p símbolos estiver no q-ésimo escopo mais externo, com relação ao escopo onde se detecta a necessidade de sua busca, esta é efetuada portanto em um tempo proporcional a q*(2p+4) unidades de tempo.

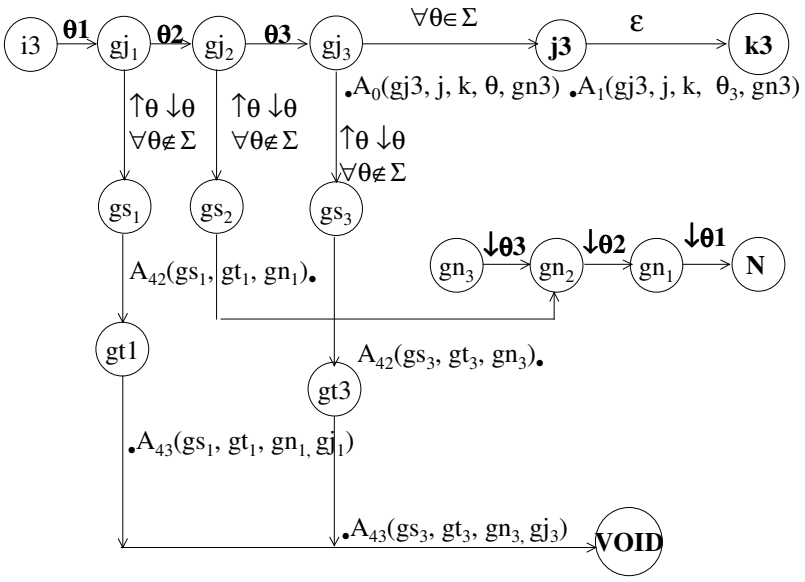


Fig. 8 – Modo "Uso": Ativação da função adaptativa A_{42} no recohecimento da cadeia $\theta_1\theta_2$

Função adaptativa J:

A função adaptativa J, corresponde à função adaptativa Remoção, anteriormente apresentada trata a finalização de um escopo, desconectando-o do processamento da derivação, uma vez que a análise do mesmo foi efetuada com sucesso. Renomeou-se o não-terminal I_0 como 3.

J( ) = {x, y, z, n_anterior, temp, m:
-[(3 → x)]
-[(SP_96 → y)]
+[(y → z)]
+[(3 → z)]

-[(temp → {Z(n_anterior)} SP_96)]
+[(SP_96 → n_anterior)]
-[(temp → {Z(u)} m]
+[(temp → {Z(u)} SP_96] }

Avaliação de complexidade:

Observe-se que a execução da ação adaptativa J não altera o número de regras de produção da gramática adaptativa. No fechamento de um escopo são executadas oito ações elementares e portanto esta função adaptativa tem um tempo de processamento constante.

2.4 Algumas considerações

Neste item são conentados alguns desdobramentos do uso das técnicas expostas anteriormente.

Concordância entre atributos de elementos imediatos

Os mecanismos adaptativos analisados neste capítulo se mostram passíveis de serem empregados na execução de tarefas de tratamento de concordâncias de atributos de símbolos que se apresentam inicialmente justapostos na fita de entrada.

Delimitação de Escopos Aninhados:

O desempenho dos mecanismos de delimitação de escopos aninhados no tempo de execução, bem como o número de não-terminais e produções (ou estados e transições) acrescentadas e até mesmo removidas pela ativação de tais mecanismos, são proporcionais ao produto dos seguintes fatores: comprimento da fita de entrada, número de ocorrências de símbolos na cadeia e em alguns casos, ao número de terminais do alfabeto da Linguagem.

Modo de operação treino e modo de operação uso – aprendizagem e automatização

Uma das características do formalismo adaptativo é o fato de ser ele próprio a realização de uma forma de aprendizado. Assim, as técnicas adaptivas podem fazer uso de dois modos de operação: o modo "treinamento", onde ocorre o aprendizado ou armazenamento de um determinado conhecimento representado em uma fita de entrada (um arquivo texto não formatado), e o modo "uso", onde é possível é possível buscar uma determinada informação armazenada (ou não) no modo "treinamento".

De fato, ainda que o alfabeto da linguagem empregada nos mecanismos apresentados anteriormente ser ASCII, estes mecanismos não dependem da natureza dos símbolos presentes na fita de entrada.

Ao longo do modo treinamento, o conhecimento a ser memorizado é disponível em uma fita de entrada (um arquivo tipo texto não formatado) e delimitado por meta-símbolos "{" e "}" que *representam* o início e o fim de escopos.

As ações adaptativas apresentadas permitem a designação *automática* dos novos não-terminais (ou estados) para cada escopo representado inicialmente na fita de entrada.

Além dos ponteiros para os modos de operação do dispositivo adaptativo, são empregados dois outros ponteiros no modo treinamento. O ponteiro N90 aponta sempre para o último símbolo memorizado em um escopo. Em tempo de treinamento, este símbolo coincide com o último símbolo lido da cadeia de entrada. O ponteiro 50_TipoAtual permite que ao longo do aprendizado seja processada a associação entre um escopo e o atributo deste mesmo escopo.

De fato, a função adaptativa A_{43}, ativada em tempo de treino, prepara o dispositivo para ser empregado no modo uso. É esta ação adaptativa que captura simultaneamente o estado final (ou último terminal) do escopo e o seu atributo.

À medida que novas informações são inseridas no dispositivo, os ponteiros, N3 para o início do escopo mais recente e o ponteiro SP_96 para a extremidade direita do escopo são *automaticamente* atualizados. O mesmo se dá, ao final do tratamento de um escopo.

A alteração do modo treinamento para o modo uso pode ser efetuado automaticamente mediante a presença do meta-símbolo ":" na informação inicialmente representada na fita de entrada e armazenada (como qualquer outro símbolo).

Representação de informação

Note-se ,que se pode representar uma informação e memorizá-la de forma parentetizada e etiquetada.

Considere-se o seguinte exemplo:

{S {DP1 {det o} {N mecânico}} {{VP {V limpou}{DP2{det o } {N automóvel}} {PP{Pcom}{NP3 {det {um}{N pano}}}}}}.

De fato, como se viu é possível representar no modo treinamento, as seguintes regras:

det = {o}, equivalente a: {det o }

dp = { det n}, equivalente a {dp {det }{n }}

Constatou-se que no modo "uso", a cadeia de símbolos a ser pesquisada é especificada na fita de entrada e sua busca se inicializa em um escopo mais interno e se orienta para o escopo mais externo.

A ação adaptativa A_{43} preparou o dispositivo adaptativo de forma que se uma informação é encontrada, o ponteiro $N_{90,}$ aponta para o último símbolo extraído da fita de entrada, enquanto que o símbolo de contexto associado ao seu tipo é inserido na fita, naquele escopo específico.

Considere-se a seqüência, presente na fita de entrada:

No modo uso, para o trecho da fita de entrada:

o mecânico

é possível se buscar e encontrar a informação:

$$\{_{DP1}\ \{_{det}\ o\}\ \{_{N}\ mecânico\}\}$$

equivalente a :

$$\{\{o\}_{det}\{mecânico\}_{N}\}_{DP1}$$

Tem-se que det e n e DP1 são símbolos de contexto inseridos na fita de entrada, ao longo da análise.

Coordenação e Subordinação

Por outro lado, se uma informação não é encontrada, as transições entre os diversos escopos são realizadas automaticamente pelas ações adaptativas A_{39}. ou A_{42} O desempenho da busca é proporcional ao produto do comprimento da cadeia a ser pesquisada pela diferença de profundidade existente entre o escopo onde a busca é especificada e onde é (ou não) encontrada.

Em uma árvore, a interligação de cada nó-pai com um conjunto correspondente de nós-filhos imita a maneira como a correspondente regra de produção de uma gramática gramática manda substituir um nó não-terminal (associado ao lado esquerdo da produção) por uma seqüência formada de terminais e não-terminais (relativa ao seu lado direito). Os nós-irmãos referem-se a elementos justapostos entre si e são subordinados ao nó-pai.

Em outras palavras, é possível estabelecer o relacionamento estrutural (coordenação e subordinação) entre elementos no modo treinamento, através da especificação da mesma e empregando-se os dos símbolos "{" e "}"

Considere-se:

dp, ip, ib e coord, símbolos de contexto, que representam os grupos determinante, flexão, primeira projeção do grupo flexão e uma entrada lexical coordenativa, respectivamente. No modo treino, é possível se especificar facilmente:

ip = { dp ib }

Duas orações coordenadas que se apresentem com grupos determinantes coordenados poderiam ser representados como:

ip = {{{dp} coord {dp}} {ib}} coord {{{dp} coord {dp}} {ib}}

Analogamente, representa-se a subordinação.

2.5 Conclusão

Este capítulo mostrou como as técnicas adaptativas utilizadas em (NETO, 1993) podem representar e tratar diversos problemas encontrados no processamento de Linguagem Natural.

Tais constatações associadas às que serão apresentadas no próximo capítulo resultaram na proposição de uma arquitetura adaptativa para processamento de linguagem natural, fundamentada no uso dessas técnicas.

3 O PROCESSAMENTO DA ESPECIFICAÇÃO DA LINGUAGEM NATURAL

No capítulo 2, estudou-se como o formalismo adaptativo permite representar de forma parentetizada e etiquetada estruturas sintáticas justapostas, subordinadas ou coordenadas. Esta representação é equivalente a uma árvore. Tais estruturas assim especificadas são passíveis de serem pesquisadas para a análise sintática de uma determinada cadeia de entrada. O processamento da busca, de uma informação além de ser automatizada (de um escopo mais interno para o escopo mais externo), apresenta um desempenho no tempo proporcional ao número de escopos somado ao dobro do número de símbolos (elementos justapostos) que constituem a informação. Há que se recordar que as técnicas empregadas por (NETO, 1993) não dependem da natureza dos símbolos. Considere-se a figura 9.

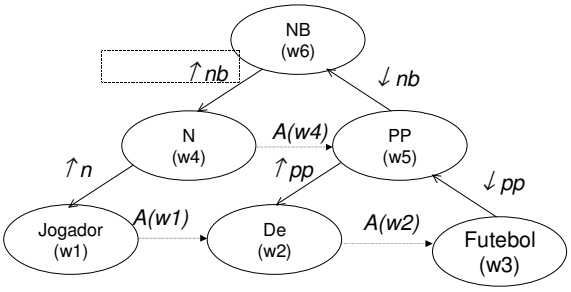


Fig. 9 – Estrutura aprendida no modo "treinamento" e utilização posterior

Nesta figura, apresenta-se esquematicamente (os detalhes foram apresentados no capítulo anterior), como uma estrutura pode ser "aprendida", a partir da extração dos símbolos nb, n e pp,(representada pelo símbolo "↑") e a sua utilização no modo uso, para a análise de

uma cadeia de entrada . As **inferências** são anotadas na cadeia de entrada e são representadas pela inserção dos mesmos símbolos. ,(representada pelo símbolo "↓")

Observe-se que na figura 9 a representação de um grupo preposicional PP está simplificada. Na verdade, o grupo preposicional é ele mesmo uma estrutura.

Os mecanismos aqui apresentados permitem a representação e tratamento de estruturas sintáticas associados aos seus atributos, segundo o seguinte formato::

identificador_regra (lista_opcional de parâmetros) =

{

lista de identificadores → lista de regras ;

}

Exemplo:

dp(&gen, &num) = { det &gen &num → { np(&gen, &num)}}

Trata-se da representação do grupo funcional determinante, constituído de um determinante, gênero e número e do grupo nominal

Observe-se a figura 10.

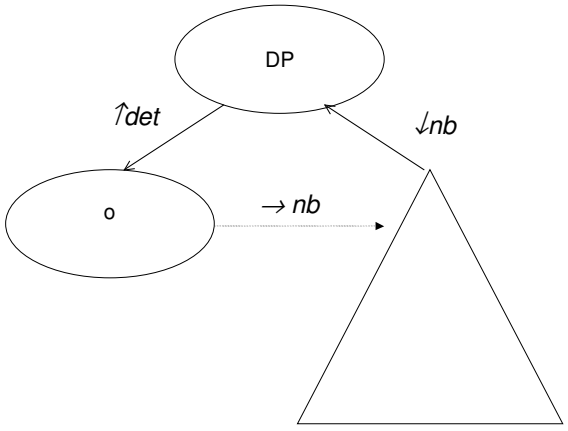


Figura 10 – Especificação do grupo determinante DP, fazendo uso do mecanismo de expansão

Obviamente a gramática da linguagem natural deve conter as diversas estruturas sintáticas, associadas aos seus atributos. Uma vez que cada uma das estruturas sintáticas sejam representadas, os mecanismos apresentados em (NETO, 1993), permitem que ***automaticamente*** a árvore seja expandida até às suas folhas, ou seja, que a estrutura seja internamente representada apenas por uma seqüência de símbolos (identificadores) em vez de estruturas, segundo as regras fornecidas pelo especialista.

Configura-se assim um modo de operação denominado em (NETO, 1993), como modo de expansão. A detecção da necessidade desse modo de operação é efetuado pela utilização do meta-símbolo "→".

Uma vez que internamete a representação da regra é constituída de apenas símbolos , a utilização dessas regras apresenta desempenho compatível com aqueles apresentados no capítulo 2.

Há ainda que se considerar a representação dos atributos. Considerados como parâmetros no modo de aprendizagem, no modo uso, a ocorrência dos argumentos segundo as regras apresentadas, configuram a concordância entre esses atributos.

Em outras palavras, o que se propõe nesta tese é que as regras de especificação de linguagem natural sejam interpretadas pelo transdutor adaptativo tal qual as macros em (NETO, 1993), empregando os mesmos mecanismos adaptativos. Justifica-se a adoção desta estratégia pelo desempenho de cada uma destas técnicas*: proporcional ao número de símbolos presentes na cadeia de entrada e ao número de símbolos presentes no alfabeto de entrada*, as quais incluem também o *tratamento dinâmico e incremental de não-determinismos e ambigüidades.*

. No caso de exánsão de macros em tratamento de linguagens de programação, as ações adaptativas associadas às declarações de macros encarregam-se de coletar o nome e os parâmetros formais da macro, efetuando alterações no autômato para que seja realizada uma

verificação de consistência entre os parâmetros e os argumentos das chamadas da macro. Além destas verificações, as ampliações do autômato promovem a associação entre os parâmetros formais e os argumentos, de forma que as ocorrências dos parâmetros formais sejam substituídos pelos correspondentes argumentos quando da expansão da macro.

Quando uma declaração de uma macro é identificada no código fonte, verifica-se sua sintaxe e o texto que lhe corresponde é simultaneamente memorizado na forma de um conjunto de transições de inserção do mesmo, associado ao identificador da macro. Em seguida, um novo símbolo na cadeia de entrada é inserido, indicando que a declaração da macro foi devidamente processada.

A seguir, relata-se o estudo detalhado do transdutor sintático adaptativo do exemplo "Expansão de Macros", de (NETO, 1993), aqui interpretado como uma máquina para o reconhecimento da sintaxe das regras de especificação de Linguagem Natural.

Inicialmente apresentam-se os mecanismos adaptativos para o modo de operação "treinamento" e em seguida, os mecanismos para o modo de operação "uso".

As funções adaptativas serão descritas através do formalismo gramatical. Também são apresentados os transdutores adaptativos em sua representação gráfica equivalente ilustrando a modificação do transdutor adaptativo quando as funções adaptativas são ativadas.

3.1 A Configuração Inicial do Transdutor Adaptativo e o Modo de Treinamento.

A figura 11. ilustra um autômato semelhante àquele apresentado em (NETO, 1993). Trata-se de um trecho do transdutor adaptativo que reconhece a estrutura sintática das regras a serem fornecidas pelo Lingüista. As transições aí indicadas consomem tokens, fornecidos pelo transdutor léxico.

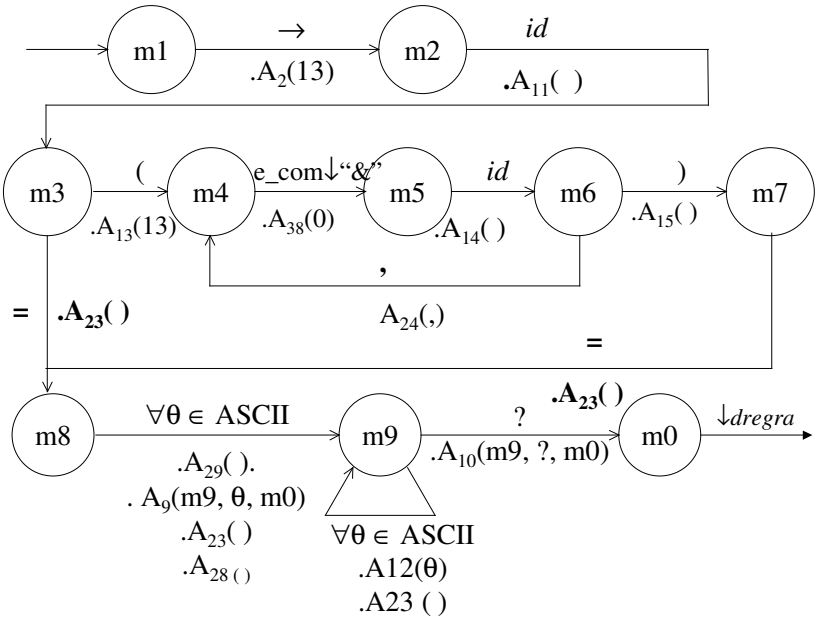


Figura 11 – Trecho de um Transdutor Sintático para aprendizagem das regras a serem fornecidas pelo Lingüista.

O transdutor léxico se encarrega de converter adequadamente os caracteres ASCII que constituem o texto original formado pelas regras fornecidas pelo Lingüista a serem memorizadas pelo sistema, e os substitui na cadeia de entrada pelos seus tokens.

Assim, por exemplo, seja um texto de entrada em que se especifica a inserção da regra:

dp(&gen, &num) = { det &gen &num → { np(gen, num) }}

o analisador léxico insere na cadeia de entrada a seqüência de tokens:

id (e_com id, e_com id) = { id e_com id e_com id → { id (e_com id, e_com id)}}

É necessário que o transdutor adaptativo responsável pela análise léxica, opere ora em modo isolado, consumindo apenas um símbolo como, &, , (e), ora opere em modo átomo consumindo uma seqüência de símbolos na extração dos identificadores de regras e dos parâmetros.

O transdutor sintático além de verificar a sintaxe das regras, configura o modo de operação do analisador léxico, mediante a execução de ações adaptativas presentes em suas

transições. A interação entre o transdutor sintático e o transdutor léxico é efetuada dinamicamente, empregando-se uma pilha explícita, que insere o estado onde o analisador sintático interrompeu sua operação e retornando ao mesmo, após a inserção do token na cadeia de entrada.

Outra tarefa realizada pelo transdutor adaptativo, através da ativação de funções nele presente no modo treinamento, consiste em promover a criação de máquinas iniciais reconhecedores dos argumentos, estabelecendo uma correspondência posicional entre os argumentos e os parâmetros, efetuando dessa forma a ampliação do transdutor adaptativo inicial.

No modo "treinamento", o corpo da regra é associado ao seu identificador e às máquinas reconhecedoras dos argumentos, na forma de produções de inserção, de forma que no modo "uso" uma vez que seu identificador seja referenciado, o texto que lhe corresponde possa ser inserido na cadeia de entrada.

Os identificadores das regras, bem como dos parâmetros, são extraídos pelo transdutor léxico conforme a estratégia para memorização de uma seqüência de símbolos, apresentada no capítulo 2 .

Assim para cada regra presente no modo treinamento é criado um transdutor conforme ilustra a figura 12. Cumpre observar que nesta figura são representados os blocos Cont_1, Cont_2 e Cont_3 que se destinam à contagem de regras em expansão, mecanismo que será detalhado posteriormente.

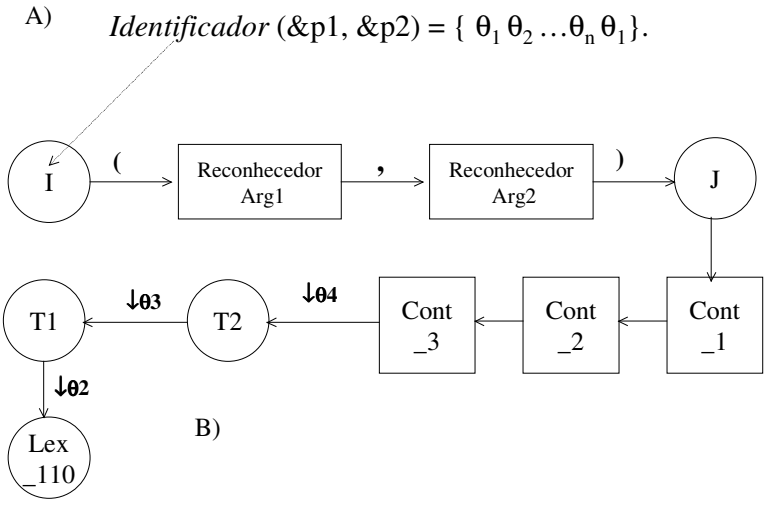
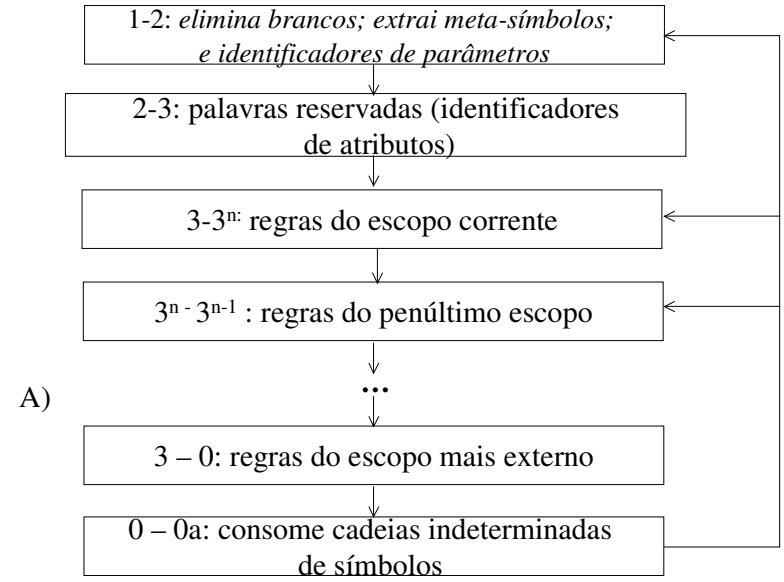


Fig.. 12- (A) Regra processada no modo treinamento
(B) Trecho de transdutor resultante da aprendizagem de uma regra

Na Fig. 13 a) transpõe-se o esquema macroscópico da estrutura do transdutor léxico apresentado em (NETO, 1993), em uma situação em que há o processamento de regras relativas a escopos subordinados presentes em uma construção frasal.

Em 13 B) apresenta-se o trecho do transdutor léxico, onde figuram transições adaptativas que consomem símbolos isolados tais como •, ⊥, ■ , &, ou ainda, transições que consomem um único símbolo ASCII com inserção do respectivo token. Também é representado o trecho com origem no estado 70, a partir do qual se dá a extração e memorização de identificadores de parâmetros. Entre os estados 2 e 3, devem figurar os



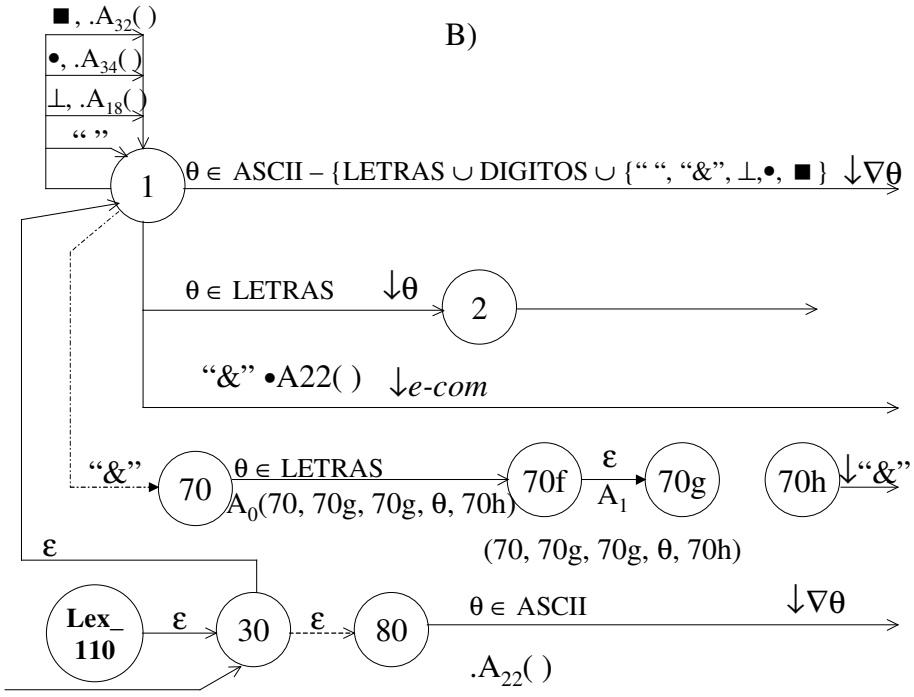


Figura 13 – Transdutor Léxico (Extraído de (NETO, 1993))
A) Esquema macroscópico representando o processamento de regras relativas a n escopos subordinados presentes em uma construção frasal.
B) Módulo 1 –2

transdutores responsáveis pelo reconhecimento de palavras reservadas. Nesta tese se propõe que tais palavras reservadas sejam os atributos gramaticais tais como gênero (masculino, feminino, neutro), pessoa (1, 2, 3), papel temático, etc.

Note-se que ao estado Lex_110 devem convergir todos os transdutores criados no modo "treinamento".

Uma vez que se descreveu a configuração inicial do transdutor léxico-sintático, prossegue-se neste item com a análise dos mecanismos adaptativos propostos por (NETO, 1993), responsáveis pela auto-expansão do transdutor ao longo da operação em modo "treinamento".

Configuração do modo de operação do Analisador Léxico: as funções adaptativas A_{22} e A_{23}

A função adaptativa A_{23} configura o próximo estado do analisador léxico para o estado 80, a partir do qual o analisador léxico consome um símbolo isolado ASCII da cadeia de entrada e o substitui pelo seu respectivo token. A inserção do token na cadeia de entrada é sucedida pela ativação da função adaptativa A_{22}, que configura o analisador léxico de forma que a extração dos símbolos se faça a partir do estado 1, origem de transições responsáveis pela extração de uma seqüência de símbolos, bem como de alguns símbolos isolados isolados.

A_{23} ( ) = {

-[N30 $\rightarrow$ N1]

+[N30 $\rightarrow$ N80] }

A_{22} ( ) = {

-[N30 $\rightarrow$ N80]

+[N30 $\rightarrow$ N1] }

Avaliação de complexidade:

A execução das ações adaptativas A_{23} e A_{22} não altera o número de regras presentes na gramática. O tempo consumido para efetuar cada uma delas é o de duas transições elementares.

Considere-se novamente o autômato representado na figura 11. À medida que se sucede a verificação da sintaxe do cabeçalho da regra, um outro trecho de transdutor adaptativo é criado. Tal transdutor é constituído de:

a) um conjunto de transdutores iniciais capazes de reconhecerem os argumentos associados aos parâmetros das regras;
b) um conjunto de transições responsáveis pela substituição do identificador de uma regra pelo texto que lhe corresponde.

O novo trecho do transdutor adaptativo é criado e incorporado ao analisador léxico de forma que esteja disponível no modo "uso" da regra.

Atribuição de Tipo: a função adaptativa $A_2(t)$

$A_2(t)$ atualiza o ponteiro N50_TipoAtual, o qual indica o tipo "default" da informação em reconhecimento. No início do processo de aceitação de uma regra, lhe é atribuído o tipo void _13

$$A_2(t) = \{ j:$$
$$-[N50\_TipoAtual \rightarrow j]$$
$$+ [N50\_TipoAtual \rightarrow t] \}$$

Avaliação de complexidade:

Esta função é ativada uma vez para cada nova regra a ser inserida no sistema. Por outro lado, não altera o número de regras da gramática adaptativa. Segue-se o autômato correspondente.

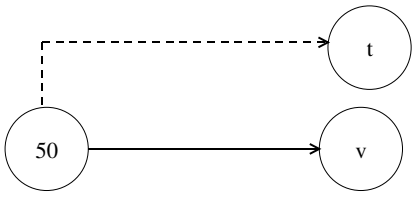


Figura 14 - Atualização do Ponteiro para o tipo de cadeia em extração

A criação de uma gramática auxiliar : função adaptativa A_{11}

Tão logo se memorize o identificador da regra, uma sub-gramática auxiliar é inicializada, pela ativação da função adaptativa A_{11}. Esta sub-gramática, que apresenta como raiz o não-terminal N15 é incrementalmente ampliada. Entre os não-terminais N18 e N15 serão criadas produções associadas à contagem dos parâmetros de uma regra.

Segue-se a descrição da função adaptativa A_{11}, bem como o mapeamento da sub-gramática inicializada, para o autômato correspondente.

A_{11} (t) = { w, x, y, z:
- [N15 $\rightarrow$ x]
- [N18 $\rightarrow$ z]
+ [N15 $\rightarrow$ N14]
+ [N18 $\rightarrow$ N15] }

Avaliação de complexidade:

Esta ação é executada uma vez para cada regra a ser inserida no modo “treinamento”.

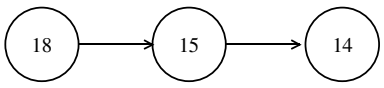


Figura 15 - Inicialização de uma Lista Auxiliar

A Função Adaptativa A_{13}

A função adaptativa A_{13} é ativada pelo consumo do token (. Tem por função disponibilizar uma regra que consome o mesmo símbolo no modo uso, porém associado às funções adaptativas A_{23} e A_SbAt.

A_{13} também atualiza o ponteiro N50_TipoAtual para N16 (corresponde ao tipo parâmetro), visto que ao consumo do símbolo (, segue-se a extração dos parâmetros da regra.

A_{13} =

{ I, X, Y, Z. J*:

/* remoção de produções para a ativação de tipo parâmetro*/

- [N50_TipoAtual $\rightarrow$ X]

- [Y $\rightarrow$ {A_SbAt(I)} N13_AUX]

- [N13_AUX $\rightarrow$ VOID_13]

/*remoção de regras que permitiriam a extração e devolução de símbolos na cadeia de entrada No transdutor em construção só se permitirá a ocorrência do símbolo)*/

- [θ Z $\leftarrow$ θ Y] $\forall$ $\theta \in$ ASCII – (LETRAS $\cup$ DIGITOS) /* 66 regras */

/*inserção de regra que consome o símbolo (, associado a ações adaptativas para o modo uso */

+ [I $\rightarrow$ {A_{23}, A_SbAt(I)} (J]

/*transdutor em construção ainda é do tipo void_13 */

+ [J $\rightarrow$ VOID_13]

/*próxima cadeia a ser extraída deverá ser do tipo parâmetro (N16) */

+ [N50_TipoAtual $\rightarrow$ N16]

/*configura léxico para operar em modo isolado para extração de "&" */

A_{23} }

Avaliação de Complexidade:

A execução da ação adaptativa é efetuada uma vez para cada regra no modo treinamento. Sua execução diminui em 66, o número de regras presentes até o instante do reconhecimento do símbolo (. Consome-se um tempo equivalente ao de 74 regras elementares. (2 regras elementares referem-se à execução de A_{23}).

A Função adaptativa A_{38}

As duas seguintes regras correspondem à transição que consome o símbolo *e_com*, presente no autômato da figura 11

$$M4 \rightarrow e\_com\ M5\_Aux$$
$$M5\_Aux \leftarrow \{A_{38}(0)\}\ \&\ M5$$

Observe-se que o símbolo *e*_com, que precede o identificador de um parâmetro, é substituído na cadeia de entrada pelo símbolo "&". Trata-se de uma estratégia para manter disponível na cadeia de entrada o símbolo "&", de forma em seqüência se extraia o parâmetro. Isto se faz necessário porque o trecho do transdutor que reconhece identificadores de parâmetros tem origem no estado 70, enquanto que o transdutor que reconhece identificadores de regras tem origem no estado 3. (Confira-se na Fig. 13 b))

Uma vez consumido o identificador de parâmetro, o token *id* é inserido na cadeia de entrada. Segue-se a descrição da função adaptativa A_{38}.

A_{38} = {

/ *"& é substituído por *e_com* */

-[N1 → {A_{22}} "&" N17]

/ *"&" é sucedido pelo reconhecimento de identificadores de parâmetros*/

+ [N1 → "&" N70] }

Avaliação de complexidade:

O número de regras não é alterado pela execução da ação adaptativa A_{38}. Sua execução é o de duas ações elementares. É efetuada para cada parâmetro de cada regra no modo Treinamento.

As Funções adaptativas A_{24} e A_{15}

Estas funções adaptativas são ativadas pelo consumo dos tokens associados aos delimitadores da lista de parâmetros "," e ")", respectivamente e promovem a criação de regras que consomem os mesmos símbolos, na lista de argumentos. Segue-se a descrição das mesmas.

$A_{15}(x) = \{I , J^*:$
$-[I \rightarrow VOID\_13]$
$+[I \rightarrow x\ J]$
$+[\ I \rightarrow VOID\_13]\ |$

Avaliação de Complexidade:

Esta função adaptativa é executada na ocorrência do símbolo) na cadeia de entrada no modo Treinamento. Acrescenta em 1 o número de regras na gramática existente. O custo no tempo é aquele de execução de 3 regras elementares.

$A_{24}(x) = \{I , J^*:$
$-[I \rightarrow VOID\_13]$
$+[I \rightarrow \{A_{23}(\ )\}\ x\ J]$
$+[\ I \rightarrow VOID\_13]\ \ \}$

Avaliação de Complexidade

Esta função adaptativa é executada sempre que o símbolo **,** é encontrado na cadeia de entrada no modo Treinamento. Acrescenta em 1, o número de regras na gramática existente. O custo no tempo de sua execução coincide com o de 3 regras elementares.

As Funções adaptativas A_{14} e A_{36}

O identificador do parâmetro, que se segue ao símbolo "&", é memorizado em um autômato e é substituído na cadeia de entrada pelo token *id.*

Uma vez que o símbolo *id* seja consumido pelo trecho sintático do transdutor adaptativo, a função adaptativa A14 é ativada e realiza as seguintes tarefas:

- Inicializa uma sub-gramática destinada à extração argumento que lhe corresponderá no tempo de uso da gramática.
- Cria e efetua a manutenção de uma lista dos não-terminais associados aos identificadores aos identificadores dos parâmetros.
- Promove a execução da ação adaptativa A_{36} que acrescenta à sub-gramática recém-inicializada, *mecanismos de tratamento de não-determinismos e ambigüidades,* no que diz respeito à extração de identificadores de argumentos.

A figura 16 ilustra o mapeamento da sub-gramática inicial referente aos argumentos, para autômato adaptativo. Observe-se o argumento w da ação adaptativa A_{27}. Este argumento refere-se ao não-terminal associado ao identificador do parâmetro recém-reconhecido, apontado portanto pelo não-terminal N90.

Segue-se o conjunto de ações que constituem a ação adaptativa A_{14}.

A14 ( ) = { W, X, N, S. M*, J*, K*, R*, T*:

/* ao consumo do próximo símbolo "&" deverá ser inserido o token e_com */

-[N1 → "&" N70]

+[N1 → {A_{22}( )} N17]

/* verifica qual é o último identificador extraído */

?[N90 → W]

/* insere identificador recém-criado na lista de parâmetros */

-[N → {A_{19}(S)} N60]

+[M → {A_{19}(W)} N60]

+[N → {A_{19}(S)} M]

/* cria autômato para reconhecimento de argumento K, associado ao parâmetro W: associação memorizada através dos argumentos da ação adaptativa A_{27} */

+[I → {A_{27} (J , θ, K, W), A_{23}( ), A_{25}( )} θ J} ∀ θ ∈ ASC II] /* 128 transições */

+[J → {A_{12} (θ), A_{23}( )} θ J] ∀ θ ∈ ASC II – {"&"} /* 127 transições */

+[J → { A_{12} ("&"), A_{23}( )}]

+[T → θ J] ∀ θ ∈ (LETRAS ∪ DIGITOS) } /* ASC II: 128 transições*/

+ [θ T ← θ J] ∀ θ ∈ ASC II - (LETRAS ∪ DIGITOS) }

/* complementa ação adaptativa A_{14} */

A_{36} (W, T, J, K, R) }

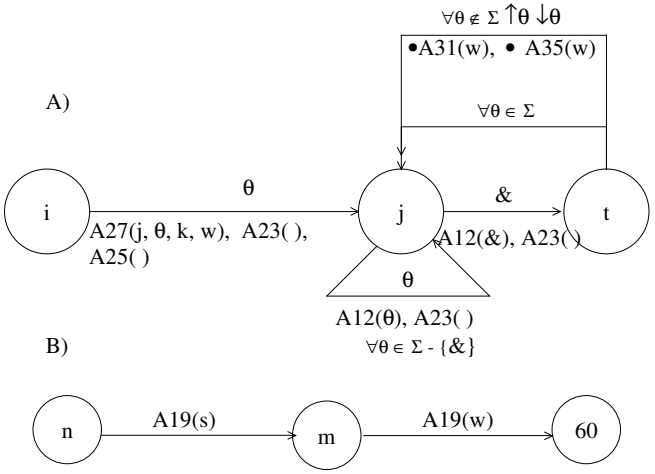


Fig. 16 - A) Configuração da Máquina M_i, destinada ao reconhecimento de um argumento A_i, criada pela execução da ação adaptativa A_{14}, obtida no modo Treinamento. B) Lista de Não - Terminais associados aos Parâmetros.

Avaliação de Complexidade:

A ação adaptativa A_{14} aumenta em 385, o número de regras para cada identificador de parâmetro presente em cada regra a ser incluída (aprendida) no sistema . O seu desempenho

coincide com o de 390 ações elementares. Esta avaliação não levou em conta a execução da ação adaptativa A_{36}, comentada no entanto, a seguir.

A função adaptativa A_{36}

Esta função adaptativa, quando ativada, realiza as seguintes tarefas:

- Cria produções que permitam que em tempo de uso, o transdutor correspondente em construção possa contabilizar o número de parâmetros presentes na lista de parâmetros;
- Acrescenta à sub-gramática recém-criada por A_{14} regras concernentes à extração de argumentos, a saber:
  - regras que finalizam a extração de argumentos;
  - a detecção de um argumento homônimo ao parâmetro formal. Esta tarefa, em (NETO, 1993) é realizada pelas ações adaptativas A_3 e A_4 , comentadas a seguir.

Detecção de um argmento homônimo ao parâmetro formal: as funções adaptativas A3 e A4.

Estas funções adaptativas são empregadas na criação de aceitador de uma cadeia de símbolos quaisquer, com tratamento de cadeias anteriormente reconhecidas e passíveis de apresentarem prefixo comum, ou até mesmo serem iguais às cadeias a serem aceitas.

Considere-se uma situação em que se deseja reconhecer uma seqüência de símbolos quaisquer entre dois estados j e t. Inicialmente a transição entre os dois estados j e t, se apresenta configurada apenas para o consumo do meta-símbolo *e-com*, que marca o início de um identificador de parâmetro ou argumento. (Observe-se que no caso geral, o símbolo inicialmente presente é irrrelevante). Considere-se no entanto que já foi detectada em algum

instante do reconhecimento uma seqüência de símbolos, armazenada em uma lista, cujo estado final de aceitação é w.

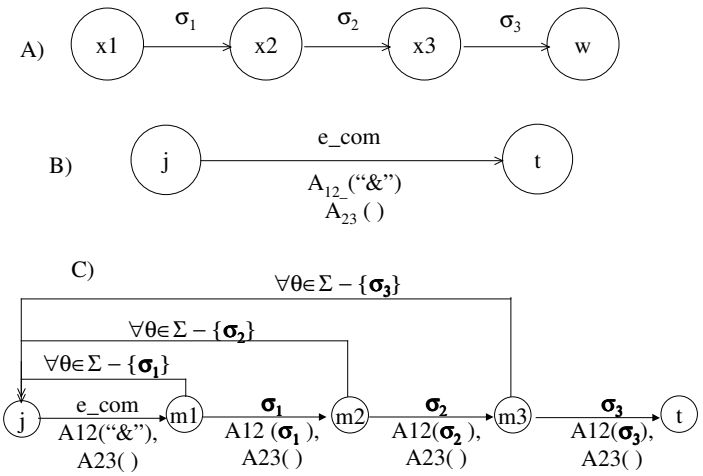


Fig. 17: Resultado da execução das ações adaptativas A3 e A4: A) W é o estado associado a um identificador de parâmetro. Antecedem-lhe três transições que consomem a cadeia de símbolos $\sigma_1\sigma_2\sigma_3$
B) Entre j e t, existe inicialmente apenas uma transição que consome o símbolo isolado e_com. C) Autômato resultante da aplicação das ações adaptativas A_3 e A_4

A técnica empregada nas funções adaptativas A_3 e A_4 permite que o trecho de autômato correspondente ao identificador de parâmetros seja copiado para o autômato para extração de argumentos. Dessa forma, ainda que o argumento seja homônimo ao parâmetro formal, em tempo de *uso*, estará disponível uma cópia deste identificador no escopo demarcado para o tratamento de argumentos.

Graças ao mecanismo de cópia, que associa cadeias de símbolos iguais de tipos distintos (no caso tipo parâmetro e tipo argumento), o autômato é construído de forma que os identificadores dos argumentos sejam extraídos ***deterministicamente*** ainda que apresentem uma sub-cadeia coincidente com a do parâmetro formal.

Tem-se que:

$A_3(W, T, J) = \{\sigma, X:$

/*verifica, incrementalmente os símbolos que constituem o identificador do parâmetro */

?[X → σ W]

/*executa ação adaptativa A_4 */

A_4 (σ, X, W, T, J) }

A_4 (σ, X, W, T, J) = { M*:

-[J → {A_{12}("&"), A_{23}( )} e_com T]

+[M → {A_{12}(σ), A_{23}( )} σ T]

+[M → θ J] ∀θ ∈ Σ - { σ }

+[J → {A_{12}("&"), A_{23}( )} e_com M]

A_3 (X, M, J) }

Avaliação de complexidade:

Se houver n transições com n símbolos, tem-se que o autômato se expande em 128*n transições elementares . e n estados.

O consumo no tempo é de (131+n) unidades de execução de transições elementares. Acrescente-se o custo no tempo de n chamadas da ação adaptativa A_4 e n chmadas da ação A_3

Além da execução das ações adaptativas A_3 e A_4, para criação de autômato que detecte argumento homônimo ao parâmetro, a ação adaptativa A_{36} acrescenta transições ao autômato para finalização da extração do argumento, bem como para contabilizar o número de parâmetros. Para a contagem de parâmetros, emprega-se o símbolo especial "⊥".

Segue-se a descrição da ação adaptativa A_{36}.

A_{36} (W, T, J, K, X, R) = {
A_3 (W, T, J)
+ [J → { A_{26}(J,•, K, W) } K]
+[K → Void_13]
-[X → N15]

+[X ← ⊥ R]

+[R → N15] }

Avaliação de Complexidade:

A execução da ação adaptativa A_{36} apresenta desempenho dependente da execução das ações adaptativas A_3 e A_4, e portanto dependente do número de símbolos presentes que constituem o identificador do parâmetro formal. Acrescido ao desempenho destas, A_{36} cria para cada parâmetro mais 3 transições: duas destinadas à finalização de extração de argumento e uma destinada à contagem de parâmetro. O desempenho no tempo é dependente da execução das ações adaptativas A_3 e A_4, acrescido do custo de execução de 5 ações elementares para cada parâmetro presente na lista de parâmetros de uma regra.

A inicialização da máquina M_i, para reconhecimento de argumento associado ao parâmetro P_i é apresentada na figura 18.

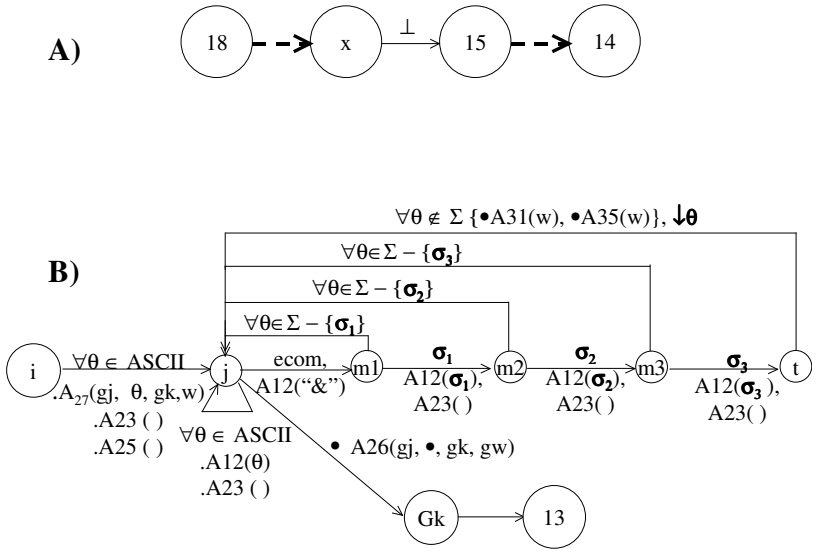


Fig. 18- A) Contagem de Parâmetros
B)Autômato Incial para Reconhecimento de um Argumento, obtido pela execução das ações adaptativas A_{14}

Como exemplo, a figura 19 ilustra a configuração do trecho do autômato criado, após a execução das ações adaptativas, A_{11}, A_{13}, A_{38}, A_{14}, A_{36}, A_{15} e A_{24}, na aprendizagem do cabeçalho de uma regra com três parâmetros.

As máquinas M_1, M_2, M_3 na figura 19 representam autômatos cuja configuração é a mesma que aquela ilustrada em 4.8 b)

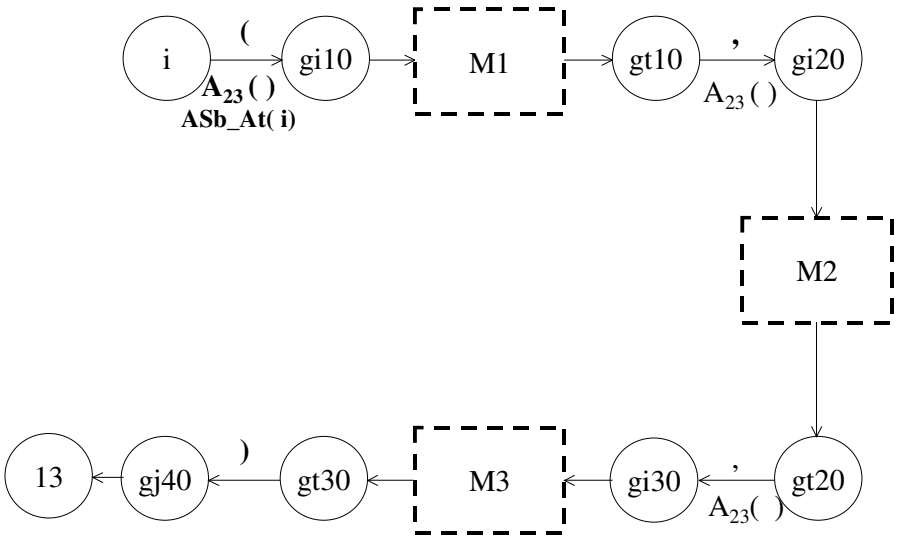


Figura 19 – Autômato para Reconhecimento dos Argumentos Associados aos Parâmetros

Os procedimentos até agora efetuados permitiram inicializar o trecho do transdutor responsável pelo reconhecimento dos argumentosss, bem como estabelecer a correspondência entre o identificador dos parâmetros recém-extraídos ao argumento a ser encontrado na cadeia de entrada no tempo de uso.

O processo de aprendizagem da regra continua, a fim de memorizar o corpo da mesma. Para tanto, o analisador léxico é preparado para extrair símbolos isolados através da execução da ação adaptativa A_{23}. Isto é necessário dado que o primeiro símbolo disponível na cadeia de entrada será interpretado como o delimitador do corpo da regra. Alterado o modo de

operação para o modo símbolo isolado, é executada função adaptativa A_{29}, que insere antes do não-terminal void_13, um mecanismo de contagem de regras a ser executado no modo uso, bem como um marcador de fim de regra.

Contagem de regras em processamento: a função adaptativa A_{29}

No corpo de uma regra, apresentam-se geralmente outras regras, as quais podem ser executadas seqüencialmente ou ainda é possível que uma regra referencie outras.

Para que, no modo "uso", haja o controle de quantas regras estão sendo aplicadas dentro do escopo de uma outra regra, a ação adaptativa A_{29} insere um mecanismo de contagem, o qual por sua vez é efetuado pelas ações adaptativas A_{16} e A_{33}. No modo uso, a informação de que o processamento de uma regra foi finalizado é obtida quando se encontrar na cadeia de entrada o meta-símbolo "•"; sendo assim a ação adaptativa A_{29} também acrescenta ao autômato em construção uma transição de inserção deste meta-símbolo.

Segue-se a descrição algébrica da função adaptativa A_{29}:

A_{29} ( ) =
{ I, K, J*, M*, N*:
-[N50_TipoAtual → K]
-[I → Void_13]
/* A_{16} e A_{33}efetuam mecanismo de contagem */
+ [I → {A_{16} (J, M) } J]
+ [J → {A_{33} (J, M)} M]
/* transição de inserção de símbolo de fim de regra */
+[M ← • N]
+[N → Void_13]
+[N50_TipoAtual → K] }

Avaliação de Complexidade:

Para cada regra são acrescentadas 3 produções e o tempo de execução é o de 7 ações elementares.

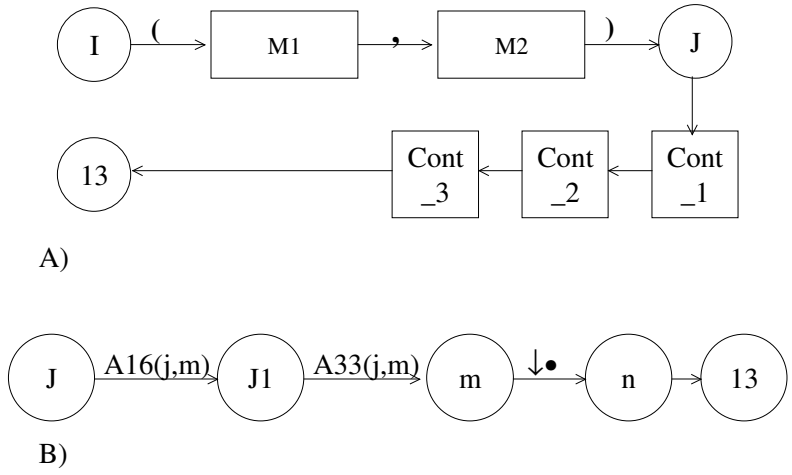


Figura 20 – Mecanismo de contagem de número de regras a serem memorizadas pelo sistema.
A) mecanismos de contagem representado em blocos;
B) Trecho de Autômato Adaptativo para contagem de regras.

Após a execução de A_{29}, o primeiro símbolo da cadeia de entrada é extraído e é memorizado como argumento de A_9.

Detecção do símbolo de delimitação de uma regra: a função adaptativa A9

A extração do primeiro símbolo θ_1 do corpo da regra ativa esta função adaptativa. Este símbolo é memorizado em seu argumento e a transição adaptativa entre M_9 e M_0, bem como o laço em M9, são alterados de forma que na próxima ocorrência do símbolo θ_1, o mesmo seja consumido nas transições entre M_9 e M_0. Implementa-se assim, a interpretação do primeiro símbolo do corpo da regra como seu delimitador.

A função adaptativa A_9 é descrita como:

$A_9(I, J, \sigma) =$

{ X:

$-[I \rightarrow \{A_{10} (I, X, J)\} X J]$

$+[I \rightarrow \{A_{10} (I, \sigma, J)\} \sigma J]$

-[I → {A_{12} (σ), A_{23}( )}σ J] }

Avaliação de Complexidade:

A ação adaptativa é efetuada apenas uma vez para cada nova regra presente no modo treinamento. Decrementa de 1 o número regras presentes na gramática. O seu desempenho no tempo é de 3 regras elementares.

A figura 21 ilustra a alteração do transdutor pela execução da da ação adaptativa A_9.

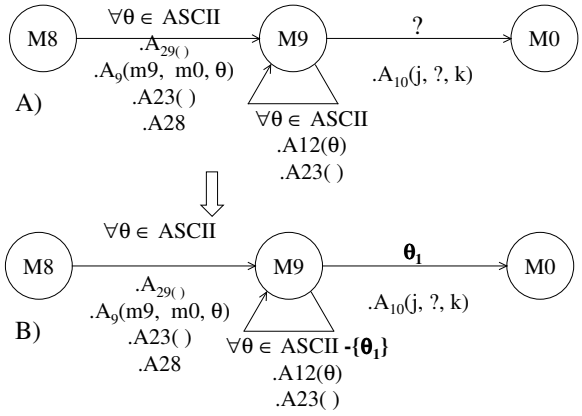


Figura 21 – O primeiro símbolo θ_1 na cadeia de entrada é interpretado como delimitador do corpo da regra

O analisador léxico é programado novamente para operar no modo símbolo isolado, através da execução da ação adaptativa A_{23} . Em seguida, a ação A_{28} é executada.

Função Adaptativa A_{28}

A função adaptativa A_{28}, simplesmente desconecta a seqüência de produções entre os não-terminais N18 e N15, indicadoras do número de parâmetros (observe-se a figura 18 -A)) e conecta tais produções ao final das produções concernentes ao reconhecimento dos identificadores dos argumentos da regra. A ação adaptativa A_{28} invoca a ação adaptativa A_{25}.

Ação Adaptativa A_{25}

Esta ação adaptativa inicializa novamente a lista auxiliar, que agora se destinará a armazenar temporariamente o texto referente à regra. A ação adaptativa A_{25} também configura o analisador léxico de forma que uma vez que seja encontrado o símbolo que inicializa um identificador de parâmetro, este seja marcado como *e_com.*

Todos os demais símbolos são lidos e armazenados entre os não-terminais 15 e 14, através da execução da ação adaptativa A_{12}, até que seja encontrado o símbolo θ_1.

Seguem-se as descrições algébricas das ações adaptativas A_{28}, A_{25} e A_{12}.

A_{28} =

{ X, σ, J:

-[I → VOID_13]

/* contagem de parâmetros é desconectado da área de rascunho e inserido no transdutor em construção para contagem de argumentos */

-[N18 ← σ X]

+[I ← σ X]

-[J → N15]

+[J → VOID_13]

/* reinicia lista auxiliar para armazenar temporariamente corpo da regra */

A_{25} }

A_{25}( ) = { X, Y:

/* reinicia lista auxiliar para armazenar temporariamente corpo da regra */

A_{11}

/* "&" é substituído por e_com */

-[N1 → "&" X]

+[X → "&" N1]

-[N1 → {A_{22}( )}"&" Y]

+[Y → {A_{22}( )} "&" N1]

+ ["&" N80 →{A_{22}( )} e_com Z]

+[Z → ε] }

Avaliação de Complexidade:

As funções adaptativas A_{28}e A_{25} são ativadas na ocorrência de um símbolo delimitador do início do corpo de uma regra na cadeia de entrada. A execução de ambas normalmente *não* acrescenta regras à gramática e apresenta o desempenho no tempo equivalente ao de 14 ações elementares, sendo 4 ações elementares associadas à execução da ação adaptativa A_{11}.

Função adaptativa A_{12}

Esta função adaptativa é ativada para cada símbolo ASCII presente no corpo da regra. e cria para cada símbolo uma produção de inserção do mesmo, na sub-gramática auxiliar delimitada pelos não-terminais N14 e N15. Segue-se sua descrição algébrica:

$A_{12}(\sigma) =$

{ X, GJ*:
-[N15 $\rightarrow$ X]
+[N15 $\rightarrow$ GJ]
+[GJ $\leftarrow$ σ X]
}

Avaliação de Complexidade:

Para cada símbolo, presente na regra, exceto os símbolos qque a delimitam a gramática é acrescentada de uma produção. O tempo de execução é constante e igual ao de 3 produções elementares

Função adaptativa A_{10}

Esta ação adaptativa é executada quando for novamente encontrada a segunda ocorrência do símbolo θ_1, que é dessa forma interpretado como delimitador do corpo da regra.

Através da execução de A_{10}, a seqüência de produções entre os não-terminais N15 e N14, é desconectada e concectada ao Analisador Léxico, resultando em um autômato semelhante ao da figura 22.

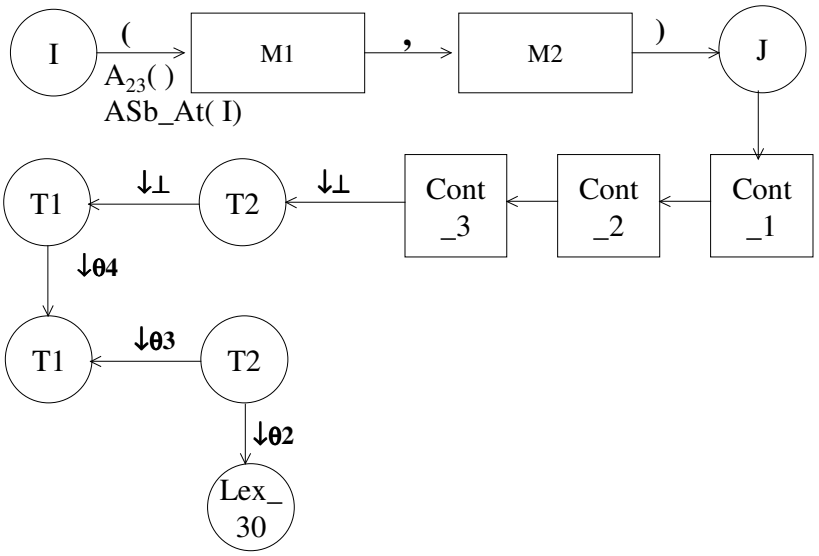


Figura 22 – Autômato conectado ao Léxico após a execução da ação adaptativa A_{10}.

Observe-se na figura que, no modo uso, na ocorrência do identificador de uma regra na cadeia de entrada resultará na substituição do texto que lhe corresponde e subseqüente ativação do analisador léxico.

Segue-se a descrição algébrica da ação adaptativa A_{10}.

A_{10}(I, σ, J) =

{ K, X, Y, Z, R, S, T, U, v, GM*:

-[K → VOID_13]

-[X → “&” N1]

+[N1 → “&” X]

-[R →{A_{22}( ) } “&” N1]

+[N1 → {A_{22}( ) }“&” R]

-[“&” N80 → {A_{22}( )} e_com Z]

+[I → {A_{12} (σ), A_{23}( )} σ J]

/* texto do corpo da regra é associado ao fim ao identificador da regra e desconectado da lista auxiliar para armazenamento temporário (entre os não terminais N15 e N14) */

-[N15 → T]

+[K → T]

-[U ← v N14]

/*final do texto realimenta léxico */

+[U ← v N30]

/*restauração da última produção referente à sintaxe de uma regra */

-[I → {A_{10} (I, s, J)} s J]

+[I → {A_{10} (I, ?, J)} ? J]

}

Avaliação de Complexidade:

Esta função é ativada apenas uma vez a cada duas ocorrências do símbolo delimitador de uma regra, visto que após sua execução, restaura o autômato de forma que a seguinte ocorrência do mesmo símbolo possa ser interpretado como se fosse a primeira. Uma vez que, apresenta 7 regras elementares de exclusão e 6 regras elementares de inserção, ela não altera o número total de regras da gramática e seu desempenho no tempo corresponde ao de 13 ações elementares.

Ao final da execução da ação adaptativa A_{10} é inserido um símbolo na cadeia de entrada, denotando que foi finalizada a aprendizagem da regra.(dregra)

A seguir, são apresentados os mecanismos adaptativos empregados no modo Uso

3.2 Execução das Ações Adaptativas do Transdutor no Modo Uso.

No item anterior foi possível constatar como no modo treinamento, um transdutor adaptativo para o processamento de uma regra é criado e incorporado ao analisador léxico para seu uso posterior.

No modo uso, os transdutores criados anteriormente, são empregados de forma que as ações adaptativas neles presentes efetuem o reconhecimento identificadores das regras e de seus argumentos, realizem a associação entre os parâmetros formais e os argumentos das regras e promovam a substituição das regras pelo texto que lhe correspondem.

A seguir, estas ações adaptativas são apresentadas.

O reconhecimento de um argumento de uma regra: as funções adaptativas A_{27}, A_{25} e A_{12}

Considere-se a figura 23. Aí apresenta-se um trecho do autômato adaptativo,

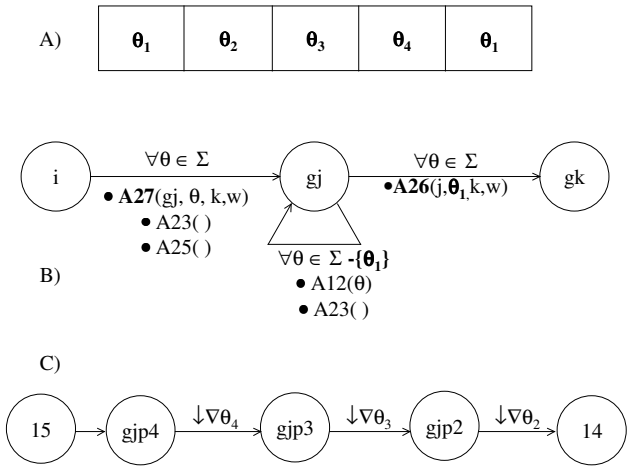


Figura 23 – Processamento de um Argumento de uma Regra
A) Cadeia de entrada original – cadeia de entrada constituída de símbolos associados ao identificador.
B) Resultado da Execução da ação adpatativa A_{27}, após o consumo do primeiro símbolo θ1, presente na cadeia de símbolos que constituem o argumento.
C) Resultado da Execução da ação adaptativa A_{12}: memorização da cadeia de símbolos que constituem o argumento

construído no modo de aprendizagem, e se destina ao reconhecimento de um argumento. Como se pode observar, faz uso das açãoes adaptativas A_{27}, A_{25}, A_{12}, além de A_{23}.

Os identificadores dos argumentos das regras são constituídos, tais como os respectivos parâmetros, de uma cadeia de caracteres ASCII..

A ativação da função adaptativa A_{27}, promove o armazenamento do primeiro símbolo θ_1 constituinte do identificador em seu argumento, elimina a transição com o símbolo especial "• ", substituindo-o pelo símbolo θ_1, constituindo-se em um mecanismo muito análogo àquele efetuado pelas ações adaptativas A_9 e A_{10}.

A ação adaptativa A_{25} inicializa a lista $18 \rightarrow 15 \rightarrow 14$ e configura o analisador léxico de forma que se o símbolo "&" for extraído da cadeia de entrada ele deverá ser entendido como um outro símbolo da cadeia de entrada qualquer. Entre os estados 15 e 14 serão criadas transições de inserção dos símbolos presentes na cadeia de entrada, até que haja uma nova ocorrência do símbolo θ_1. Isto será possível graças à execução da ação adaptativa A_{12}, responsável por este empilhamento. Há que se notar que os símbolos das transições de inserção estão presentes na forma de tokens, visto que serão empregadas na parte sintática do transdutor. Este efeito é obtido, graças às execuções da ação adaptativa A_{23} para cada símbolo do argumento. Em particular, a presença de um símbolo "&" será mapeado para o seu token e_com. Na nova ocorrência do símbolo θ_1, será executada a ação adaptativa A_{26} .

Na figura 23 C), observa-se que o estado 15 aponta para o último símbolo da cadeia de entrada referente ao argumento, enquanto que o estado 14 será apontado pelo primeiro símbolo da cadeia de entrada.

A seguir é apresentada a descrição algébrica da ação adaptativa A_{27}

$$A_{27}(J, \sigma, K, W) = \{ X:$$
$$-[J \rightarrow \{A_{26}(J, X, K, W)\}\ X\ K]$$
$$+[J \rightarrow \{A_{26}(J, \sigma, K, W)\}\ \sigma\ K]$$
$$-[J \rightarrow \{A_{12}(\sigma), A_{23}(\ )\}\ \sigma\ J]\ \}$$

Avaliação de Complexidade:

A função adaptativa A_{27} é ativada para cada argumento presente na referência a uma determinada regra. O número de produções da gramática é decrementado de 1. O seu desempenho no tempo é aquele de 3 produções elementares.

Uma vez executada a ação adaptativa A_{27}, para a detecção do símbolo delimitador do identificador de um argumento, a extração dos demais símbolos é efetuada. No modo treinamento, o reconhecedor foi construído de tal forma, que ao longo da extração é verificada a ocorrência de um argumento homônimo ao parâmetro formal. No caso de se encontrar um argumento homônimo ao parâmetro formal, são efetuadas as ações adaptativas A_{31} e A_{35}.

Estas funções adaptativas são ativadas na ocorrência de um argumento homônimo ao parâmetro formal. Observe-se na figura 18 que neste caso, o argumento não é memorizado na lista auxiliar. Daí, a necessidade de se efetuar uma cópia do parâmetro formal nesta lista auxiliar.

As funções adaptativas A_{31} e A_8

Inicialmente, a função adaptativa A_{31} desconecta o analisador léxico e executa a ação adaptativa A_8, a qual por sua vez efetivamente retira o último identificador.que foi armazenado entre os não-terminais N15 e N14 Seguem-se as descrições algébricas das ações adaptativas A_{31} e A_8

$A_{31}(W) =$

{ -[N110_Lex $\rightarrow$ N30_Lex]

-[N90 $\rightarrow$ W]

$A_8(1)$

}

Avaliação de Complexidade

Esta função adaptativa é ativada somente quando um argumento é homônimo ao parâmetro formal. O seu custo é igual ao da função adaptativa A_8, exceto que se deve reduzir em 2, o número de regras na gramática, bem como incluir o tempo de execução desta redução.

$A_8(f) = \{ X, Y, Z:$

-[N15 $\rightarrow$X]

/* remove qalquer símbolo ASCII, exceto aquele que inicializa novo argumento */

-[X $\leftarrow \theta$ Y] $\forall \theta \in$ ASCII – {"&"}

/* marca não-terminal associado a novo argumento em Z */

-[X $\leftarrow$ "&" Z]

/*reconecta extremidade */

+[N15 $\rightarrow$Y]

/* idem no caso ter eliminado "&" */

+[N15 $\rightarrow$Z]

/* caso contrário, volta a eliminar outro símbolo */

$A_8(Y)$

Avaliação de desempenho

Para cada símbolo a eliminar, são executadas ações para remover e reconectar o terminal N15 e um teste para verificar se se trata de um símbolo ASCII, distinto de "&". Ao final de n símbolos constituintes do argumento que não o homônimo mais recente, são removidas n+1 transições (os símbolos e aquela que consome "&") e o não terminal N15, adequadamente reconectado.

Após a ativação das ações adaptativas A_{31} e conseqüentemente de A_8, é ativada a função adaptativa A_{35} que dispara a execução das funções adaptativas A_5, A_6 e A_7 a fim de se obter uma uma reprodução do parâmetro formal. O transdutor em construção que fora

anteriormente desconectado do analisador léxico pela ativação de A_{31} é novamente conectado.

Seguem-se as descrições destas funções adaptativas.

Funções Adaptativas A_5 , A_6 e A_7

Estas funções adaptativas constituem um mecanismo adaptativo de cópias de regras que empilham símbolos na cadeia de entrada. Considere-se um conjunto de regras, como o ilustrado pelo exemplo seguinte:

$S \leftarrow \sigma_1 K_1$
$K_1 \leftarrow \sigma_2 K_2$
$K_2 \leftarrow \sigma_3 K3$
$R0 \rightarrow \{A\_SbAt(X)\}R1$
$R1 \rightarrow S$
$N15 \rightarrow U$
$Pont \rightarrow X$
$Pont \rightarrow R0$

A execução da ação adaptativa A5 e a primeira execução da ação adaptativa A6 permitem que se verifique a existência desse conjunto de regras.

As posteriores execuções da ação adaptativa A6 em conjunto com a ação adaptativa A7 permitem que se efetuem cópias dos n símbolos de empilhamento.

Observe-se que após a execução das ações adaptativas acima, é possível criar o seguinte conjunto de regras:

$N15 \rightarrow GV1$
$GV1 \leftarrow \sigma_1 GM_1$
$GM_1 \leftarrow \sigma_2 GM_2$
$GM_2 \leftarrow \sigma_3 GM_3$

A5 (X) =
{ Y, R1, S, U, *GV:
?[$P \rightarrow X$]

?[Y → X]

?[Y → R0]

?[R0 → R1 {A_SbAt(X)}]

?[R1 → S]

-[N15 → U]

+[N15 → GV]

+[GV → U] }

A6(S, GV, U) =

{X, σ:

?[(GS ← σ X]

A7(σ, GV, U, X) }

A7(σ, GV, U, X)=

{GM*:

-[(GV→ U]

+[GV ← σ GM]

+[GM → U]

A6 (X, GM, U) }

A figura seguinte ilustra o efeito da execução das ações adaptativas A_5, A_6 e A_7 sobre um conjunto de três transições de empilhamento dos símbolos σ_1, σ_2, e σ_3.

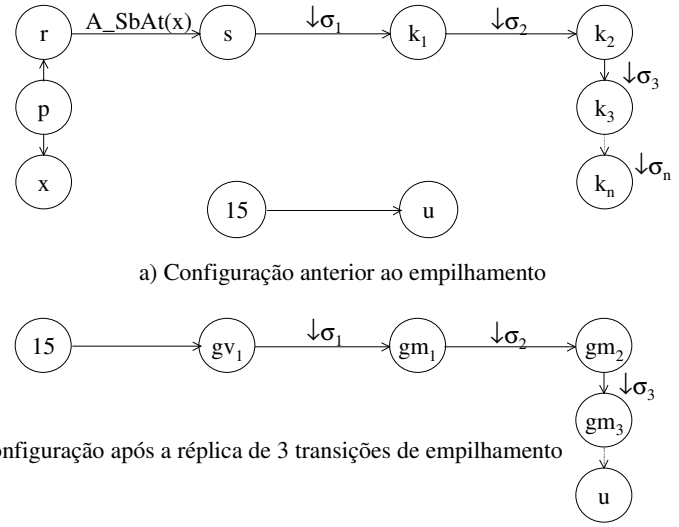


Fig. 24 - Resultado da ativação das funções A_5, A_6 e A_7 para a réplica de 3 símbolos

Avaliação de complexidade:

Se houver n regras de inserção de símbolos de contexto, tem-se que a gramática se expande em n regras (não adaptativas). São criados (n+1) símbolos não-terminais. A obtenção do novo conjunto de regras, implica na aplicação de (8 + n*4) regras.

Função adaptativa A_{26}:

Esta função adaptativa é ativada quando se detecta o fim da cadeia de símbolos que identificam um argumento, ou seja na ocorrência do símbolo delimitador (θ_1). A transição que a acionou é substituída por aquela inicial marcada com o símbolo "•", removida anteriormente pela ocorrência do mesmo símbolo. O símbolo "&" é passa a ser interpretado como delimitante de um parâmetro formal. A cadeia de inserção de símbolos concernentes ao argumento é associada ao seu respectivo parâmetro formal através da ligação de uma transição entre o estado onde é finalizado o identificador de parâmetro e aquele apontado pelo estado 15. Dessa forma, a extração de um parâmetro da cadeia de entrada, cujo identificador é finalizado no não-terminal W, no modo uso é sucedida pela inserção dos símbolos que constituem o identificador do argumento que lhe corresponde. Observe-se a figura 25.

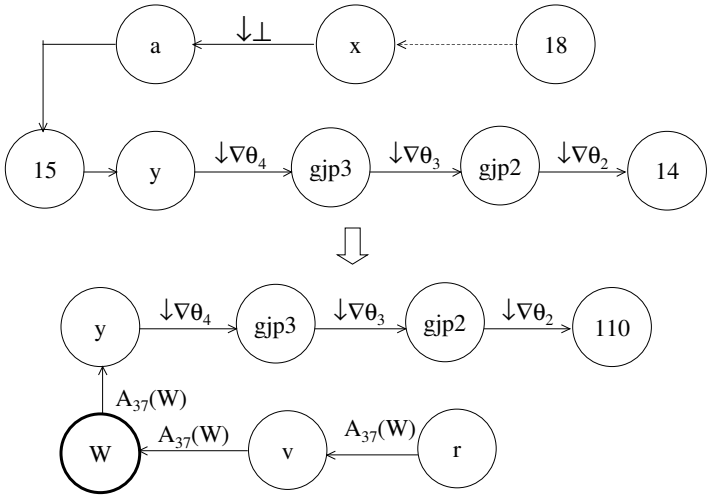


Figura 25 – Configuração de trecho do autômato após a execução da ação adaptativa A_{26}.

A_{26}(J, σ, K, W) =

{ X, Y, Z, R, S, t, V:
/*restaura transição em laço com o consumo do símbolo delimitador */
+[J → {A_{12} (σ), A_{23}( )} σ J] }
/*restaura transição consumindo símbolo •*/
-[J → {A_{26} (J, σ, K, W)} σ K]
+[J → {A_{26} (J, •, K, W)} • K]
/*restaura consumo do símbolo "&"
-[X → "&" N1]
+[N1 → "&" X]
-[S → {A_{22}( )} "&" N1]
+[N1 → {A_{22}( )} "&" S]
-["&" N80 → {A_{22}( )} e_com Z]
/*argumento associado ao parâmetro é desconectado da lista auxiliar:
é conectado ao identificador da regra e ao analisador léxico */
-[N15 → Y]
-[W → {A_{37}(W)} V]
-[R → {A_{37}(W)} W]
+[W → {A_{37}(W)} Y]
/*o identificador de parâmetros é apontado por uma lista de argumentos*/
+[V → {A_{37}(W)} W]
+[R → {A_{37}(W)} V]
-[W → {A_{37}(W)} V]
/*conexão ao léxico */
-[Z ← t N14]
+[Z ← t N110] }

Avaliação de Complexidade:

Esta função adaptativa é ativada sempre que for detectado o símbolo delimitador que finaliza o identificador de um argumento. Apresenta 8 ações elementares de inserção e 8 ações elementares de remoção e portanto, não altera o número de regras da gramática. A

execução da mesma consome o tempo de 16 ações elementares para cada parâmetro presente em uma regra.

Função Adaptativa A_{37}

Esta função adaptativa equivale a A_SbAt, ou seja mantém um ponteiro para o estado final de um identificador , que em A_{37} diz respeito a um parâmetro formal de regra. Tem-se:

$A_{37}(w) = \{X:$
$-[N90 \rightarrow X]$
$+[N90 \rightarrow W]\}$

Avaliação de Complexidade

$A_{37}(w)$ não altera o número de regras. O tempo de execução é o de duas regras elementares.

Com a conclusão de A_{26}, é finalizado também o reconhecimento do argumento. Sucede-se a inserção do corpo da regra na cadeia de entrada. Por outro lado, dado que a sintaxe das regras permite que estas se encontrem ora coordenadas entre si ora numa relação de subordinação, .prevê-se um mecanismo de contagem de regras em andamento associado ao corpo da regra., conforme a Fig. 12. Tal mecanismo adaptativo de contagem está associado às ações adaptativas A_{16} e A_{33}

As funções adaptativas A_{16}, A_{21} e A_{20}

A função adaptativa A_{16} é ativada em tempo de expansão, ou seja, sempre que no modo uso das regras ocorrer uma referência a uma determinada regra, esta é substituída pelo texto que lhe corresonde. Neste caso, o contador de regras em andamento é incrementado. A_{16}

o que o faz, mediante a ativação da função adaptativa A_{20}. Na primeira vez, A_{16} também executa A_{21}, responsável por configurar o analisador léxico para extrair nomes dos argumentos da regra.

Segue-se a descrição algébrica destas funções adaptativas.

A_{16}(J, K) = { f:

?[N100 →f N100]

{ A_{21} (f, j, k), A_{20}( ) } }

A_{21}(f, J, K) = {

-[N1 → {A_{22}( )} "&" N17]

+[N1 → "&" N70]

/* as duas alterações seguintes substituem a regra associada à execução da ação adaptativa A_{33}, por uma de inserção do meta-símbolo ■, associada à mesma ação. Este símbolo indica que não há nenhuma outra regra em expansão */

-[J → {A_{33}(J, K) } K]

+[J ← {A_{33}(J, K) } ■ K]

}

A_{20}( ) = { X, GJ*:

-[N100 → X]

+[N100 → GJ]

+[GJ → X]}

Avaliação de complexidade:

Estas ações acrescentam uma produção à gramática. Se se tratar da primeira ativação de A_{16}, o tempo de execução é o de 8 produções elementares, senão consome o tempo de exeucção de 4 produções elementares.

Função adaptativa A_{33}

O símbolo ■ deve ser inserido, para cada regra a ser processada, apenas uma vez na cadeia de entrada.. A ação adaptativa A_{33} substitui a transição responsável pela inserção do símbolo na cadeia de entrada, por uma transição em vazio. Tem-se:

$A_{33}(J, K) = \{ \sigma :$

$-[J \leftarrow \{A_{33}(J, K)\ \sigma\ K]$

$+ [J \rightarrow \{A_{33}(J, K)\ \ K]$

Avaliação de Complexidade:

A ação adaptativa A_{33} não altera o número de produções presentes na gramática. Consome o tempo de execução de duas ações elementares. É executada, apenas ao final do processamento de uma regra.

Para as situações de fim da substituição de um parâmetro por seu argumento, fim do tratamento da regra corrente e fim do processamento da regra em andamento, são inseridos na cadeia de entrada, para marcarem tais eventos, os seguintes símbolos: ⊥, ● e ■, a serem consumidos pelo analisador léxico e devidamente interpretados através das ações adaptativas, conforme o que se expõe a seguir.

Função adaptativa A_{32}

Quando é finalizado tratamento da regra corrente, há apenas que se decrementar o contador de macros em andamento.

$A_{32}(\ ) =$

$\{ X, y:$

$-[N100\ \rightarrow X]$

- [X → Y]
+[N100 → Y] }

Avaliação de Complexidade:

A função adaptativa A_{32} diminui em 1 o número de regras da gramática e seu desempenho é o de 3 regras elementares. É executada para toda a regra empregada ao longo da análise da sentença de entrada.

Função Adaptativa A_{18}

Esta função adaptativa é ativada pelo consumo do símbolo ⊥, indicador que foi finalizada a substituição de um parâmetro pelo seu respectivo argumento. A_{18} remove o parâmetro e o argumento de suas respectivas pilhas. Observe-se a Fig. 26

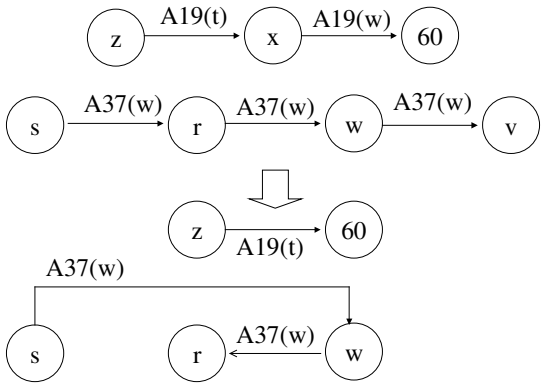


Fig. 26 – Funalização da Substituição de um Parâmetro por seu argumento

Segue-se a descrição algébrica de A_{18}

$A_{18}( ) =$ { X, Z, R, S, t, V, W:

/*remove da pilha de ponteiros de parâmetros, o último parâmetro */

-[X → {A_{19} (W)} N60]

-[Z → {A_{19} (T)} X]

+[Z → {A_{19} (T)} N60]

/*remove da pilha de argumentos homônimos associado ao parâmetro W, o último argumento*/

-[W → {A_{37} (W)} V]

-[R → {A_{37} (W)} W]

-[S → {A_{37} (W)} R]

+[W → {A_{37}(W)} R]

+[S → {A_{37} (W)} W]

}

Avaliação de Complexidade:

Esta função adaptativa é executada para cada argumento presente no corpo da regra. A cada execução, o número de regras da gramática é decrementado em 2 e o custo no tempo é o de 8 ações elementares

Função Adaptativa A_{34}

Uma vez finalizado o processamento de uma regra, o próximo símbolo que ocorrer na cadeia de entrada deverá marcar o início de um parâmetro formal e portanto deverá ser mapeado como e_com pelo analisador léxico. A_{34} configura o analisador léxico para esta situação. Tem-se:

A_{34}( ) = {

-[1 → "&" N70]

+[1 ⇒ {A_{22}( ) } "&" N17]

}

3.3 Conclusões

Neste item teceram-se considerações sobre os desdobramentos da utilização dos mecanismos de expansão apresentados em (NETO, 1993) e empregados para o processamento da especificação da Linguagem Natural no tratamento de Dependências de Contexto.

A avaliação de complexidade de cada uma das técnicas aqui apresentadas revela desempenho sempre proporcional ao comprimento do texto de entrada, número de argumentos parâmetros e ocorrências de determinados símbolos.

Trata-se de mecanismo que efetua sucessivas cópias de regras originais, fundamentado na existência de um analisador léxico original e de uma máquina reconhecedora da sintaxe das regras de representação. O uso iterativo destas duas máquinas permitem a expansão do analisador léxico, criando uma camada capaz de tratar as ocorrências no modo uso de um conjunto de regras originais, cujo aprendizado foi anteriormente efetuado.

A utilização destes mecanismos permitem a implementação do tratamento de não-determinismos e ambigüidades apresentados em (NETO; MORAES, 2002). Por outro lado se a esta época a descrição da gramática a partir de uma aproximação livre de contexto, nos mecanismos aqui estudados, os atributos das estruturas e a concordância entre os mesmos são facilmente representados e processados automáticamente.

Como já afirmado na introdução deste capítulo, a conversão ***automática*** da representação de uma estrutura em uma seqüência de símbolos, permite inferir que a representação interna destas estruturas sejam processadas com o mesmo desempenho indicado no capítulo 2.

Levando-se em consideração os resultados apresentados neste capítulo e no que lhe antecede, o próximo tece algumas considerações adicionais sobre a proposta de uma arquitetura adaptativa para processamento de Linguagem Natural.

4 UMA ARQUITETURA ADAPTATIVA PARA PROCESSAMENTO DE LINGUAGEM NATURAL

Este capítulo apresenta uma proposta de arquitetura para um software destinado a efetuar o processamento – mais especificamente, a análise – de linguagem natural, empregando para isso técnicas de análise que fazem uso da tecnologia adaptativa.

Para isso, na proposta descrita nesta tese, a representação externa da linguagem natural se faz com o emprego de uma meta-linguagem, a qual é transformada em um dispositivo reconhecedor que opera como um transdutor baseado em autômatos de pilha estruturados adaptativos.

Esse transdutor opera em quatro níveis sobre o texto de entrada:

- a extração de palavras do texto, sua classificação segundo a categoria a que pertence, sua decomposição em partículas elementares e sua etiquetação ficam a cargo de um autômato adaptativo;
- a parte correspondente à verificação sintática das regras de colocação fica a cargo de outro autômato adaptativo, encarregado da verificação da parte estática da sintaxe da linguagem. Embora neste nível seja, a rigor, desnecessário empregar autômatos adaptativos, sua utilização facilita muito a verificação sintática de textos cujos componentes estejam elipticamente omitidos, ou então, textos cujos constituintes não estejam apresentados em sua ordenação primária;
- a verificação de aspectos mais complexos da linguagem, representados principalmente por aqueles ligados à concordância, à regência, às anáforas, aos pronomes, etc. Aqui também os autômatos adaptativos se apresentam como uma

alternativa interessante para a resolução desses problemas de dependências contextuais apresentados pelas linguagens naturais. Neste caso específico, o tratamento das dependências contextuais é feito adicionando-se ações adaptativas ao autômato utilizado para a verificação das regras de colocação.

- em um quarto estágio, são tratados problemas associados à presença de ambigüidades e não-determinismos decorrentes das inúmeras combinações legítimas permitidas pelas linguagens naturais através de permutações na ordenação de seus componentes, dando assim margem a múltiplas interpretações válidas para o texto de entrada, o que acarreta correspondentemente o aparecimento de muitos caminhos válidos simultâneos nos autômatos que implementam a análise do mesmo.

Assim, como se pode notar, é possível, com um único modelo, baseado no autômato adaptativo, efetuar todo o tratamento do texto de entrada, desde seus aspectos morfológicos mais elementares, passando pela ordenação de seus constituintes, até a verificação dos aspectos mais complexos das possíveis transposições e da interação entre as diversas partes estruturais de que é formado.

4.1 Considerações Gerais

A figura 27 representa a arquitetura de uma ferramenta que disponibiliza a um lingüista, especialista em linguagem natural, mas não necessariamente conhecedor do paradigma computacional adaptativo, uma interface através da qual possa descrever a linguagem natural através de uma gramática.

Essa ferramenta destina-se também para ser utilizada principalmente por um usuário final, seja ele homem ou máquina, interessado em obter uma floresta de árvores sintáticas,

associadas a um texto de entrada por ele fornecido, na qual cada árvore representa uma possível interpretação legítima do texto de entrada.

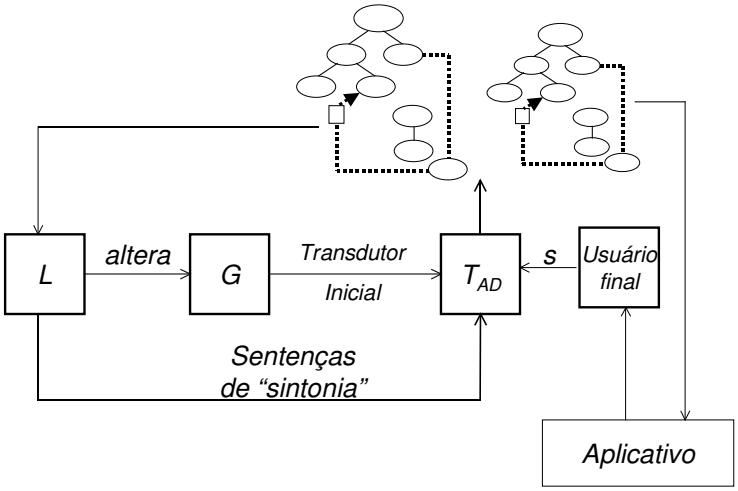


Figura 27 – Arquitetura adaptativa para processamento de linguagem natural

Nesta figura, L representa um lingüista, G uma gramática da linguagem natural e T_{AD} representa um transdutor adaptativo.

O lingüista, que é especialista em linguagem natural, tem acesso a operações através das quais pode inserir, remover ou simplesmente alterar um conjunto de regras que define a gramática G, a qual, por sua vez, descreve a linguagem L desejada.

Também é possível ao lingüista apresentar à ferramenta sentenças de "sintonia", destinadas a facilitar o ajuste da gramática, obter o conjunto das árvores de sintaxe correspondentes a essas sentenças, avaliar a gramática disponível e eventualmente efetuar modificações no seu conjunto de regras.

Em sua ***configuração inicial***, o transdutor deve ser capaz de memorizar ***um dicionário*** juntamente com as ***regras da gramática*** fornecidas pelo lingüista, estando então o transdutor

adaptativo T_{AD} em um modo de operação denominado "**treinamento**", e promovendo como resultado sua automodificação para uma nova configuração.

Isso se repete enquanto novas alterações forem sendo inseridas pelo lingüista, até que o transdutor adaptativo atinja uma ***configuração final***, na qual ele seja considerado, pelo lingüista, apto a analisar, à luz do dicionário e da gramática, textos de entrada fornecidos por um usuário final, escritos na linguagem natural a que se referem o dicionário e a gramática.

O usuário final é o elemento interessado na análise de um texto de entrada, e, do ponto de vista da ferramenta, distingue-se do lingüista porque não tem direito de efetuar alterações no conjunto de regras que definem a gramática da linguagem natural.

A seguir, os elementos da ferramenta e sua função no processamento de linguagem natural são descritos, com alguns pormenores adicionais.

4.2 Elementos Principais da Ferramenta

A ferramenta aqui proposta tem como elemento central um transdutor adaptativo, que incorpora, em sua configuração final, atingida após o devido treinamento, as funções de análise léxica e sintática, o dicionário e a gramática da linguagem, o texto de entrada a ser analisado, e as árvores de sintaxe correspondentes ao texto de entrada, de acordo com a gramática da linguagem.

Esses elementos, ligeiramente comentados a seguir, constituem as partes mais importantes da ferramenta proposta.

4.2.1 Analisadores léxico e sintático

Inicialmente identificam-se, para o transdutor adaptativo, dois grandes módulos funcionais: o analisador léxico e o analisador sintático.

O analisador léxico efetua a separação e classificação dos componentes léxicos do texto de entrada e identifica os vínculos sintáticos impostos pelos mesmos aos demais componentes da sentença.

Cabe também ao analisador léxico decompor eventualmente as palavras em seus componentes mais primitivos, separando assim os elementos constituintes das contrações, isolando os radicais das palavras, identificando seus prefixos e sufixos, e assim por diante, e dando um tratamento individualizado para cada uma dessas partículas.

É também da alçada do analisador léxico consultar o dicionário, determinando, para a palavra extraída, informações adicionais necessárias para uma eventual eliminação de ambigüidade.

Ao analisador sintático cabe primariamente o papel de reconhecedor, através do qual efetua a verificação da validade da colocação dos elementos léxicos, observada no texto de entrada.

Considerando que linguagens naturais toleram inúmeras permutações e omissões para os elementos constituintes de suas sentenças, e que tais elementos em geral podem ter estruturas similares, encontra-se sempre um problema de porte razoável quando se deseja determinar de forma rigorosa o papel de cada componente do texto analisado.

Decorre imediatamente que a convivência com essas múltiplas interpretações (ao menos, parcialmente, durante a análise) é necessária, e isso exige do reconhecedor que tenha a possibilidade de lidar com situações não-determinísticas, e do usuário, que aceite como um fato as ambigüidades da língua, recebendo eventualmente por isso mais de uma interpretação para um mesmo texto de entrada.

Superados esses problemas de não-determinismo e ambigüidade, o analisador sintático deve gerar, com base em uma descrição formal (geralmente gramatical) da sintaxe da

linguagem, e também, naturalmente, no próprio texto a ser analisado, uma ou mais árvores de sintaxe a ele correspondentes.

4.2.2 Dicionário e gramática

São bases de dados que disponibilizam à ferramenta elementos para que esta possa conduzir adequadamente o trabalho de análise da sentença em linguagem natural.

O analisador léxico se serve principalmente do dicionário para efetuar a identificação das palavras válidas da língua, a extração, a classificação morfológica e a etiquetação das palavras do texto de entrada (como resposta, pode-se esperar eventualmente mais de um resultado válido desta análise).

Já o analisador sintático se utiliza dos elementos léxicos extraídos pelo analisador léxico, bem como das regras que compõem a gramática da linguagem, para verificar a qual (eventualmente pode-se aqui também obter mais de um resultado válido) das possíveis construções sintáticas legítimas da linguagem pertence o texto em análise.

Primariamente, o dicionário pode ser visto como uma coleção de todas as palavras que a língua natural permite utilizar nas suas sentenças.

Entretanto, na prática a isso se acrescentam dados de caráter outro que não puramente morfológico, incluindo informações sintáticas e semânticas sobre a palavra em questão.

As gramáticas incluem informação sobre as seqüências que a linguagem permite utilizar na construção das sentenças.

Devem conter elementos que identifiquem os grandes componentes das sentenças, sua colocação relativa, seus elementos opcionais, a estrutura interna de cada um desses elementos e as suas regras de formação, as flexões esperadas para cada um dos componentes léxicos, etc.

Como complemento, as gramáticas devem informar acerca da interação esperada entre seus elementos, como é o caso de concordâncias e regências, omissões de elementos, etc.

4.2.3 Texto de entrada e árvores de sintaxe

Através de um arquivo de texto não-formatado, o usuário final pode alimentar o sistema com um texto de entrada, escrito em linguagem natural, esperando que em resposta o sistema construa automaticamente uma correspondente árvore de sintaxe (pode eventualmente haver mais de uma), de acordo com as regras da gramática disponível.

Naturalmente, para que haja uma correta aceitação do texto, este deve ser constituído integralmente de palavras pertencentes ao conjunto representado pelo dicionário existente, e essas palavras devem estar flexionadas de acordo com as regras de flexão impostas pela categoria lexical a que pertence, e às regras de flexão indicadas no dicionário e previstas na gramática da linguagem em questão.

Uma palavra deve ser separada de suas vizinhas por espaçadores específicos de cada linguagem, geralmente espaços em branco, sinais de pontuação, fim de linha e similares.

Cabe, obviamente, ao analisador léxico identificar os limites de cada palavra, de acordo com tais regras, e isolar essa palavra para depois iniciar sua análise.

Uma vez isolada a palavra, esta pode ser classificada e adequadamente etiquetada, para ser então empregada na construção da árvore de sintaxe do texto de entrada.

Neste ponto, a análise sintática inicia seu processo de conversão de uma seqüência de palavras classificadas e etiquetadas em uma árvore de sintaxe construída de acordo com as regras de colocação estabelecidas pela gramática da linguagem de entrada.

Árvores de sintaxe estabelecem o relacionamento estrutural entre palavras ou entre sub-árvores de sintaxe da sentença a que se referem, e são caracterizadas por apresentarem

cada qual um único nó principal, correspondente ao não-terminal raiz da gramática, nós-folha representando cada um dos terminais encontrados no texto de entrada, e nós intermediários que promovem a interligação dos demais nós.

A interligação de cada um dos nós-pai com um conjunto correspondente de nós-filhos imita a maneira como a correspondente regra de produção da gramática manda substituir um nó não-terminal (associado ao lado esquerdo da produção) por uma seqüência formada de terminais e não-terminais (relativa ao seu lado direito).

Uma árvore de sintaxe deve ter como contorno exatamente a cadeia de terminais correspondente ao próprio texto de entrada a que se refere a árvore.

Pode haver mais de uma árvore de sintaxe para uma mesma sentença da linguagem, e quando isso acontece, indica que a gramática utilizada para a definição dessa linguagem é uma gramática ambígua.

Nesse caso, árvores diferentes indicam formas diferentes (mas legítimas) de interpretação dessa mesma sentença, e a diferença entre elas está exatamente nos diferentes conjuntos de nós intermediários que as constituem, e na variedade de interconexões utilizadas na formação das diferentes árvores.

A ausência de uma árvore de sintaxe que corresponda ao texto de entrada segundo a gramática da linguagem mostra que o texto de entrada não é uma sentença corretamente formada segundo o conjunto regras gramaticais adotado para definir a linguagem natural nessa ferramenta.

4.3 Modos de operação da ferramenta

A ferramenta proposta apresenta, como foi sugerido anteriormente, dois modos de operação, a saber: o modo **treinamento** e o modo **uso**.

No modo **treinamento** a ferramenta exibe, no transdutor adaptativo que a realiza, uma configuração inicial, que permite uma primeira memorização de regras de uma gramática que caracteriza formalmente a linguagem desejada, estabelecidas por um lingüista.

Graças aos mecanismos adaptativos presentes, com base nas regras gramaticais recebidas do lingüista, o transdutor se auto-modifica em resposta a este processo de aprendizagem, de tal forma que a ferramenta resultante, nele apoiada, se vá tornando capaz de realizar cada vez mais das tarefas de análise léxica e sintática de um texto de entrada.

Sucessivas modificações nas regras vão assim sendo introduzidas através da alteração do conjunto de regras gramaticais, e as evoluções correspondentes vão sendo realizadas sobre o transdutor, até que a ferramenta se torne finalmente capaz de reconhecer e analisar um texto de entrada em todos os aspectos lingüísticos considerados essenciais da particular linguagem natural desejada, ficando assim a ferramenta pronta para ser utilizada por um não-especialista, para a análise dos seus textos específicos, em linguagem natural.

Aqui, portanto, a ferramenta está preparada para ser utilizada em modo **uso**. Naturalmente, agora se pode considerar que todas as informações importantes acerca da linguagem já são de conhecimento da ferramenta, por ter sido esta previamente treinada pelo lingüista.

O usuário final pode então fornecer à ferramenta seus textos de entrada, escritos na linguagem natural, e, com base na gramática da linguagem em questão, conhecida do sistema, pode finalmente ser produzida uma ou mais árvores de sintaxe correspondentes aquele texto.

O modo **uso**, portanto, dispensa o lingüista, e pode ser exercitado diretamente pelo usuário final, o qual faz dos resultados fornecidos pela ferramenta o uso que for mais apropriado, tais como: verificar a correção ortográfica e gramatical de seu texto de entrada; conhecer as possíveis interpretações do texto; utilizar as estruturas sintáticas possíveis, levantadas pela ferramenta, para alimentar um estágio adicional de processamento da

linguagem natural que promova a extração do conteúdo semântico do texto de entrada; alimentar um estágio de processamento encarregado de efetuar uma tradução do texto de entrada para algum outro idioma, etc.

Existe um terceiro modo de operação: trata-se do modo **expansão**. Nas linguagens naturais, são profusas as relações de subordinação. Estas dependências se dão em diferentes graus de complexidade por meio de diversos expedientes. A ocorrência de um item lexical, de uma categoria gramatical, de uma determinada flexão, etc. é gramaticalmente correta, somente se após a ocorrência do termo subordinado a este primeiro. Esta situação é facilmente tratada pelo formalismo adaptativo. Uma vez que na regra se especifique a ocorrência de um determinado símbolo é possível representar o elemento a ele subordinado. Em tempo de uso desta regra, tão logo seja detectada na cadeia de entrada o símbolo, núcleo da subordinação, pelo analisador léxico, o transdutor adaptativo se expande de forma que seja acrescentado um trecho do analisador sintático, responsável pela verificação do termos que ao núcelo se subordinam.

O mecanismo de expansão consiste em criar uma cópia do trecho da gramática que geraa o termo subordinado . A utilização do mecanismo de expansão é pois a técnica empregada para o tratamento destes não-determinismos profusos na gramática natural. É também adotado no tratamento de ambigüidades.

4.4 O tratamento de não-determinismos e ambigüidades na ferramenta proposta

Um problema recorrente no estudo de linguagens complexas, como é o caso das linguagens naturais, é a presença de situações para as quais, em resposta a um dado estímulo, o sistema nem sempre evolui obrigatoriamente para uma única situação seguinte.

Em alguns casos, é possível evitar tais situações revendo a formulação adotada para a caracterização da linguagem, e redescrevendo-a de alguma outra maneira que evite o problema.

Contudo, essa manobra raramente é possível no âmbito das linguagens naturais, e se pode afirmar com segurança que é relativamente grande a freqüência dos casos que exigem, infelizmente, que se conviva com tais situações.

Situações com múltiplas evoluções possíveis surgem todas as vezes em que for detectada, durante o tratamento da linguagem, alguma condição de ambigüidade ou de não-determinismo.

Uma das formas mais intuitivas de considerar, na prática, tais situações consiste em se estudar, em paralelo, todas as combinações alternativas por elas suscitadas, de modo que, se alguma dessas combinações for válida, tal combinação leve à construção de uma árvore de derivação completa e correta.

Equivalentemente, é possível obter o mesmo resultado fazendo-se um a um, seqüencialmente, o estudo individual de apenas uma dessas possibilidades de cada vez, como se as demais não existissem, coletando ao final de cada um desses estudos, a eventual árvore completa que este caso particular origina.

Como no caso paralelo, caso se adote essa técnica, o processamento sucessivo das diversas combinações válidas de alternativas deve ser feito até que todas elas tenham sido verificadas, e todas as árvores válidas, coletadas.

Naturalmente, nessa opção seqüencial, os casos de insucesso na construção da árvore leva a descartar aquela combinação de possibilidades, e não à rejeição da entrada.

A rejeição do texto de entrada só deverá acontecer, neste caso, se todas as combinações consideradas inicialmente como válidas falharem em produzir árvores corretas.

Do ponto de vista de implementação, é possível considerar caso a caso tais condições indesejáveis de não-determinismo ou ambigüidade, ou então, criar um mecanismo geral para efetuar seu tratamento de maneira mais automática.

Neste último caso, disponibiliza-se um procedimento geral de tratamento de tais fenômenos, o qual pode então ser ativado sempre que necessário, sem que haja necessidade que se faça individualmente um detalhamento específico dos pormenores de cada caso particular.

Em (NETO; MORAES, 2002), é apresentada uma proposta adaptativa para o tratamento de não-determinismos e ambigüidades onde se busca o conjunto de todas as interpretações possíveis para uma dada sentença.

Para isso, o transdutor adaptativo deve ser construído de tal forma que as construções normais sejam aceitas da forma usual, e que as situações de não-determinismo ou de ambigüidade, uma vez identificadas, criem, para serem executadas em paralelo, cópias da gramática, cada uma das quais capaz de reconhecer uma das possíveis construções sintáticas válidas.

O paralelismo da operação desses autômatos é simulado através da sua execução alternada, passo a passo, em que cada um deles efetua uma transição por vez, correspondente ao consumo de um símbolo da cadeia de entrada.

As diversas versões da gramática vão sendo executadas simultaneamente, sendo descartadas aquelas que não permitirem o prosseguimento do reconhecimento, e sendo consideradas vitoriosas aquelas que conseguirem esgotar a cadeia de entrada em uma situação final.

Havendo mais de uma máquina vitoriosa, para a mesma sentença, isso será indicativo de que a sentença em questão tem mais de uma interpretação válida, decorrente da ambigüidade presente na linguagem.

Uma das premissas fundamentais em (NETO; MORAES; 2002) é pois que o tratamento adequado das situações de não-determinismos e ambigüidades faça uso de novas instâncias de uma sub-gramática original. Essa situação corresponde nas linguagens de programação ao tratamento de macros. A proposta adaptativa para o tratamento de macros é apresentada na seção 4.4.8 em (NETO, , 1993).

Suscitada pela busca de um mecanismo adaptativo de cópia, necessário ao tratamento de não-determinismos e ambigüidades efetuou-se uma análise dos referidos mecanismos. Concluiu-se que as mesma técnicas podem ser empregadas em processamento de Linguagem Natural. Tal análise é relatada no capítulo 4 da presente tese.

4.5 Descrição dos Componentes da Ferramenta

A figura 28 ilustra, em um primeiro nível de detalhamento, os blocos funcionais da ferramenta destinados ao tratamento sintático da linguagem natural, a saber: o analisador léxico, o analisador sintático e o módulo que efetua o tratamento das ambigüidades e não-determinismos detectados pela análise léxica ou pela análise sintática.

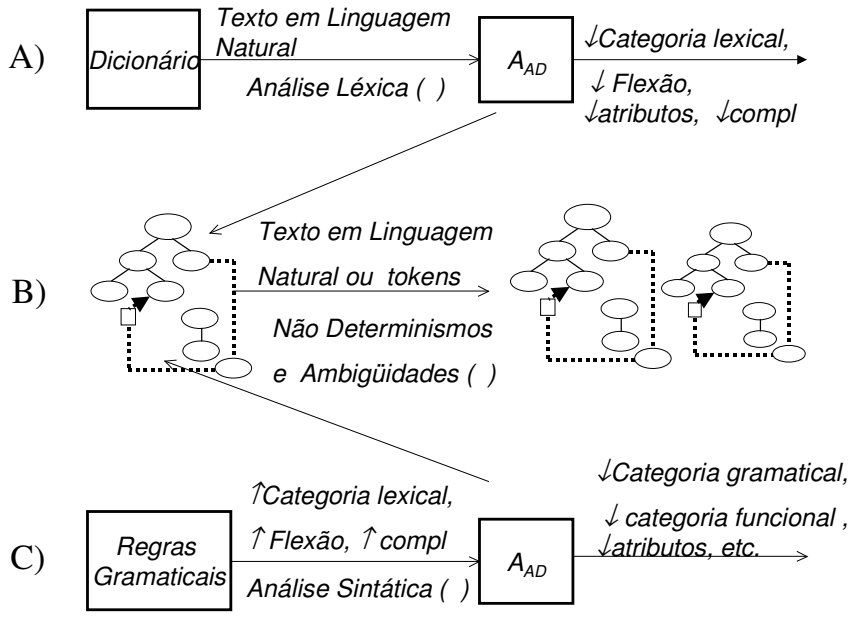


Figura 28 – Blocos Funcionais: A) Análise Léxica B) Tratamento de Não-Determinismos e Ambigüidades C) Análise Sintática

Observe-se que, neste nível superior da abstração, nada é imposto acerca da natureza da linguagem de entrada da ferramenta, e, portanto, o raciocínio aqui desenvolvido, embora seja melhor aplicado a linguagens de alta complexidade, pode, na realidade, ser aplicado a qualquer tipo de linguagem, mesmo que não seja uma linguagem natural.

Em modo **treinamento**, os blocos funcionais podem ser empregados para tratarem a meta-linguagem através da qual a ferramenta recebe do lingüista as informações sobre a linguagem natural a ser processada, e essa meta-linguagem, cuja estrutura é livre de contexto, é com muita certeza muito mais simples que qualquer linguagem natural que através dela costuma ser definida.

A descrição, a seguir, dos componentes representados pelos blocos funcionais da ferramenta, referem-se mais propriamente ao seu funcionamento em modo **uso**.

4.5.1 O analisador léxico

Em 28.A) representa-se o ***analisador léxico***, bloco funcional da ferramenta que é responsável pela extração das palavras do texto fornecido pelo usuário final, pela sua classificação segundo a categoria lexical a que pertence, e sua decomposição em partes menores, correspondentes aos desmembramentos de contrações e de flexões, prefixos e sufixos incorporados nas palavras extraídas.

Para reconhecer e classificar as palavras, o analisador léxico faz uso de um dicionário cujo conteúdo deve conter informações que permitam determinar todas as categorias lexicais possíveis para cada palavra.

O resultado da execução do analisador léxico compõe-se de um ou mais tokens, relativos à primeira das palavras ainda não extraídas do texto, devidamente separada do texto de entrada, desmembrada em seus componentes primários, apropriadamente classificada (categoria lexical, informação sobre o tipo de complementos exigidos) e acompanhada dos seus atributos de flexão (gênero, número, grau, pessoa, tempo, modo, etc.), e de informações relativas ao quadro de subcategorização do item lexical.

É conveniente notar que no dicionário diversas informações adicionais, de cunho não puramente morfológico, costumam ser obtidas sobre a palavra extraída, tais como as exigências contextuais da palavra, em matéria de concordância e de regência, informações sobre seus possíveis significados em diversos contextos, informações sobre os seus usos mais freqüentes, expressões idiomáticas nas quais costuma figurar, etc.

É importante notar a forte interação necessária que deve existir entre o analisador léxico e o dicionário, por um lado – é no dicionário que o analisador léxico encontra a maior parte das informações lingüísticas sobre a palavra analisada, e entre o analisador léxico e o

analisador sintático, por outro – o analisador sintático é um forte usuário das informações obtidas na análise léxica.

4.5.2 O analisador sintático

Em 28.C) representa-se o analisador sintático, bloco funcional da ferramenta que se responsabiliza pela análise da estrutura do texto de entrada, e pela construção de todas as possíveis árvores de sintaxe, que possam ser geradas pela gramática da língua natural em questão a partir das saídas do bloco funcional que executa a análise léxica da sentença fornecida como texto de entrada pelo usuário final.

Há na literatura muitos métodos relatados para a realização da análise sintática para uma linguagem descrita por uma gramática. (AHO, ULLMAN, 1972); (ALLEN, J., 1995). Em (JURAFSKY; MARTIN, 2000) e (MATTHEWS, 1998.) são citados inúmeros trabalhos.

Entre as estratégias de análise sintática mais utilizadas, destacam-se: a análise descendente, a análise ascendente e a análise com o auxílio de reconhecedores ou autômatos.

Em qualquer das análises, tem-se uma área de trabalho, na qual se vai construindo a árvore à medida que isso for possível a partir da configuração corrente da árvore, do conteúdo do texto de entrada e de alguma regra gramatical aplicável no momento.

Distinguem-se as análises descendente e ascendente quanto ao não-terminal específico que, com base nas regras gramaticais aplicáveis, é escolhido para ser substituído, na área de trabalho, num determinado momento da análise.

Assim, ***analisadores descendentes*** escolhem sempre o não-terminal mais à esquerda ainda não substituído para sobre ele aplicar uma regra gramatical em que ele conste como membro esquerdo, substituindo-o na área de trabalho, pelo lado direito da regra em questão.

Isso faz com que a árvore de sintaxe seja construída partindo sempre da raiz, em direção às folhas, e que o ponto em que é feita a substituição seja um ponto de limite entre a parte esquerda da árvore, cujas folhas são todas constituídas de terminais, e a parte direita da mesma, cujo contorno não é formado apenas de terminais, mas portanto apresenta ainda nós não-terminais a serem substituídos.

Para dar partida à operação de um analisador descendente, inicia-se a área de trabalho apenas com o não-terminal que corresponde à raiz da gramática da linguagem desejada, correspondendo a iniciar o trabalho apenas com a raiz da árvore de sintaxe.

No prosseguimento da análise descendente, escolhe-se para ser substituído sempre o nó não-terminal cuja posição na árvore seja aquela mais à esquerda no contorno da árvore.

Em outras palavras, visto pela perspectiva da forma sentencial que constitui o conteúdo da área de trabalho em cada momento, procura-se substituir o não-terminal mais à esquerda dessa forma sentencial.

A escolha da substituição a ser aplicada consiste em consultar a gramática, em busca de uma regra de substituição para tal não-terminal, mas que seja compatível com o próximo token extraído do texto de entrada pelo analisador léxico.

Por outro lado, ***analisadores ascendentes*** escolhem primeiramente para ser substituído o não-terminal mais à direita da forma sentencial contida na sua área de trabalho.

Inicialmente, preenche-se a área de trabalho com o texto de entrada devidamente convertido pelo analisador léxico à forma de uma seqüência de tokens.

Um dispositivo tomador de decisão, específico para cada gramática e linguagem, e construído (usando-se técnicas clássicas, (cuja discussão extrapola o escopo deste trabalho) geralmente com base em um autômato finito, varre a área de trabalho, em busca de reduções a aplicar sobre a forma sentencial corrente.

Uma *redução* consiste na substituição, na forma sentencial, de uma ocorrência mais à direita de algum padrão correspondente ao lado direito de alguma regra gramatical da gramática, pelo não-terminal associado ao lado esquerdo da regra que estiver sendo aplicada.

Após cada redução, repete-se o processo, procurando-se aplicar sucessivas reduções sobre a forma sentencial contida na área de trabalho, até que esta seja reduzida, finalmente, a um único não-terminal, correspondente à raiz da gramática.

A coleta das reduções efetuadas permite a construção da árvore de sintaxe do texto analisado pela aplicação da seqüência coletada de todas as decisões tomadas pelo dispositivo tomador de decisões do analisador sintático ascendente.

Uma terceira possibilidade, que foi adotada para a ferramenta aqui proposta, consiste na utilização de um reconhecedor como elemento básico do analisador.

Pelo fato de que em sua operação um reconhecedor se limita a aceitar ou não suas cadeias de entrada como como sendo parte da linguagem, torna-se conveniente enriquecê-lo com algumas funções adicionais, em particular, com a capacidade de também gerar saídas, o que caracteriza o dispositivo assim construído como sendo um transdutor.

Para a ferramenta proposta neste trabalho, escolheu-se como técnica a aplicar a utilização de um transdutor baseado em autômatos de pilha estruturados adaptativos como elemento primário responsável pelo reconhecimento sintático da linguagem.

Devido à natureza complexa das linguagens naturais (ambigüidades, recursividade à esquerda), nem todos os tipos de autômato se servem para essa função, razão pela qual se optou pelo emprego de autômatos de pilha estruturados adaptativos.

Estes dispositivos (NETO, 1993), apresentam poder de máquina de Turing, e têm a desejável propriedade de permitir o tratamento simplificado de linguagens regulares ou livres de contexto apenas utilizando suas transições básicas e sua pilha sintática.

Por outro lado, caso sejam necessários maiores recursos do autômato, por causa da complexidade da linguagem, os autômatos adaptativos permitem também representar o tratamento de dependências contextuais, necessárias ao processamento de linguagens dependentes de contexto, como é o caso das linguagens naturais.

Para que isso se torne possível, esses dispositivos fazem uso de ações adaptativas, destinadas a proporcionar ao autômato a capacidade de se auto-modificar, podendo eles, através desse recurso, memorizar informações e alterar seu próprio comportamento, adaptando-o às necessidades de cada particular situação.

Disso resulta um dispositivo que, a partir de uma configuração inicial genérica, em que qualquer texto de entrada poderia, potencialmente, ser aceito, tenha a capacidade de modificar-se à medida que elementos específicos do texto de entrada vão sendo encontrados, adaptando-se assim às suas particulares necessidades.

Através da incorporação ao autômato das informações adequadas, extraídas dos elementos do texto, à medida que a análise desse texto de entrada vai se desenvolvendo, o autômato vai se especializando correspondentemente.

Dessa maneira, torna-se possível projetar as ações adaptativas de auto-modificação do autômato concebidas de tal forma que opções lingüísticas não utilizadas possam ir sendo excluídas das atenções do autômato, enquanto aquelas que estejam em uso sejam enriquecidas com as opções viáveis apenas.

O resultado do uso dessa estratégia é que, dinamicamente, o autômato adaptativo assim construído permanece sempre em sintonia com o particular texto de entrada que está sendo analisado, e que partes do autômato que não seriam percorridos nem sequer são mantidas no dispositivo, evitando assim o gasto desnecessário de áreas de armazenamento.

4.5.3 O tratamento de ambigüidades e não-determinismos

Na figura 28.C) representou-se o módulo que efetua o tratamento das ambigüidades e não-determinismos.

Embora não se trate de uma componente cuja necessidade seja evidente num primeiro momento, é de suma importância a sua presença no sistema, pois permite uniformizar e, até um certo ponto, automatizar o tratamento de todas as situações em que mais de uma decisão válida possa ser tomada a partir de um mesmo estímulo, em algum ponto específico da análise (léxica ou sintática).

Ao ser acionado, este módulo cria as instâncias que forem necessárias, da análise em andamento, para se levar em consideração, separadamente, cada um dos casos possíveis detectados.

No caso da análise léxica, este componente de tratamento de ambigüidades e não-determinismos deve ser acionado sempre que o token extraído apresentar mais de uma classificação possível.

No caso da análise sintática, o tratamento de ambigüidades e não-determinismos deve ser acionado sempre que mais de uma regra gramatical puder ser aplicada, ou, equivalentemente, quando, a partir do estado corrente do autômato, mais de um caminho legítimo for encontrado para o token em uso.

Convém lembrar que o uso automático desses procedimentos de tratamento de ambigüidades e não-determinismos só se torna possível se seu acionamento for implicitado no funcionamento geral dos dispositivos de análise.

Assim, o transdutor que implementa as análises léxica e sintática subentende a ativação automática do analisador léxico todas as vezes que um token é consumido por

alguma transição do autômato subjacente do transdutor sintático, de tal modo que sempre se tenha disponível um próximo token a ser analisado.

Da mesma maneira, deve-se subentender a ativação dos mecanismos para o tratamento adequado dos não-determinismos ou das ambigüidades correspondentes às situações em que o analisador léxico ou o analisador sintático se depararem com múltiplas decisões válidas, igualmente aplicáveis no momento.

Isto é o que acontece todas as vezes que uma resposta múltipla for gerada pelo analisador léxico (mais de uma classificação possível para uma mesma palavra) ou então pela consulta à gramática, em busca das regras aplicáveis (mais de uma regra, todas igualmente aplicáveis), ou ainda pela detecção de uma situação não-determinística no autômato subjacente ao transdutor utilizado).

Em qualquer desses casos, cada uma das possibilidades deve ser analisada, qualquer que seja a técnica adotada: uso de paralelismo ou da serialização das situações, ou o uso de threads na implementação, permitindo explorar algum paralelismo disponível na plataforma existente ou então promovendo a simulação de tal paralelismo. Como já citado, em (NETO; MORAES, 2002) adotou-se a simulação do paralelismo

4.6 As bases de dados

Para ser possível à ferramenta efetuar adequadamente a análise de uma linguagem, uma descrição completa de todos os aspectos desta linguagem que sejam relevantes para a análise desejada deve estar formalmente representada no sistema, disponível em alguma notação por ele conhecida.

Tais descrições podem ser apresentadas de inúmeras maneiras alternativas, como, por exemplo, entre muitas outras possibilidades, na forma de functores, na forma de expressões

regulares, na notação de Wirth, na forma de gramáticas adaptativas, na de conjuntos de produções livres ou dependentes de contexto, na forma de autômatos finitos, ou de pilha, ou adaptativos, etc.

Todas essas formas são válidas, podem ser indiferentemente utilizadas, e se equivalem mutuamente, guardados os seus níveis relativos de expressividade, e a escolha dentre elas pode ser feita com base em algum critério arbitrário, usualmente baseado nas disponibilidades e nas conveniências do projeto.

A equivalência teórica dos diversos formalismos computacionais anteriormente citados, permite converter, direta ou indiretamente, cada um deles em uma versão equivalente, denotada no formato dos demais formalismos, entre os quais, os autômatos de pilha estruturados adaptativos.

Em (TANIWAKI, 2000) encontram-se alguns resultados obtidos nesse sentido, que mostram algumas das equivalências e constatam a viabilidade do uso de técnicas adaptativas no processamento de linguagens naturais.

Outro esforço neste sentido, no qual se demonstrou viável a utilização de tecnologia adaptativa na construção de analisadores léxicos ou etiquetadores automáticos para a língua portuguesa, encontra-se em (MENEZES, 2000)

O procedimento da análise de linguagem natural aqui utilizado, empregando os transdutores adaptativos, inspira-se nos métodos inicialmente apresentados em (NETO, 1993) para a resolução de dependências de contexto em linguagens de programação, detalhado no capítulo anterior.

Ainda, convém, naturalmente, destacar que uma ferramenta como a que está sendo proposta disponibiliza a seus usuários uma interface em linguagem natural que pode ser utilizada por outros aplicativos, tal como foi apresentado esquematicamente na figura 27.

Com a ajuda deste tipo de recurso, o usuário final dos aplicativos pode operá-los fornecendo-lhes sentenças em linguagem natural em lugar de instruções especiais, expressas em alguma linguagem de comandos especialmente desenvolvida para o aplicativo.

Por sua vez, para o lingüista, que manipula os aspectos ligados à formalização da linguagem natural utilizada, a ferramenta deve dar acesso a alguma meta-linguagem, tipicamente alguma notação gramatical, empregada para expressar a estrutura da linguagem.

Nessas meta-linguagens, regras de construção de sentenças permitem desenvolver uma descrição rigorosa da estrutura sintática da linguagem, da qual é possível determinar com precisão o comportamento esperado dos seus analisadores léxico e sintático.

Adicionalmente, a existência, em tais gramáticas, de informações ligadas à forma de identificação de dependências contextuais, no texto de entrada, permite escolher a forma mais adequada para sua detecção em um particular texto de entrada, e subseqüente tratamento.

Para a ferramenta proposta, são identificados os seguintes repositórios principais de informação: o dicionário, a gramática ou autômato, o texto de entrada e a árvore de sintaxe.

Nesta proposta, embora tais elementos estejam apresentados conceitualmente de forma clássica, fazem extensivo uso da capacidade de auto-modificação característica dos dispositivos adaptativos, e se apresentam portanto com uma conotação dinâmica.

Discorre-se em seguida sobre os elementos em questão.

Dicionário:

No *dicionário* estão coletadas e organizadas todas as palavras da linguagem natural com as quais se pode construir sentenças. Muitos dicionários não colecionam todas as possíveis palavras, mas apenas as chamadas "formas de dicionário", excluindo assim um número muito grande de repetições devidas às inúmeras possíveis flexões que as palavras

podem sofrer ao serem empregadas em sentenças da língua, bem como os principais atributos (etiquetas) associados, para uso da análise léxica.

No dicionário figuram também as categorias gramaticais que a palavra costuma assumir nas sentenças que a empregam, bem como os principais atributos (etiquetas) associados, para uso da análise léxica.

Dessa maneira, o analisador léxico consulta o dicionário em busca de uma palavra, e uma vez localizada essa palavra, extrai as correspondentes informações sobre as categorias gramaticais a que pode pertencer, suas possíveis flexões em cada caso, bem como sobre suas exigências léxicas e sintáticas.

Gramáticas e autômatos:

Para descrever formalmente a linguagem, pode-se escolher um mecanismo formal gramatical ou então um dispositivo reconhecedor (autômato).

A teoria garante que tais formalismos têm o mesmo poder descritivo, e portanto são equivalentes, logo usar um ou outro é apenas uma questão de conveniência.

Naturalmente, o uso simultâneo das duas formas de descrição da linguagem não é conveniente por causa das redundâncias inevitáveis, além das dificuldades de manutenção que essa prática certamente acarreta, no caso de alguma alteração se mostrar necessária.

Caso se opte por uma gramática, podem-se dividir as regras gramaticais em dois níveis (em geral na prática não são apresentados em separado, mas formam um único conjunto de regras): um nível superior, que se ocupa das grandes estruturas da língua, e um outro nível, inferior, que descreve os detalhes das construções sintáticas permitidas em cada caso.

Caso seja escolhido um reconhecedor (autômato), considerando que a dificuldade de construir diretamente um autômato é bem maior que o de construir uma gramática, os lingüistas em geral preferem descrever linguagens usando formulações gramaticais, e portanto

na prática costuma-se partir de uma gramática, fornecida pelo lingüista, convertendo-a se necessário em autômato, usando para isso algum algoritmo disponível.

Se isso for verdade, torna-se possível automatizar esse processo, incluindo-se na ferramenta uma implementação de tal algoritmo, e fornecendo à ferramenta, em seu modo de **treinamento**, a gramática desejada, para ser convertida em autômato.

Posteriormente, um autômato, equivalente à gramática, construído e instalado pela própria ferramenta enquanto operava em modo **treinamento**, irá comandar, em modo **uso**, a operação dos procedimentos de análise do texto de entrada de acordo com o estabelecido na gramática originalmente fornecida à ferramenta.

Texto de entrada:

O processamento de textos em linguagem natural exige, obviamente, que lhe seja fornecido algum texto escrito em linguagem natural, para ser processado.

Esse texto deve ser apresentado à ferramenta pelo usuário final, sempre em modo **uso**, e o único módulo que dele extrai informações é o analisador léxico.

Tais informações são codificadas pelo analisador léxico para uso subseqüente na análise sintática.

O analisador sintático, por sua vez, pode efetuar outras transformações sobre esse conjunto de informações, tendo em vista a construção da árvore sintática da sentença em análise.

Na codificação feita pelo analisador léxico, as palavras são convertidas em categorias lexicais e estas, associadas a outros atributos extraídos da própria palavra e de informações complementares, obtidas nos dicionários.

Árvore de sintaxe:

Uma das metas mais importantes que se deseja alcançar quando se constrói um programa para o processamento automático de linguagem natural é a obtenção de uma árvore de sintaxe para cada texto de entrada fornecido.

À luz de uma gramática, pode-se visualizar a árvore sintática como uma árvore de derivação do texto de entrada, derivação essa feita com base na gramática.

Assim, a raiz da gramática será o nó principal da árvore de derivação, e as folhas dessa árvore corresponderão às palavras extraídas do texto de entrada.

Os nós intermediários correspondem a não-terminais da gramática, que tenham sido utilizados para a derivação da sentença.

Esses não-terminais intermediários são escolhidos sempre que uma regra gramatical for aplicada: o lado esquerdo da regra determina o não-terminal, e o seu lado direito, a substituição indicada pela regra.

Na árvore, o nó correspondente ao não-terminal presente no lado esquerdo da regra terá como nós-filhos os elementos contidos no lado direito da regra.

4.7 Representação gramatical da linguagem natural

Adota-se neste trabalho, para a representação gramatical da linguagem natural, uma notação similar à própria declaração de uma Função Adaptativa.

No caso particular aqui considerado, o problema a resolver consiste em determinar a estrutura sintática de um texto de entrada fornecido, e as premissas que devem ser levadas em conta nesta determinação têm como base os fatos contidos no dicionário, as palavras extraídas na forma de tokens pelo analisador léxico, e as maneiras legítimas de encadeamento, permitidas para esses elementos.

As regras que definem a linguagem prestam-se primordialmente a descrever as formas permitidas para a construção de sentenças válidas, de acordo com as estruturas que caracterizam a linguagem que interessa, em particular, ao usuário da ferramenta.

A raiz da gramática deve ser especificada da seguinte forma:

identificador.

Exemplo:

periodo.

As regras, que definem uma estrutura sintática devem ser especificadas da seguinte forma:

identificador_regra (lista opcional de parâmetros) =

{ lista_identificadores → { lista_regras }

O identificador_regra e o escopo onde a mesma é armazenada identificam juntos uma determinada regra.

Os parâmetros destinam-se a representar os atributos de uma determinada categoria, tais como gênero, número, grau, pessoa, papel temático, etc. Seus identificadores são precedidos pelo símbolo "&".

Os identificadores representam as categorias lexicais e gramaticais. Assim, quando se deseja representar uma estrutura sintática, sua especificação pode ser indicada como no exemplo seguinte.

periodo ( ) = { ip ponto_final }

periodo ( ) = { ip α_coord → { periodo ( ) } ponto_final }

Estas regras especificam um período simples e um período composto por coordenação, respectivamente.

Os parâmetros podem ser empregados quando se deseja especificar, por exemplo, atributos dos constituintes da oração ou mesmo dos itens lexicais. Seguem-se alguns exemplos:

dp (&gen, &num) = { det &gen &num → {np(&gen, &num)} }

Esta regra especifica o grupo determinante, com a presença obrigatória do determinante, o qual subcategoriza uma instância do grupo nominal, o qual por sua vez concorda em gênero e número com o determinante.

np(&gen, &num) = {adj &gen &num coord → {np(&gen, &num)}}

np(&gen, &num) = { adj &gen &num subst &gen &num}

np(&gen, &num) = {subst &gen &num}

Estas regras especificam o grupo nominal, com a presença de adjetivos que podem ocorrer de forma coordenada, ou um adjetivo justaposto a um substantivo, ou ainda um único substantivo.

Representação da componente léxica da linguagem.

A representação da componente léxica pode fazer uso da mesma sintaxe da metalinguagem empregada para a componente sintática da linguagem. Naturalmente depende apenas dos fenômenos lingüisticos que se deseja representar. O exemplo que se segue, apresenta uma forma simplificada de se representar um determinante. No capítulo seguinte apresenta-se a representação do fenômeno de subcategorização na representação da componente lexical

o ( ) = {det masc sing }

o ( ) = {pron pessoal acusativo masc sing }

o = {pron demonstrativo masc sing}

4.8 Primeiro detalhamento do nível de implementação

A "implementação" da ferramenta capaz de realizar as tarefas de análise léxica e sintática em linguagem natural é efetuada automaticamente pelo transdutor adaptativo em sua ***configuração inicial***, quando no sistema são inseridos o dicionário e o conjunto de regras gramaticais, ou seja, no modo treinamento.

Estas bases de dados são fornecidas através de um texto de entrada e são interpretadas pela ferramenta como um conjunto de identificadores (terminais e não-terminais da gramática da linguagem natural) e meta-símbolos. Tais identificadores e meta-símbolos são armazenados e associados a ações adaptativas adequadas para que possam ser executadas no modo uso da gramática.

A alternância entre o modo de treinamento e o modo uso se faz de forma muito simples no formalismo adaptativo. Ainda, faz parte do formalismo adaptativo, ações de inserção, remoção e consulta a regras. Dessa forma, é possível alternar o modo de operação treinamento e uso sempre que o Lingüista assim achar conveniente e ao mesmo será possível inserir ou remover as regras existentes.

Seguem-se detalhes de implementação para análise sintática e léxica

4.8.1 Implementação da análise sintática

Em geral, a análise sintática se baseia na composição da informação fornecida pelo analisador léxico, e proveniente do texto de entrada, com a informação contida nas regras que definem a gramática da linguagem natural.

Na presente proposta, a gramática original da linguagem, tal como inserida pelo Lingüista, é associada a um conjunto de ações adaptativas de forma que:

- no modo "uso" da gramática, a cada acesso a uma regra seja criada uma instância da mesma. Na arquitetura que se propõe, quando o analisador sintático tiver de acessar uma nova regra gramatical, esta não é meramente empregada para a verificação da estrutura sintática da cadeia de entrada. Na verdade, em cada acesso a uma regra gramatical, é criada uma nova instância da mesma, a qual efetivamente se prestará à execução da análise sintática. A criação de novas instâncias de regras gramaticais, já é uma tarefa que por si colabora no tratamento eficiente dos não-determinismos e ambigüidades sintáticas.
- seja criada uma lista de ponteiros que associa os lados esquerdo e direito da produção; Esta lista de ponteiros, orienta o processo de derivação da análise sintática e portanto representa um trecho da árvore; Considerando-se todas as regras fornecidas por um Lingüista, tem-se um grafo conectado representando todas as possibilidades disponíveis. No modo uso, a sentença de entrada selecionará trechos desse grafo, criando cópias do mesmo e construindo dessa forma a árvore.
- As ações adaptativas devem promover a comutação automática entre o processo de derivação e a leitura da cadeia de entrada. Isto significa que se tarefa de derivação é bem conduzida, inserirá um símbolo não-terminal na cadeia de entrada e por outro lado, a leitura do mesmo ativará o prosseguimento da derivação da sentença.
- ambigüidades e não-determinismos que se manifestam na cadeia de entrada sejam detectados e tratados convenientemente. Para isto, na leitura da cadeia de entrada, o primeiro símbolo presente na cadeia de entrada é detectado e interpretado como o delimitador de uma nova construção sintática. Na próxima ocorrência do mesmo, que é inicializada pelo mesmo símbolo. As ocorrências dos símbolos presentes entre a

ocorrência destas duas instâncias de símbolos, são copiados em uma esturutra auxiliar e reconduzidos ao analisador léxico (da configuração inical).

Na análise sintática, no modo uso, a ocorrência de um identificador na forma não marcada, implica que a regra por ele representada deverá promover a ampliação da gramática, seguida da leitura da cadeia de entrada. Ao final da leitura da cadeia de entrada, o símbolo na forma marcada é inserido na cadeia de entrada.

A alternância entre o acesso ao texto de entrada e a criação de uma instância de uma regra gramatical, configura uma operação que se faz ora de forma redutiva, ora de forma descendente.

As produções fornecidas pelo Lingüista no modo treinamento são convenientemente transformadas em Autômatos de Pilha Adaptativos. Mais precisamente, assumindo-se que o lado esquerdo da produção seja constituída de um só não-terminal correspondente ao nó pai em uma árvore, a este é associado uma lista de ponteiros correspondentes aos nós filhos. Assim, em tempo de uso da regra, esta lista de ponteiros poderá efetivamente orientar a construção da árvore.

4.8.1.1 OPERAÇÃO EM MODO "TREINAMENTO" :

Em geral, a análise sintática se baseia na composição da informação fornecida pelo analisador léxico, e proveniente do texto de entrada, com a informação contida nas regras que definem a gramática da linguagem natural.

4.8.1.2 OPERAÇÃO EM MODO "USO":

Tendo a ferramenta sido previamente treinada com as informações léxico-sintáticas da linguagem, o analisador sintático deverá estar pronto para analisar os textos de entrada que lhe forem submetidos, pois nesta ocasião o dicionário e a gramática já serão de seu conhecimento.

É aqui que todas as partes da ferramenta colaboram para a análise do texto de entrada, conforme esboçado a seguir.

Toda vez que for solicitado, o analisador léxico extrairá e classificará a próxima palavra do texto de entrada, disponibilizando o conjunto de todas as possíveis interpretações morfológicas para a palavra, acompanhadas dos atributos de flexão e das exigências morfo-sintáticas associadas a cada interpretação morfológica.

Toda vez que receber do analisador léxico informações sobre a próxima palavra do texto de entrada, o analisador sintático irá pesquisar, para cada uma de suas possíveis interpretações, a(s) possível(is) regra(s) gramatical(is) ou, correspondentemente, a(s) possível(is) transição(ões) do autômato reconhecedor à(s) qual(is) essa nova palavra se ajuste, respeitado o histórico de análise das palavras já analisadas pela ferramenta.

Toda vez que o analisador léxico ou o analisador sintático se deparar com mais de um caminho a seguir, que seja legítimo para a cadeia de entrada correntemente em análise, os mecanismos de tratamento de não-determinismos e ambigüidades deve ser ativado para que todos os casos possíveis sejam independentemente analisados.

Toda vez que o mecanismo de tratamento de não-determinismos e ambigüidades for ativado, deve receber do elemento que o ativou informação suficiente sobre as diversas opções a serem simultaneamente consideradas, de modo tal que seja promovida a instanciação de todas elas para o prosseguimento da análise: caso uma (ou mais) das instâncias consiga analisar integralmente o texto de entrada, a(s) árvore(s) sintática(s) correspondente(s) é(são) considerada(s) válida(s) para aquele texto de entrada. Se, em qualquer momento, alguma delas

não conseguir prosseguir na análise, então deverá ser desconsiderada, visto que se trata de um texto de entrada que não é aderente à gramática adotada.

Apresenta-se a seguir um método de análise para cada um dos elementos do software de processamento de linguagem natural.

4.8.2 Implementação da análise léxica

O Analisador Léxico, é implementado como uma parte do transdutor adaptativo, que é ativado sempre que o próximo símbolo na cadeia de entrada for uma palavra do dicionário, e não um símbolo não-terminal.

As palavras são buscadas e classificadas segundo na seqüência e conforme as regras fornecidas pelo Lingüista.

Em outras palavras, a especificação da arquitetura do etiquetador ou analisador morfológico. é determinada pelas regras fornecidas pelo especialista em Linguagem Natural.

Uma possibilidade para esse etiquetador é de que ele venha a fornecer o conjunto de todas as possíveis classificações para uma dada palavra, e em diversas situações, estabelecer, para cada caso, as possíveis exigências que a palavra impõe acerca de concordâncias ou de regências (sub-categorização). Exemplo

De qualquer maneira, a cada entrada pode-se obter um conjunto vazio, unitário ou múltiplo de saídas. Aqui o analisador léxico pode interagir com o mecanismo de tratamento de ambigüidades e não-determinismos, para que eventuais múltiplas interpretações possam ser tratadas

4.8.3 Comunicação entre a analise léxica e a sintática:

Determinado o conjunto possível de classes para uma palavra, o analisador léxico substitui a palavra recém analisada pela classe correspondente, acrescida das etiquetas correspondentes, deixando essa informação disponível para uso do analisador sintático.

Uma possivel opção é a de utilizar para isso a própria cadeia de entrada (que agora se transformou em uma cadeia de trabalho, ou de rascunho: considerando que duas ou mais diferentes interpretações léxicas de uma mesma palavra devem, cada uma de sua vez, provocar diferentes alterações sobre essa cadeia de entrada, e que portanto tais alterações devem ser mutuamente exclusivas, então na ocasião em que uma das opções estiver sendo adotada, as demais devem ser ignoradas, como se não existissem). Isto é possível graças aos mecanismos de delimitação de escopos indicadas no capítulo 2.

De qualquer modo, mais uma vez, para evitar complexidades desnecessárias no método empregado, é preciso escolher, com base em algum critério sólido, uma opção apenas de cada vez, e desconsiderar definitivamente todas as demais possibilidades.

4.8.4 Implementação dos mecanismos de dependências (sintáticas) de contexto:

Para que todas essas operações possam ser realizadas de forma interativa, e para que os mecanismos de análise possam ser executados adequadamente, modificando dinamicamente o formalismo (autômato ou gramática adaptativa) escolhido para a representação da linguagem, uma série de ações adaptativas devem existir que garantam a modificação apropriada do formalismo subjacente, de acordo com as necessidades determinadas pelos mecanismos de reconhecimento e análise adotados.

Assim, as diversas operações adaptativas que foram descritas nos capítulos anteriores preenchem exatamente essa função, e cada uma delas, na sua ocasião mais propícia, irá efetuar as alterações no formalismo subjacente de forma que as informações, de caráter sintático ou então referentes às dependências de contexto, que estiverem sendo tratadas possam ser devidamente incorporadas ao formalismo que implementa a análise da linguagem, na forma de ampliações ou de alterações da gramática ou do autômato correspondente.

4.9 Considerações Finais

Este capítulo apresentou diversas considerações complementares no que diz respeito a uma arquitetura adaptativa.

A proposta desta tese é que a sua implementação é aquela apresentada em (NETO, 1993), de forma que se possa auferir o desempenho diferenciado do tratamento de dependências de contexto advindos do formalismo adaptativo, conforme minuciosamente identificado nos capítulos anteriores.

Assim, na configuração incial do transdutor adaptativo, existe um analisador léxico, constituído de:

a) transdutores responsáveis pelo reconhecimento de palavras reservadas. Nesta tese se propõe que tais palavras reservadas sejam os atributos gramaticais tais como gênero (masculino, feminino, neutro), pessoa (1, 2, 3), papel temático, etc.
b) um transdutor léxico, onde figuram transições adaptativas que consomem símbolos isolados, tais como •, ⊥, ■ , &, ou ainda, transições que consomem um único símbolo ASCII com inserção do respectivo token.
c) um transdutor capaz de tratar identificadores de regras
d) um transdutor capaz de tratar identificadores de parâmetros

Ainda, o sistema deve apresentar um transdutor sintático capaz de verificar a sintaxe da metalinguagem utilizada para a representação da Linguagem Natural, memorizar e criar cópias das regras fornecidas pelo Lingüista. Uma pilha explícita pode ser utilizada para que seja possível fazer uma comutação entre este transdutor sintático e o transdutor léxico.

O uso desses transdutores aliados aos seus mecanismos adaptativos detalhados nos capítulos anteriores garantem o processamento das regras fornecidas pelo Lingüista. O transdutor se expande, conferindo a si mesmo uma camada capaz de processar a análise sintática de uma sentença fornecida pelo usuário do sistema, conforme descrito no início deste capítulo.

As técnicas adaptativas apresentadas, são capazes de identificar e decompor todos os não-determinismos e ambigüidades presentes em uma sentença apresentada pelo usuário, segundo a gramática fornecida pelo Lingüista, criando trechos de transdutores finitos determiníticos destinados à análise da sentença em questão. Aufere-se assim um desempenho linear por partes.

Deparou-se com o fato que cada mecanismo adaptativo subjacente ao tratamento destes não-determinismos e eventualmente ambigüidades apresenta isoladamente desempenho proporcional ao comprimento da fita de entrada, ocorrências de particulares símbolos na fita de entrada, o alfabeto e o número de regras da gramática da Linguagem Natural disponíveis no sistema.

Vislumbrou-se uma sintaxe da metalinguagem para a representação da Linguagem Natural. Considerações a respeito da mesma são apresentadas no próximo capítulo, bem como a conlusão deste trabalho.

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma proposta de arquitetura adaptativa empregando os mecanismos adaptativos apresentados em (NETO, 1993). A proposta da representação da Linguagem Natural foi imediata.

Foi possível verificar que problemas complexos em análise de Linguagem Natural, tais como os não-determinismos e ambigüidades presentes em situações de concordância, subcategorização, coordenação podem ser resolvidos com eficiência. De fato, todos os mecanismos adaptativos para solucionar estes problemas apresentam desempenho O(n).

Considere-se como exemplo, o seguinte estudo de caso:

Subcategorização

As entradas lexicais em um Léxico devem conter a minimamente uma informação que diz respeito à sua categoria e uma informação relativa ao quadro de subcategorização.

Subcategorização diz respeito ao fato lingüístico que os itens lexicais podem selecionar complementos que co-ocorrem obrigatória ou facultativamente dentro de uma categoria gramatical. Assim para o grupo verbal VP, o verbo seleciona. os seus complementos, completando o seu significado. Todas as categorias lexicais podem subcategorizar complementos.

Tal como num sistema de lógica de predicados, as expressões lingüísticas podem ser analisadas num predicador central e num determinado número de argumentos que lhe completam o sentido, convertendo o predicador numa expressão semanticamente completa.

Além do quadro de subcategorização, que especifica a categoria gramatical dos complementos categorizados, cada entrada lexical apresenta sua estrutura argumental que associa a informação relativa à seleção semântica, com a informação relativa à subcategorização.

Os seguintes exemplos ilustram as estruturas argumentais de três verbos:

entrada lexical	classe lexical	argumento externo (sujeito)	argumento interno (subcategorizado)
entregar	V	Agente	<__DP [$_{PP}$ a] > Tema Alvo
dormir	V	Tema	<__ >
assustar	V	Tema	<__DP > Experienciador

A representação e o processamento do fenômeno de subcategorização e seleção semântica enunciado devem levar em conta os seguintes fatores:

- o caráter obrigatório ou opcional dos elementos subcategorizados pela entrada lexical;
- a subcategorização propriamente dita: ou seja a representação de que uma vez que ocorra um determinado item lexical, seus complementos devem ocorrer na sentença de entrada.

A representação da estrutura argumental dos verbos pode ser feita conforme se segue:

a. verbo com dois complementos obrigatórios:

{entrada_lexical (&token, &classe_gramatical, &tema_sujeito, &complemento1, &tema1, &complemento2, &prep, &tema2) = {&token &classe_gramatical, &tema_sujeito →(&complemento1(&gen, &num, &tema1)) →(&complemento2(&gen, &num, &prep, &tema2))}}

b. verbo com um complemento:

{entrada_lexical (&token, &classe_gramatical, &tema_sujeito, &complemento1, &tema1) = {&token &classe_gramatical &tema_sujeito →(&complemento1(&gen, &num, &tema1)) }}

c. verbo sem complementos:

entrada_lexical (&token, &classe_gramatical, &tema_sujeito) = {&token &classe_gramatical &tema_sujeito }

Uma regra como a que se segue poderia ainda ser criada:

subcat_sel (entregar) = { ∇subcat_sel

{entregar (&token, &classe_gramatical, &tema_sujeito, &complemento1, &tema1, &complemento2, &tema2) = **{**&token &classe_gramatical, &tema_sujeito →(&complemento1(&gen, &num, &tema1)) →(&complemento2(&gen, &num, &tema2))**}**

}

{&entregar (&token, &classe_gramatical, &tema_sujeito, &complemento1, &tema1) = **{**&token &classe_gramatical &tema_sujeito →(&complemento1(&gen, &num, &tema1)) }**}**

}

Considere-se também a seguinte regra:

VP (&entrada_lexical, &verbo) = { &entrada_lexical → { subcat_sel(&entrada_lexical }

O processamento desta regra em um sistema dotado dos mecanismos adaptativos apresentados ao longo desta pesquisa, permitem que na presença do verbo entregar na sentença de entrada, no modo uso, toda a informação a respeito de sua subcategorização e seleção semântica esteja disponível para o prosseguimento da análise. Se por um lado, inicialmente existem duas árvores possíveis e a sentença fornecida pelo usuário estiver correta, apenas uma delas será efetivamente selecionada pela mesma sentença..

Como se viu estas tarefas são executadas automaticamente, sempre deterministicamente e com desempenho linear, ainda que na presença do não-determinismo inicial presente na gramática, associado às duas possibilidades de subcategorização e seleção semântica do verbo entregar.

Ainda, pode-se verificar que facilmente se representa a concordância do papel temático do complemento com aquele selecionado pelo verbo.

Ainda, constata-se que na presença do verbo entregar, a análise sintática foi efetuada de forma ascendente.

A regra associada ao grupo verbal VP poderia ser alterada, de forma que pudesse receber os atributos do verbo e fazer a concordância com um sujeito ainda que estivesse à esquerda da sentença de entrada.

O estudo exaustivo, por um lado, dos mecanismos apresentados não permitiram que se elencasse a representação e, **automaticamente** o processamento de demais dependências sintáticas, tal como inicialmente se propôs nesta pesquisa. Trata-se no entanto, de tarefa que pode ser realizada a curto prazo.

Por conseguinte, a obtenção de uma ferramenta tal como aquela descrita em (NETO, 1993) e aqui analisada, pode ter como aplicação imediata não somente a obtenção de compiladores, mas também de processadores de Linguagem Natural, tal como aqui analisado.

O paradigma adaptativo, ainda que não exclua a utilização de métodos estatísticos, é por sua natureza a expressão de que, uma vez que se possa representar o conhecimento humano através de uma linguagem dependente de contexto, seu processamento é capaz de processar todos os não-determinismos e elencar todas as ambigüidades.

REFERÊNCIAS

AHO, A.V.; ULMANN, J. D. **The Theory of Parsing, Translation, and Compiling**, Vol. 1. New Jersey: Prentice-Hall, Englewood, 1972. 542p.

CHIANG, D. **Evaluating Grammar Formalisms for Applications to Natural Language Processing and Biological Sequence Analysis.** 2004., Faculties of the University of Pennsylvania,. Dissertação.

CHOMSKY, N**. Aspects of the Theory of Syntax**. Cambridge: MIT Press, 1965. 251p.

CHOMSKY, N. **Lectures on Government and Binding** Dordrecht: Foris, 1986 371p..

CHOMSKY, N. A Minimalist Programming for Linguistic Theory. In HALE, K.. KEYSER, S. J. . **The View from Building 20**. Cambidge: MIT Press, 1993, p

CHRISTOPHER Manning Assistant Professor on Computer Science and Linguistics Natural Language processing group, Stanford University. Disponível em: <http://nlp.stanford.edu/~manning/>. Acesso em: 27 ago. 2005.

DOUGHERTY, R. C. **Natural Language Computing***:* An English Generative Grammar in Prolog, 1994. 400p.

FRANK, R.; K. VIJAY-SHANKER. *Primitive C-Command*. Syntax 4:164-204., 2001.

FRANK, R. **Phrase Structure Composition and Syntactic Dependencies**. Massachussets: MIT Press, 2002. 340p.

FRANK, R., D. MATHIS; W. BADECKER [on-line]. *The Acquisition of Anaphora by Simple Recurrent Networks*. Manuscript., 2004. Disponível em <http://www.cog.jhu.edu/faculty/rfrank/#3>

GAZDAR, G. , MELLISH C. **Natural Language Processing in LISP**, Addison-Wesley, 1989. 524 p.

IWAI, M. K.. **Um formalismo gramatical adaptativo para linguagens dependentes de contexto**. 2000. 191p. Tese – Escola Politécnica, Universidade de São Paulo, São Paulo, 2000.

JOSHI, A. K. How much context-sensitivity is required to provide reasonable structurall descriptions: Tree Adjoining Grammars. In DOWTY D., KARTTUNEN L., ZWICKY A. **Natural Language Parsing: Psychological, Computational and Theoretical Perspectives.** Cambridge: Cambridge Univerity Press, 1985.

JURAFSKY, D. e MARTIN, J. H. **Speech and Language Processing:** An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recogniton. New Jersey: Prentice Hall, 2000. 934p.

KARTTUNEN, L. **Applications of Finite-State Transducers in Natural Language Processing.** In Implementation and Application of Automata, Yu, S. and Paun, A. (eds.). Lecture Notes in Computer Science Volume 2088, pages 34-46, Springer Verlag, Heidelberg, 2001.

LABORAT [on-line] Laboratório de Tecnologias Adaptativas. Escola Politécnica. Universidade de São Paulo: Neto, J. J. [2005-08-28]. Available from Internet <http://www.pcs.usp.br/~lta/>

LEXICON OF LINGUISTICS [on-line]. Utrecht Institute of Linguistics OTS. Utrecht University: Kerstens J.; Ruys, E. Zwarts, J., 1996-2001 [2005-08-02]. Avaiable from Internet <http://www2.let.uu.nl/UiL-OTS/Lexicon/>.

LINGUATECA [on-line].[2005-08-02] Available from Internet <http://www.linguateca.pt/>.

MATTHEWS, C. *An Introduction to Natural Language Processing through Prolog* London and New York: Longman, 1998 306p. ISBN 0-582-066220

MENEZES, C. E. D.: *Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos* São Paulo: Escola Politécnica; Universidade de São Paulo; São Paulo, 2000. Dissertação de Mestrado.

MENEZES, C. E. D., NETO, J. J.: *Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos*, Proc. of V PROPOR, Encontro para o Processamento Computacional de Português Escrito e Falado, 19 a 22 de novembro de 2000, Atibaia, (2000)

NETO, J.J. Introdução à compilação. Editora LTC, Rio de Janeiro, 1987

NETO, J.J *Contribuições à metodologia de construção de compiladores.* São Paulo: Escola Politécnica; Universidade de São Paulo, 1993. 306p. Tese de Livre-Docência.

NETO, J. J. *Adaptive Automata for Context-Sensitive Languages*. SIGPLAN NOTICES, Vol. 29, n. 9, pp. 115-124, 1994

NETO, J. J. *Solving complex problems with Adaptive Automata*. In Lecture Notes in Computer Science. S. Yu, A. Paun (Eds.): Implementation and Application of Automata 5th International Conference, CIAA 2000, Vol.2088, London, Canada, Springer-Verlag, 2000, pp.340.

NETO, J.J *Adaptive rule-driven devices - general formulation and case study* - CIAA'2001 Sixth International Conference on Implementation and Application of Automata. Lecture Notes in Computer Science, Springer-Verlag, Pretoria (2001)

NETO, J.J. *Autômatos em Engenharia de Computação - uma Visão Unificada*. Primera Semana de Ciencia y Tecnología de la Sociedad Chotana de Ciencias y la Red Mundial de Científicos Peruanos Ciudad de Chota, Perú, Junio 22-27, 2003.

NETO, J.J e MORAES, M.: *Formalismo adaptativo aplicado ao reconhecimento de linguagem natural* - Anais da Conferencia Iberoamericana en Sistemas, Cibernética e Informática, 19-21 de Julio, 2002, Orlando, Florida (2002)

NETO, J.J e MORAES, M. *Using Adaptive Formalisms to Describe Context-Dependencies in Natural Language*. Lecture Notes in Artificial Intelligence. N.J. Mamede, J. Baptista, I. Trancoso, M. das Graças, V. Nunes (Eds.): Computational Processing of the Portuguese Language 6th International Workshop, PROPOR 2003, Volume 2721, Faro, Portugal, June 26-27, Springer-Verlag, 2003, pp 94-97

NETO, J.J., PARIENTE, C. B. and Leonardi, F. *Compiler Construction - a Pedagogical Approach*. Proceedings of the V International Congress on Informatics Engineering ICIE 99, Buenos Aires, Argentina, 1999.

PENN-HELSINKI Parsed Corpus of Middle English. University of Pennsylvania [on-line] [2005-08-27]. Available from internet http://www.ling.upenn.edu/hist-corpora/PPCME2-RELEASE-2/.

PENN-HELSINKI PARSED Corpus of Early Modern English Kroch, A., Santorini B., Delfs L. 2004 University of Pennsylvania [on-line] [2005-08-27]. Available from internet <http://www.ling.upenn.edu/hist-corpora/PPCEME-RELEASE-1>

PEREIRA, JOEL CAMARGO DIAS PEREIRA. *Ambiente integrado de desenvolvimento de reconhecedores sintáticos, baseado em autômatos adaptativos*. São Paulo: Escola Politécnica; Universidade de São Paulo, 1999.

PEREIRA, J. C. D.; NETO, J. J. *Um Ambiente de Desenvolvimento de Reconhecedores Sintáticos Baseado em Autômatos Adaptativos.* II Simpósio Brasileiro de Linguagens de Programação - SBLP97. pp. 139-150, Campinas, 1997

RAPOSO, E. P. *Teoria da Gramática*: A Faculdade da Linguagem. 2. ed. Lisboa: Caminho, 1998. 527p. ISBN 972-21-0713-5

TANIWAKI, C. Y. O.: *Formalismos Adaptativos na Análise Sintática de Linguagem Natural* - MSc Dissertation, Escola Politécnica da Universidade de São Paulo (2001)