MARGARETE KEIKO IWAI

UM FORMALISMO GRAMATICAL ADAPTATIVO PARA LINGUAGENS DEPENDENTES DE CONTEXTO

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

São Paulo
2000

MARGARETE KEIKO IWAI

UM FORMALISMO GRAMATICAL ADAPTATIVO PARA LINGUAGENS DEPENDENTES DE CONTEXTO

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
2000

Aos meus pais, pela inesgotável paciência e compreensão

AGRADECIMENTOS

Ao meu amigo, professor e orientador Prof. Dr. João José Neto pelo amplo e efetivo acompanhamento, pelos conselhos, incentivos e ensinamentos e pela grande amizade.

Aos meus amigos Laís do Nascimento Salvador, Aurélio Akira Mello Matsui, Izaura Cristina Araújo, Paulo Eduardo Santos, e às minhas irmãs Olga e Márcia pelo apoio que me deram em todos os momentos.

Às funcionárias do departamento Fátima, Célia, Cristina e Tânia pelo auxílio na obtenção da infra-estrutura para a realização deste trabalho.

Ao CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico pelos recursos fornecidos.

A todos que direta ou indiretamente colaboraram para que este trabalho se tornasse possível.

RESUMO

Este trabalho procurou dar uma contribuição à área das linguagens formais e autômatos, com a apresentação do desenvolvimento de um formalismo gramatical adaptativo para linguagens dependentes de contexto, denominado Gramáticas Adaptativas.

Este formalismo possui como característica principal a capacidade de se alterar a medida que é feita a geração da sentença pertencente à linguagem que é representada pela gramática adaptativa.

Esta tese procurou fazer uma compilação de alguns trabalhos referentes às gramáticas adaptáveis, bem como de alguns formalismos correlatos dinâmicos que são utilizados na representação de linguagens, como por exemplo autômatos.

O presente trabalho faz um estudo da equivalência da gramática adaptativa com o seu formalismo dual, conhecido como Autômatos Adaptativos. São apresentados, também, alguns algoritmos que permitem o mapeamento de um formalismo para o outro e vice-versa.

ABSTRACT

The present work is a contribution to the field of automata and formal languages. It describes the development of a grammatical adaptive formalism for describing context-sensitive languages.

Our formalism, named Adaptive Grammar, shows the important property of self-modification, in the sense that the set of rules of an adaptive grammar changes in response to the application of existing rules to derive the sentences of the language.

In this thesis, we have made a compilation of the some related works on adaptable grammars, as well as some others dynamic formalisms used to represent languages.

This work also presents a study on the equivalence between adaptive grammars and their dual formalism, represented by Adaptive Automata. Some mapping algorithms are also shown, which allow us to convert each formalism into its dual one.

Iwai, Margarete Keiko
Um formalismo gramatical adaptativo para linguagens dependentes de contexto. São Paulo, 2000.
191 p.
Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Linguagens formais 2. Gramáticas
3.Autômatos I. Universidade de São Paulo,
Escola Politécnica, Departamento de Engenharia de Computação e Sistemas Digitais II. t

SUMARIO

Capítulo 1

1. Introdução

Uma das formas de comunicação entre os seres humanos é a que utiliza as linguagens escrita e falada. Através deste veículo, é possível expressar as mais diversas informações, tais como idéias, acontecimentos, fatos, entre outros, com conteúdos das mais variadas complexidades.

Com o surgimento dos computadores digitais, foi preciso estudar métodos formais que permitissem a comunicação entre estas máquinas e os seres humanos através de utilização da linguagem escrita.

Normalmente, as informações podem ser facilmente transmitidas e entendidas quando o transmissor e o receptor de uma mensagem utilizam uma linguagem comum. A perfeita compreensão de uma informação pode ser prejudicada quando se utiliza uma linguagem que não é perfeitamente compreendida por uma, ou por ambas as partes.

Este problema tornou-se bastante evidente com o surgimento dos computadores digitais. Neste momento, foi preciso estudar métodos formais e rigorosos que permitissem facilitar a comunicação entre a máquna e o ser humano, como por exemplo, uma forma de representação da linguagem, e métodos que permitissem a geração e o entendimento das sentenças escritas nesta linguagem.

Em meados na década de 50, o lingüista Noam Chomsky propôs uma forma de representação de linguagens através de um mecanismo abstrato e finito que gerava as sentenças pertencentes a uma linguagem, as chamadas gramáticas. Chomsky classificou essas gramáticas em quatro tipos: gramáticas regulares (tipo 3), gramáticas livres-de-contexto (tipo 2), gramáticas sensíveis ao contexto (tipo 1) e as gramáticas irrestritas (tipo 0). As linguagens geradas por esta forma de representação são denominadas respectivamente: regulares (tipo 3), livres-de-contexto (tipo 2), sensíveis ao contexto (tipo 1) e os conjuntos recursivamente enumeráveis (tipo 0). Este último inclui as linguagens naturais.

Uma outra forma encontrada para representar linguagens é através de um mecanismo que realiza a identificação de cadeias de símbolos como sendo sentenças de uma dada linguagem. Este mecanismo utiliza máquinas de estados finitos, ou autômatos,

que representam a linguagem e permitem decidir se uma sentença pertence ou não a esta linguagem. Uma tradicional classificação destes autômatos é feita de acordo com o tipo de linguagem que ele reconhece e segue a ordem definida no parágrafo anterior: autômato finito, autômato a pilha, máquina de Turing com fita limitada e máquina de Turing sem limite de fita. As linguagens que despertam maior interesse no presente trabalho são as de tipo 0 e 1, que representam a maioria das linguagens de programação.

Essas formas de representação de linguagens permitiram o desenvolvimento dos meios de comuniçação entre o homem e o computador através da criação das linguagens de programação, e também o desenvolvimento dos compiladores e interpretadores, que são programas que traduzem textos escritos em uma linguagem, compreensível para os seres humanos, para outra linguagem, compreensível para a máquina.

Em geral, o mecanismo de tradução de uma linguagem para a outra é constituída de uma série de etapas que executam tarefas separadas, como por exemplo, o reconhecimento dos componentes básicos (análise léxica), a verificação da sintaxe da sentença (análise sintática), e também a validade do significado da informação que está sendo transmitida (análise semântica).

À medida que a linguagem vai se tornando mais complexa, a dificuldade em desenvolver métodos que permitam a sua representação e manipulação também cresce.

Uma das maiores complexidades encontradas nesta área de pesquisa diz respeito à especificação dos aspectos sensíveis ao contexto, características fundamentais das linguagens de tipos 0 e 1.

Existem alguns formalismos clássicos que procuram expressar os aspectos sensíveis ao contexto de uma linguagem de programação, como por exemplo as gramáticas de atributos [Alb91], [Knu68], [Knu71] e as gramáticas de dois níveis [Wij75], [Pag81].

Outros formalismos existentes, que também tratam dos aspectos sensíveis ao contexto de uma linguagem de programação, são os autômatos adaptativos [Jos93] e [Jos94], os autômatos finitos auto-modificáveis (*Self-Modifying Finite Automata*) [Rub93], [Rub95a], [Rub95b] e [Shu95], e as gramáticas do tipo adaptável [Chr90] e [Shu93].

O presente trabalho representa uma continuidade à linha de pesquisa iniciada em adaptativos [Jos93], em que são apresentados os autômatos adaptativos, formalismo cognitivo utilizado para o tratamento de linguagens sensíveis ao contexto. Este modelo

constitue uma formalização de reconhecedores de topologia dinâmica para tais linguagens, permitindo que a usualmente chamada semântica estática da linguagem de programação, identificada em construções tais como estruturas de blocos, escopo de variáveis, tratamento de macros, verificação da consistência do uso dos tipos das variáveis, etc, seja propriamente expressa como parte integrante da sintaxe.

A apresentação das motivações para o desenvolvimento deste trabalho, bem com os objetivos que conduzem a realização deste projeto serão apresentados a seguir.

1.1. Motivação e objetivos

Na teoria formal de linguagens, pode-se notar uma escassez de formalismos que representem de um modo prático os aspectos dependentes de contexto, de forma puramente sintática.

Os formalismos existentes, ou são muito complexos, como por exemplo, as gramáticas de dois níveis, ou utilizam recursos secundários, externos ao formalismo, que auxiliam no tratamento de dependências de contexto. Como exemplo, tem-se as gramáticas de atributos, que representam um formalismo bastante difundido e utilizado, mas que perde clareza ao representar dependências muito complexas, sobrecarregando, assim, as suas regras semânticas.

Os autômatos adaptativos são um formalismo bastante prático e com um grande potencial de aplicação em diversas áreas de pesquisa, como por exemplo em compiladores, engenharia de *software*, construção de ferramentas, meta-programação, inteligência artificial, etc. No entanto, como formalismo cognitivo ele foi projetado mais propriamente para auxiliar na construção da implementação de linguagens, e não tanto na sua fase de projeto e concepção estrutural.

Esta tese tem por motivação dar continuidade ao desenvolvimento dos fundamentos da tecnologia adaptativa, propondo como dispositivo generativo, uma gramática adaptativa, do tipo sensível ao contexto, aderente ao formalismo implementado pelos autômatos adaptativos e mais própria para as etapas de projeto e estruturação de linguagens.

Como finalidade adicional, esta notação deverá servir como metalinguagem para ferramentas automáticas de auxílio à implementação das linguagens através do uso de autômatos adaptativos.

Esta tese tem o objetivo de avaliar o poder e as limitações computacionais deste modelo gramatical e estabelecer sua relação com modelos formais consagrados, tais como autômatos finitos, autômatos a pilha e a máquina de Turing.

O presente trabalho propõe, portanto, um formalismo dual ao dos autômatos adaptativos, visando a facilitar o desenvolvimento de linguagens complexas ou outras aplicações que necessitem especificar linguagens dependentes de contexto na forma de gramáticas.

Para tanto, é apresentada uma notação para este formalismo gramatical, que sirvirá como metalinguagem de entrada para a definição de linguagens em uma ferramenta que transforma gramática para autômato e vice-versa.

Apresenta ainda um formalismo gramatical que seja facilmente mapeado para os autômatos adaptativos, deste modo, desenvolve-se um método alternativo de representação de linguagens sensíveis ao contexto

São apresentados, neste trabalho, alguns teoremas que provam a equivalência entre os autômatos adaptativos e as gramáticas adaptativas.

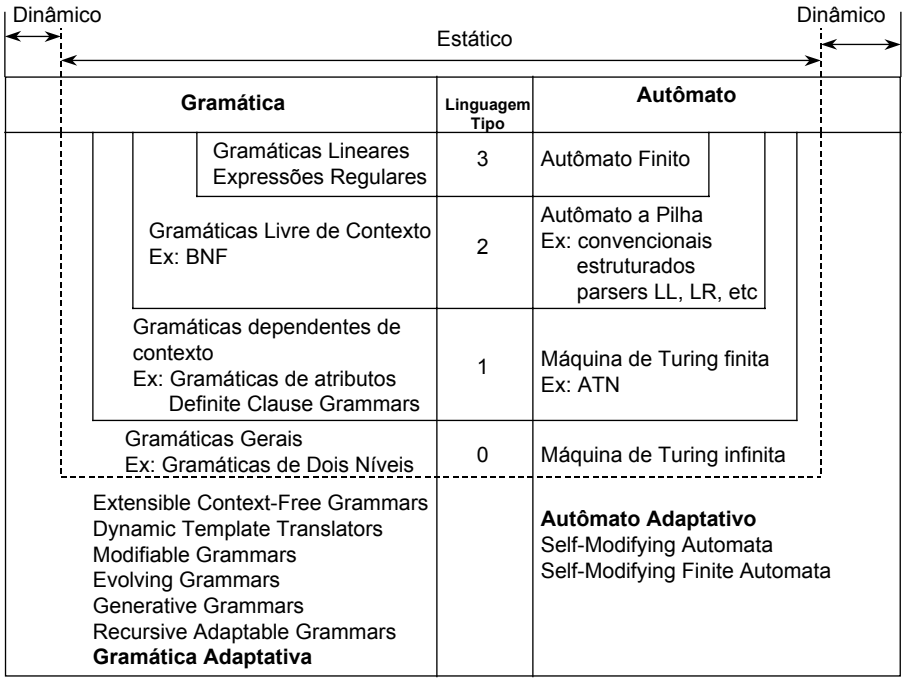
A experiência adquirida no desenvolvimento deste formalismo permitiu ainda estabelecer alguns métodos importantes para a utilização desta notação para resolver problemas importantes da área.

1.2. Revisão bibliográfica

Esta seção apresenta uma visão geral da área das linguagens formais e autômatos, através de uma compilação de publicações importantes. A elaboração deste texto foi baseada em dois importantes trabalhos de pesquisa, de autoria de Henning Christiansen [Chr90] e John Shutt [Shu93].

Antes de iniciarmos a apresentação da literatura, convém apresentar uma pequena ilustração representativa do panorama geral da área, posicionando as referências citadas no contexto das linguagens formais.

Neste quadro dividem-se e classificam-se alguns dos diversos formalismos existentes em conjuntos de gramáticas, do lado esquerdo e de autômatos, do lado direito, discriminados pelo tipo de linguagem da qual eles fazem parte, segundo a hierarquia de Chomsky.



No conjunto mais restrito, tem-se as linguagens de tipo 3, que são as linguagens regulares, representadas pelas gramáticas lineares, expressões regulares e pelos autômatos finitos. Por se referirem a formalismos bastante conhecidos e difundidos, não serão tratados aqui com maiores detalhes, podendo ser facilmente encontrados na literatura, por exemplo, nas seguintes referências [Hop69], [Lew81], entre muitasoutras.

As linguagens livres de contexto, ou do tipo 2, estão aqui representadas pelas gramáticas livres de contexto, em notações tais como, o BNF (Backus-Naur Form), e os autômatos de pilha convencionais, os autômatos de pilha estruturados, as técnicas de análise sintática LL, LR, etc.

No grupo das linguagens dependentes de contexto, também denominadas sensíveis ao contexto, podemos citar as gramáticas de atributos e as *definite clause grammars*, que representam as gramáticas, e a máquina de Turing com fita limitada, representando os formalismos de autômato

Por fim, temos alguns representantes das linguagens do tipo 0, ou conjuntos recursivamente enumeráveis. Podemos citar neste grupo as gramáticas de dois níveis e o Máquina de Turing com fita ilimitada.

Todos os formalismos apresentados até este ponto são do tipo estático, isto é, eles não se alteram durante o processo de execução da geração (no caso de gramáticas) ou do reconhecimento (no caso de autômatos) de sentenças da linguagem que representam.

O conjunto mais externo do quadro apresenta alguns formalismos que são do tipo dinâmico e que são utilizadas para representar linguagens do tipo 0.

Apesar da não uniformidade da terminologia nesta literatura, pode-se dizer que estes formalismos são do tipo adaptável, isto é, eles possuem uma estrutura dinâmica que se altera ao longo do seu processo de utilização. O foco destas alterações está centrado nos conjuntos de regras e de transições das gramáticas e dos autômatos respectivamente.

Para que um formalismo seja adaptável, as principais estruturas que serão alteradas serão justamente os conjuntos de regras de produção e o conjunto de transições. Estas alterações envolvem basicamente a adição ou remoção de regras ou de transições, nas gramáticas ou nos autômatos, respectivamente.

No restante deste capítulo, apresentamos um levantamento dos trabalhos mais importantes em cada um desses assuntos, que desempenham um papel significativo para a área ou para esta tese.

1.2.1. Gramáticas adaptáveis

Em função do modo como manipulam os seus conjuntos de regras de produção, as gramáticas adaptáveis podem ser classificadas como sendo imperativas ou declarativas [Shu93].

Gramáticas adaptáveis imperativas

Gramáticas adaptáveis imperativas atuam sobre o conjunto de suas regras de produção de forma explícita, isto é, de maneira imperativa, alterando algoritmicamente o conjunto de regras durante a geração das sentenças da linguagem que definem. Como conseqüência, nas gramáticas desta classe podem ser identificadas diversas sucessivas configurações da gramática original, que se alteram durante o processo de derivação das sentenças.

Em [For63] surgiram publicadas as primeiras idéias referentes ao princípio da adaptabilidade das gramáticas. Embora tal publicação não o tenha demonstrado formalmente, menciona que as declarações são os mecanismos que possibilitam a

extensibilidade da linguagem, e que também são as ferramentas naturais para a criação de novas e explícitas regras de sintaxe em uma gramática.

Posteriormente, a linguagem Algol68 mostrou ser uma linguagem de programação que permitia extensões, tais como recursos para definir novos operadores e novos tipos de dados, e que inspirou muitas linguagens que surgiram posteriormente, deixando como herança, entre tantos outros, alguns conceitos de extensibilidade, tais como definição das estruturas de dados pelo próprio usuário e também a definição e utilização de macros.

Assim, foram surgindo algumas linguagens de programação extensíveis. Um delas foi a EL1 [Weg80]. Esta linguagem fez parte de um sistema de programação que apresentou uma classe de gramáticas livres de contexto do tipo extensível associado a um transdutor de estados finitos que traduz o programa em paralelo ao processamento de sua análise sintática, eventualmente executando comandos para remover ou adicionar regras à gramática.

Um outro modelo que seguia a filosofia das gramáticas de dois níveis com uma terminologia específica foi desenvolvida em [Mas87], que apresentou uma classe de gramáticas adaptáveis denominadas *Dynamic Template Translator*, ou DTT, um tradutor dirigido por sintaxe que modificava o conjunto de suas próprias regras de produção. Segundo [Chr90], este modelo pode ser visto como uma generalização do trabalho de Wegbreit. Enquanto o sistema EL1 utilizava um transdutor, o DDT anexava instruções de modificação ao conjunto de regras de produção, que eram executadas quando da aplicação da regra durante a derivação.

Um outro trabalho, que utilizava uma gramática livre de contexto e um transdutor, que no caso era uma máquina Turing-compatível, está apresentada na trilogia [Bur90a], [Bur90b] e [Bur92], e é comentado em [Rob91] que faz algumas observações acerca deste desenvolvimento.

Neste trabalho, desenvolveu-se um formalismo denominado gramática modificável, que gerou a meta-linguagem USSA (*Universal Syntax and Semantics Analyser*), uma linguagem de descrição para linguagens de programação, que é uma extensão das gramáticas de atributos.

Segundo [Chr90], o analisador sintático desenvolvido em tal pesquisa permite adicionar novas regras à gramática ou remover regras do seu conjunto de produções, enquanto é feito o reconhecimento de uma sentença da linguagem pelo transdutor.

Por último, em um trabalho paralelo, um analisador sintático dinâmico e uma gramática evolutiva são apresentados em [Cab92]. Trata-se de uma técnica que apresenta uma série de gramáticas geradas a partir de uma mesma gramática inicial. Esta gramáticas vão sendo substituídas à medida que a análise sintática do programa vai evoluindo, através da alteração do conjunto das regras de produção originais, com a adição de novas regras de produção.

Gramática adaptáveis declarativas

No modelo das gramáticas adaptáveis classificadas como declarativas, a manipulação do conjunto de regras de produção é feita de modo declarativo, não impondo seqüência ao mecanismo de geração de sentenças da linguagem, nem utilizando um estado interno global, como é feito no caso imperativo, mas permitindo uma atuação mais livre sobre o conjunto de regras de produção [Shu93].

Em alguns modelos declarativos, esta atuação pode ser feita sobre uma estrutura de dados, como por exemplo, nos nós da árvore de derivação, e não diretamente no processo de derivação, como ocorre no modelo imperativo.

As principais publicações sobre gramáticas adaptáveis são [Hanf73], [Chri85] e [Shut93]. Os dois últimos modelos são baseados em gramáticas de atributos e em lógica.

Em [Hanf73] é utilizado um modelo matemático baseado em cálculo λ no desenvolvimento de meta-linguagens para mostrar a estrutura dinâmica da sintaxe de uma linguagem de programação.

Os modelos de gramáticas adaptáveis que apresentaram formalizações mais completas foram os de Christiansen e Shutt.

[Chr85], entre uma série de artigos e relatórios técnicos, introduziu um formalismo para uma gramática adaptável denominada *generative grammar*, que foi baseada nos conceitos das gramáticas livre de contexto associada à semântica denotacional, sendo apresentada como uma simples generalização das gramáticas de atributos. Esta notação também é conhecida como Gramática de Christiansen [Shu93].

Em [Chr86a] e [Chr86b] foi apresentada uma introdução informal das gramáticas e linguagens denominadas generativas, uma classe que abrange as linguagens de programação extensíveis. Este artigo também descreve técnicas do tipo *top-down* para o reconhecimento das linguagens definidas por tais gramáticas.

[Chr88a] apresentou uma grande evolução deste modelo, com a formalização da notação baseada em gramáticas de atributos. Além disso, são feitas as primeiras considerações deste modelo gramatical com a linguagem de programação Prolog, originando uma nova reformulação da Gramática de Christiansen baseada na notação DCG (*definite clause grammar*). Em [Chr88b] é mostrada toda a evolução e desenvolvimento do trabalho deste autor.

Posteriormente, em [Chr90] é apresentado um bom *survey* desta área relacionada a gramáticas adaptáveis.

Finalmente, em [Shu93] é apresentado o desenvolvimento de uma gramática adaptável recursiva, cuja formalização está baseado em *one-sorted algebra* [Gog79], também conhecida na literatura como álgebra universal. O trabalho do Shutt procurou também seguir os conceitos da Gramática de Christiansen, além de apresentar também um ótimo *survey* de algumas das gramáticas existentes na área, classificando-as e tecendo comentários sobre cada modelo.

1.2.2. Autômatos de topologia dinâmica

Apresentam-se nesta seção alguns tipos de autômatos cuja estrutura varia à medida que vai sendo executado algum tipo de processamento com ele.

As publicações referentes aos modelos de autômatos de topologia dinâmica são muito mais escassas que a das gramáticas correspondentes. Nesta seção estão relacionados os poucos tipos de autômatos de topologia dinâmica encontrados na literatura.

Os autômatos em questão são estruturas conceitualmente adaptáveis. Foram encontrados dois tipos de modelos que seguem tam princípio, e que são os autômatos adaptativos, que é a base deste trabalho, e os *self modifying finite automata* (SMFA).

Os SMFA foram desenvolvidos como uma continuação de uma pesquisa prévia do seu autor sobre gramáticas adaptáveis recursivas [Shu93]. Trata-se de autômatos finitos que modificam seu conjunto de transições de forma dinâmica durante a computação, tornam-se por esta razão capaz de reconhecer linguagens dependentes de contexto. Este trabalho foi

desenvolvido em [Rub93], [Rub95a], [Rub95b] e [Shu95], que são relatórios técnicos que apresentam as diferentes fases do desenvolvimento do trabalho.

Em paralelo ao desenvolvimento do SMFA, surgiram os autômatos adaptativos, introduzidos em [Jos93], [Jos94], que constitui um formalismo capaz de reconhecer linguagens dependentes de contexto. Este modelo tem seu funcionamento baseado nos autômatos a pilha estruturados, uma variante dos autômatos de prilha tradicionais formada de um conjunto de máquinas de estados finitos mutuamente recursivas [Jos87]. Este formalismo é representado por uma ferramenta denominada RSW [Per97], [Per99],que é capaz de simular mecanismos baseados em autômatos adaptativos.

Tanto os SFMA como os Autômatos Adaptativos são equivalentes à máquina de Turing e suas topologias podem ser modificadas à medida que é feito o reconhecimento de uma cadeia de entrada. Ambos os modelos constituem formas equivalentes de autômatos de topologia dinâmica.

1.2.3.Outros formalismos correlatos

Alguns formalismos, embora não sejam adaptáveis, possuem um núcleo livre de contexto acrescido por alguns recursos que permitem o tratamento da dependência de contexto.

Podemos citar nesta categoria as gramáticas de atributos, as gramáticas de dois níveis e as linguagens extensíveis.

A gramática de atributos foi apresentada inicialmente por Donald Knuth em [Knut68] e [Knut71], como um formalismo capaz de descrever os aspectos dependentes de contexto de uma linguagem de programação. Tornou-se uma notação muito popular e bastante utilizada, podendo ser aplicada em diversas implementações, como, por exemplo, editores, interpretadores, compiladores e em sistemas de construção de compiladores, ou *compiler-compilers*, entre outros.

As gramáticas de dois níveis, também conhecidas como gramáticas W, foram desenvolvidas por van Wijngaarden em meados da década de 60, como uma notação metalingüistica para a definição da linguagem Algol 68 [Wijn75].

O conceito de linguagem extensível foi uma tendência que surgiu nas décadas de 60 e 70, e que pode ser definida como sendo um método que permite aos usuários da

linguagem de programação extensível definir novas características para tal linguagem, como por exemplo novas notações, novas estruturas de dados, novas operações, e possivelmente novas estruturas de controle, entre outros [Sta75]. Exemplos históricos deste tipo de linguagem são o Algol 68 e o SIMULA (*SIMUlation LAnguage*), que surgiu em 1967.

Voltando aos formalismos adaptáveis, podemos citar um outro modelo de sistema dinâmico, baseado no formalismo dos statecharts [Har87 apud Alm95] que foi recentemente desenvolvido e apresentado em [Alm95] e se denomina statechart adaptativo. Este sistema é capaz de modificar sua configuração à medida que processa as entradas impostas a ele.

Em continuidade a este trabalho, [San97] apresenta um formalismo para especificação de sistemas reativos complexos, sincronizados entre si, baseado numa composição de Rede de Petri, Statechart convencional e Statechart Adaptativo.

Um trabalho mais recente está em curso envolvendo um formalismo adaptativo baseado não em gramáticas genrativas ou em autômatos, mas em redes de Markov, que são autômatos estocásticos. As redes de Markov adaptativas estão sendo utilizadas com sucesso para a modelagem do processo de composição musical [Bas99].

Na área da engenharia de software, estão surgindo alguns métodos evolutivos Podemos citar nesta categoria um método de desenvolvimento de software adaptativo orientado a objetos, chamado Método de Demeter [Lie96]. Este método consiste de um dicionário de classes para definir a estrutura dos objetos e define uma propagação de padrões para implementar o comportamento dos mesmos.

Na área da inteligência artificial, os autômatos adaptativos podem ser utilizados como modelo formal em aplicações de inferência gramatical [Jos98]. Este trabalho apresenta um pequeno exemplo de dispositivo capaz de aprender a sintaxe de uma linguagem regular.

Além das áreas voltadas para a teoria da computação e da engenharia de software, tendências do uso de técnicas similares têm surgindo desde meados da década de 90 também na área de hardware, como se pode observar pelo surgimento de eventos nacionais e internacionais relacionados ao assunto, [Cor00], [Fpl00], cujas áreas de interesse abrangem desde aplicações em geral, até sistemas dinamicamente reconfiguráveis. Em

particular, [Fpl00] é uma conferência internacional cujo tema central são sistemas reconfiguráveis e hardware evolutivo.

1.3. Estrutura da tese

Este capítulo foi dedicado a uma apresentação global do que irá compor esta tese e procurou dar uma visão geral da literatura existente na área e aos tópicos relacionados.

No próximo capítulo, serão apresentados os conceitos teóricos básicos de alguns tipos de representação de linguagens, mais precisamente, as gramáticas e os autômatos do tipo convencional, bem como os do tipo adaptável que foram citados nesta revisão bibliográfica.

No capítulo 3 será apresentado o desenvolvimento e a formalização da gramática adaptativa e a apresentação dos teoremas que demonstram a equivalência entre a gramática adaptativa e o autômato adaptativo.

O capítulo 4 apresentará, em linhas gerais, os algoritmos que permitem o mapeamento da gramática adaptativa com o autômato adaptativo e vice-versa.

E por fim, no capítulo 5, está registrado um resumo do resultado desta pesquisa, as suas contribuições e alguns possíveis trabalhos futuros.

Capítulo 2 - Conceitos

Este capítulo tem como objetivo apresentar, de maneira geral, alguns assuntos teóricos básicos relacionados ao estudo formal das linguagens de programação.

A teoria, discutida ao longo deste capítulo, concentra-se basicamente em dois assuntos: gramáticas e reconhecedores. Estes dois assuntos estão diretamente relacionados à proposta deste trabalho. Serão apresentados alguns métodos convencionais de representação de linguagens, bem como os formalismos adaptativos, que possibilitam uma forma alternativa para a especificação das mesmas.

2.1 Linguagens

Com o advento do computador digital, muitos estudos foram feitos no sentido de desenvolver modelos formais que representassem a comunicação entre o homem e a máquina. Essas pesquisas levaram ao desenvolvimento de diversas técnicas, entre elas, a criação das linguagens de programação.

Em se tratando de computadores digitais, em que se exige precisão na interpretação dos comandos dados pelo usuário, é preciso que sejam evitadas as imprecisões naturais que acontecem na comunicação entre os seres humanos, e que se exteriorizam como o mau entendimento das frases, as ambigüidades, entre outros. Portanto, é necessário que seja feita uma formalização rigorosa deste modelo computacional de comunicação, o que se obtém através do tratamento abstrato e matemático dado às técnicas de especificação das linguagens de programação.

Assim, utilizando métodos formais bem definidos, é possível especificar conceitos, definições, propriedades, entre outros, de modo que a manipulação dessas informações seja feita através da utilização de operações bem estabelecidas no modelo.

O intuito deste trabalho não é o de esgotar os conceitos referentes à teoria das linguagens formais, os quais são facilmente encontráveis na literatura, tais como, por exemplo, [Hop69], [Aho86], [Har78] e [Sal69]. Serão apresentadas, a seguir, somente algumas definições, baseadas em [Hop69], que têm maior ligação com os assuntos desenvolvidos neste trabalho.

Um *alfabeto* ou *vocabulário* é qualquer conjunto finito de símbolos e será denotado por V.

Uma *sentença* de uma linguagem é uma cadeia de símbolos pertencentes ao seu alfabeto, V, e que tem comprimento finito. Por convenção, uma sentença é normalmente denotada pelas letras minúsculas do alfabeto grego e o comprimento de uma sentença qualquer, α, será representada por $|\alpha|$.

Uma *linguagem* é um conjunto de sentenças escritas sobre um alfabeto.

Um dos interesses da teoria das linguagens é estudar as formas de representação das linguagens, as quais, se possível, devem ser computacionalmente finitas, muito embora a maioria das linguagens de interesse possua um número infinito de sentenças.

Existem basicamente três formas de representação de linguagens: *enumeração*, *geração* e *reconhecimento* das sentenças da linguagem.

A enumeração é feita através da apresentação de uma lista de todas as cadeias de símbolos que formam a linguagem. Neste caso, esse método permite representar apenas linguagens finitas.

O processo de geração das sentenças é feita através de um conjunto de *leis de formação* das cadeias, denominado gramática.

O reconhecimento das sentenças de uma linguagem é descrito através de um conjunto de *regras de aceitação* de cadeias, que especifica um dispositivo *reconhecedor* (usualmente, *máquinas de estados* finitos ou *autômatos*). Essas formas de representação de linguagens utilizam um procedimento ou um algoritmo o qual procura determinar se uma cadeia pertence ou não à linguagem. Um *procedimento* é uma seqüência finita de *instruções* que podem ser executadas mecanicamente, sendo que esta seqüência pode terminar ou não. Um algoritmo é um procedimento que sempre termina.

Segundo [Hop69], se houver um procedimento que gere as sentenças de uma linguagem, então pode-se construir um procedimento que reconheça as sentenças desta linguagem, não sendo necessariamente um algoritmo. Para determinar se uma sentença x pertence a uma linguagem L, enumeram-se todas as sentenças de L e compara-se x com cada sentença. Se x foi gerada, o procedimento termina, tendo reconhecido x como pertencente a L. Se x não pertence a L, então o procedimento nunca terminará.

Desse modo, pode-se dizer que existem dois tipos de linguagens, as chamadas linguagens *recursivas*, que são reconhecidas por algoritmos, e as *recursivamente enumeráveis*, que são reconhecidas por procedimentos.

A seguir serão apresentados, de forma mais rigorosa, dois tipos de representação de linguagens, as gramáticas e os reconhecedores.

2.2. Gramáticas convencionais

Esta seção tem por objetivo apresentar, em linhas gerais, um formalismo gerador das sentenças de uma linguagem, as *gramáticas*. Foram utilizadas, na composição deste texto, [Hop69], [DeR76] e [Lew81].

Nas definições usuais, uma gramática G é definida como uma quádrupla (V_N, V_T, *P*, *S*), onde:

$V = V_N \cup V_T$ é o *vocabulário* da gramática G

V_N é um conjunto finito de símbolos *não-terminais* (ou *variáveis*)

V_T é um conjunto finito de símbolos *terminais*, tal que $V_N \cap V_T = \varnothing$

S é o *símbolo inicial* da gramática, $S \in V_N$

P é um conjunto finito de *regras de produção*, em que cada regra é escrita na forma

$$\alpha \rightarrow \beta \text{ , onde } \alpha \in V^*V_N V^* \text{ e } \beta \in V^*$$

V^* denota o conjunto de todas as possíveis cadeias compostas por símbolos do vocabulário V da gramática, incluindo a cadeia vazia, denotada por ε.

V^+ denota o conjunto de todas as possíveis cadeias não vazias compostas por símbolos do vocabulário V.

Uma gramática conforme estabelecido acima também é conhecida como sendo um *sistema de re-escrita irrestrita* (*unrestricted rewriting system*), pois as sentenças da linguagem são geradas a partir de um símbolo inicial S e sucessivamente substituídas ou re-escritas de acordo com o conjunto de *regras de produção* ou de *regras de re-escrita* [DeR76].

Se $\alpha \rightarrow \beta$ é uma produção de *P* e $\mu, \nu \in V^*$ são cadeias de símbolos do vocabulário da gramática, então $\mu\alpha\nu \Rightarrow \mu\beta\nu$ é dita uma *derivação imediata* da gramática.

A relação de derivação imediata informa que a produção $\alpha \rightarrow \beta$ foi aplicada à cadeia $\mu\alpha\nu$ para obter $\mu\beta\nu$.

Uma *derivação* corresponde à obtenção de uma seqüência de cadeias α_0 , α_1 ,….., $\alpha_n \in V^*$, $n \geq 0$ tal que

$\alpha_0 \Rightarrow \alpha_1$, $\alpha_1 \Rightarrow \alpha_2$, ….., $\alpha_{n-1} \Rightarrow \alpha_n$, com $n \geq 0$, ou seja, através da aplicação sucessiva das regras de produção da gramática.

Notação: $\alpha_0 \Rightarrow^* \alpha_n$; ou se $n \geq 1$, então $\alpha_0 \Rightarrow^+ \alpha_n$

Se $S \Rightarrow^* \alpha$, isto é, se uma cadeia é derivada do símbolo inicial *S*, então α é deta uma *forma sentencial*. Se α é formada apenas por símbolos terminais, então ela é chamada *sentença*.

Uma *linguagem gerada por uma gramática* é denotada por L(G) e é constituída por um conjunto de todas as sentenças derivadas do símbolo inicial *S*, ou seja, pelo conjunto $L(G) = \{ \omega \in V_T^* \mid S \Rightarrow^* \omega \}$

Existe um outro método visual para descrever o processo de derivação de uma sentença em uma gramática $G= (V_N, V_T, P\ S)$, que é uma *árvore de derivação*. Ela é composta por um conjunto finito de *nós* e *arestas* orientadas.

Todo nó da árvore tem um *rótulo*, que é um símbolo de $V= V_N \cup V_T$. Se um nó m possui uma aresta que parte dele e chega a um nó n com $n \neq m$, então dizemos que n é *descendente direto* de m.

Se uma aresta liga dois nós, n_1 e n_2, da árvore e se o essa aresta se orienta no sentido do nó n_1 para o nó n_2, dizemos que a aresta *parte* do nó n_1 e *chega* no nó n_2.

O nó situado mais ao topo da árvore é a raiz da árvore e deverá estar rotulado com o não-terminal correspondente à *raiz da gramática*.

Se existir um caminho n_1, n_2,..... n_k de nós conectados por arestas então n_k diz-se descendente de n_1. Os nós que não possuem descendentes diretos são chamados folhas. Estes nós são símbolos terminais. Se o nó folha for formado por um não-terminal, dizemos que a árvore está incompleta e a árvore não deriva nenhuma sentença da gramática G.

Os nós que possuem descendentes diretos além deles mesmos são símbolos não-terminais da gramática. Cada nó é descendente direto dele mesmo.

As sentenças são formadas pela concatenação das folhas da árvore, começando da esquerda e indo para a direita, formando o contorno da árvore.

Existe uma classificação dos tipos de gramáticas definida por Chomsky, que corresponde à obtenção de classes mais simples de linguagens pela imposição de restrições progressivas à forma das regras de produção.

A gramática definida acima é denominada como sendo do *tipo 0* ou *irrestrita*. As linguagens por elas representadas são conhecidas como *conjuntos recursivamente enumeráveis*. As linguagens geradas por esta classe de gramáticas são as mais difíceis de serem representadas, formalizadas e implementadas. As linguagens naturais fazem parte desta classe de linguagens.

As regras de produção de P são da forma $\alpha \rightarrow \beta$, onde $|\beta| \geq |\alpha|$, isto é, o comprimento da cadeia α não pode ultrapassar o comprimento de β.

Estas regras de produção também podem ser escritas na forma $\gamma A \delta \rightarrow \gamma \omega \delta$, onde $A \in V_N$ e $\gamma\omega\delta \in V^+$, sendo que ω não é a cadeia vazia ε

Uma gramática livre de contexto $G = (V_N, V_T, P, S)$ ou do tipo 2 é aquela em que as regras de produção de P são da forma $\alpha \rightarrow \beta$, onde $\alpha \in V_T$ e $\beta \in V^*$.

Uma gramática regular $G = (V_N, V_T, P, S)$ ou do tipo 3 é aquela em que as regras de produção de P é da forma $A \rightarrow aB$ ou $A \rightarrow a$, onde A e B são símbolos não terminais e a é um símbolo terminal.

Gramáticas regulares e livres de contexto geram as linguagens regulares e livres de contexto, respectivamente. Maiores detalhes da teoria envolvendo estas gramáticas e suas linguagens podem ser encontradas em diversos livros da área, tais como [Hop69] e [Lew81]

As linguagens geradas por estas gramáticas são chamadas linguagens sensíveis ao contexto e nesta categoria de enquadra a maioria das linguagens de programação de interesse.

A seguir serão descritas com mais detalhes as gramáticas sensíveis ao contexto, e que despertam maior interesse para os assuntos tratados neste trabalho.

Gramáticas sensíveis ao contexto

Serão consideradas nesta seção as gramáticas sensíveis ao contexto, ou do tipo 1, e sua aplicação ao estudo das linguagens de programação.

Pode-se estudar as linguagens de programação segundo três aspectos, léxico, sintático e semântico.

O aspecto léxico trata da decomposição dos textos em seus componentes mais elementares, e da classificação de tais cadeias de caracteres, isto é, das palavras da linguagem.

A *sintaxe* de uma linguagem é descrita pelas regras da gramática, e a *semântica* corresponde ao significado que as construções geradas por estas regras possam ter, dentro do contexto em que estão inseridas.

Geralmente, a descrição dos aspectos sintáticos livres de contexto de uma linguagem de programação costuma ser feita simplesmente através de gramáticas livres de contexto. Esta especificação é realizada usualmente através de meta-linguagens tais como a notação BNF (Backus-Naur Form), a notação de Wirth e os diagramas de sintaxe [José87].

Estas notações procuram formalizar as regras de produção da gramática, sem levar em consideração os aspectos sensíveis ao contexto das linguagens descritas, os quais costumam ser descritas informalmente ou como parte da descrição da semântica.

A descrição da semântica é uma tarefa mais complexa, pois não envolve somente a análise de construções lingüísticas isoladas, mas o significado da sentença como um todo, considerados todos os aspectos de sua interpretação.

Algumas técnicas, que serão citadas adiante, procuram especificar, através de uma gramática-base livre de contexto, a sintaxe de uma linguagem, e incluem como extensões os demais aspectos dependentes de contexto.

Estas técnicas são descritas detalhadamente em [Shut93], que propõe uma gramática adaptável recursiva, incluindo um bom *survey* sobre as gramáticas convencionais e as do tipo adaptável, além de formalizar as gramáticas adaptáveis recursivas e apresentar uma boa coleção de referências bibliográficas.

2.3 Autômatos convencionais

Esta seção discute um outro tipo de técnica para a representação de linguagens. Trata-se dos reconhecedores, em especial, alguns tipos de máquinas de estados finitos, isto é, os tradicionais autômatos.

Descrições e definições relativas a reconhecedores, máquinas de estados, redes de transição, e outros conceitos afins, podem ser encontrados na literatura, em publicações como [Jos87], [Gri86], [Pet81], entre muitos outros. Neste trabalho será apresentado apenas um resumo de alguns tópicos das referências mencionadas, que se mostram relevantes ao assunto da tese.

Um reconhecedor é composto basicamente por:

um texto de entrada, representado por uma cadeia de símbolos

um cursor, que faz a leitura dos símbolos do texto de entrada

uma máquina de estados finita, que manipula o cursor e seus movimentos e também o consumo ou não dos símbolos do texto de entrada.

A máquina de controle de estados também manipula a utilização da memória auxiliar e as mudanças de estados da máquina.

e eventualmente uma memória auxiliar, podendo ser uma pilha ou outra estrutura, que armazena informações colhidas durante o reconhecimento.

A *configuração* de um reconhecedor caracteriza-se por três elementos, o estado corrente, a posição do cursor no texto de entrada, e o conteúdo da memória auxiliar.

Uma *configuração inicial* é determinada por um *estado inicial* bem definido, o *cursor* no início do texto a ser analisado, e a *memória* previamente preenchida com conteúdo que indica uma situação inicial (usualmente vazia).

Uma *configuração final* é determinada por um *estado final*, pertencente a um conjunto de estados finais, o texto completamente analisado e a memória com conteúdo que indique uma situação final pré estabelecida (usualmente vazia).

O *reconhecimento* bem sucedido de uma sentença é definido se o reconhecedor partir de uma configuração inicial e terminar em uma configuração final.

Os reconhecedores podem ser classificados em: *determinísticos*, em que, para qualquer movimento no processo de reconhecimento de uma cadeia, existe apenas uma configuração possível a ser atingida, e *não-determinísticos*, em que, para cada

configuração, pode existir um conjunto finito de movimentos possíveis da máquina de estados finita.

Uma máquina de estados finita pode ser representada por um *diagrama de estados* finito, representado por um grafo orientado, composto por um conjunto finito de estados, conectados por arcos orientados. Cada arco representa uma transição entre dois estados, e é rotulado por símbolos, que variam de acordo com a aplicação.

Um estado do grafo é escolhido como sendo o estado inicial; e um ou mais estados são marcados como sendo os seus estados finais.

Para cada linguagem, classificada de acordo com a hierarquia de Chomsky, existe um tipo adequado de autômato que a reconhece. Discute-se a seguir cada uma dessas alternativas.

Autômatos Finitos

Um *autômato finito* é um dispositivo sem memória, que reconhece linguagens regulares, ou do tipo 3.

Um autômato finito M é uma quíntupla (Q, Σ, P, q_0, F) onde

Q é um conjunto finito e não vazio de *estados*

Σ é um conjunto finito e não vazio de *símbolos de entrada*, e é denominado o *alfabeto de entrada* do autômato.

P é uma *função de transição de estados* do autômato, é definida como

$$P: Q \times \Sigma \rightarrow Q$$

Esta função mapeia o estado corrente e o símbolo corrente de entrada em um próximo estado.

q_0 é o estado inicial do autômato

F é um subconjunto não vazio de Q, constituído por todos os estados finais do autômato.

O *reconhecimento* de uma cadeia de entrada, ω, pelo autômato M é feito sempre a partir do único estado inicial q_0. O consumo esta cadeia por M é feita símbolo a símbolo através da aplicação das produções de P, até o seu total esgotamento. O reconhecimento é

bem sucedido se esta situação for atingida e se o estado atingido pertencer a *F*, caso contrário, ω não pertence à linguagem.

Transdutores

Um formalismo bastante citado neste trabalho diz respeito ao conceito de transdutores. Neste ponto apresentamos uma idéia geral deste conceito.

Transdutor é um dispositivo estruturalmente similar a um autômato convencional, mas que, ao invés de simplesmente reconhecer cadeias de entrada, transforma tais cadeias, de acordo com regras estabelecidas, nas cadeias de saída correspondentes. Assim, a partir do seu estado inicial, o transdutor muda de estado para estado de acordo com a cadeia de entrada, e, em cada passo, emite opcionalmente uma saída, na forma de uma seqüência de símbolos de saída, de acordo com uma regra associada à transição executada.

O diagrama de estado é naturalmente idêntico ao do autômato subjacente exceto que os rótulos das transições contêm, além da indicação do símbolo de entrada, também informações da saída correspondente, por exemplo, a | ω. [Lew81]

Autômatos de Pilha

Um tipo de autômato muito utilizado, que reconhece linguagens livres de contexto, são os autômatos de pilha estruturados [Jos93] que são reconhecedores com uma memória organizada em forma de pilha, e constituídos de uma coleção de submáquinas que operam como autômatos finitos e podem ativar-se mutuamente como se fossem submáquinas mutuamente recursivas.

Um autômato de pilha estruturado M é definido como

M = (Q, A, Σ, Γ, P, Z_0, q_0, F), onde:

Q é um conjunto finito de estados

A é um conjunto de submáquinas do tipo a_i = (Q_i , Σ_i, P_i, q_{0i}, Fi) para i = 1…n, onde

$Q_i \subseteq Q$ é o subconjunto de estados q_{ij} da submáquina a_i

$\Sigma_i \subseteq \Sigma$ é o subconjunto de símbolos de entrada da submáquina a_i

$P_i \subseteq P$ é o subconjunto de transições da submáquina a_i

$q_{0i} \in Q_i$ é o estado de entrada da submáquina a_i

$F_i \subseteq Q_i$ é o subconjunto dos estados de saída da submáquina a_i

Σ é um conjunto finito de símbolos teminais

Γ é um conjunto finito não vazio de símbolos da pilha

Z_0 é o símbolo da pilha vazia, $Z_0 \in \Gamma$

q_0 é o estado inicial, $q_0 \in Q$

F é um subconjunto não vazio de Q, constituído dos estados finais da submáquina cujo estado inicial é q_0

P é um conjunto finito de transições do tipo:

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

onde:

γ - parte irrelevante do conteúdo da pilha

g - topo da pilha (que é desempilhado pela transição)

e - estado corrente

s - símbolo consumido pela aplicação da transição

α - parte da entrada irrelevante a esta transição

g' - novo topo da pilha (empilhada pela execução da transição)

e' - novo estado

s' - símbolo inserido na cadeia de entrada pela transição

Cada submáquina $a_i \in A$ é representada por uma quíntupla ai = (Q_i, Σ_i, P_i, $q_{i,0}$, F_i), onde:

$Q_i \subseteq Q$ é o conjunto de estados $q_{i,j}$ da submáquina a_i

$\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada de a_i

$P_i, \subseteq P$ é o conjunto de regras de produção $p_{i,j}$ de a_i

$q_{i,0} \in Q_i$ é o estado inicial de a_i

$F_i \subseteq Q_i$ é o conjunto de estados finais de a_i

O reconhecimento de uma cadeia ω por M é feita partindo de $q_0 \in Q_0$. Essa cadeia vai sendo consumida pelas transições, que podem usar a pilha para promover chamadas/retornos de submáquinas, até que ω se esgote com a pilha vazia em estado final.

Máquina de Turing

Por fim, temos as máquinas de Turing que são reconhecedores em que uma máquina de estados utiliza como memória auxiliar uma fita, com possibilidade de leitura e escrita e movimentação do cursor de leitura/gravação nos dois sentidos desta fita.

Um modelo básico de máquina de Turing consiste em uma máquina de controle finito, uma fita de entrada que é dividida em células, um cursor que se movimenta pela fita uma célula por vez e nas duas direções (direita e esquerda). Este cursor pode escrever ou remover símbolos nesta fita, de acordo com os comandos dados pela máquina de controle.

Quando a fita de entrada tem tamanho limitado, a máquina de Turing reconhece linguagens do tipo sensível ao contexto, caso contrário, a linguagem reconhecida é do tipo recursivamente enumerável.

Definição (Máquina de Turing) [Lew81]

Uma Máquina de Turing M é uma quádrupla $(K, \sum, \delta, s)$, onde

K é um conjunto finito de *estados*, que não contém o estado de parada h (*halt*);

$\sum$ é um *alfabeto*, onde $\# \in \sum$, mas L e $R \notin \sum$

$s \in K$ é o *símbolo inicial*

δ é a função de $K \times \sum$ para $(K \cup \{h\}) \times (\sum \cup \{L, R\})$.

Se $q \in K$, $a \in \sum$, e $\delta(q, a) = (p, b)$, então a máquina M, quando estiver no estado q, lendo o símbolo a, irá para o estado p, e

se b é um símbolo pertencente a $\sum$, reescreverá o símbolo a como b, ou

se b é L ou R, moverá o seu cabeçote para a esquerda ou direita, respectivamente.

Como δ é uma função, uma operação de M é determinística e irá parar somente quando M estiver no estado de parada (*halt*) ou tentar mover-se para a esquerda, fora do limite esquerdo da fita.

Definição: Uma configuração de uma máquina de Turing $M = (K, \sum, \delta, s)$ é um elemento pertencente ao produto cartesiano:

$$(K \cup \{h\}) \times \sum^* \times \sum \times (\sum^* (\sum - \{\#\}) \cup \{\varepsilon\}).$$

Cada elemento desta configuração corresponde a seguinte seqüência: estado corrente, cadeia à esquerda do símbolo corrente, símbolo correntemente apontado pelo cursor e restante da cadeia, à direita do símbolo corrente.

Uma configuração com estado corrente *halt* indica que a máquina *M* está em sua configuração final.

Não serão apresentados maiores detalhes a respeito deste formalismo, podendo o leitor encontrar muitas referências sobre o assunto em textos clássicos, tais como [Hopc69], [Lewi81], etc.

2.4 Formalismos gramaticais não-convencionais

Nesta seção serão comentados alguns formalismos utilizados para representar os aspectos dependentes de contexto das linguagens.

Gramáticas de dois níveis

O formalismo conhecido como *gramática de dois níveis* ou *gramáticas W* é composto de dois conjuntos de regras. O primeiro é constituído de um número finito de elementos, as meta-regras, que formam uma gramática livre de contexto, e de meta-variáveis. Este conjunto pode ser interpretado como uma meta-gramática finita, a qual gera um segundo conjunto de regras, que pode ser infinito, e que é responsável pela geração propriamente dita da linguagem que se quer definir.

Apesar da simplicidade conceitual deste formalismo, ele não se mostrou tão popular quanto as *gramáticas de atributos*, que serão comentadas a seguir. Talvez, isto se deve à sua notação mais complexa e conseqüentemente de legibilidade e aprendizado mais difícil.

Segundo [Shu93], uma vantagem das gramáticas W é que este formalismo não faz uso, para a representação das dependências de contexto, de quaisquer dispositivos adicionais, como, por exemplo, as funções semânticas usadas pelas gramáticas de atributos, nem de vários mecanismos usados em gramáticas adaptáveis do tipo imperativo.

As gramáticas de dois níveis apresentaram algumas idéias que foram utilizadas na teoria das linguagens extensíveis, tal como, a utilização de uma meta-gramática que permite ao usuário a criação de sua própria gramática.

Textos acerca deste assunto podem ser encontrados em algumas referências clássicas, como por exemplo, [Pag81].

Gramáticas de atributos

Um outro formalismo utilizado para representar aspectos dependentes de contexto são as *gramáticas de atributos* que são, na realidade, extensões das gramáticas livres de contexto, no sentido de que a informação associada às construções da linguagem que se está definindo é incorporada através da vinculação de atributos aos símbolos da gramática que definem tais construções. Os valores dos atributos são determinadas através da execução de regras de avaliação de atributos, associadas às produções da gramática livre de contexto subjacente.

Os atributos associados aos símbolos da gramática podem ser divididos em duas classes disjuntas, os *atributos herdados* e os *atributos sintetizados*. As regras de avaliação de atributos, associadas a uma dada produção da gramática, definem os *atributos sintetizados* ligados ao símbolo da gramática que consta no lado esquerdo da produção, enquanto que os *atributos herdados* estão ligados aos símbolos que figuram no seu lado direito.

Um bom texto acerca deste formalismo e de suas aplicações se encontra em [Paa95], que apresenta alguns paradgimas de programação, bem como algumas linguagens de especificação baseadas em gramáticas de atributos.

Outras referências sobre o assunto, podem ser encontradas em [Der88], [Alb91] que pertencem à coleção Lecture Notes in Computer Science, e apresentam uma série de referências sobre linguagens, ferramentas, bem como artigos que tratam de assuntos correlatos, e também em [Pag81] apresenta uma utilização prática deste formalismo através da especificação de exemplos de mini-linguagens.

Definite clause grammars

Um outro formalismo, também utilizado para representar dependências de contexto, é conhecido como *definite clause grammar*, ou DCG, foi desenvolvido em

[Per80], a partir do conceito de *metamorphosis grammar* [Col78], no qual as as gramáticas são expressas como cláusulas de lógica de predicados de primeira ordem.

Segundo [Per80], DCG é uma extensão natural das gramáticas livres de contexto, semelhante às gramáticas de atributos, e permite a especificação de linguagens do tipo dependente de contexto.

A notação DCG foi fortemente influenciada pela linguagem Prolog, a qual serve como ambiente para a análise e execução de especificações escritas em DCG. Maiores detalhes podem ser encontrados em [Per80].

Gramáticas adaptáveis

A revisão bibliográfica apresentada no capítulo 1 procurou dar uma visão geral de algumas gramáticas do tipo adaptável, bem como de algumas linguagens extensíveis. Nesta seção serão reapresentados, a título ilustrativo, alguns destes formalismos.

Extensible Context-Free Grammar

O primeiro formalismo que apresentamos foi desenvolvido por Wegbreit [Wegb80] e consiste em uma gramática livre de contexto associado a um transdutor de estados finitos. Essa gramática é denominada ECFG (*Extensible Context-Free Grammar*).

O transdutor recebe um texto de entrada para ser analisado e sua saída é interpretada com um série de instruções para a modificação do conjunto de regras da gramática. Sua saída pode não alterar a gramática, ou indicar uma instrução para adicionar regras ao conjunto de regras da gramática, ou ainda, remover regras da gramática.

Em [Wegb80] está provado que a classe das linguagens geradas por ECFG é um superconjunto próprio das linguagens livre de contexto, e também, é um subconjunto próprio dos conjuntos recursivamente enumeráveis.

Dynamic Template Translator

Em seguida, temos o formalismo conhecido com *Dynamic Template Translator* que foi desenvolvido por Mason [Maso87] e é uma generalização do esquema de tradução livre-de-contexto.

A aplicação de um regra de tradução dispara uma seqüência de ações, as quais agem como efeitos colaterais sobre o esquema corrente de tradução. Estas ações podem adicionar ou remover regras de tradução ou ainda modificar sua seqüência de ações. A natureza imperativa deste mecanismo de extensão acarreta a necessidade de impor uma ordem fixa de execução das ações para as regras.

Em [Mas87] está provado que este formalismo tem o poder das máquinas de Turing, isto é, para toda máquina de Turing *M*, existe uma gramática *G Dynamic Template Translator* tal que L(*M*) = L(*G*) e vice-versa.

Modifiable Grammar

Um outro formalismo adaptável é conhecido como *modifiable grammar* [Burs92]. Consiste de uma gramática livre de contexto associada a um transdutor equivalente à máquina de Turing. De forma semelhante aos formalismos apresentados anteriormente, ao receber uma derivação parcial de uma sentença como entrada, o transdutor apresenta como saída uma lista de regras que devem ser adicionadas ao conjunto corrente de regras e uma outra lista de regras que devem ser removidas do mesmo.

Por ser também um tipo de gramática adaptável imperativa, este modelo orienta as decisões do transdutor, e para permitir isto, Burshteyn apresenta em seu artigo dois modelos gramaticais, o *Top-Down Modifiable Grammar* e o *Bottom-Up Modifiable Grammar*.

Em [Burs90b] é provado que esta gramática também tem a potência das máquinas de Turing.

Evolving Grammar

Além destes trabalhos temos um outro formalismo gramatical denominado *Evolving Grammar* [Caba92] que é definido como uma sucessão de gramáticas estáticas e de analisadores sintáticos dinâmicos, que seguem a evolução de uma gramática inicial à medida que vai sendo realizada a análise sintática de um texto por ela representada.

O crescimento de uma gramática inicial, do tipo livre de contexto, é feito através da incorporação de novos símbolos não-terminais e de novas regras de produção, específicos de cada texto considerado, de acordo com o seu contexto.

A idéia básica destas gramáticas assemelha-se à dos formalismos adaptativos, no sentido de que ambos empregam uma técnica que permite a alteração da configuração

inicial de seus modelos de representação da linguagem que descrevem, gerando, através da execução de operações especiais, uma sucessão de novas configurações.

Entretanto, os Evolving Grammars alteram a gramática inicial somente com a incorporação de não-terminais e de regras de produção nos seus respectivos conjuntos iniciais, enquanto que as operações dos formalismos adaptativos permitem, além desta operação, também a eventual remoção de regras.

Generative Grammar

Um outro formalismo foi desenvolvido por Christiansen, denominado como *generative grammar*, que é uma generalização das gramáticas de atributos, em que as relações de derivação são definidas de acordo com um atributo herdado e especializado. As declarações podem ser modeladas através de regras sintáticas que modificam ou estendem o conteúdo deste atributo.

Segundo [Shut93], esta gramática é uma gramática de atributos estendida, tal que o primeiro atributo de cada símbolo não-terminal é herdado e tem, como domínio semântico, todo o conjunto de gramáticas desta categoria. Este atributo especial é chamado *language attribute*.

Em trabalhos mais recentes, as *generative grammars* foram reformuladas em termos de *definite clause grammars,* e essa modificação pode ser encontrada em [Chr90].

Recursive Adaptable Grammar

Por fim, encontra-se a gramática adaptável recursiva, ou *Recursive Adaptable Grammar* (RAG), [Shut93] que é uma gramática adaptável do tipo declarativo, baseado em álgebra universal ou *one-sorted algebra* [Gog79].

2.5 Formalismos (reconhecedores) não-convencionais

Augmented Transition Network

Da mesma forma que ocorre no caso das gramáticas, há também autômatos não-convencionais que se mostram úteis em diversas aplicações. Alguns destes formalismos são aplicados na especificação de linguagens naturais, como por exemplo, os ATN (*Augmented Transition Network*), outras podem ser utilizadas no modelamento de sistemas, tais como as Redes de Petri [Pet81] e os Statecharts [Har87 apud Alm95].

O ATN é um formalismo bastante antigo, tendo sido inicialmente introduzido em [Woo70]. É um exemplo bem próximo dos autômatos convencionais de estrutura estática. A apresentação desta notação, que será apresentada a seguir, foi baseada nos livros [Gri86] e [Bro90].

Uma rede de transição pode ser vista como um padrão para o reconhecimento e geração de seqüências de palavras de uma linguagem e pode ser representada por um grafo orientado, em que os nós do grafo correspondem aos estados da rede.

Um arco em uma rede de transição pode ser rotulado por símbolos da linguagem. Um estado da rede é destacado para designar o seu estado inicial, e um ou mais estados são os estados finais.

Um ATN é uma extensão de uma rede de transição convencional utilizada como representação de uma linguagem regular.

As redes de transição possuem uma outra extensão que são as redes de transição recursivas. Essa é basicamente definida pela rotulação dos estados e pelos arcos, estes são rotulados com símbolos da linguagem livre de contexto.

Um ATN é uma rede de transição recursiva com a adição de procedimentos, geralmente codificados em Lisp, que permitem a formalização das restrições dependentes de contexto da linguagem.

Existe também um tipo de ATN recursivo que inclui registradores que podem armazenar dados que agem como variáveis locais; e funções, que podem ler e escrever sobre estes registradores [Bro90].

A adição destas funções pode ser usada para controlar os caminhos que podem ser percorridos na rede, por exemplo, através de um registrador que armazena a freqüência com que um dado ponto da rede é freqüentado. Assim, com uma função associada a este registrador, o reconhecedor pode controlar a direção do caminho a ser percorrido, baseado neste valor. Estas extensões permitem que o ATN recursivo se torne um reconhecedor sensível ao contexto.

Em [Per80] encontra-se uma comparação do *Definite Clause Grammar* com o ATN, e apresenta um método que permite a associação do DCG com o ATN. Assim, esse método permite que o ATN seja especificado em termos de uma gramática, o que facilita o desenvolvimento de linguagens

Adicionalmente existem diversos outros formalismos não convencionais, entre os quais destacam-se os Statecharts Adaptativos, Statecharts Adaptativos e Reds de Petri Adaptativos.

O *Statechart* convencional constitui uma versão mais abrangente do formalismo das máquinas de estados finitos e dos diagramas de transição de estados, e são capazes de representar o comportamento dinâmico ou os aspectos de controle dos sistemas que descrevem. Basicamente, pode-se dizer que um Statechart consiste de uma rede aninhada de bolhas, graficamente representadas como retângulos com os cantos arredondados, ligados por arcos direcionados, que representam as transições que podem ser disparadas na ocorrência de eventos e condições restritivas.

A partir deste modelo, foi desenvolvido o *Statechart Adaptativo* [Alm95] como uma ferramenta utilizada para a modelagem de sistemas, o qual segue as idéias básicas que regem os formalismos adaptativos.

Um Statechart Adaptativo é constituído de um Statechart convencional inicial, que transita entre os seus estados de maneira convencional até a execução de alguma transição que contenha uma chamada a uma função adaptativa.

O Statechart Adaptativo é composto por uma seqüência de eventos externos independentes, por um statechart convencional inicial, um statechart convencional final e uma seqüência de statecharts intermediários gerados a partir da execução das transições adaptativas.

Embora esse modelo esteja formalizado para a modelagem de sistemas, intuitivamente, ele também pode ser utilizado na formalização de sistemas de reconhecimento de linguagens.

Um outro formalismo que procura seguir as técnicas adaptativas é são os Statecharts adaptativos sincronizados [San97], um formalismo gráfico baseado em Statechart Adaptativo, e Redes de Petri, o que permite representar explicitamente através de redes de Petri as sincronizações existentes entre diferentes Statecharts adaptativos.

A seguir serão apresentados alguns autômatos com estrutura dinâmica.

Autômatos com estrutura dinâmica

Esta seção tem o objetivo de apresentar alguns formalismos que representam estruturas dinâmicas, as quais alteram sua configuração durante a sua execução.

a. Autômatos Adaptativos

O objetivo desta seção é o de apresentar as principais idéias do modelo dos Autômatos Adaptativos, formalismo que foi mencionado anteriormente neste texto. A apresentação desta formalização mostra algumas modificações em relação a sua formalização original, [Jos93], [Jos94] e [Alm95].

Os Autômatos Adaptativos são reconhecedores de linguagens sensíveis ao contexto. Constituem um modelo de máquina de estados finitos com características dinâmicas, que permitem que sua estrutura e comportamento seja alterado sempre que necessário, em particular quando são localizadas as dependência de contexto.

Essa alteração se dá através de sucessivas operações de modificação do autômato, tais como, adição, remoção de transições, acionadas pela execução de transições adaptativas.

Os autômatos adaptativos podem operar como autômatos finitos, para o reconhecimento de linguagens regulares; como autômatos de pilha estruturados, para aceitar linguagens livres de contexto; e como máquina de Turing, utilizando o recurso das ações adaptativas, para tratar linguagens sensíveis ao contexto.

A seguir será apresentada uma definição mais rigorosa dos autômatos adaptativos bem como uma descrição de sua operação.

Um **Autômato Adaptativo** M pode ser definido como uma tripla ordenada (E^0 , A, F^0), onde:

A é o conjunto de ações adaptativas, responsáveis pela dinâmica da topologia do autômato, e

E^0 é a máquina de estados inicial, basicamente um autômato de pilha estruturado [Jos93], que implementa M :

$E^0 = (Q^0 , SM^0 , \Sigma, \Gamma, P^0 , Z_0 , q_0 , F)$ onde

Q^0 é o conjunto de estados pré-definidos, contendo o estado inicial, $q_0 \in Q_0$.

F é o conjunto de estados finais, $F \subseteq Q_0$.

Σ e Γ são alfabetos de entrada e de pilha, respectivamente, e $Z_0 \in \Gamma$ representa a pilha vazia.

SM_0 é a coleção das submáquinas que implementam E_0 .

P^0 é o conjunto inicial de produções da forma $(\gamma g, e, s\alpha) : \rightarrow (\gamma g', e', s'\alpha)$, simbolizando a evolução do autômato de pilha estruturado da situação (g, e, s) para a situação

(g', e', s')

F^0 é uma relação do tipo (t, A, B), onde t $\in P^0$ e A, B $\in A$.

$F^0 \subseteq P^0 \times (A \cup \{\varepsilon\})^2$

Para cada transição t a relação F^0 associa uma ação adaptativa anterior A e uma ação adaptativa posterior B .

O autômato de pilha estruturado que implementa o autômato adaptativo, em cada momento, pode ser representado por um conjunto de produções da forma:

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

onde:

$(\gamma g, e, s \alpha)$ representa a situação do autômato antes da execução da transição:

g representa o conteúdo do topo da pilha antes da execução da transição

e representa o estado do autômato antes da execução da transição

s representa o símbolo da cadeia de entrada a ser consumido pela execução da transição

$(\gamma g', e', s' \alpha)$ representa a situação do autômato depois da execução da transição:

g' representa o conteúdo do topo da pilha depois da execução da transição

e' representa o estado do autômato depois da execução da transição

s' representa o símbolo inserido na cadeia de entrada pela execução da transição

As ações adaptativas associadas às produções do autômato de pilha estruturado acima são A e B (opcionais), e são executadas pelo autômato adaptativo em resposta à execução da transição à qual está associado, onde A e B $\in A$.

A representa a ação adaptativa a ser executada antes da mudança de estado

B representa a ação adaptativa a ser executada depois da mudança de estado

De forma análoga à que foi estudada no caso das gramáticas, restrições progressivas ao formato das produções acima definidas permitem representar da forma mais simples possível cada uma das diversas possíveis classes de linguagens. Conseqüentemente, também neste caso um único formalismo pode ser usado progressivamente, permitindo o emprego da forma mais simples em cada caso, conforme a necessidade de cada linguagem:

Para linguagens sensíveis ao contexto, usar produções da forma geral:

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

Para linguagens livres de contexto, limitá-las à forma geral seguinte, sem ações adaptativas:

$$(\gamma g, e, s\,\alpha) : \rightarrow (\gamma g', e', \alpha)$$

Para linguagens regulares, impor a limitação suplementar abaixo, eliminando a pilha:

$$(e, s\,\alpha) : \rightarrow (e', \alpha)$$

Operação

A evolução do autômato adaptativo ocorre através da modificação do conjunto de suas transições pela evolução gradual do seu conjunto de regras, com a adição ou remoção de produções, efetuadas por meio da execução das ações adaptativas. Genericamente, essa evolução ocorre da forma descrita pelo algoritmo abaixo:

Iniciar $i = 0$ (i é um índice que identifica as diversas transições particulares)

Dada uma situação $(g, e, s)_i$, procura-se em P_i uma transição $(\gamma g, e, s\alpha)_i : \rightarrow (\gamma g', e', s'\alpha)_i$

Determinando-se, assim, g', e', s', que vai ser a nova situação $(g', e', s')_i = (g, e, s)_{i+1}$

Aplica-se $\varphi_i : P_i \rightarrow \mathbb{A} \times \mathbb{A}$. Esta função determina, a partir da transição P_i , o par $(\mathtt{A}_i, \mathtt{B}_i)$ de ações adaptativas que a ela se associa:

$(\gamma g, e, s\,\alpha)_i \mathtt{A}_i : \rightarrow (\gamma g', e', s'\,\alpha)_{i+1} , \mathtt{B}_i$

Executar $\mathtt{A}_i : P_i \rightarrow P_i'$

Procura-se novamente, agora em P_i', uma transição $(\gamma g, e, s\alpha)_i : \rightarrow (\gamma g'', e'', s''\alpha)_i$

se $(\gamma g'', e'', s''\alpha)_{i+1} \neq (\gamma g', e', s'\alpha)_i$, então a transição que estava sendo executada desapareceu por efeito da execução da ação adaptativa A_i. Nada mais há a ser feito neste caso, devendo-se voltar ao passo 1. para a escolha da próxima transição a executar.

Caso contrário, volta-se ao passo 2. para consumar a execução da transição corrente.

Executar B_i : $P_i' \rightarrow P_{i+1}$.

Incrementar i, e voltar ao passo 1. para prosseguir o reconhecimento

Todo autômato adaptativo pode ser visto como uma máquina de estados que, ao início de sua operação, apresenta uma topologia fixa, pré-determinada, representada pela máquina de estados inicial E_0.

Sobre a topologia dada por uma máquina de estados qualquer E_i , o autômato adaptativo opera como um autômato usual, efetuando uma seqüência de transições não-adaptativas, sendo tal seqüência finalizada ou pelo término do reconhecimento da sentença, ou então pela execução de alguma transição adaptativa, responsável por alterar sua topologia.

Neste caso, pode-se dizer que o autômato se modifica, assumindo uma nova topologia, representada por alguma outra máquina de estados E_{i+1} , cujo estado inicial de operação deverá ser determinado pela transição executada.

Esta operação se repete até que, na topologia dada por uma máquina de estados E_n o reconhecimento se encerre com êxito, ou que a cadeia de entrada seja rejeitada pelo autômato. Assim, sendo $\omega \in \Sigma^*$ a cadeia de entrada, que pode ser decomposta em uma seqüência de subcadeias $\alpha_0, \alpha_1, \ldots \alpha_n \in \Sigma^*$, pode-se dizer que o reconhecimento de uma cadeia $\omega = \alpha_0 \alpha_1 \ldots \alpha_n$ por um autômato adaptativo pode ser vista como uma trajetória de reconhecimento, dada por

$$\langle E_0 , \alpha_0 \rangle \rightarrow \langle E_1 , \alpha_1 \rangle \ldots \rightarrow \langle E_n , \alpha_n \rangle$$

que representa o consumo da subcadeia α_0 pela máquina E_0, a qual consome a subcadeia α_1, etc.

O autômato adaptativo inicia sua operação em uma situação inicial dada por (Z_0 , e_0 , ω), ou seja, com a pilha vazia, com a cadeia de entrada completa, e com o autômato, representado pela máquina de estados E_0, em seu estado inicial e_0 .

Estando o autômato adaptativo, em uma dada ocasião, na situação ($\gamma \pi$, e , $\sigma \alpha$), é feita uma busca no conjunto de produções do autômato, com o objetivo de localizar alguma produção aplicável a tal situação.

Diz-se que uma produção é aplicável quando a situação corrente do autômato não conflitar com o especificado pelas informações que compõem o lado esquerdo da mesma (conteúdo do topo da pilha, estado corrente e átomo corrente da cadeia de entrada). Podem ocorrer diversos casos:

Existe uma só produção aplicável. Neste caso a transição correspondente é executada determinística e incondicionalmente

Há mais de uma produção aplicável. Transições internas a uma sub-máquina têm precedência sobre outras transições aplicáveis, e, entre estas, transições com consumo de átomos precedem as transições em vazio aplicáveis. Nesta situação, restando apenas uma produção aplicável, esta deve ser executada deterministicamente. Restando mais de uma produção aplicável, constata-se um não-determinismo na operação do autômato, devendo todas as transições correspondentes ser executadas em paralelo.

Nenhuma das produções é aplicável. Caso isto ocorra em um estado não-final, constata-se a rejeição da cadeia pelo autômato, devendo-se considerar que a cadeia não pertence à linguagem definida pelo autômato adaptativo, a não ser que haja, em paralelo, outras tentativas de reconhecimento em curso no autômato (disparados de forma não-determinística). Caso isto ocorra em algum estado final, a cadeia de entrada é aceita se tiver sido completamente consumida e se a pilha estiver vazia, caso contrário será rejeitada se não houver tentativas de reconhecimento em curso no autômato.

As três regras acima devem ser aplicadas, repetidamente, até que ou seja atingida uma situação final (γ , e_f , ε) ou então que a cadeia de entrada seja rejeitada pelo autômato.

Descreve-se a seguir a maneira como são utilizadas as produções na execução das transições do autômato adaptativo. Para uma produção aplicável da forma

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

executam-se, na seqüência abaixo indicada, as atividades seguintes:

Se `A` estiver presente, a ação adaptativa correspondente deverá ser executada em primeiro lugar. Se esta ação eliminar a produção em execução, esta será descontinuada, voltando o autômato adaptativo a buscar, no seu novo conjunto de produções, outra produção que seja aplicável à situação corrente.

Se *g* estiver especificado, deverá ser eliminado do topo da pilha

Se *g*' estiver especificado, deverá ser empilhado

Se *s* estiver especificado, deverá ser consumido da cadeia de entrada, passando a ser o átomo corrente aquele que figurar à sua direita na cadeia de entrada, se existir

Se *s*' estiver especificado, deverá ser inserido imediatamente à esquerda do átomo corrente da cadeia de entrada

o estado *e* deixa de ser o estado corrente, que passa a ser o estado *e*'

Se `B` estiver presente, a ação adaptativa correspondente deverá ser executada em último lugar.

Para completar esta especificação do funcionamento do autômato adaptativo, resta apresentar a notação com que as ações adaptativas são especificadas, bem como a sua interpretação.

Para que seja possível utilizar uma ação adaptativa, é necessário definir a correspondente função adaptativa `F` , através de uma ênupla cujas componentes são:

o nome `F` da função adaptativa

a lista de parâmetros formais τ_1 , τ_2 , ... , τ_p

uma lista de variáveis (opcional)

uma lista de geradores (opcional)

uma chamada de ação adaptativa anterior (opcional)

uma lista de ações básicas: de inspeção, eliminação e inclusão de produções

uma chamada de ação adaptativa posterior (opcional)

As variáveis, os parâmetros e os geradores são representados por nomes simbólicos que representam valores a serem utilizados pelas produções. Ao longo de cada

execução da função esses elementos, inicialmente indefinidos, não assumem mais de um único valor, o qual, uma vez atribuído, não se altera até o final da execução da função.

Variáveis oferecem uma forma de se referenciar por nome algum valor que não seja constante. As variáveis recebem valores como efeito da execução de ações adaptativas básicas de inspeção ou de eliminação de transições.

Análogos às variáveis, os geradores permitem associar automaticamente valores não repetidos aos nomes que os representam, a cada ocasião em que a função adaptativa é ativada. Tal preenchimento ocorre também ao início da execução da função adaptativa.

Os parâmetros, associados a valores indefinidos na ocasião do início da execução da função adaptativa, são sempre parâmetros de entrada.

São preenchidos, ao início da execução da função adaptativa, pelo mecanismo de passagem de parâmetros, que atribui a cada um o valor correntemente associado ao correspondente argumento, indicado na chamada da função, valor este que será mantido por toda a execução da função adaptativa.

.

As ações adaptativas básicas de inspeção, eliminação e inclusão de produções assumem respectivamente as seguintes formas

$$?\,[\,(\gamma g, e, s\,\alpha) :\ \mathtt{A}, \rightarrow (\gamma g', e', s'\,\alpha), \mathtt{B}\,]$$

$$-\,[\,(\gamma g, e, s\,\alpha) :\ \mathtt{A}, \rightarrow (\gamma g', e', s'\,\alpha), \mathtt{B}\,]$$

$$+\,[\,(\gamma g, e, s\,\alpha) : \mathtt{A}, \rightarrow (\gamma g', e', s'\,\alpha),\ \mathtt{B}\,]$$

onde as expressões entre colchetes representam respectivamente formas de produções a serem pesquisadas, eliminadas ou inseridas no conjunto de produções que define a autômato adaptativo, e a respectiva associação às ações adaptativas a serem executadas quando de sua aplicação.

No caso das ações básicas de inspeção, o conjunto de produções do autômato adaptativo é pesquisado quanto à ocorrência de produções da forma indicada entre colchetes. Eventuais variáveis, utilizadas para denotar g , e , s , g' , e' , s', bem como nomes e argumentos das ações adaptativas `A` e `B`, são preenchidas com os valores correspondentes, indicados na(s) produção(ões) eventualmente encontrada(s). Caso não seja encontrada nenhuma produção do formato indicado, as correspondentes variáveis são

marcadas como indefinidas. Havendo mais de uma produção nestas condições, constata-se um não-determinismo, e os vários casos possíveis são tratados em paralelo. Ações básicas de inspeção não alteram o conjunto de produções do autômato adaptativo.

Para ações básicas de eliminação, procede-se inicialmente como no caso de ações básicas de inspeção, eliminando-se do conjunto de produções que na ocasião definem o autômato adaptativo, todas as produções encontradas como resultado da busca realizada. Caso não seja encontrada nenhuma produção desta forma, nenhuma alteração será executada.

Ações básicas de inclusão promovem, no conjunto de produções do autômato, a inserção da produção cuja forma vem indicada entre colchetes. Caso nesta ocasião tal produção já exista no conjunto de produções que define o autômato, nenhuma inserção será efetuada.

As ações adaptativas `A` ou `B` são incluídas nas produções do autômato na forma `A` (φ_1 , φ_2 , ... , φ_p) ou `B` (φ_1 , φ_2 , ... , φ_p) respectivamente.

onde φ_1 , φ_2 , ... , φ_p são os p argumentos passados à função adaptativa de nome `A`. ou `B` .

Nas chamadas paramétricas de ações adaptativas, impõem-se as seguintes condições:

a referência à função adaptativa deve ser feita usando-se o mesmo o nome `F` com que a função adaptativa foi declarada

os argumentos φ_1 , φ_2 , ... , φ_p deverão guardar correspondência posicional com os parâmetros formais a que se referem: τ_1 , τ_2 , ... , τ_p

cada um dos argumentos φ_1 , φ_2 , ... , φ_p poderá ser: um nome de função adaptativa, um nome de parâmetro formal, um nome de variável, um nome de gerador, um símbolo de pilha, um símbolo do alfabeto de entrada, um nome de estado.

As ações adaptativas anterior e posterior são normalmente representadas por meio de chamadas de funções adaptativas, em geral paramétricas.

Define-se uma configuração $T_k = (E_k, \gamma_k, q_k, \omega_k)$ em um instante k como sendo as informações que representam o autômato naquele instante k. Dizemos que neste instante, o autômato é representado pela máquina de estados $E_k \in SM^0$, com $\gamma_k \in \Gamma^*$ sendo o conteúdo completo da pilha, $q_k \in Q$ o estado corrente da máquina de estados E_k, e $\omega_k \in \Sigma^*$ a parte da cadeia de entrada que ainda não foi consumida.

A configuração inicial é representada por $T_0 = (E_0, Z_0, q_0, \omega)$, onde E0 é a maquina de estados inicial do autômato adaptativo, Z_0 indica a pilha vazia, q_0 é o estado inicial e ω é a cadeia de entrada completa.

A configuração final é representada por $T_f = (E_f, Z_0, q_f, \varepsilon)$, onde E_f é a máquina de estados final que implementa a autômato adaptativo, Z_0 é a pilha vazia, q_f é um estado final e ε indica a cadeia vazia.

O passo de derivação de uma configuração de um instante k para o próximo instante k+1 é representado por $T_k \vdash T_{k+1}$.

Seja $T_k \vdash T_{k+1} \vdash T_{k+2}. \vdash ..... \vdash T_{k+m}$ então podemos denotar esta seqüência de derivações como $T_k \vdash^* T_{k+m}$.

Dizemos que a linguagem aceita por um autômato adaptativo M= (E_0 , A) é $L(M) = \{ \omega \mid T_0 \vdash^* T_f \}$, isto é a linguagem formada por todas as cadeias ω tal que a partir da configuração inicial $T_0 = (E_0, Z_0, q_0, \omega)$ o autômato adaptativo atinge, após um número adequado de passos, a configuração final $T_f = (E_f, Z_0, q_f, \varepsilon)$.

Linguagem reconhecida pelo autômato a partir de um estado i é definida por

$L(i) = \{ \omega \mid \omega \in \Sigma^* \text{ e } (E^x, \gamma, i, \omega) \vdash^* (E^y, Z_0, f, \varepsilon) \}$, onde $f \in F^y$ e E^y é a submáquina final que implementa o autômato adaptativo.

Seja A um símbolo não-terminal da gramática adaptativa

$L(A)$ é o conjunto das sentenças reconhecidas por A, sendo A considerada como sendo a máquina principal.

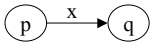
$L(q_0, j)$ é o conjunto de prefixos reconhecido pelo autômato partindo do estado q_0 e chegando ao estado j.

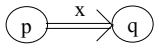
Caso particular, se $j \in F^y$ então $L(q_0, j)$ é um subconjunto da linguagem reconhecida pelo autômato.

A figura seguinte apresenta a notação gráfica utilizada para denotar as transições do autômato adaptativo.

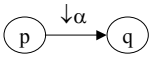
Notação gráfica para Autômatos Adaptativos

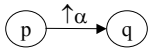
1. Estado inicial: q_0 

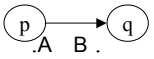
2. Transição interna: $(\gamma, p, x\alpha) :\rightarrow (\gamma, q, \alpha)$ 

3. Transição de chamada de submáquina: $(\gamma, p, x\alpha) :\rightarrow (q, N_0, \varepsilon)$, onde N_0, é o estado inicial da submáquina 

4. Retorno de submáquina: 

5. Empilhamento de informação contextual: $(\gamma, p, \varepsilon) :\rightarrow (\gamma, q, \alpha)$ 

6. Desempilhamento de informação contextual: $(\gamma, p, \alpha) :\rightarrow (\gamma, q, \varepsilon)$ 

7. Transição com ação adaptativa: $(\gamma, p, x\alpha) : A , \rightarrow (\gamma, q, \alpha) , B$
B : ação adaptativa anterior
A : ação adaptativa posterior 

Esta apresentação dos Autômatos Adaptativos procurou mostrar as idéias principais deste modelo. Maiores detalhes sobre o funcionamento e aplicações podem ser encontradas nas referências citadas anteriormente.

Autômato Auto Modificável (SMA)

Os autômatos auto-modificáveis ou SMA (*self-modifying automata*) [Rub93], [Rub95], [Shu95] foram desenvolvidos como um modelo formal para a representação de linguagens do tipo 0, na hierarquia de Chomsky. Da mesma forma que os autômatos adaptativos, os SMA permitem a alteração do seu conjunto de transições durante a computação, através de operações do tipo adição de estados e de transições, e remoção de estados e de transições.

O funcionamento dos SMA é semelhante ao de um autômato finito padrão, e a sua principal diferença está nas transições que estão associadas às ações de modificação, que alteram a configuração do autômato à medida que é executado o reconhecimento de uma sentença da linguagem. Esta técnica é utilizada também no reconhecimento de contruções

livres de contexto, como ocorre, por exemplo no aninhamento de comandos. No caso dos autômatos adaptativos, esta forma gramatical pode ser reconhecida com o auxílio de uma pilha, não sendo necessário recorrer a ações adaptativas.

Em [Rub95] está demonstrado que os SMFA's são equivalentes à máquina de Turing, e portanto reconhecem linguagens recursivamente enumeráveis.

No próximo capítulo será apresentada o formalismo proposto neste trabalho, que são as gramáticas adaptativas.

Capítulo 3 – Gramáticas Adaptativas

Este capítulo é dedicado à apresentação da gramática adaptativa. O desenvolvimento deste formalismo foi baseado nos fundamentos que deram origem aos autômatos adaptativos, com o intuito de propiciar a dualidade entre os dois modelos e facilitar a migração entre os dois tipos de formalismos adaptativos.

Este capítulo é dividido em diversas seções. A primeira seção fornece um panorama geral dos formalismos adaptativos. Em seguida, é feita a apresentação da definição da gramática adaptativa, mostrando suas principais características, além de estabelecer a equivalência entre as gramáticas adaptativas e os autômatos adaptativos. Por último, é apresentado um exemplo de uma pequena linguagem especificada através da uma gramática adaptativa.

3.1.Formalismos Adaptativos

Usualmente, os formalismos que são utilizados para representar linguagens são constituídos por arranjos estáticos de elementos que, uma vez estabelecidos, não se alteram durante a sua operação.

As gramáticas convencionais são representadas por conjuntos estáticos e finitos de símbolos terminais e não-terminais e por um conjunto estático de regras de produção.

Algo similar se observa com os autômatos convencionais, que utilizam conjuntos estáticos e finitos de estados e de regras de transições.

Nos formalismos adaptativos é permitido que aconteçam modificações estruturais nos arranjos que constituem estes modelos, o que pode acarretar alterações no comportamento do dispositivo durante o processamento de uma sentença. Por exemplo, no caso dos autômatos de natureza adaptativa, estas alterações podem modificar a sua topologia, permitindo que novos caminhos sejam criados e que outros sejam removidos.

O agente responsável por tais alterações, em um formalismo adaptativo, é a *ação adaptativa*. No caso das gramáticas, a ação adaptativa está associada às suas regras de produção. Durante a geração de uma sentença, se uma regra de produção, associada a alguma ação adaptativa, for escolhida para substituir algum não-terminal da forma

sentencial, então esta ação adaptativa será ativada, sendo então capaz de alterar o conjunto dos símbolos não-terminais e e das regras de produção da gramática.

Uma ação adaptativa é capaz de consultar, remover e adicionar regras de produção, no caso de gramáticas, e regras de transição, no caso de autômatos, alterando assim as formas dos modelos originais.

Em outros formalismos, analogamente, a inclusão de ações adaptativas associadas às suas regras operacionais caracteriza a criação do formalismo adaptativo correspondente. É o que ocorre, por exemplo, no caso dos Statecharts Adaptativos e das Redes de Markov Adaptativos.

3.2.Gramáticas Adaptativas

Em idéias gerais, o conceito de uma gramática adaptativa é de um formalismo generativo capaz de representar linguagens sensíveis ao contexto. O que distingue estas gramáticas das convencionais é a sua capacidade de se auto-modificar à medida que uma sentença da linguagem vai sendo derivada.

As modificações acontecem durante a geração da sentença, quando da aplicação de regras de produção às quais estejam associadas ações adaptativas, cuja execução acarreta alterações no conjunto de regras de produção e, possivelmente, no conjunto de símbolos não-terminais.

Uma sentença ω, pertencente à linguagem representada por uma gramática adaptativa, é gerada a partir de uma gramática original G_0 e de uma sucessão de gramáticas G_1, ... G_{n-1} intermediárias, criadas sempre que alguma ação adaptativa for ativada durante o geração da sentença, e se encerra utilizando G_n como sendo a gramática final.

Pode-se definir uma gramática adaptativa G como segue:

Uma gramática adaptativa G é constituída por uma tripla ordenada (G^0, T, R^0), onde T é um conjunto finito, possivelmente vazio, de funções adaptativas

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0 , S)$ é uma *gramática inicial*, onde

V_N^0 é um conjunto finito e não vazio de *símbolos não terminais*,

V_T é um conjunto finito e não vazio de *símbolos terminais*,

$V_N^0 \cap V_T = \varnothing$

V_C é um conjunto finito de *símbolos de contexto*.

$V^0 = V_N{}^0 \cup V_T \cup V_C$

$V_N{}^0$, V_T e V_C são conjuntos disjuntos dois a dois

$S \in V_N{}^0$ é o símbolo inicial da gramática,

$P_L{}^0$ é o conjunto de regras de produção aplicável às situações livres de contexto.

$P_D{}^0$ é o conjunto de regras de produção aplicável às situações dependentes de contexto.

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

Tipo 1 ou pertencentes ao conjunto $P_L{}^i$, onde $i \in \mathbb{N}$:

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

onde $\alpha \in (V_T \cup V_N)^*$, $N \in V_N{}^i$ e A é uma ação adaptativa opcional associada a regra de produção. Essa ação adaptativa é opcional e pode ser omitida.

Tipo 2 ou pertencentes ao conjunto $P_L{}^i$, onde $i \in \mathbb{N}$:

$$N \rightarrow \phi$$

onde ϕ é um meta-símbolo que indica o conjunto vazio.

Esta produção 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 ou pertencentes ao conjunto $P_D{}^i$, onde $i \in \mathbb{N}$:

$$\alpha N \leftarrow \{ \texttt{A} \} \ \beta M$$

onde $\alpha \in V_C \cup \{\varepsilon\}$ e $\beta \in V_C$,

ou

$$\alpha N \rightarrow \{ \texttt{A} \} \ \beta M$$

onde $\alpha \in V_C$, $\beta \in V_T \cup \{\varepsilon\}$, N e $M \in V_N{}^i$, com A opcional.

A primeira produção tem a seta ao contrário, indicando que β está sendo injetada na cadeia de entrada. Esta produção troca αN por βM, inserindo informações de contexto.

A segunda produção tem a seta no sentido correto, mas possui no seu lado esquerdo um

símbolo de contexto seguido por um símbolo não-terminal, isto indica que αN está sendo trocado por βM e gerando β na cadeia de saída.

R^0 é uma relação do tipo (r, A), onde $r \in (P_L{}^0 \cup P_D{}^0)$ e $A \in T$.

$R^0 \subseteq P^0 \times (T \cup \{\varepsilon\})$

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

Definição (regra de produção adaptativa)

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

Note que 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, a qual, em conjunto com as produções em $P_D{}^0$, responde pela representação das dependências de contexto nesta gramática.

Declaração de uma função adaptativa

Ações adaptativas são chamadas de funções adaptativas.

O formato de declaração de uma função adaptativa é o seguinte, inspirado nas funções adaptativas utilizadas para os autômatos adaptativos:

Nome da ação (lista de parâmetros formais)

{ lista de variáveis, lista de geradores:
ação adaptativa opcional
ação elementar 1
ação elementar 2
.....
ação elementar n
ação adaptativa opcional
}

No cabeçalho da função adaptativa consta o nome da função adaptativa, seguido, entre parênteses, por uma lista de parâmetros formais, separados por vírgulas. Ao ser chamada a função adaptativa, esta lista será preenchida com os correspondentes valores dos

argumentos passados na particular chamada da função, valores esses que serão preservados intactos durante toda a execução da função.

O corpo da função adaptativa está representado entre chaves. Ele é constituído por uma lista de variáveis (separadas por vírgulas), uma lista de geradores (separados por vírgulas), seguidos por dois pontos. A seguir, a função adaptativa pode conter, opcionalmente, a chamada de alguma função adaptativa. Esta ação adaptativa é processada antes da execução da função adaptativa que está sendo declarada.

Em seguida, são listadas diversas *ações adaptativas elementares*, podendo haver ainda, ao final, outra chamada de função adaptativa, a ser processada após a execução de todas as ações elementares da função adaptativa que está sendo declarada.

Variáveis são símbolos que dão nome a elementos cujos valores são desconhecidos na ocasião da chamada da função adaptativa, e que devem ser preenchidos durante a execução da função adaptativa, como resultado de ações adaptativas de consulta.

Geradores são elementos semelhantes às variáveis, mas que são automaticamente preenchidas ao início da execução da função adaptativa, com valores que não se repetem, preservando então este valor durante toda a execução da função.

Há três tipos de ações adaptativas elementares: consulta, adição e remoção, as quais podem ser representadas da seguinte forma, respectivamente:

? [N → {ação_adaptativa_opcional } M]

+ [N → {ação_adaptativa_opcional } M]

- [N → {ação_adaptativa_opcional } M]

onde $N \in V_N^i$, $M \in V^{i*}$, para algum $i \in \square$, em que i representa o passo da evolução da gramática, e para alguma ação_adaptativa, se existir.

Ação elementar de consulta

Esta ação verifica a existência, no conjunto corrente de produções da gramática, de alguma regra de produção que apresente o formato indicado. Quando um ou mais símbolos da expressão estiverem representados por alguma variável, esta ação irá preencher essas variáveis com os valores correspondentes que forem encontrados. Conseqüentemente, variáveis são preenchidas como resultado da execução das ações elementares de consulta. Deve-se notar que as variáveis, inicialmente indefinidas, são preenchidas no máximo uma

única vez durante a execução de uma função adaptativa.

Ação elementar de adição

Esta ação elementar inclui na gramática uma nova regra de produção com o formato indicado, podendo utilizar variáveis preenchidas, e também geradores para a definição de símbolos não existentes até o momento da sua aplicação. A adição de uma regra já existente é inócua. A adição de regras que referenciam variáveis indefinidas também é inócua.

Ação elementar de remoção

A execução desta ação elementar é feita em dois passos. O primeiro é a consulta da existência da regra se quer excluir com o conseqüente preenchimento das variáveis. Em caso afirmativo, a remoção é feita, caso contrário, nada é removido.

Observe-se que, havendo mais de uma ação adaptativa elementar a ser executada independentemente da ordem em que foram declaradas, têm precedência as regras de consulta. Entre as remoções e adições, têm precedência as remoções. As adições são sempre executadas por último. Ações elementares que referenciam variáveis indefinidas não serão executadas.

Definição (ação adaptativa)

Em fase de execução, as ações adaptativas são responsáveis pelas alterações da gramática.

Quando uma ação adaptativa é executada, a gramática evolui, alterando os seus conjuntos de símbolos não-terminais e de regras de produção.

Exemplo de chamada de função adaptativa:

$$\begin{aligned} \mathtt{A}\,(A_1, B_1, C_1) \;=\; \{\; & A_2^{*}, B_2^{*}, C_2^{*}: \\ & + [A_1 \rightarrow \{\mathtt{A}\,(A_2, B_2, C_2)\}\; aA_2] \\ & + [A_1 \rightarrow \{\mathtt{B}\;\; (B_2, C_2)\}\; \varepsilon] \\ & + [B \rightarrow b\; B_2\,] \\ & + [C \rightarrow c\; C_2\,] \end{aligned}$$

}

Derivação (seqüência de derivação) $\Rightarrow_G$

Assim como as gramáticas convencionais, a geração das sentenças de uma linguagem representada por uma gramática adaptativa se dá através de sucessivas substituições dos símbolos não-terminais, de acordo com as regras da gramática, partindo de um símbolo inicial S.

A diferença primordial deste procedimento é a presença das ações adaptativas que, quando ativadas, alteram a gramática que vinha sendo utilizada até o momento em que a regra adaptativa foi aplicada.

Uma outra importante diferença em relação às gramáticas convencionais refere-se às produções envolvendo contexto, as quais podem utilizar-se de símbolos de contexto na derivação, de forma que as substituições que eles dependem sejam efetuadas apenas quando o referido símbolo de contexto estiver explícitamente presente na forma sentencial quando da substituição do não-terminal por ele afetado.

Assim, para cada passo de derivação, existe alguma gramática adaptativa intermediária G^i sendo utilizada, e isto é identificado da seguinte forma nos símbolos de relação de derivação:

Se $\alpha \rightarrow \beta$ é uma produção em $P_L{}^i$ e γ é uma cadeia de símbolos de $V_T{}^*$ e δ é uma cadeia de símbolos de V^*, então $\gamma\alpha\delta \Rightarrow_{G^i} \gamma\beta\delta$, isto é, $\gamma\alpha\delta$ deriva diretamente $\gamma\beta\delta$ na gramática G^i.

Define-se este tipo de derivação como sendo uma *derivação interna*.

Se $\alpha \rightarrow \phi$ é uma produção em $P_L{}^i$ e γ é uma cadeia de símbolos de $V_T{}^*$ e δ é uma cadeia de símbolos de V^* e se não houver outra produção aplicável, então $\gamma\alpha\delta \Rightarrow_{G^i} \phi$, isto é, $\gamma\alpha\delta$ não deriva nenhuma sentença (nem sequer a cadeia vazia).

Se $\alpha N \leftarrow \beta M$ é uma produção em $P_C{}^i$, onde $\alpha \in V_C{}^*$ e $\beta \in V_C$, N e $M \in V_N{}^i$, γ é uma cadeia de símbolos de $V_T{}^*$ e δ é uma cadeia de símbolos de V^*, então $\gamma\alpha N\delta \Rightarrow_{G^i} \gamma\beta M\delta$, isto é, $\gamma\alpha N\delta$ deriva diretamente $\gamma\beta M\delta$ na gramática G^i.

Se $\alpha N \rightarrow \beta M$ é uma produção em $P_C{}^i$, onde $\alpha \in V_C$ e $\beta \in V_T{}^*$, N e $M \in V_N{}^i$, γ é uma

cadeia de símbolos de V_T^* e δ é uma cadeia de símbolos de V^*, então $\gamma\alpha N\delta \Rightarrow_{G^i} \gamma\beta M\delta$, isto é, $\gamma\alpha N\delta$ deriva diretamente $\gamma\beta M\delta$ na gramática G^i.

A relação $\Rightarrow_{G^i}$ envolve duas cadeias de símbolos. A segunda é gerada a partir da primeira pela aplicação de uma única regra de produção pertencente a gramática G^i.

Quando a seqüência de derivação é descrita da forma:

$..... \alpha_i \Rightarrow_{G^k} \{ A \} \alpha_{i+1} \Rightarrow_{G^{k+1}} \alpha_{i+2}....$

isto indica que uma regra de produção adaptativa foi aplicada, ou seja, primeiro executa-se a ação adaptativa { A }, que gera a gramática G^{k+1}; em seguida aplica-se a regra de produção que atua no ambiente da gramática G^k. Finalmente, é feita a migração para a gramática G^{k+1}, descartando-se a gramática G^k.

A forma sentencial α_{i+1} foi gerada pela gramática G^k, e α_{i+2}, pela gramática G^{k+1}. Designa-se este tipo de derivação como sendo uma *derivação adaptativa*.

Seja $\alpha_k \Rightarrow_{G^i} \alpha_{k+1} \Rightarrow_{G^i} \alpha_{k+2} \Rightarrow_{G^i} ..... \Rightarrow_{G^i} \alpha_{k+m}$, onde $\alpha_k, \alpha_{k+1}, \alpha_{k+2}, ... , \alpha_{k+m}$ são cadeias de símbolos de V^{i*}, para uma gramática G^i, então esta expressão pode ser denotada da seguinte maneira $\alpha_k \Rightarrow_{G^i}^* \alpha_{k+m}$,

Definição ($\rightarrowtail$):

a.) Define-se uma derivação da forma $\{ A \} \alpha \Rightarrow_{G^i} \beta$ como sendo $\alpha \rightarrowtail_{(A, i)} \beta$

b.) Define-se uma derivação da forma $\alpha \Rightarrow_{G^i} \beta$ como sendo $\alpha \rightarrowtail_{(\varepsilon, i)} \beta$

c.) Define-se derivação da forma $\{ A \} \alpha \Rightarrow_{G^i} \beta$ ou $\alpha \Rightarrow_{G^i} \beta$ como sendo $\alpha \rightarrowtail \beta$

d.) Uma seqüência de zero ou mais derivações da forma

$\alpha \rightarrowtail \alpha_1 \rightarrowtail \alpha_2 \rightarrowtail ..... \rightarrowtail \alpha_n$

como sendo $\alpha \rightarrowtail^* \alpha_n$

Uma linguagem gerada por uma gramática adaptativa G (denota-se L(G)) é definida como sendo $\{ \omega \mid \omega \in V_T^* \text{ e } S \rightarrowtail^* \omega \}$. Isto é, uma sentença pertence a L(G) se e somente

se:

1. A sentença consiste somente de terminais.

2. A sentença deve ser derivada a partir de S aplicando-se sucessivas derivações da gramática.

Uma forma sentencial α é uma cadeia de símbolos formada por terminais, não-terminais e símbolos de contexto e deve ser derivada de S, isto é, $S \overset{*}{\rightarrow} \alpha$.

Havendo mais de um não-terminal na forma sentencial, convencionou-se efetuar substituições sempre partindo do não-terminal mais à esquerda, da mesma forma que é feito no caso do *parsing top-down* para gramáticas livres de contexto. A derivação é sempre referente ao não-terminal mais à esquerda da forma sentencial.

A derivação é sensível à ordem de substituição dos não-terminais, devido aos efeitos colaterais provados pelas ações adaptativas.

Notar que a posição da ação adaptativa é sempre após a regra, sugerindo que a sua execução seja feita sempre após a substituição do não-terminal correspondente.

Árvore de derivação

Uma árvore de derivação da gramática adaptativa é similar a uma árvore de derivação convencional, exceto pela presença de regras de produção especiais do tipo 3, definidos anteriormente, as quais envolvem símbolos de contexto.

Existe uma correspondência direta deste tipo de regra de produção com um caso particular de transição do autômato adaptativo:

$$(\gamma, a, \sigma\alpha) \rightarrow (\gamma, b, \sigma'\alpha), \texttt{A} ,$$

onde $\sigma \in \Sigma \cup \{\varepsilon\}$ e $\sigma' \in \Sigma$.

A esta transição corresponde o seguinte par de produções gramaticais:

$$E' \rightarrow \sigma E''$$

e

$$E'' \leftarrow \{\texttt{B} \} \sigma' E'$$

onde E' e E' são os não-terminais correspondentes aos estados a e b respectivamente.

O não-terminal E'' é um não-terminal auxiliar.

Exemplo: Seja dada uma gramática adaptativa $G = (G^0, T, R^0)$ onde $T = \phi$ e

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0 , S)$

$P_L^0 \cup P_D^0 = \{$

$S \rightarrow aA$	$C \leftarrow yD$
$A \leftarrow xB$	$yD \rightarrow cE$
$xB \rightarrow bC$	$E \rightarrow \varepsilon$

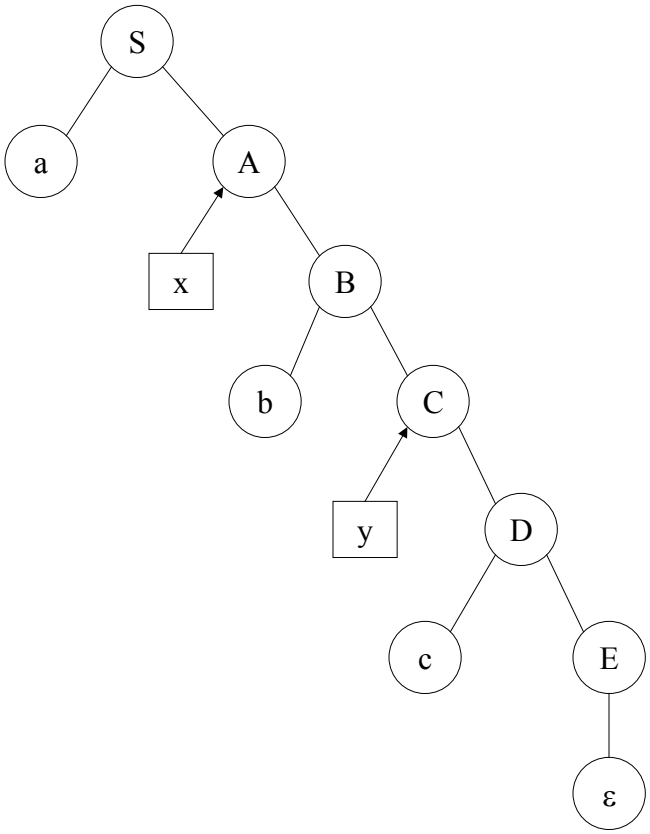
$\}$

Onde $V_N^0 = \{S, A, B, C, D, E\}$, $V_T = \{a, b, c\}$ e $V_C = \{x, y\}$, $V_C = \phi$

e quisermos gerar a cadeia ω = abc, temos a seguinte derivação:

$S \Rightarrow aA \Rightarrow axB \Rightarrow abC \Rightarrow abyD \Rightarrow abcE \Rightarrow abc\varepsilon$

Temos portanto a seguinte árvore de derivação, com nós diferenciados (contextuais) para o caso das regras de produção especiais.



Os símbolos x e y estão denotados em nós representados por retângulos, para indicar que eles não serão gerados na cadeia de saída. x e y são somente informações de contexto, que foram inseridos na forma sentencial durante a derivação da sentença, mas que depois foram substituidos por ação da aplicação da regra cujo lado esquerdo contém a informação contextual correspondente.

Definições:

A linguagem gerada pela gramática é o conjunto de todas as sentenças obtidas através de derivações sucessivas partindo do símbolo inicial S até a obtenção de uma forma sentencial que contenha somente símbolos terminais.

Linguagem gerada a partir de um não-terminal

$\mathtt{L}(i) = \{ \omega \mid \omega \in V_T^* \text{ e } i \Rightarrow_{G^x} \ldots\ldots \Rightarrow_{G^y} \omega\}$

Seja A um símbolo não-terminal da gramática adaptativa

$\mathtt{L}$ (A) é o conjunto das sentenças geradas como contorno das árvores de derivação tendo A como nó principal.

$\mathtt{L}$ (S, j) é o conjunto das formas sentenciais geradas a partir do não-terminal S e chegando a uma forma sentencial com prefixo da forma α j, onde $\alpha \in V_T^*$ e j é um não-terminal.

Considerações sobre o meta-símbolo ϕ

Para melhor ilustrar a utilização do meta-símbolo ϕ, será dado um pequeno exemplo de gramática adaptativa que gera a linguagem com um única sentença abc.

Exemplo:

As regras de produção da gramática adaptativa inicial G^0 são as seguintes:

1^0. $S \rightarrow \{\mathtt{A}\ (C)\}\ ABC$

2^0. $A \rightarrow a$

3^0. $B \rightarrow b$

4^0. $C \rightarrow \phi$

A ação adaptativa $\mathtt{A}$ é definida como:

$\mathtt{A}\ (x) = \{ +[x \rightarrow c\ ]\ \}$

Os conjuntos de símbolos nãoterminais e de símbolos terminais são respectivamente $V_N^0 = \{ S, A, B, C\}$ e $V_T = \{ a, b\}$

Para gerar a sentença abc, inicia-se o processo de derivação pelo símbolo inicial S.

$S \Rightarrow \{\ \mathtt{A}\ (C)\ \}\ ABC$

Ao aplicar esta derivação, a ação adaptativa $\mathtt{A}$ (C) é executada e ela adiciona a regra $C \rightarrow c$ ao conjunto de regras de produção. Assim, a gramática evolui de G^0 para G^1.

O conjunto de regras de produção fica assim:

$1^0.\ S \rightarrow \{\mathtt{A}\ (C)\}\ ABC$

$2^0.\ A \rightarrow a$

$3^0.\ B \rightarrow b$

$4^0.\ C \rightarrow \phi$

$5^1.\ C \rightarrow c$

e a derivação da sentença continua de forma convencional até a sua geração completa

$S \Rightarrow \{\ \mathtt{A}\ (C)\ \}\ ABC \Rightarrow aBC \Rightarrow abC \Rightarrow abc$

Se a ação adaptativa $\mathtt{A}$ (C) não existisse, a gramática não evoluiria de G^0 para G^1.

Com o conjunto de regras de produção da gramática G^0, teríamos a seguinte derivação da sentença:

$S \Rightarrow ABC \Rightarrow aBC \Rightarrow abC \Rightarrow \phi$

Na tentativa de substituir o símbolo não-terminal C na forma sentencial abC, a única regra que define C é a $4^0.\ C \rightarrow \phi$

A derivação resultou no meta-símbolo ϕ, que representa o conjunto vazio.

Se as produções fossem as seguintes

$1^0.\ S \rightarrow ABC$

$2^0.\ A \rightarrow \phi$

$3^0.\ B \rightarrow b$

$4^0.\ C \rightarrow c$

a derivação seria:

$S \Rightarrow ABC \Rightarrow \phi$

resultando novamente o conjunto vazio.

Isto significa que, para que a gramática adaptativa possa gerar perfeitamente a linguagem desejada, os símbolos não-terminais definidos nas regras de produção como ϕ, devem ter a precedência mínima entre as produções aplicáveis nas derivações, e devem ser alvo de alterações por parte das ações adaptativas para que possam incorporar substituições que levem à obtenção de sentenças da linguagem.

Nesta situação, as alterações devem visar a adição de regras de produção que defina o não-terminal de forma diferente de ϕ, permitindo assim, a opção por alternativas que levem a derivações completas das sentenças.

Utiliza-se também o meta-símbolo ϕ quando se removem todas as alternativas existentes de um não-terminal. Por exemplo:

Se tivéssemos o seguinte conjunto de regras de produção:

1^0. $S \rightarrow A\ A\ A$

2^0. $A \rightarrow \{S\ (a)\}\ a$

3^0. $A \rightarrow \{S\ (b)\}\ b$

4^0. $A \rightarrow \{S\ (c)\}\ c$

Onde $S\ (\alpha) = \{ - [A \rightarrow \{S\ (\alpha)\}\ \alpha]$

$+[A \rightarrow \phi]$

$\}$

e quisermos gerar a sentença abc, teríamos a seguinte seqüência de derivações:

$S \Rightarrow AAA \Rightarrow \{\ S\ (a)\ \}\ aAA$

Neste momento, a ação adaptativa $S\ (a)$ remove a regra de produção que foi aplicada, adaptando a gramática de G^0 para G^1 e resultando no seguinte conjunto de produções

1^0. $S \rightarrow A\ A\ A$

3^0. $A \rightarrow \{S\ (b)\}\ b$

4^0. $A \rightarrow \{S\ (c)\}\ c$

5^0. $A \rightarrow \phi$

A derivação continua, desta vez com a aplicação da regra 3^0. $A \rightarrow \{S\ (b)\}b$, e novamente com a execução da ação adaptativa S .

$S \Rightarrow AAA \Rightarrow \{ \ S \ (a) \} \ aAA \Rightarrow \{ \ S \ (b) \} \ abA$

A gramática se adapta novamente, passando de G^1 para G^2 e com as seguintes regras de produção:

$1^0. S \rightarrow A \ A \ A$

$4^0. A \rightarrow \{S \ (c)\} \ c$

$5^0. A \rightarrow \phi$

E por fim a derivação termina, desta vez com a aplicação da regra $4^0. A \rightarrow \{S \ (c)\}c$ e novamente com a execução da ação adaptativa S .

$S \Rightarrow AAA \Rightarrow \{ \ S \ (a) \ \} \ aAA \Rightarrow \{ \ S \ (b) \} \ abA \Rightarrow \{ \ S \ (c) \} \ abc$

A gramática se adapta novamente, passando de G^2 para G^3 e com as seguintes regras de produção:

$1^0. S \rightarrow A \ A \ A$

$5^0. A \rightarrow \phi$

Isto mostra que o meta-símbolo ϕ é utilizado como base de indução para a remoção e para a adição de regras de produção.

Notação

Por questões de conveniência, a notação adotada para especificar as gramáticas adaptativas é a notação de Wirth modificada [Jos93], assim descrita em sua própria notação:

gramatica-adaptativa-baseada-em-notação-de-Wirth-modificada = (regra \ ε).

regra = não-terminal "→"("{" adaptativa "}" | ε) expressão

| (símbolo-contexto | ε) não-terminal ("←" | "→") ("{" adaptativa "}" | ε) expressão-contexto

| ϕ .

adaptativa = id-ação "(" (ε | (argumento \ ",")) ")" .

argumento = não-terminal

| terminal.

expressão = (termo \ " | ").

expressão-contexto = (símbolo-contexto | ε) expressão

termo = (fator \ ε).

fator = não-terminal

| terminal

| “ε”

| “(” expressão (“\” expressão | ε) “)”.

Esta notação é composta dos seguintes elementos:

adaptativa: corresponde à porção adaptativa da parte direita da regra de produção.

id-ação: identificador da ação adaptativa formado por cadeias de caracteres, iniciada por uma letra maiúscula, seguido por letras, dígitos ou hífens.

Argumento são os valores passados para a ação adaptativa. São formados por terminais e não-terminais.

Terminal: cadeias de caracteres situada entre aspas duplas ou apóstrofes.

Não-terminal: cadeia de caracteres iniciadas por letra e seguido por letras, dígitos ou hífens.

Cadeia vazia: indicada por ε.

símbolo-contexto: cadeia de letras minúsculas.

Fator: elemento básico desta notação. Os fatores são: a cadeia vazia, os terminais, os não-terminais e as expressões entre parentêses, com ou sem repetição.

Um estrutura de repetição utiliza o meta-símbolo “\”, por exemplo:

(ε \ a) indica a repetição arbitrária (0 ou mais vezes) do símbolo a.

(a \ ε) indica a repetição arbitrária (1 ou mais vezes) do símbolo a.

(a \ b) significa que o símbolo a ocorre uma vez, seguido por uma repetição arbitrária (inclusive vazia) da cadeia ba.

Termo: elemento composto pela concatenação de um ou mais fatores

Expressão: elemento composto por um ou mais termos alternativos. Estes termos são separados pelo símbolo |.

Regra: define a expressão que especifica o não-terminal.

3.3. Exemplo

Para ilustrar o funcionamento da gramática adaptativa será apresentado a seguir um exemplo de linguagem sensível ao contexto especificada neste formalismo.

Um exemplo clássico de linguagem sensível ao contexto é representada pela expressão $a^n b^n c^n$. Cada sentença desta linguagem é formada por uma seqüência de terminais a's seguido por b's e por fim por uma seqüência de c's. A quantidade de símbolos de cada terminal é a mesma. Isto significa que as sentenças são do tipo abc, aaabbbccc, etc.

a. Gramática convencional:

$G = (V_N, V_T, P, S)$

$V_N = \{ S, A, C\}$

$V_T = \{ a, b, c \}$

$P = \{ \; S \rightarrow aAbc$

$S \rightarrow \varepsilon$

$A \rightarrow aAbC$

$A \rightarrow \varepsilon$

$Cb \rightarrow bC$

$Cc \rightarrow cc$

$\}$

b. Gramática adaptativa:

Intuitivamente, esta linguagem pode ser entendida como sendo constituída por três linguagens concatenadas, a^n, b^n e c^n, sendo que cada uma delas é representada por uma gramática que gera a correspondente seqüência de terminais. Por exemplo, a primeira gramática gera a seqüencia de a's, a segunda gramática gera a seqüência correspondente de b's e a última gramática gera os c's restantes.

Uma gramática linear para gerar a seqüência de a's pode ser a seguinte:

$G_A = (V_N, N_T, P, S)$ onde $V_N = \{ S, A\}$, $V_T = \{ a \}$, S é o símbolo inicial e

$P = \{ S \rightarrow A$

$A \rightarrow aA$

$A \rightarrow \varepsilon$

$\}$

Para gerar seqüências de b's e c's pode-se construir gramáticas semelhantes.

Sabemos que apenas definindo estas três linguagens e concatenando-as isto não gera a linguagem desejada, pois não teríamos como controlar a quantidade exata de elementos para cada seqüência de símbolos.

Com a gramática adaptativa, podemos controlar a quantidade exata de elementos à medida que a sentença vai sendo gerada. Isto é possível, por exemplo, com a adição de novas regras de produção enquanto se gera a seqüência de a's.

A regra de produção responsável pela geração dos a's terá uma ação adaptativa associada que permite a adição de novas regras de produção responsáveis pela geração dos b's e c's correspondentes.

As regras que definem os a's são ligeiramente diferentes dos da gramática linear definida anteriormente, e assumem a forma seguinte:

$A \rightarrow \{ \texttt{A} (A_1, B_1, C_1)\} aA_1$

$A \rightarrow \{\texttt{B} \ (B_1, C_1)\} \varepsilon$

Note-se que não existe uma recursividade sobre o símbolo A, mas a referência a um novo símbolo não-terminal A_1 cujo índice representa a quantidade de a's gerados até o momento. Para cada A_i, onde $i \geq 1$, o índice i indica a quantidade de regras adicionadas ao conjunto de regras de produção, à medida que a seqüência de a's vai sendo gerada. Isto acontece em virtude da execução de duas ações adaptativas `A` e `B` , cada uma associada a uma das regras apresentadas.

Estas duas regras que descrevem o não-terminal A significam que, quando a sentença estiver sendo gerada, ela irá gerar apenas o símbolo a ou a cadeia vazia. Esta última indica que terminou a seqüência de a's. Quando a primeira regra é aplicada, ela irá gerar o terminal a, e depois, vai ativar a execução da ação adaptativa `A`, que permite a adição de regras de produção para dar continuidade a geração dos demais a's da sentença, bem como

as regras responsáveis pela geração dos símbolos b's e c's, de acordo com a quantidade de símbolos a's gerados.

A ação adaptativa `B` é responsável pelo término da geração das seqüências: quando esta ação for executada, ela irá gerar as regras de produção cujo símbolo não-terminal gera a cadeia vazia, finalizando a sentença produzida.

Para que estas três seqüências possam ser geradas da forma correta e na ordem esperada, a raiz da gramática irá executar esta tarefa em duas etapas. Na primeira etapa, a sentença é dividida em suas partes esquerda e direita. A parte esquerda gera as seqüências de a's e de b's e a parte direita gera a seqüência de c's. Isto pode ser representado pelas seguinte conujunto de produções:

$S \rightarrow EC$

$E \rightarrow AB$

O não-terminal E vai dar origem às seqüências de a's e b's, enquanto que o não-terminal C vai dar origem à seqüência de c's.

A gramática adaptativa que representa esta linguagem é:

$G = (G^0, T, F^0)$, onde

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0 , S)$

$V_N^0 = \{ S, E, A, A_1, B, B_1, C, C_1\}$

$V_T = \{ a, b, c \}$

$V_C = \phi$

Os conjuntos de regras de produção são

$P_C^0 = \phi$

$P_L^0 = \{ \quad S \rightarrow EC$

$E \rightarrow AB$

$A \rightarrow \{A \ (A, A_1, B_1, C_1)\} \ aA_1$

$A \rightarrow \{B \quad (B_1, C_1)\} \ \varepsilon$

$B \rightarrow B_1$

$C \rightarrow C_1$

$A_1 \rightarrow \phi$

$B_1 \rightarrow \phi$

$C_1 \rightarrow \phi$

}

O conjunto de ações adaptativas é

T = {

A (t, x , y, z) = { x'*, y'*, z'*:

+ [x → {A (x, x', y', z')} ax']

+ [x → {B (y', z')} ε]

+ [y → by']

+ [z → cz']

+ [x' → ϕ]

+ [y' → ϕ]

+ [z' → ϕ]

– [t → { A (t, x, y, z)} ax]

+ [t → ax]

- [t → {B (y', z')}ε]

};

B (x, y) = { +[x → ε]

+ [y → ε]

}

}

Para mostrar como é feita a geração das sentenças será apresentado um exemplo de como a gramática adaptativa gera a seguinte sentença ω = aaabbbccc

Para referenciar mais facilmente as regras de produção, iremos enumerá-las utilizando um índice no canto superior direito para indicar qual foi a gramática adaptativa que deu origem a tal produção.

Parte-se da gramática adaptativa inicial G_0.

G^0:

1^0. $S \rightarrow E\ C$

2^0. $E \rightarrow AB$

3^0. $A \rightarrow \{\mathtt{A}\ (A, A_1, B_1, C_1)\}\ aA_1$

4^0. $A \rightarrow \{\mathtt{B}\ \ (B_1, C_1)\}\ \varepsilon$

5^0. $B \rightarrow B_1$

6^0. $C \rightarrow C_1$

7^0. $A_1 \rightarrow \phi$

8^0. $B_1 \rightarrow \phi$

9^0. $C_1 \rightarrow \phi$

Aplica-se as derivações seguintes:

1. (regra 1^0.) $S \rightarrow E\ C$
2. (regra 2^0) $E \rightarrow AB$
3. (regra 3^0) $A \rightarrow \{\mathtt{A}\ (A, A_1, B, C)\}\ aA_1$

Neste ponto, a regra 3^0. foi aplicada, e a ação adaptativa é instanciada da seguinte maneira:

$\mathtt{A}\ (A, A_1, B_1, C_1) = \{ A_2^*, B_2^*, C_2^*$:

$+ [A_1 \rightarrow \{\mathtt{A}\ (A_1, A_2, B_2, C_2)\}\ aA_2\ ]$

$+ [A_1 \rightarrow \{\mathtt{B}\ \ (B_2, C_2)\}\ \varepsilon]$

$+ [B \rightarrow b\ B_2\ ]$

$+ [C \rightarrow c\ C_2\ ]$

$+ [\ A_2 \rightarrow \phi\ ]$

$+ [B_2 \rightarrow \phi\ ]$

$+ [C_2 \rightarrow \phi\ ]$

$- [A \rightarrow \{\mathtt{A}\ (A, A_1, B_1, C_1)\}\ aA_1\ \ ]$

$+ [A \rightarrow aA_1]$

$- [A \rightarrow \{\mathtt{B}\ \ (B_1, C_1)\}\ \varepsilon]$

}

A nova gramática fica assim:

$G^1 = (V_N^1, V_T, V_C, P_L^1, P_C^1, S)$

$P_L^1 = P_L^0 - \{ \; A \rightarrow \{A (A, A_1, B_1, C_1)\} \; aA_1$

$A \rightarrow \{B \; (B_1, C_1)\} \; \varepsilon$

$\}$

$\cup \{ A \rightarrow aA_1 \}$

$\cup \{ \; A_1 \rightarrow \{A (A_1, A_2, B_2, C_2) \} \; aA_2$

$A_1 \rightarrow \{B \; (B_2, C_2)\} \; \varepsilon$

$B \rightarrow b \; B_2$

$C \rightarrow c \; C_2$

$A_2 \rightarrow \phi$

$B_2 \rightarrow \phi$

$C_2 \rightarrow \phi$

$\}$

$V_N^1 = V_N^0 \cup \{ A_2, B_2, C_2 \}$

Enumerando as novas produções adicionadas, que foram geradas pela ação adaptativa, temos:

$1^0. \; S \rightarrow E \; C$

$2^0. \; E \rightarrow AB$

$3^1. \; A \rightarrow aA_1$

$5^0. \; B \rightarrow B_1$

$6^0. \; C \rightarrow C_1$

$7^0. \; A_1 \rightarrow \phi$

$8^0. \; B_1 \rightarrow \phi$

$9^0. \; C_1 \rightarrow \phi$

$10^1. \; A_1 \rightarrow \{A (A_1, A_2, B_2, C_2) \} \; aA_2$

$11^1. \; A_1 \rightarrow \{B \; (B_2, C_2)\} \; \varepsilon$

$12^1.B_1 \rightarrow b\ B_2$

$13^1\ C_1 \rightarrow c\ C_2$

$14^1.A_2 \rightarrow \phi$

$15^1.\ B_2 \rightarrow \phi$

$16^1.C_2 \rightarrow \phi$

Continuando a geração da sentença, temos novamente a aplicação de uma regra de produção adaptativa.

4. (regra 10^1.) $A_1 \rightarrow \{\mathtt{A}\ (A_1, A_2, B_2, C_2)\}\ aA_2$

A partir deste ponto, omitiremos os detalhes da execução da ação adaptativa irrelevantes para o entendimento do leitor. No entanto, manteremos a apresentação das novas gramáticas que forem sendo geradas.

A nova gramática assim obtida é:

$G^2 = (\ V_N^2, V_T, V_C, P_L^2, P_C^2,\ S)$

$P_L^2 = P_L^1 - \{\ A_1 \rightarrow \{\mathtt{A}\ (A_1, A_2, B_2, C_2)\}\ aA_2$

$A_1 \rightarrow \{\mathtt{B}\ (B_2, C_2)\}\ \varepsilon$

$\}$

$\cup \{\ A_1 \rightarrow aA_2\ \}$

$\cup \{\ A_2 \rightarrow \{\mathtt{A}\ (A_2, A_3, B_3, C_3)\}\ aA_3$

$A_2 \rightarrow \{\mathtt{B}\ (B_3, C_3)\}\ \varepsilon$

$B_2 \rightarrow b\ B_3$

$C_2 \rightarrow c\ C_3$

$A_3 \rightarrow \phi$

$B_3 \rightarrow \phi$

$C_3 \rightarrow \phi$

$\}$

$V_N^2 = V_N^1 \cup \{\ A_3, B_3, C_3\ \}$

17^2. $A_2 \rightarrow \{\mathtt{A}\ (A_2, A_3, B_3, C_3)\}\ aA_3$

18^2. $A_2 \rightarrow \{\mathtt{B}\ \ (B_3, C_3)\}\ \varepsilon$

19^2. $B_2 \rightarrow b\ B_3$

20^2.$C_2 \rightarrow c\ C_3$

21^2. $A_3 \rightarrow \phi$

22^2.$B_3 \rightarrow \phi$

23^2. $C_3 \rightarrow \phi$

O próximo passo da derivação é:

5. (regra 17^2.) $A_2 \rightarrow \{\mathtt{A}\ (A_2,A_3, B_3, C_3)\}\ aA_3$

A nova gramática:

$G^3 = (\ V_N{}^3, V_T, V_C, P_L{}^3, P_C{}^3,\ S)$

$P_L{}^3 = P_L{}^2 - \{\quad A_2 \rightarrow \{\mathtt{A}\ (A_2, A_3, B_3, C_3)\}\ aA_3$

$A_2 \rightarrow \{\mathtt{B}\ \ (B_3, C_3)\}\ \varepsilon$

$\}$

$\cup\ \{\ A_2 \rightarrow aA_3\ \}$

$\cup\ \{\quad A_3 \rightarrow \{\mathtt{A}\ (A_3, A_4, B_4, C_4)\}\ aA_4$

$A_3 \rightarrow \{\mathtt{B}\ \ (B_4, C_4)\}\ \varepsilon$

$B_3 \rightarrow b\ B_4$

$C_3 \rightarrow c\ C_4$

$A_4 \rightarrow \phi$

$B_4 \rightarrow \phi$

$C_4 \rightarrow \phi$

$\}$

$V_N{}^3 = V_N{}^2\ \cup\ \{\ A_4, B_4, C_4\ \}$

Enumerando as novas regras, temos:

24^3. $A_3 \rightarrow \{\mathtt{A}\ (A_3, A_4, B_4, C_4)\}\ aA_4$

25^3. $A_3 \rightarrow \{\mathtt{B}\ (B_4, C_4)\}\ \varepsilon$

26^3. $B_3 \rightarrow b\ B_4$

27^3. $C_3 \rightarrow c\ C_4$

28^3.$A_4 \rightarrow \phi$

29^3. $B_4 \rightarrow \phi$

30^3. $C_4 \rightarrow \phi$

Continuando a derivação, temos o seguinte passo:

6. (regra 25^3.) $A_3 \rightarrow \{\mathtt{B}\ (B_4, C_4)\}\ \varepsilon$

Esta ação adaptativa é aplicada da seguinte forma:

$$\mathtt{B}\ (B_4, C_4) = \{ + [B_4 \rightarrow \varepsilon] + [C_4 \rightarrow \varepsilon] \}$$

A nova gramática:

$G^4 = (V_N^4, V_T, P^4, S, A)$

$$P^4 = P^3 \cup \{ B_4 \rightarrow \varepsilon,\ C_4 \rightarrow \varepsilon \}$$

$V_N^4 = V_N^3$

Enumerando as novas regras, temos:

31^4. $B_4 \rightarrow \varepsilon$

32^4. $C_4 \rightarrow \varepsilon$

O restante da geração da sentença é apresentado a seguir:

7. (regra 5^0.) $B \rightarrow B_1$

8. (regra 12^1.)$B_1 \rightarrow b\ B_2$

9. (regra 19^2.) $B_2 \rightarrow b\ B_3$

10. (regra 26^3.) $B_3 \rightarrow bB_4$

11. (regra 31^4.) $B_4 \rightarrow \varepsilon$

12. (regra 6^0.) $C \rightarrow C_1$

13. (regra 13^1.) $C_1 \rightarrow c\ C_2$

14. (regra 20^2.) $C_2 \rightarrow c\ C_3$

15. (regra 27^3.) $C_3 \rightarrow cC_4$

16 (regra 32^4.) $C_4 \rightarrow \varepsilon$

Assim, a derivação completa da cadeia aaabbbccc pode ser escrito como segue:

$S \Rightarrow_{G^0} ED \Rightarrow_{G^0} ABC \Rightarrow_{G^0} \{\ A\ (A, A_1, B, C)\ \}\ aA_1BC \Rightarrow_{G^1} \{\ A\ (A_1, A_2, B_2, C_2)\ \}\ aaA_2BC$

$\Rightarrow_{G^2} \{\ A\ (A_2, A_3, B_3, C_3)\ \}\ aaaA_3BC \Rightarrow_{G^3} \{B\ \ (B_4, C_4)\ \}\ aaa\varepsilon BC \Rightarrow_{G^4} aaaB_1C \Rightarrow_{G^4}$

$aaabB_2C \Rightarrow_{G^4} aaabbB_3C \Rightarrow_{G^4} aaabbbB_4C \Rightarrow_{G^4} aaabbb\varepsilon C \Rightarrow_{G^4} aaabbbC_1 \Rightarrow_{G^4} aaabbbcC_2 \Rightarrow_{G^4}$

$aabbbccC_3 \Rightarrow_{G^4} aaabbbcccC_4 \Rightarrow_{G^4} aaabbbccc\varepsilon$

3.4 Equivalência da gramática adaptativa com o autômato adaptativo

Lema3.1: O resultado da execução de qualquer ação adaptativa é equivalente a uma seqüência não vazia de ações adaptativas elementares

Sem perda de generalidade, considerando que uma ação adaptativa qualquer executa, a despeito do valor assumido por seus argumentos, as mesmas ações adaptativas elementares (pois os parâmetros só servem para instanciar as ações elementares declaradas na função adaptativa e isto não influencia o comportamento da função adaptativa quanto às ações elementares que ele executa), podemos especificar o resultado da execução de qualquer ação adaptativa em termos da seguinte gramática na notação de Wirth modificada.

ação = (ação | ε)(ação_elementar \ ε) (ação | ε)

onde

ação_elementar = consulta | remoção | adição

defatorando esta expressão, temos

ação = (ação) (ação_elementar\ ε) (ação)

| (ação) (ação_elementar\ ε)

| (ação_elementar\ ε) (ação)

| (ação_elementar\ ε)

pela fórmula de eliminação de recursões (ver 4.3), temos:

ação = (ε \ (ação_elementar \ ε)) (ação_elementar \ ε) (ε \ (ação_elementar \ ε)) \
(ação_elementar \ ε)

= (ação_elementar \ ε) \ (ação_elementar \ ε)

= (ação_elementar \ ε)

□

Para facilitar o mapeamento da gramática adaptativa para o autômato adaptativo será apresentada a seguir uma forma de normalização da gramática, que será utilizado mais adiante, na demonstração do teorema da equivalência entre os dois formalismos

Forma normal das gramáticas adaptativas

Seja G = (G^0, T, R^0) uma gramática adaptativa, onde T é o conjunto de ações adaptativas, G^0 = ($V_N{}^0$, V_T, V_C, $P_L{}^0$, $P_D{}^0$, S) a gramática inicial que implementa G e R^0 é a relação que associa regras de produção às ações adaptativas.

Vamos contruir uma nova gramática G_N = ($G^{0'}$, T', $R^{0'}$) que é a gramática G normalizada, onde $G^{0'}$= ($V_N{}^{0'}$, V_T', V_C', $P_L{}^{0'}$, $P_D{}^{0'}$, S) é a nova gramática inicial que implementa G_N, T' é o seu conjunto de ações adaptativas e $R^{0'}$ é a relação que associa suas regras de produção com as ações adaptativas.

Para simplificar o tratamento, caso a raiz da gramática S seja recursiva, criamos uma produção S' → S, garantindo que todas as raizes da gramática não sejam auto-recursivas.

Tomemos uma regra de produção com a seguinte forma:

A→ {A } a_1 a_2....a_n

a.) quando n = 0 ou seja A→ ε a forma normal correspondente é A→ {A } ε

b.) quando n>0, a forma normal correspondente é a seguinte:

1. A→ {A '} a_1 A_1
2. α_1 A_1 → a_2 A_2

.....

n. α_{n-1} A_{n-1} → a_n A_n

n+1. α_n A_n ← α A_{n+1}

n+2. A_{n+1} → ε

com $\alpha \in V_C$,

$\alpha_i \in V_C$ se $a_i \in V_N$, e

$\alpha_i = \varepsilon$ se $\alpha_i \in V_T$

A ' é a ação adaptativa normalizada equivalente a A . Se A = ε, então A '= ε

Se $a_1 \in V_N$, então a primeira regra do conjunto de regras normalizadas é alterada para

0. $A \rightarrow \{A'\} A_0$

e a regra 1 passa a ficar da seguinte forma:

1. $A_0 \rightarrow a_1 A_1$

Se A for o símbolo inicial da gramática, então não existe a regra n+1, pois não é necessário inserir o símbolo de contexto para o símbolo inicial. As últimas regras normalizadas passam a ficar da seguinte forma:

n. $\alpha_{n-1} A_{n-1} \rightarrow a_n A_n$

n+1. $\alpha_n A_n \rightarrow \varepsilon$

A normalização de produções do tipo $X \rightarrow (\{A_1\} A_1 \setminus \{A_2\} A_2)$ é feita da seguinte forma:

$X \rightarrow \{A_1'\} A_1 X_1$

$\alpha_1 X_1 \rightarrow X_2$

$X_2 \rightarrow \{A_2'\} A_2 X_3$

$\alpha_2 X_3 \rightarrow X$

$X_2 \rightarrow \varepsilon$

com $\alpha_i \in V_C$ se $A_i \in V_N$, e

$\alpha_i = \varepsilon$ se $A_i \in V_T$

A_i' é a ação adaptativa normalizada equivalente a A_i. Se $A_i = \varepsilon$, então $A_i' = \varepsilon$

Se cada A_i for uma expressão $(V_N \cup V_T)^*$ então aplicar as regras a.) e b.) apresentadas anteriormente.

A normalização de produções do tipo $X \rightarrow (\{A_1\} A_1 \mid \{A_2\} A_2 \mid .... \mid \{A_n\} A_n)$ é feita da seguinte forma:

$X \rightarrow \{ A_1' \} A_1 X_1$

$\alpha_1 X_1 \rightarrow X_2$

$X \rightarrow \{A_2'\} A_2 X_1$

$\alpha_2 X_1 \rightarrow X_2$

.....

$X \rightarrow \{A_n'\} A_n X_1$

$\alpha_n X_1 \rightarrow X_2$

$X_2 \rightarrow \varepsilon$

com $\alpha_i \in V_C$ se $A_i \in V_N$, e

$\alpha_i = \varepsilon$ se $A_i \in V_T$

A_i' é a ação adaptativa normalizada equivalente a A_i. Se $A_i = \varepsilon$, então $A_i' = \varepsilon$

Se cada A_i for uma expressão $(V_N \cup V_T)^*$ então aplicar as regras a.) e b.) apresentadas anteriormente.

Exemplo:

Seja $G = (G^0, T, R^0)$ uma gramática adaptativa em que:

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$ é a gramática inicial que implementa G, R^0 é a relação que associa regras de produção às ações adaptativas, T é o conjunto de ações adaptativas. Para simplificar, neste exemplo, tomaremos $T = \phi$.

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, X)$

X é o símbolo inicial da gramática

$V_N^0 = \{ X, Y, Z\}$

$V_T = \{ a, b, c, d, e\}$

$V_C = \phi$

$P_L^0 = \{ \; X \rightarrow a\ Y\ b\ Z$

$Y \rightarrow c\ d\ a$

$Z \rightarrow d\ e$

$\}$

$P_D^0 = \phi$

Para normalizar esta gramática, tomaremos cada uma das regras de produção e as

transformaremos nas expressões conforme foi exemplificado anteriormente.

Comecemos com a regra

$X \rightarrow a\ Y\ b\ Z$

As novas expressões que definem o não-terminal X são as seguintes:

$$X \rightarrow a\ X_1$$

$$X_1 \rightarrow Y\ X_2$$

$$y\ X_2 \rightarrow b\ X_3$$

$$X_3 \rightarrow Z\ X_4$$

$$z\ X_4 \rightarrow X_5$$

$$X_5 \rightarrow \varepsilon$$

Em seguida, transformaremos a regra

$Y \rightarrow c\ d\ a$

As novas expressões que definem o não-terminal Y serão as seguintes:

$$Y \rightarrow c\ Y_1$$

$$Y_1 \rightarrow d\ Y_2$$

$$Y_2 \rightarrow a\ Y_3$$

$$Y_3 \leftarrow y\ Y_4$$

$$Y_4 \rightarrow \varepsilon$$

E por fim, transformaremos a última regra

$Z \rightarrow d\ e$

As novas expressões que definem o não-terminal Z serão:

$$Z \rightarrow d\ Z_1$$

$$Z_1 \rightarrow e\ Z_2$$

$$Z_2 \leftarrow z\ Z_3$$

$$Z_3 \rightarrow \varepsilon$$

Para mostrar como estas regras são utilizadas, vamos derivar a seguinte sentença: ω = a c d a b d e:

Obs: Estão sublinhados na derivação a seguir os símbolos que serão substituídos no próximo passo da derivação.

$X \Rightarrow a\ \underline{X_1} \Rightarrow a\ \underline{Y}\ X_2 \Rightarrow a\ c\ \underline{Y_1}\ X_2 \Rightarrow a\ c\ d\ \underline{Y_2}\ X_2 \Rightarrow a\ c\ d\ a\ \underline{Y_3}\ X_2 \Rightarrow a\ c\ d\ a\ y\ \underline{Y_4}\ X_2 \Rightarrow a\ c$

d a $\underline{y\ \varepsilon\ X_2} \Rightarrow$ a c d a b $\underline{X_3} \Rightarrow$ a c d a b $\underline{Z}\ X_4 \Rightarrow$ a c d a b d $\underline{Z_1}\ X_4 \Rightarrow$ a c d a b d e $\underline{Z_2}\ X_4 \Rightarrow$ a c d a b d e z $\underline{Z_3}\ X_4 \Rightarrow$ a c d a b d e $\underline{z\ \varepsilon\ X_4} \Rightarrow$ a c d a b d e $\underline{X_5} \Rightarrow$ a c d a b d e ε

Normalização das ações adaptativas

As ações adaptativas também devem ser normalizadas.

Pelo lema 3.1, apresentado anteriormente, as ações adaptativas poderiam ser escritas como uma seqüência de ações adaptativas elementares.

A normalização de ações adaptativas é feita de forma semelhante à das regras de produção anteriormente apresentadas. No entanto, os não-terminais intermediários são variáveis que serão preenchidas à medida que os argumentos forem sendo passados. Por exemplo:

Se tivermos uma ação adaptativa que contenha ações elementares de consulta, de remoção e de adição, e sendo ela da seguinte maneira representada:

A (x, y) = { ? [x → a b c]

- [y → d e]

+[z → f]

}

A normalização desta ação fica da seguinte maneira:

B (x, y) = { $u_1, u_2, u_3, u_4, v_1, v_2, v_3, t_1, t_2$:

? [x → a u_1]

? [u_1 → b u_2]

? [u_2 → c u_3]

? [u_3 ← x' u_4]

? [u_4 → ε]

- [y → d v_1]

- [v_1 → e v_2]

- [v_2 ← y' v_3]

- [v_3 → ε]

+[z → f t_1]

+[t_1 ← z' t_2]

+[t_2 → ε]

}

As produções de contexto, da forma

$A \leftarrow \alpha B$

$\alpha A \rightarrow B$

$\alpha A \leftarrow \beta B$

$\alpha A \rightarrow bB$

não mudam, pois são assim definidas, somente com o lado direito da produção apresentando, no máximo, apenas dois elementos.

Teorema3.2: Se uma linguagem é gerada por uma gramática adaptativa, então ela é aceita por algum autômato adaptativo.

Seja $G = (G^0, T, R^0)$ uma gramática adaptativa, onde T é o conjunto de ações adaptativas, $G^0 = (V_N{}^0, V_T, V_C, P_L{}^0, P_D{}^0, S)$ a gramática inicial que implementa G e R^0 é a relação inicial que associa regras de produção com as ações adaptativas.

Devemos construir um autômato adaptativo $M = (E^0, A, F^0)$, tal que L(G)= L(M), onde

A é o conjunto de ações adaptativas e E^0 é o autômato de pilha estruturado inicial que implementa M e F^0 a relação inicial que associa transições com as ações adaptativas anterior e posterior.

Para construir o autômato M a partir da gramática G é conveniente que esta última seja transformada para o formato apropriado desejado, através de uma operação de normalização, conforme foi descrito anteriormente, para então efetuar-se o mapeamento da gramática para o autômato.

Com a gramática G normalizada para $G'= (G^{0'}, T', R^{0'})$, podemos aplicar as regras apresentadas para mapear todos os elementos de G' para os elementos correspondentes do autômato. Os conjuntos $G^{0'}$, T'e $R^{0'}$ são G^0, T e R^0 normalizados.

A gramática inicial que implementa a gramática adaptativa é $G^{0'} = (V_N{}^{0'}, V_T', V_C', P_L{}^{0'}, P_D{}^{0'}, S)$

Vamos construir o autômato de pilha estruturado inicial $E^0= (Q^0, SM^0, \Sigma, \Gamma, P^0, Z_0, q_0, F)$ que implementa o autômato adaptativo.

Cada elemento do conjunto dos símbolos não-terminais da gramática normalizada corresponderá a um elemento do conjunto de estados do autômato. ($V_N' = Q^0$)

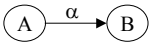
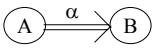
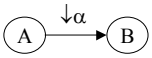
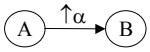
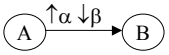
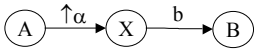
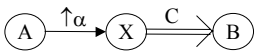
O estado inicial q_0 corresponde ao símbolo não terminal inicial S.

Cada elemento do conjunto de símbolos terminais da gramática corresponde a um elemento do vocabulário do autômato. ($V_T' \subset \Sigma$)

Cada elemento do conjunto de símbolos de contexto da gramática corresponde a um elemento do vocabulário do autômato ($V_C' \subset \Sigma$)

Os conjuntos V_T' e V_C' são disjuntos, e sua união forma o vocabulário do autômato ($V_T' \cap V_C' = \phi$ e $V_T' \cup V_C' = \Sigma$)

O mapeamento dos conjuntos de regras de produção normais e de contexto da gramática para o conjunto de regras de transições do autômato é representado de acordo com a seguinte tabela, que ilustra o tipo da regra de produção da gramática com o correspondente diagrama de máquina de estados e com a regra de transição equivalente:

1: $A \rightarrow \alpha B$, onde $\alpha \in V_T$		$(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \varepsilon)$
2: $A \rightarrow \alpha B$, onde $\alpha \in V_N$		$(\varepsilon, A, \alpha) : \rightarrow (B, \alpha_0, \varepsilon)$ onde α_0 é o estado inicial da submáquina α
3: $A \leftarrow \alpha B$, onde $\alpha \in V_C$		$(\varepsilon, A, \varepsilon) : \rightarrow (\varepsilon, B, \alpha)$
4: $\alpha A \rightarrow B$, onde $\alpha \in V_C$		$(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \varepsilon)$
5: $\alpha A \leftarrow \beta B$, onde $\alpha, \beta \in V_C$		$(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \beta)$
6: $\alpha A \rightarrow bB$, onde $\alpha \in V_C$ e $b \in V_T$		$(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, X, \varepsilon)$ $(\varepsilon, X, b) : \rightarrow (\varepsilon, B, \varepsilon)$
7: $\alpha A \rightarrow CB$, onde $\alpha \in V_C$ e $C \in V_N$		$(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, X, \varepsilon)$ $(\varepsilon, X, \varepsilon) : \rightarrow (B, C_0, \varepsilon)$ onde C_0 é o estado inicial da submáquina
8: $A \rightarrow \varepsilon$		$(X, A, \varepsilon) : \rightarrow (\varepsilon, X, \varepsilon)$ p/ $X \in$ { retornos de submáquinas chamadoras}
9: $A \rightarrow \phi$		estado inicial de submáquina

As ações adaptativas `A` da gramática também vão ser transformadas nas correspondentes ações adaptativas `B` do autômato. Pelo lema apresentado anteriormente, esta transformação é feita apenas nas ações adaptativas elementares, que também devem ser normalizadas e ficam da seguinte forma:

Nos casos seguintes o símbolo $\otimes$ representa os operadores ? ou + ou –.

As ações adaptativas `A` e são `B` opcionais, no sentido de que, existindo `A` , `B` existe, caso

contrário A e B são omitidos:

1. Para cada $\otimes[A \rightarrow \{A\ \}\ \alpha B]$ onde $\alpha \in V_T$ temos $\otimes[(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \varepsilon), B\ ]$

2. Para cada $\otimes[\ A \rightarrow \{A\ \}\ \alpha B]$ onde $\alpha \in V_N$ temos $\otimes[\ (\varepsilon, A, \varepsilon) : \rightarrow (B, \alpha, \varepsilon), B\ ]$

3. Para cada $\otimes[\ A \leftarrow \{A\ \}\ \alpha B]$ onde $\alpha \in V_C$ temos $\otimes[\ (\varepsilon, A, \varepsilon) : \rightarrow (\varepsilon, B, \alpha), B\ ]$

4. Para cada $\otimes[\ \alpha A \rightarrow \{A\ \}\ B]$ onde $\alpha \in V_C$ temos $\otimes[\ (\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \varepsilon), B\ ]$

5. Para cada $\otimes[\ \alpha A \leftarrow \{A\ \}\ \beta B]$ onde $\alpha, \beta \in V_C$ temos $\otimes[\ (\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \beta), B\ ]$

6. Para cada $\otimes[\ \alpha A \rightarrow \{A\ \}\ bB]$ onde $\alpha \in V_C$ e $b \in V_T$ temos

$\otimes[\ (\varepsilon, A, \alpha) : \rightarrow (\varepsilon, X, \varepsilon)]$

$\otimes[\ (\varepsilon, X, b) : \rightarrow (\varepsilon, B, \varepsilon), B\ ]$

7. Para cada $\otimes[\ \alpha A \rightarrow \{A\ \}\ CB]$, onde $\alpha \in V_C$ e $C \in V_N$ temos

$\otimes[\ (\varepsilon, A, \alpha) : \rightarrow (\varepsilon, X, \varepsilon)]$

$\otimes[\ (\varepsilon, X, b) : \rightarrow (B, C, \varepsilon), B\ ]$

8. Para cada $\otimes[\ A \rightarrow \{A\ \}\ \varepsilon\ ]$ temos $\otimes[\ (X, A, \varepsilon) : \rightarrow (\varepsilon, X, \varepsilon), B\ ]$

p/ $X \in$ { retornos de submáquinas chamadoras}

9. Para cada $\otimes[\ A \rightarrow \phi\ ]$ equivale ao estado inicial da submáquina A.

O autômato adaptativo em sua primeira máquina de estados E^0, inicia a execução a partir de o estado inicial q_0 equivalente a S, o símbolo inicial da gramática G^0.

Em cada passo do reconhecimento, o autômato executa alguma transição representada pelos tipos de 1 a 9.

Se a transição for adaptativa, então a ação adaptativa será executada após a aplicação da transição.

Na execução da ação adaptativa, pelo lema anterior, podemos reduzir o problema a uma seqüência de ação elementares. Assim, existem alguns casos a considerar.

Se a ação elementar da gramática for do tipo consulta, então a execução da ação elementar equivalente do autômato não altera o conjunto de transições (nada há a considerar).

Se a ação elementar adaptativa for do tipo adição então a execução de uma ação deste tipo pode:

a. acrescentar um novo não-terminal que não existia anteriormente. No autômato, esta ação acrescenta uma nova submáquina, correspondente ao não-terminal adicionado que não

existia anteriormente

b. acrescentar uma nova regra definindo uma opção para um não-terminal que já existia. No autômato, esta ação acrescenta uma nova cadeia de transições, correspondente à seqüência de símbolos contidos no lado direito da regra adicionada.

c. acrescentar uma regra já existente (neste caso, nada há a fazer)

Se a ação elementar adaptativa for do tipo remoção então:

a. eliminar uma regra inexistente (neste caso, nada há a fazer)

b. eliminar uma regra existente. No autômato deve ser eliminada toda a cadeia de transições correspondente à seqüência de símbolos indicada no lado direito da regra a ser eliminada

c. eliminar uma última regra que define um não-terminal. No autômato, devem ser eliminadas todas as transições internas da submáquina associada ao não-terminal, preservando-se apenas os seus estados inicial e final, para efeito de referência.

Após a execução de uma ação adaptativa, a configuração do autômato adaptativo evolui para a próxima máquina de estados que o implementa, seguindo o seguinte ciclo:

Enquanto nenhuma regra de transição adaptativa é aplicada, todas as transições são executadas como no caso não adaptativo livre de contexto. Quando ocorre a aplicação de uma transição adaptativa, o conjunto de regras de transição é alterado. Logo, para manter a equivalência entre os dois modelos, o autômato deve se modificar correspondentemente, passando a ser implementado por uma nova máquina de estados, que irá prosseguir executando o ciclo de operação, até que a cadeia de entrada se esgote, que um estado final seja atingido e que a pilha fique vazia (condição de aceitação). Enquanto uma das três consições não for atendida, a cadeia de entrada não será aceita.

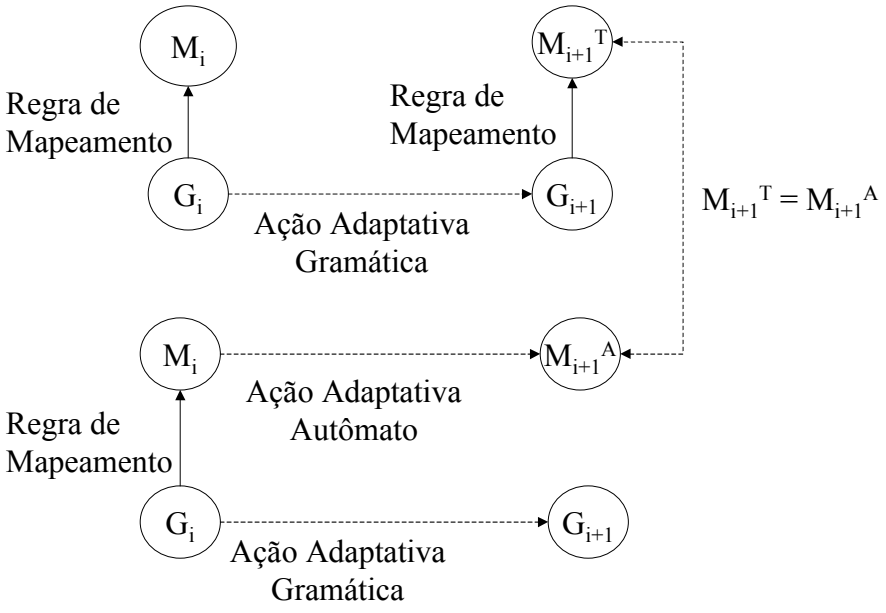
As transições do autômato assim construído foram projetadas de tal modo que a execução do reconhecimento de uma certa cadeia imita os movimentos da derivação da mesma cadeia pela gramática.

Para verificar que a linguagem gerada pela gramática adaptativa e a linguagem reconhecida pelo autômato adaptativo assim construído é a mesma, basta mostrar que quaisquer que sejam as alterações ocorridas na gramática, alterações equivalentes deverão ocorrer no autômato.

Seja G_i a gramática adaptativa produzida após i execuções de transições com ações adaptativas, e seja M_i o autômato obtido a partir desta gramática através do mapeamento da regras de produção da gramática para as regras de transição do autômato construído conforme apresentado neste teorema.

Enquanto não ocorre a aplicação de alguma regra adaptativa, a execução da sentença ocorrerá como no caso livre de contexto. No momento em que é feita a aplicação de uma regra adaptativa, a gramática evolui de G_i para G_{i+1}.

Consideraremos dois autômatos obtidos de formas diferentes e vamos mostrar que eles são os mesmos. Veja a figura seguinte



a.) O autômato $M_{i+1}{}^T$ que será considerado é obtido a partir das regras de produção da gramática G_{i+1} usando as regras de mapeamento 1 a 4, descritas anteriormente.

b.) O autômato $M_{i+1}{}^A$ que será considerado é obtido pela execução passo a passo das ações adaptativas elementares resultantes da execução da ação adaptativa associada à transição de M_i correspondente à regra de produção da gramática G_i que provocou sua evolução para

G_{i+1}

Para constatar que M_{i+1}^{T} e M_{i+1}^{A} são o mesmo autômato basta verificar que a evolução da gramática G_i para G_{i+1} ocorreu através da aplicação de uma regra de produção associada a alguma ação adaptativa `A` . No autômato M_i, a transição correspondente à regra de produção aplicada foi associada (por construção) a uma ação adaptativa `B` que por sua vez é equivalente a `A` (também por construção). Assim, à evolução de G_i para G_{i+1} corresponderá a evolução de M_i para M_{i+1} causada pela execução da transição correspondente à produção aplicada. Como a ação adaptativa `B` corresponde no autômato exatamente à ação adaptativa `A` da gramática, então as alterações de M_i para M_{i+1} corresponderão exatamente às alterações de G_i para G_{i+1}.

Assim sendo, constatamos que a aplicação de ações adaptativas preserva a equivalência entre as várias versões da gramática adaptativa e as correspondentes versões do autômato adaptativo, ou seja, que a gramática adaptativa e o correspondente autômato adaptativo representam sempre a mesma linguagem ao longo de sua operação.

□

Lema3.3: Se uma linguagem gerada por um não-terminal correspondente a uma submáquina, então essa sublinguagem pode ser definida por uma gramática cuja raiz é o não-terminal em questão.

Prova:

Seja A um não-terminal

1° caso:

Se $A \rightarrow \delta$, onde $\delta \in V_T^*$ e $A \in V_N$ então $L(A) = \{\delta\}$. Trivial.

2° caso:

Se $A \rightarrow \alpha B \beta$, onde α e $\beta \in (V_T \cup V_N)^*$ e A e $B \in V_N$, então

$L(A) = L(\alpha)\, L(B)\, L(\beta)$

3° caso:

Se $A \rightarrow \alpha A \beta$ e $A \rightarrow \gamma$ onde α, β e $\gamma \in (V_T \cup V_N)^*$ e $A \in V_N$, então

$L(A) = L(\alpha)^n L(\gamma) L(\beta)^n$, onde $n \geq 0$.

4° caso:

Se $A \rightarrow X$ e $A \rightarrow Y$, onde X e $Y \in (V_T \cup V_N)^*$ então $L(A) = L(X) \cup L(Y)$

5° caso:

Se $A \rightarrow XY$, onde X e Y $\in (V_T \cup V_N)^*$ então L (A) = L (X) L (Y).

Como a cadeia de símbolos é finita e cada uma das produções acima deriva um número finito de símbolos, então em algum momento a produção do 1° caso vai ser aplicada. Caso contrário ou a cadeia é infinita, o que é um absurdo, ou a cadeia não é formada só por terminais, o que também é um absurdo.

Logo, a sublinguagem definida por um não-terminal é formada por sentenças finitas constituídas somente por terminais

Teorema3.4: Se uma linguagem é aceita por um autômato adaptativo, então ela pode ser gerada por alguma gramática adaptativa

Prova:

Para a demostração deste teorema, vamos inicialmente efetuar uma transformação no autômato de forma a converter todas as ações adaptativas anteriores das regras de transição adaptativas em ações equivalente posteriores.

Seja Q o conjunto de estados deste autômato adaptativo antes da transformação. Seja inicialmente $Q^0 = Q$ o conjunto de estados do autômato transformada a ser construído.

Se uma transição for do tipo

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

ela será transformada, em dois passos, nas seguintes regras de transição:

$(\gamma, e, s\alpha) \rightarrow (\gamma, e'', s\alpha)$, A , onde $e'' \notin Q$ é um estado intermediário adicional, a ser acrescentado ao conjunto Q^0.

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

Se a ação adaptativa A (ou B) for do tipo auto-eliminável, então a decomposição em duas transições transforma a ação A em A ' (ou B em B ') tal que, A ' (ou B ') deve eliminar tanto a transição $(\gamma, e, s\alpha) \rightarrow (\gamma, e'', s\alpha)$, A ' como a transição $(\gamma g, e'', s\alpha) \rightarrow (\gamma g', e', s'\alpha)$, B '

Seja $M = (E^0, A, \Phi^0)$ um autômato adaptativo, onde A é o conjunto de todas as ações adaptativas usadas por M; $E^0 = (Q^0 , SM^0 , \Sigma, \Gamma, TR^0 , Z_0 , q_0 , F^0)$ é o autômato de pilha

estruturado inicial que implementa M e Φ^0 a relação que associa as transições de E^0 com as ações adaptativas anterior e posterior a elas associadas.

Devemos construir uma gramática adaptativa, $G = (G^0, T, R^0)$ tal que L(G)= L(M), onde T é o conjunto de ações adaptativas de G, $G^0 = (V_N^0, V_T, P^0, S)$ a gramática inicial que implementa G e R^0 é a relação que associa regras de produção contidas em P^0 com as ações adaptativas correspondentes do conjunto T.

O conjunto V_N^0 contém o símbolo S que representa o estado inicial do autômato ($S= q_0$); contém também os símbolos não-terminais correspondentes a cada estado inicial de submáquina; e contém ainda os símbolos não-terminais correspondentes a cada estado em Q^0.

As regras em P^0 são de 4 tipos:

1. Para cada transição interna $(\gamma, a, \sigma\alpha) \rightarrow (\gamma, b, \alpha)$, $A , \in TR^0$, onde $\sigma \in \Sigma \cup \{\varepsilon\}$ criamos uma produção

$$a' \rightarrow \{B\}\ \sigma b' \quad \text{para } \sigma \in V_T, \text{ ou então}$$

$$\sigma a' \rightarrow \{B\}\ b' \quad \text{para } \sigma \in V_C$$

sendo a' e b' não-terminais correspondentes respectivamente aos estados origem a e destino b da transição e com $B = \varepsilon$ se $A = \varepsilon$, e $B \in A$ se $A \in T$. A relação entre A e B está descrita adiante.

2. Para cada transição interna $(\gamma, a, \sigma\alpha) \rightarrow (\gamma, b, \sigma'\alpha)$, $A , \in TR^0$, onde $\sigma \in \Sigma \cup \{\varepsilon\}$ e $\sigma' \in \Sigma$ criamos duas produções

$$a' \rightarrow \sigma c$$

e

$$c \leftarrow \{B\}\ \sigma' b'$$

sendo a' e b' não-terminais correspondentes respectivamente aos estados origem a e destino b da transição e com $B = \varepsilon$ se $A = \varepsilon$, e $B \in A$ se $A \in T$ e c é um não-terminal auxiliar injetado por causa da alteração da cadeia.

3. Para cada chamada de submáquina $(\gamma, a, \alpha) \rightarrow (\gamma b, N_0, \alpha)$, $\in TR^0$, onde N_0 é o estado inicial da submáquina N, criamos uma produção

$$a' \rightarrow N_0'b'$$

N_0' representa neste caso um não-terminal correspondente ao estado N_0 da submáquina N. Associadas a cada submáquina existem transições de retorno às submáquinas chamadoras, do tipo $(\gamma b, N_i, \alpha) \rightarrow (\gamma, b, \alpha)$, onde N_i são estados finais da submáquina associada ao não terminal N. Para cada uma delas, cria-se uma produção

$$N_i' \rightarrow \varepsilon$$

4. Para cada $a \in F^0$, criamos uma produção

$$a' \rightarrow \varepsilon$$

As ações adaptativas `A` do autômato também vão ser transformadas nas ações adaptativas correspondentes `B` da gramática. Usando o mesmo esquema de correspondência empregado anteriormente, temos:

Para cada ação adaptativa elementar da forma $\otimes$ [$(\gamma g, e, \sigma\alpha) \rightarrow (\gamma g', e', \sigma'\alpha)$, `B`], pertencente ao corpo de cada uma das ações adaptativas do autômato, onde $\otimes \in \{?, +, -\}$, temos a consioderar os seguintes casos, correspondentes às transformações acima descritas:

1.Para ações elementares da forma $\otimes$ [$(\gamma, a, \sigma\alpha) \rightarrow (\gamma, b, \alpha)$, `A`] no autômato, criar a ação correspondente na gramática:

$$\otimes [a' \rightarrow \{\texttt{B}\ \} \sigma b'] \text{ para } \sigma \in V_T, \text{ ou então}$$

$$\otimes\ [\sigma a' \rightarrow \{\texttt{B}\ \} b']\ \text{ para }\ \sigma \in V_C$$

2.Para ações elementares da forma $\otimes$ [$(\gamma, a, \sigma\alpha) \rightarrow (\gamma, b, \sigma'\alpha)$, `A`] no autômato, (envolvendo empilhamento de símbolo na cadeia de entrada do autômato) criar o seguinte par de ações elementares correspondentes na gramática, nas quais σ' opera como símbolo de contexto:

$$\otimes [a' \rightarrow \sigma c\ ]$$

e

$$\otimes [c \leftarrow \{\texttt{B}\ \} \sigma' b']$$

Sendo c um não-terminal auxiliar que não existia anteriormente, é preciso declará-lo como

um gerador, na ação adaptativa da gramática.

3 Para cada ação elementar da forma $\otimes [(\gamma, a, \alpha) \rightarrow (\gamma b, N_0, \alpha)]$ no autômato, (envolvendo transições de chamada de submáquina) criar na gramática a ação elementar correspondente:

$$\otimes [a' \rightarrow N_0 'b']$$

4. Para ações elementares da forma $\otimes [(\gamma b, N_i, \alpha) \rightarrow (\gamma, b, \alpha)]$ no autômato, (envolvendo transições de retorno da submáquina) criar na gramática a ação correspondente:

$$\otimes[N_i' \rightarrow \varepsilon]$$

Devemos mostrar que a gramática assim construída imita os movimentos do autômato, gerando portanto as mesmas sentenças por ele reconhecidas, e nada mais.

Vamos provar essa afirmação por indução, no comprimento da derivação e no comprimento da trajetória do autômato, que a linguagem gerada pela gramática e a linguagem reconhecida pelo autômato são a mesma.

Base da indução

Seja N um estado final do autômato e seja N' o não-terminal correspondente ao estado N, na gramática

Se o autômato estiver no estado final N então, por construção, a produção correspondente na gramática é $N' \rightarrow \varepsilon$. Assim,

$L(N) = \{\varepsilon\}$ e $L(N') = \{\varepsilon\}$

Portanto, $L(N) = L(N')$

Hipótese de indução

Suponhamos que $L(q_0, j)$ seja um conjunto de todos os prefixos reconhecidos pelo autômato partindo do estado q_0 e chegando ao estado j.

Suponhamos também que $L(S, j')$ gera o mesmo conjunto de prefixos α^*, onde $\alpha \in V_T$ e j' é o não-terminal da gramática correspondente ao estado j do autômato

Passo indutivo

Vamos provar que, após o próximo movimento do autômato e da gramática as respectivas linguagens reconhecida e gerada continuam sendo iguais.

Se o autômato for não-determinístico, então poderá existir mais de uma transição possível a ser percorrida pelo autômato a partir do estado j.

Tomando uma transição t, dentro do conjunto de possíveis transições a serem percorridas, alguns casos poderão ocorrer

a.) a tentativa do reconhecimento da sentença a partir desta transição t falhou. Neste caso, toma-se uma outra possível transição até atingir alguma transição cujo reconhecimento seja bem sucedido.

b.) a tentativa do reconhecimento da sentença a partir desta transição t (e somente desta transição) foi bem sucedida.

c.) a tentativa do reconhecimento sentença a partir desta transição t e de outra transição u, foram bem sucedidas. Neste caso, ocorreu uma ambiguidade.

Tendo em vista estas três situações, haverá um conjunto de transições válidas para o reconhecimento da sentença.

Dando prosseguimento a demonstração do teorema, tomaremos um estado k, que é alcançado por uma transição válida que parte de j.

Seja δ a subcadeia reconhecida pelo autômato a partir do estado j e chegando ao estado k.

Seja δ' a subcadeia acrescentada ao prefixo α^* , onde $\alpha \in V_T$, da forma sentencial derivada pela gramática a partir da substituição do não-terminal j.

Vamos provar que $\delta = \delta$'.

Temos alguns casos a analisar.

a.) Suponhamos que $(\gamma, j, \sigma\alpha) \rightarrow (\gamma, k, \alpha)$ seja uma transição do autômato onde $\sigma = \varepsilon$. A subcadeia reconhecida por esta transição é $\sigma = \varepsilon$, logo $\delta = \sigma = \varepsilon$.

No caso da gramática, suponhamos que seja aplicada a regra de produção $j \rightarrow \sigma k'$, onde $\sigma = \varepsilon$ e k' é o não terminal correspondente ao estado k. Logo $j \rightarrow \varepsilon k$, isto é $j \rightarrow k$. Portanto, $\delta' = \varepsilon$, neste primeiro caso.

Concluimos que $\delta = \delta' = \varepsilon$.

b.) Suponhamos que $(\gamma, j, \sigma\alpha) \rightarrow (\gamma, k, \alpha)$ seja uma transição do autômato onde $\sigma \in V_T$, $\sigma \neq \varepsilon$. A subcadeia reconhecida por esta transição é σ, logo $\delta = \sigma$.
No caso da gramática, suponhamos que seja aplicada a regra de produção $j \rightarrow \sigma k'$, onde $\sigma \in V_T$, $\sigma \neq \varepsilon$ e k' é o não terminal correspondente ao estado k.
Logo $j \rightarrow \sigma k$. Portanto, $\delta' = \sigma$
Concluimos que $\delta = \delta' = \sigma$.

c.) Suponhamos que $(\gamma, j, \alpha) \rightarrow (\gamma k, A_0, \alpha)$ seja transição do tipo chamada de submáquina, onde A_0 é o estado inicial da submáquina A.
No caso da gramática, suponhamos que seja aplicada a correspondente regra de produção $j \rightarrow A'k$, onde $A' \in V_N$ e A' é o não terminal correspondente ao estado A.
O δ acrescentado à cadeia reconhecida pelo autômato deve ser um δ pertencente a linguagem reconhecida pela submáquina A. δ é deduzido através da aplicação recursiva deste teorema como se $A_0 = q_0$, onde A_0 é o símbolo inicial da submáquina e q_0 é o símbolo inicial do autômato e a cadeia de entrada iniciando-se no próximo símbolo a ser consumido pelo autômato.
O δ' acrescentado ao prefixo da cadeia gerada pela gramática é o contorno da árvore montada tendo A' como raiz desta árvore e com as folhas todas formadas com terminais.
Pelo lema 3.3, $\delta' = \mathtt{L}(A')$.
Portanto $\delta = \delta'$.

Nos casos a.) e b.) pode haver ainda uma ação adaptativa associada às regras de produção da gramática e às regras de transição do autômato.
Nestes casos, após a operação indicada acima, deverão ser feitas as alterações nos conjuntos de transições do autômato, ou de produções da gramática.
Usando o lema 3.1, que diz que "qualquer ação adaptativa é equivalente a uma seqüência finita não vazia de ações elementares", temos 3 casos a considerar, um para cada tipo de ação adaptativa elementar executada.

1.) Ação de consulta

Neste caso nada há a fazer, já que estas ações não alteram a gramática nem o autômato.

2.) Ação de remoção

Seja a ação elementar considerada

$$-[\ (\gamma, i, \sigma\alpha) \rightarrow (\ \gamma, j, \sigma'\alpha), A\ \ ]$$

Observação 1: O autômato foi previamente transformado de forma tal que não apresenta ações anteriores, somente ações posteriores.

Observação 2: Não existem ações adaptativas associadas às transições de chamada e retorno de submáquinas.

No autômato, esta ação elementar remove a transição que leva do estado i ao estado j.

Temos aqui, dois casos a considerar

1° caso: Quando $\sigma' = \varepsilon$.

Na gramática, a regra de produção é a seguinte:

$$-[\ i \rightarrow \{A\ \ \}\ \sigma j]$$

esta ação elementar é equivalente, por construção, à ação elementar do autômato.

No autômato, a transição removida é (lembrando que $\sigma' = \varepsilon$)

$$(\gamma, i, \sigma\alpha) \rightarrow (\ \gamma, j, \alpha), A$$

Na gramática, a produção correspondente removida é

$$i \rightarrow \{A\ \}\ \sigma j$$

Estas duas regras são equivalentes, por construção.

Não existindo mais a transição no autômato, e não existindo mais a produção correspondente na gramática, e preservando todas as demais, a nova linguagem aceita pelo autômato e a nova linguagem gerada pela gramática deverão ser a mesma, por construção.

2° caso: Quando $\sigma' \neq \varepsilon$.

Na gramática, temos o correspondente par de regras de produção a considerar:

$$-[i \rightarrow \sigma j\ \ ]$$

e

$$-[\ i \leftarrow \{A\ \ \}\ \sigma' j]$$

No autômato, a transição correspondente removida é $(\gamma, i, \sigma\alpha) \rightarrow (\gamma, j, \sigma'\alpha)$, A

Na gramática, as produções removidas são

$$i \rightarrow \sigma\, i'$$

e

$$i' \leftarrow \{A\}\ \sigma' j$$

Estas três regras são equivalentes, por construção.

Não existindo mais a transição e não existindo mais o par correspondente de produções, e como todas as demais regras permanecem preservadas, a nova linguagem aceita pelo autômato e a nova linguagem gerada pela gramática deverão ser a mesma, por construção.

3.) raciocínio idêntico pode ser feito em relação à ação de acréscimo de transição em relação à ação de acréscimo da produção correspondente, concluindo a justificativa deste teorema.

□

Breve estudo informal da complexidade da gramática adaptativa

Considerando que as gramáticas que contêm ciclos de definições de não-terminais não são úteis para a geração de sentenças, deve-se eliminar inicialmente todos os ciclos da gramática antes de efetuar o estudo de sua complexidade. A eliminação dos ciclos em uma gramática pode ser feita através de algoritmos clássicos de remoção de ciclos em grafos.

Seja $G = (G^0, T, F^0)$ uma gramática adaptativa, em que $G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$ é a gramática inicial. Vamos definir $P^0 = P_L^0 \cup P_D^0$ como sendo o conjunto das regras de produção da gramática. Adotemos, como uma medida muito simples da complexidade de uma gramática adaptativa q, o comprimento desta gramática, $q = q^0$ é a quantidade de regras de produção em P^0. Isto é $q^0 = |G^0| = \text{card}(P^0)$.

Seja ω uma cadeia de comprimento n, tal que $\omega \in L(G)$.

Suponhamos que a ação adaptativa mais complexa da gramática G acrescente ao todo no máximo t regras ao conjunto P^i, para $i \geq 0$.

Consideremos como hipótese que toda derivação partindo de um não-terminal gera pelo menos 1 símbolo terminal.

Teremos então, como pior caso, a situação em que todas as regras da gramática executam a ação adaptativa mais complexa. Neste caso extremo, n * t + q^0 será a quantidade de regras de produção ao final da geração da sentença.

Teremos, como melhor caso, a situação em que nenhuma regra é adaptativa, e nesta situação t = 0. Assim, não haverá variação no número de produções, e a quantidade de regras no conjunto de produções ao final da geração da sentença será portanto q^0.

Analisando mais detalhadamente o comportamento das ações adaptativas, consideremos que em uma ação adaptativa `X` haja a chamada de duas outras ações adaptativas (uma anterior `A` e outra posterior `B`). Cada uma destas ações podem por sua vez chamar duas outras ações adaptativas, e assim por diante, gerando uma árvore binária de chamada de ações adaptativas. Seja r a máxima profundidade desta árvore.

A quantidade total de chamadas de ações adaptativas será, neste caso dada por

$$1 + 2 + 4 + 8 + ... + 2^r = 2^{r+1} - 1$$

Se cada ação adaptativa acrescenta explícitamente, no máximo, p produções ao conjunto P^i, para $i \geq 0$, teremos portanto, $p*(2^{r+1} - 1)$ produções acrescentadas ao todo, no pior caso, como resultado da chamada de `X` .

Se existirem chamadas iterativas ou recursivas da ação adaptativa, e se esta ação for executada, no pior caso, k vezes, e supondo ainda como situação de piro caso, que todas as ações adaptativas tenham este mesmo comportamento e sejam repetidas essas k vezes, teremos então $t = p*(2^{r+1} - 1)*k$ acréscimos de produções ao todo.

Se o processo a geração de uma sentença for iniciada por uma gramática adaptativa G^0, e a cada símbolo derivado na forma sentencial for gerado a partir da aplicação de uma regra de produção adaptativa que acrescenta t regras de produção à gramática, então, no processo de derivação de uma sentença de tamanho n, teremos G^0, G^1, ... G^n como gramáticas intermediárias.

Como a gramática G^i não apresenta ciclos de definição de não-terminais, pode-se considerar que, possuindo ela q^i regras ao todo, então haverá no máximo q^i não-terminais. Assim, no pior caso, a substituição de um não-terminal até que derive finalmente um símbolo da sentença deverá envolver, no pior caso, q^i substituições gramaticais para cada símbolo σ_i da sentença $\omega = \sigma_1 .... \sigma_n$.

Como limitante superior, para a pior situação possível, portanto, a geração dos n

símbolos da sentença deverá acarretar a execução de $n*q^n$ substituições ao todo.

Na realidade, a situação é ligeiramente melhor que esta, pois a geração dos símbolos mais à esquerda ocorrem para valores menores que i, ou seja, para gramáticas de comprimento menor. Tem-se então

para a obtenção de σ_1:

q^0 substituições no pior caso.

A nova gramática terá, no pior caso, q^1 regras, com $q^1 = q^0 + t$

para a obtenção de σ_2:

q^1 substituições no pior caso.

A nova gramática terá no pior caso q^2 regras, com $q^2 = q^0 + t + t$

para um símbolo genérico σ_i, tem-se q^i regras, com $q^i = q^0 + i * t$

para um símbolo genérico σ_n, tem-se q^n regras, com $q^n = q^0 + n * t$

Ao todo, para gerar $\omega = \sigma_1 \ \sigma_2 ..... \sigma_n$, tem-se

$q^1 + q^2 + ... + q^n$ substituições

ou seja,

$$(q^0 + t) + (q^0 + 2t) + ... + (q^0 + n * t) =$$

$$n * q^0 + t (1 + 2 + ... + n) =$$

$$n * q^0 + t * (n/2) (1 + n) =$$

$$n * q^0 + n* (t/2) + n^2 * (t/2) =$$

$$n^2 * (t/2) + n*(q^0 + t/2), \text{ com } t = p*(2^{r+1} – 1) *k$$

Conclui-se que, mesmo para as hipótes adversas e pouco prováveis de pior caso aqui impostas, o crescimento da gramática é linear com o comprimento da sentença, pois $q^i = q^0 + i*t$, e que o esforço computacional corresponde às substituições necessárias para a geração da sentença, em gramáticas sem ciclos, mas com regras que exijam substituições múltiplas para a obtenção de cada símbolo da sentença, é quadrático com o comprimento da sentença, pois exige $n^2 (t/2) + n(q^0 + t/2)$ substituições ao todo.

No entanto, para gramáticas projetadas de tal forma que cada substituição gere um símbolo da sentença, apenas uma substituição será feita de cada vez. Para gerar toda a sentença serão executadas n substituições. Se cada substituição for adaptativa, no pior caso, teremos n*t regras acrescentadas ao todo, o que nos indica um comportamneto linear neste caso mais realístico.

Podemos concluir que, mesmo dentro das piores hipóteses formuladas, a gramática sempre cresce em ordem quadrática, o que representa, em termos computacionais, um bom resultado. Portanto, a gramática adaptativa representa um modelo computacionalmente viável.

Capítulo 4 – Algoritmos

Este capítulo é dedicado à apresentação dos algoritmos de conversão entre gramáticas adaptativas e autômatos adaptativos. As duas primeiras seções apresentam métodos canônicos de conversão, baseados nos teoremas apresentados no capítulo anterior.

Em seguida, é feita a apresentação de um método eficiente de conversão da gramática adaptativa para o autômato adaptativo. Este método envolve a construção de um reconhecedor descendente (estilo LL) e um transdutor que converte a sentença na árvore de derivação. Este método estende o algoritmo apresentado em [Jos93] para comportar adaptabilidade, o meta-símbolo ϕ, bem como o caso especial de regra de produção dependente de contexto, envolvendo símbolos de contexto.

4.1 Algoritmo para a conversão canônica da gramática adaptativa para o autômato adaptativo

Seja dada uma gramática adaptativa $G = (G^0, T, R^0)$, onde T é o conjunto de ações adaptativas, $G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$ a gramática inicial que implementa G e R^0 é a relação que associa ações adaptativas às regras de produção.

A primeira parte do algoritmo trata da normalização da gramática, transformando a gramática G na gramática normalizada $G' = (G^{0'}, T', R^{0'})$, onde $G^{0'}= (V_N^{0'}, V_T', V_C', P_L^{0'}, P_D^{0'} , S)$. Se $S \rightarrow \alpha$ for recursivo, substitui-se por

$S \rightarrow S'$

e

$S' \rightarrow \alpha$

Passo 1: Para a regra de produção que define o símbolo inicial S, onde

$S \rightarrow \{A\}\ a_1\ a_2\ a_3\ ....\ a_n$

onde $a_i \in V$. Normalizá-la da seguinte forma:

Se $a_1 \in V_T$, então	Se $a_1 \in V_N$, então
$S \rightarrow$ {A '} $a_1\ A_1$	$S \rightarrow$ {A '} A_0
	$A_0 \rightarrow a_1\ A_1$
$\alpha_1\ A_1 \rightarrow a_2\ A_2$	$\alpha_1\ A_1 \rightarrow a_2\ A_2$
$\alpha_2\ A_2 \rightarrow a_3\ A_3$	$\alpha_2\ A_2 \rightarrow a_3\ A_3$
.....	....
$\alpha_{n-1}\ A_{n-1} \rightarrow a_n\ A_n$	$\alpha_{n-1}\ A_{n-1} \rightarrow a_n\ A_n$
$\alpha_n\ A_n \rightarrow A_{n+1}$	$\alpha_n\ A_n \rightarrow A_{n+1}$
$A_{n+1} \rightarrow \varepsilon$	$A_{n+1} \rightarrow \varepsilon$

com $\alpha_i \in V_C$ se $A_i \in V_N$, e

$\alpha_i = \varepsilon$ se $A_i \in V_T$

A ' é a ação adaptativa normalizada equivalente a A . Se A = ε, então A '= ε

Passo 2: Para cada cada elemento em P_L^0 que seja do tipo :

$A \rightarrow$ { A } $a_1\ a_2\ a_3$ .... a_n

onde $a_i \in V_T \cup V_N \cup \{\varepsilon\}$, com $1 < i \leq n$,

transformar em

Se $a_1 \in V_T$, então	Se $a_1 \in V_N$, então
$A \rightarrow$ {A '} $a_1\ A_1$	$A \rightarrow$ {A '} A_0
	$A_0 \rightarrow a_1\ A_1$
$\alpha_1\ A_1 \rightarrow a_2\ A_2$	$\alpha_1\ A_1 \rightarrow a_2\ A_2$
$\alpha_2\ A_2 \rightarrow a_3\ A_3$	$\alpha_2\ A_2 \rightarrow a_3\ A_3$
.....	.....
$\alpha_{n-1}\ A_{n-1} \rightarrow a_n\ A_n$	$\alpha_{n-1}\ A_{n-1} \rightarrow a_n\ A_n$
$A_n \leftarrow \alpha\ A_{n+1}$	$A_n \leftarrow \alpha\ A_{n+1}$
$A_{n+1} \rightarrow \varepsilon$	$A_{n+1} \rightarrow \varepsilon$

com $\alpha_i \in V_C$ se $A_i \in V_N$, e

$\alpha_i = \varepsilon$ se $A_i \in V_T$

A ' é a ação adaptativa normalizada equivalente a A . Se A = ε, então A '= ε

Passo 3: As regras de produção que envolvem símbolos de contexto não se alteram.

Normalização das ações adaptativas

No passo 4 substituir o símbolo ⊗ pelo símbolo ? ou + ou -, consistentemente

Passo 4: Para cada ação elementar do tipo ⊗ [A → a_1 a_2 a_3 .... a_n {A }], onde $a_i \in V_T$ e $a_1 \in V_T$ e $a_i \in V_T \cup V_N \cup \{\varepsilon\}$, com $1 < i \leq n$.

criar o seguinte grupo equivalente de ações elementares:

Se $a_1 \in V_T$, então	Se $a_1 \in V_N$, então
⊗ [A→ {A } a_1 A_1]	⊗ [A→ {A } A_0]
	⊗ [A_0→ a_1 A_1]
⊗ [$\alpha_1 A_1 \rightarrow a_2$ A_2]	⊗ [$\alpha_1 A_1 \rightarrow a_2$ A_2]
⊗ [$\alpha_2 A_2 \rightarrow a_3$ A_3]	⊗ [$\alpha_2 A_2 \rightarrow a_3$ A_3]
.....	.....
⊗ [$\alpha_{n-1} A_{n-1} \rightarrow a_n$ A_n]	⊗ [$\alpha_{n-1} A_{n-1} \rightarrow a_n$ A_n]
⊗ [$A_n \leftarrow \alpha$ A_{n+1}]	⊗ [$A_n \leftarrow \alpha$ A_{n+1}]
⊗ [$A_{n+1} \rightarrow \varepsilon$]	⊗ [$A_{n+1} \rightarrow \varepsilon$]

com $\alpha_i \in V_C$ se $A_i \in V_N$, e

$\alpha_i = \varepsilon$ se $A_i \in V_T$

A ' é a ação adaptativa normalizada equivalente a A . Se A = ε, então A '= ε

Correspondência entre os símbolos da gramática e os estados do autômato

Passo 5: Para S o símbolo não-terminal inicial da gramática, criar q^0, o estado inicial do autômato.

Para cada estado $q \in Q^0$, criar o correspondente não-terminal $q' \in V_N^0$.

Passo 6: Os símbolos terminais correspondem aos símbolos do vocabulário do autômato.

Passo 7: Os símbolos de contexto da gramática correspondem aos símbolos do vocabulário do autômato.

Correspondência entre as regras de produção e as regras de transição

Passo 8: Para cada regra de produção do tipo $A \rightarrow \{A\}\ \alpha B$, onde $\alpha \in V_T$, tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a regra de transição $(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \varepsilon)$, B .

As ações adaptativas A e B serão descritos no passo 17 em diante.

Passo 9: Para cada regra de produção do tipo $A \rightarrow \alpha B\ \{A\}$, onde $\alpha \in V_N$, tomar os estados A', B' e α' que são correspondentes aos não-terminais A, B e α e criar a regra de transição $(\varepsilon, A', \varepsilon) : \rightarrow (B', \alpha', \varepsilon)$, B , onde α é o estado inicial de uma submáquina

As ações adaptativas A e B serão descritos no passo 17 em diante.

Passo 10: Para cada regra de produção do tipo $A \leftarrow \{A\}\ \alpha B$, onde $\alpha \in V_C$, tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a regra de transição $(\varepsilon, A', \varepsilon) : \rightarrow (\varepsilon, B', \alpha)$, B

As ações adaptativas A e B serão descritos no passo 17 em diante.

Passo 11: Para cada regra de produção do tipo $\alpha A \rightarrow \{A\}\ B$, onde $\alpha \in V_C$, tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a regra de transição $(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \varepsilon)$, B

As ações adaptativas A e B serão descritos no passo 17 em diante.

Passo 12: Para cada regra de produção do tipo $\alpha A \leftarrow \{A\}\ \beta B$, onde $\alpha, \beta \in V_C$, tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a regra de transição $(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \beta)$, B.

As ações adaptativas A e B serão descritos no passo 17 em diante.

Passo 13: Para cada regra de produção do tipo $\alpha A \rightarrow \{A\}\ bB$, onde $\alpha \in V_C$ e $b \in V_T$, tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar as regras de transição

$(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)$ e

$(\varepsilon, X, b) : \rightarrow (\varepsilon, B', \varepsilon)$, `B`

As ações adaptativas `A` e `B` serão descritos no passo 17 em diante.

Passo 14: Para cada regra de produção do tipo $\alpha A \rightarrow$ {`A` } CB, onde $\alpha \in V_C$ e $C \in V_N$, tomar os estados A', B' e C' que são correspondentes aos não-terminais A, B e C e criar as regras de transição

$(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)$ e

$(\varepsilon, X, b) : \rightarrow (B', C', \varepsilon)$, `B`

onde X é um estado auxiliar não pertencente ao conjunto de estados Q^i

As ações adaptativas `A` e `B` serão descritos no passo 17 em diante.

Passo 15: Para cada regra de produção do tipo $A \rightarrow$ {`A` }ε, tomar o estado A' que é correspondente ao não-terminal A e criar a regra de transição $(X, A', \varepsilon) : \rightarrow (\varepsilon, X, \varepsilon)$, `B` , para $X \in$ { retornos de submáquinas chamadoras}

As ações adaptativas `A` e `B` serão descritos no passo 17 em diante.

Passo 16: Para cada regra de produção do tipo $A \rightarrow \phi$, criar um estado inicial da submáquina A.

A tabela abaixo sintetiza as regras apresentadas anteriormente:

$A \rightarrow$ {`A` } αB, onde $\alpha \in V_T$	$(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \varepsilon)$, `B`
$A \rightarrow$ {`A` } αB, onde $\alpha \in V_N$	$(\varepsilon, A', \varepsilon) : \rightarrow (B', \alpha', \varepsilon)$, `B`
$A \leftarrow$ {`A` } αB, onde $\alpha \in V_C$	$(\varepsilon, A', \varepsilon) : \rightarrow (\varepsilon, B', \alpha)$, `B`
$\alpha A \rightarrow$ {`A` } B, onde $\alpha \in V_C$	$(\varepsilon, A, \alpha) : \rightarrow (\varepsilon, B, \varepsilon)$, `B`
$\alpha A \leftarrow$ {`A` } βB, onde $\alpha, \beta \in V_C$	$(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \beta)$, `B`.
$\alpha A \rightarrow$ {`A` } bB, onde $\alpha \in V_C$ e $b \in V_T$	$(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)$ e $(\varepsilon, X, b) : \rightarrow (\varepsilon, B', \varepsilon)$, `B`
$\alpha A \rightarrow$ {`A` } CB, onde $\alpha \in V_C$ e $C \in V_N$	$(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)$ e $(\varepsilon, X, b) : \rightarrow (B', C', \varepsilon)$, `B`

$A \rightarrow$ {A } ε	$(X, A', \varepsilon) :\rightarrow (\varepsilon, X, \varepsilon)$,B , para $X \in$ { retornos de submáquinas chamadoras}
$A \rightarrow \phi$	criar um estado inicial da submáquina

Correspondência entre as ações adaptativas

Para mapear as ações adaptativas da gramática para o autômato, basta mapear as ações elementares de consulta, adição e remoção. Nos passos seguintes o símbolo $\otimes$ representa ? ou + ou –. As ações adaptativas A e são B opcionais.

Passo 17: Para cada $\otimes[A \rightarrow$ {A } $\alpha B]$ onde $\alpha \in V_T$ tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a ação elementar

$\otimes[(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \varepsilon),$ B]

Passo 18: Para cada $\otimes[A \rightarrow$ {A } $\alpha B]$ onde $\alpha \in V_N$ tomar os estados A', B' e α' que são correspondentes aos não-terminais A, B e α' e criar a ação elementar

$\otimes[(\varepsilon, A', \varepsilon) : \rightarrow (B', \alpha', \varepsilon),$ B]

Passo 19: Para cada $\otimes[A \leftarrow$ {A } $\alpha B]$ onde $\alpha \in V_C$ tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a ação elementar

$\otimes[(\varepsilon, A', \varepsilon) : \rightarrow (\varepsilon, B', \alpha),$ B]

Passo 20: Para cada $\otimes[\alpha A \rightarrow$ {A } B] onde $\alpha \in V_C$ tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a ação elementar

$\otimes[(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \varepsilon),$ B]

Passo 21: Para cada $\otimes[\alpha A \leftarrow$ {A } $\beta B]$ onde $\alpha, \beta \in V_C$ tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar a ação elementar

$\otimes[(\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \beta),$ B]

Passo 22: Para cada $\otimes[\ \alpha A \rightarrow \{A\ \}\ bB]$ onde $\alpha \in V_C$ e $b \in V_T$ tomar os estados A' e B' que são correspondentes aos não-terminais A e B e criar as ações elementares

$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)]$

$\otimes[\ (\varepsilon, X, b) : \rightarrow (\varepsilon, B', \varepsilon), B\ ]$

Passo 23: Para cada $\otimes[\ \alpha A \rightarrow \{A\ \}\ CB]$, onde $\alpha \in V_C$ e $C \in V_N$ tomar os estados A', B' e C' que são correspondentes aos não-terminais A, B e C e criar as ações elementares

$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)]$

$\otimes[\ (\varepsilon, X, b) : \rightarrow (B', C', \varepsilon), B\ ]$

Passo 24: Para cada $\otimes[\ A \rightarrow \{A\ \}\ \varepsilon]$ tomar o estado A' que é correspondente ao não-terminal A e criar a ação elementar $\otimes[\ (X, A', \varepsilon) : \rightarrow (\varepsilon, X, \varepsilon), B\ ]$

para $X \in$ { retornos de submáquinas chamadoras}

Passo 25: Para cada $\otimes[\ A \rightarrow \phi\ ]$ equivale a criar um estado inicial para a submáquina A.

A tabela abaixo sintetiza as regras apresentadas anteriormente:

$\otimes[A \rightarrow \{A\ \}\ \alpha B]$ onde $\alpha \in V_T$	$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \varepsilon), B\ ]$
$\otimes[A \rightarrow \{A\ \}\ \alpha B]$ onde $\alpha \in V_N$	$\otimes[\ (\varepsilon, A', \varepsilon) : \rightarrow (B', \alpha', \varepsilon), B]$
$\otimes[\ A \leftarrow \{A\ \}\ \alpha B]$ onde $\alpha \in V_C$	$\otimes[\ (\varepsilon, A', \varepsilon) : \rightarrow (\varepsilon, B', \alpha), B\ ]$
$\otimes[\ \alpha A \rightarrow \{A\ \}\ B]$ onde $\alpha \in V_C$	$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \varepsilon), B\ ]$
$\otimes[\ \alpha A \leftarrow \{A\ \}\ \beta B]$ onde $\alpha, \beta \in V_C$	$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, B', \beta), B\ ]$
$\otimes[\ \alpha A \rightarrow \{A\ \}\ bB]$ onde $\alpha \in V_C$ e $b \in V_T$	$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)]$ $\otimes[\ (\varepsilon, X, b) : \rightarrow (\varepsilon, B', \varepsilon), B\ ]$
$\otimes[\ \alpha A \rightarrow \{A\ \}\ CB]$, onde $\alpha \in V_C$ e $C \in V_N$	$\otimes[\ (\varepsilon, A', \alpha) : \rightarrow (\varepsilon, X, \varepsilon)]$ $\otimes[\ (\varepsilon, X, b) : \rightarrow (B', C', \varepsilon), B\ ]$
$\otimes[\ A \rightarrow \{A\ \}\ \varepsilon]$	$\otimes[\ (X, A', \varepsilon) : \rightarrow (\varepsilon, X, \varepsilon), B\ ]$ para $X \in$ { retornos de submáquinas chamadoras}
$\otimes[\ A \rightarrow \phi\ ]$	criar um estado inicial para a submáquina A

Exemplo de conversão de gramática adaptativa para autômato adaptativo

Seja dada a gramática adaptativa $G = (G^0, T, R^0)$, onde $G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$

$V_N^0 = \{ S, A, B, C\}$

$V_T = \{ a, b, c, d\}$

$V_C = \phi$

$P_L^0 = \{ S \rightarrow aABc$

$A \rightarrow \{A\ (B, C)\}\ bBC \}$

$B \rightarrow \phi$

$C \rightarrow cd \mid daC$

$\}$

$P_D^0 = \phi$

$T = \{ A\ (x, y) = \{ +[x \rightarrow ac]$

$- [y \rightarrow cd]$

$+[y \rightarrow cda]$

$\}$

$\}$

Normalizando a gramática G, temos $G' = (G^{0'}, T', R^{0'})$ onde

$G^{0'} = (V_N^{0'}, V_T', V_C', P_L^{0'}, P_D^{0'}, S)$

$V_N^{0'} = \{ A, X_1, X_2, X_3, X_4, B, Y, Y_1, Y_2, Y_3, Y_4, Y_5, C, Z_1, Z_2, Z_3, T_1, T_2, T_3, T_4, T_5\}$

$V_T' = \{ a, b, c, d\}$

$V_C' = \{ \alpha, \beta, \gamma \}$

$P_L^{0'} \cup P_C^{0'} = \{$

$S \rightarrow a\ X_1$	$A \rightarrow \{A\ '(B, C)\}\ \varepsilon Y$	$C \rightarrow cZ_1$	$C \rightarrow dT_1$
$X_1 \rightarrow A\ X_2$	$Y \rightarrow bY_1$	$Z_1 \rightarrow d\ Z_2$	$T_1 \rightarrow a\ T_2$
$\alpha X_2 \rightarrow BX_3$	$Y_1 \rightarrow B\ Y_2$	$Z_2 \leftarrow \gamma\ Z_3$	$T_2 \rightarrow CT_3$
$\beta X_3 \rightarrow cX_4$	$\beta Y_2 \rightarrow C\ Y_3$	$Z_3 \rightarrow \varepsilon$	$\gamma T_3 \rightarrow T_4$
$X_4 \rightarrow \varepsilon$	$\gamma\ Y_3 \rightarrow Y_4$		$T_4 \leftarrow \gamma T_5$

	$Y_4 \leftarrow \alpha Y_5$		$T_5 \rightarrow \varepsilon$
$B \rightarrow \phi$	$Y_5 \rightarrow \varepsilon$		

}

T' = {

A '(x, y) = {u_1, u_2, u_3, v_1, v_2, v_3, v_4 :

+ [$x \rightarrow a\ u_1$]	- [$y \rightarrow cv_1$]	+ [$y \rightarrow cv_1$]
+ [$u_1 \rightarrow c\ u_2$]	- [$v_1 \rightarrow dv_2$]	+ [$v_1 \rightarrow dv_2$]
+ [$u_2 \leftarrow x'\ u_3$]	- [$v_2 \leftarrow y'\ v_3$]	+ [$v_2 \rightarrow av_3$]
+ [$u_3 \rightarrow \varepsilon$]	- [$v_3 \rightarrow \varepsilon$]	+ [$v_3 \leftarrow y'\ v_4$]
		+ [$v_4 \rightarrow \varepsilon$]

}

Transformando cada regra de produção da gramática adaptativa normalizada nas correspondentes regra de transição do autômato adaptativo, temos:

$S \rightarrow a\ X_1$	$(\varepsilon, S', a) :\rightarrow (\varepsilon, X_1', \varepsilon)$
$X_1 \rightarrow A\ X_2$	$(\varepsilon, X_1', \varepsilon) :\rightarrow (X_2', A', \varepsilon)$
$\alpha X_2 \rightarrow BX_3$	$(\varepsilon, X_2', \alpha) :\rightarrow (\varepsilon, P', \varepsilon)$ $(\varepsilon, P', \varepsilon) :\rightarrow (X_3, B', \varepsilon)$
$\beta X_3 \rightarrow cX_4$	$(\varepsilon, X_3', \beta) :\rightarrow (\varepsilon, Q', \varepsilon)$ $(\varepsilon, Q', \varepsilon) :\rightarrow (\varepsilon, X_4', \varepsilon)$
$X_4 \rightarrow \varepsilon$	$(HALT, X_4', \varepsilon) :\rightarrow (\varepsilon, HALT, \varepsilon)$
$B \rightarrow \phi$	
$A \rightarrow \{B\ (B, C)\}\ \varepsilon Y$	$(\varepsilon, A', \varepsilon) :\rightarrow (\varepsilon, Y', \varepsilon), B\ (B', C')$
$Y \rightarrow bY_1$	$(\varepsilon, Y', b) :\rightarrow (\varepsilon, Y_1', \varepsilon)$
$Y_1 \rightarrow B\ Y_2$	$(\varepsilon, Y_1', \varepsilon) :\rightarrow (Y_2, B', \varepsilon)$
$\beta Y_2 \rightarrow C\ Y_3$	$(\varepsilon, Y_2', \beta) :\rightarrow (\varepsilon, R', \varepsilon)$

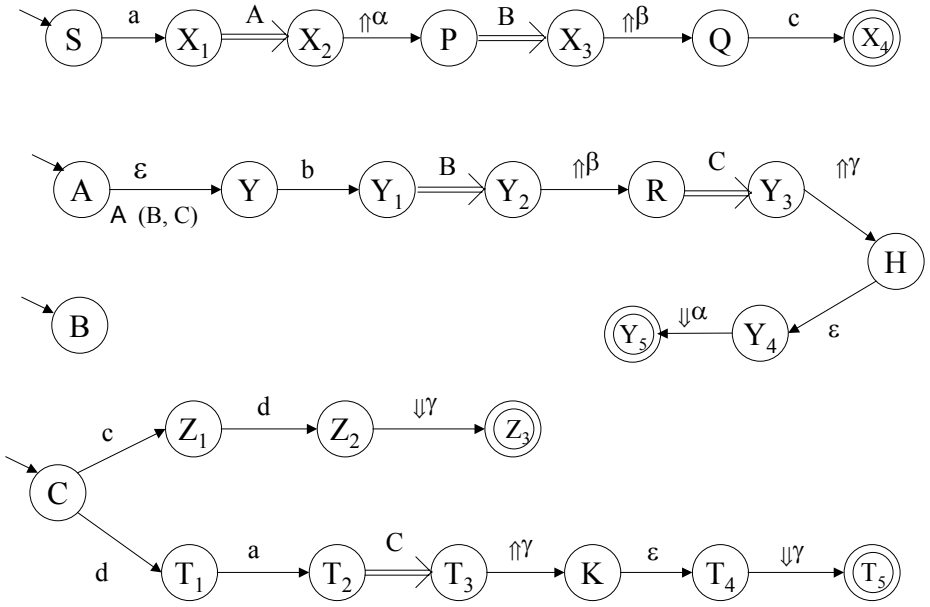
	$(\varepsilon, R', \varepsilon) :\rightarrow (Y_3', C', \varepsilon)$
$\gamma\ Y_3 \rightarrow Y_4$	$(\varepsilon, Y_3', \gamma) :\rightarrow (\varepsilon, Y_4', \varepsilon)$
$Y_4 \leftarrow \alpha Y_5$	$(\varepsilon, Y_4', \varepsilon) :\rightarrow (\varepsilon, Y_5', \alpha)$
$Y_5 \rightarrow \varepsilon$	$(X_2', Y_5', \varepsilon) : \rightarrow (\varepsilon, X_2', \varepsilon)$ $(T_3', Z_3', \varepsilon) : \rightarrow (\varepsilon, X_2', \varepsilon)$
$C \rightarrow cZ_1$	$(\varepsilon, C', c) :\rightarrow (\varepsilon, Z_1', \varepsilon)$
$Z_1 \rightarrow d\ Z_2$	$(\varepsilon, Z_1', d) :\rightarrow (\varepsilon, Z_2', \varepsilon)$
$Z_2 \leftarrow \gamma\ Z_3$	$(\varepsilon, Z_2', \varepsilon) :\rightarrow (\varepsilon, Z_3', \gamma)$
$Z_3 \rightarrow \varepsilon$	$(Y_2', Z_3', \varepsilon) :\rightarrow (\varepsilon, Y_2', \varepsilon)$
$C \rightarrow dT_1$	$(\varepsilon, C', d) :\rightarrow (\varepsilon, T_1', \varepsilon)$
$T_1 \rightarrow a\ T_2$	$(\varepsilon, T_1', a) :\rightarrow (\varepsilon, T_2', \varepsilon)$
$T_2 \rightarrow CT_3$	$(\varepsilon, T_2', \varepsilon) :\rightarrow (T_3', C', \varepsilon)$
$\gamma T_3 \rightarrow T_4$	$(\varepsilon, T_3', \gamma) :\rightarrow (\varepsilon, T_4', \varepsilon)$
$T_4 \leftarrow \gamma T_5$	$(\varepsilon, T_3', \varepsilon) :\rightarrow (\varepsilon, T_5', \gamma)$
$T_5 \rightarrow \varepsilon$	$(Y_2', Z_2', \varepsilon) :\rightarrow (\varepsilon, Y_2', \varepsilon)$ $(T_3', Z_3', \varepsilon) :\rightarrow (\varepsilon, T_3', \varepsilon)$

Transformando também as ações elementares da ação adaptativa `A '` da gramática nas correspondentes ações elementares da ação adaptativa `B` do autômato, temos:

$+ [x \rightarrow a\ u_1]$	$+ [(\varepsilon, x, a) :\rightarrow (\varepsilon, u_1, \varepsilon)]$
$+ [u_1 \rightarrow c\ u_2]$	$+ [(\varepsilon, u_1, c) :\rightarrow (\varepsilon, u_2, \varepsilon)]$
$+ [u_2 \leftarrow x'\ u_3]$	$+ [(\varepsilon, u_2, \varepsilon) :\rightarrow (\varepsilon, u_3, x')]$
$+ [u_3 \rightarrow \varepsilon]$	$+ [(\gamma, u_3', \varepsilon) :\rightarrow (\varepsilon, \gamma, \varepsilon)]$
$- [y \rightarrow cv_1]$	$- [(\varepsilon, y, c) :\rightarrow (\varepsilon, v_1, \varepsilon)]$
$- [v_1 \rightarrow dv_2]$	$- [(\varepsilon, v_1, d) :\rightarrow (\varepsilon, v_2, \varepsilon)]$
$- [v_2 \leftarrow y'\ v_3]$	$- [(\varepsilon, v_2, \varepsilon) :\rightarrow (\varepsilon, v_3, y')]$
$- [v_3 \rightarrow \varepsilon]$	$- [(\gamma, v_3', \varepsilon) :\rightarrow (\varepsilon, \gamma, \varepsilon)]$
$+ [y \rightarrow cv_1]$	$+ [(\varepsilon, y, c) :\rightarrow (\varepsilon, v_1, \varepsilon)]$

+ [$v_1 \rightarrow dv_2$]	+ [(ε, v_1, d) :→ (ε, v_2, ε)]
+ [$v_2 \rightarrow av_3$]	+ [(ε, v_2, a) :→ (ε, v_3, ε)]
+ [$v_3 \leftarrow y' v_4$]	+ [(ε, v_3, ε) :→ (ε, v_4, y')]
+ [$v_4 \rightarrow \varepsilon$]	+ [(γ, v_4', ε) :→ (ε, γ, ε)]

O diagrama do autômato adaptativo resultante é o seguinte:



O autômato adaptativo resultante é M = (E^0, A, Φ^0), onde

E^0 = (Q^0, SM^0, Σ, Γ^0, TR^0, Z^0, q^0, F^0)

q^0 = S'

Q^0 = { S', X_1', X_2', P', X_3', Q', X_4', A' , Y', Y_1', Y_2' , R', Y_3', H' , Y_5', Y_4', Z_1', Z_2', Z_3', C', T_1', T_2', T_3', K', T_4', T_5'}

SM^0 = { }

Σ ={ a, b, c, d, α, β, γ}

Γ^0 = { HALT, X_2', X_3, Y_2, Y_3','T_3', Y_2'}

$F^0 = \{ X_4 \}$

$TR^0 = \{$ $(\varepsilon, S', a) :\rightarrow (\varepsilon, X_1', \varepsilon)$

$(\varepsilon, X_1', \varepsilon) :\rightarrow (X_2', A', \varepsilon)$

$(\varepsilon, X_2', \alpha) :\rightarrow (\varepsilon, P', \varepsilon)$

$(\varepsilon, P', \varepsilon) :\rightarrow (X_3, B', \varepsilon)$

$(\varepsilon, X_3', \beta) :\rightarrow (\varepsilon, Q', \varepsilon)$

$(\varepsilon, Q', \varepsilon) :\rightarrow (\varepsilon, X_4', \varepsilon)$

$(HALT, X_4', \varepsilon) :\rightarrow (\varepsilon, HALT, \varepsilon)$

$(\varepsilon, A', \varepsilon) :\rightarrow (\varepsilon, Y', \varepsilon),$ B (B', C')

$(\varepsilon, Y', b) :\rightarrow (\varepsilon, Y_1', \varepsilon)$

$(\varepsilon, Y_1', \varepsilon) :\rightarrow (Y_2, B', \varepsilon)$

$(\varepsilon, Y_2', \beta) :\rightarrow (\varepsilon, R', \varepsilon)$

$(\varepsilon, R', \varepsilon) :\rightarrow (Y_3', C', \varepsilon)$

$(\varepsilon, Y_3', \gamma) :\rightarrow (\varepsilon, Y_4', \varepsilon)$

$(\varepsilon, Y_4', \varepsilon) :\rightarrow (\varepsilon, Y_5', \alpha)$

$(X_2', Y_5', \varepsilon) : \rightarrow (\varepsilon, X_2', \varepsilon)$

$(T_3', Z_3', \varepsilon) : \rightarrow (\varepsilon, X_2', \varepsilon)$

$(\varepsilon, C', c) :\rightarrow (\varepsilon, Z_1', \varepsilon)$

$(\varepsilon, Z_1', d) :\rightarrow (\varepsilon, Z_2', \varepsilon)$

$(\varepsilon, Z_2', \varepsilon) :\rightarrow (\varepsilon, Z_3', \gamma)$

$(Y_2', Z_3', \varepsilon) :\rightarrow (\varepsilon, Y_2', \varepsilon)$

$(\varepsilon, C', d) :\rightarrow (\varepsilon, T_1', \varepsilon)$

$(\varepsilon, T_1', a) :\rightarrow (\varepsilon, T_2', \varepsilon)$

$(\varepsilon, T_2', \varepsilon) :\rightarrow (T_3', C', \varepsilon)$

$(\varepsilon, T_3', \gamma) :\rightarrow (\varepsilon, T_4', \varepsilon)$

$(\varepsilon, T_3', \varepsilon) :\rightarrow (\varepsilon, T_5', \gamma)$

$(Y_2', Z_2', \varepsilon) :\rightarrow (\varepsilon, Y_2', \varepsilon)$

$(T_3', Z_3', \varepsilon) :\rightarrow (\varepsilon, T_3', \varepsilon)$

$\}$

$A = \{ B\ (x, y) = \{ + [(\varepsilon, x, a) :\rightarrow (\varepsilon, u_1, \varepsilon)]$

$+ [(\varepsilon,u_1, c) :\rightarrow (\varepsilon, u_2, \varepsilon)]$

$+ [(\varepsilon, u_2, \varepsilon) :\rightarrow (\varepsilon, u_3, x')]$

$+ [(\gamma, u_3', \varepsilon) :\rightarrow (\varepsilon, \gamma, \varepsilon)]$

$- [(\varepsilon, y, c) :\rightarrow (\varepsilon, v_1, \varepsilon)]$

$- [(\varepsilon,v_1, d) :\rightarrow (\varepsilon, v_2, \varepsilon)]$

$- [(\varepsilon, v_2, \varepsilon) :\rightarrow (\varepsilon, v_3, y')]$

$- [(\gamma, v_3', \varepsilon) :\rightarrow (\varepsilon, \gamma, \varepsilon)]$

$+ [(\varepsilon, y, c) :\rightarrow (\varepsilon, v_1, \varepsilon)]$

$+ [(\varepsilon, v_1, d) :\rightarrow (\varepsilon, v_2, \varepsilon)]$

$+ [(\varepsilon, v_2, a) :\rightarrow (\varepsilon, v_3, \varepsilon)]$

$+ [(\varepsilon, v_3, \varepsilon) :\rightarrow (\varepsilon, v_4, y')]$

$+ [(\gamma, v_4', \varepsilon) :\rightarrow (\varepsilon, \gamma, \varepsilon)]$

$\}$

$\}$

4.2 Algoritmo para a conversão canônica do autômato adaptativo para a forma da gramática adaptativa

Seja $M = (E^0, A, \Phi^0)$ um autômato adaptativo, onde $E^0 = (Q^0, SM^0, \Sigma, \Gamma^0, TR^0, Z^0, q^0, F^0)$ é a máquina inicial que implementa M, A é o conjunto de ações adaptativas e Φ^0 é a relação que associa as regras de transição com as ações adaptativas.

Correspondência dos estados do autômato com os não-terminais da gramática.

Passo 1: Para q^0, o estado inicial do autômato, criar o símbolo não-terminal inicial S da gramática.

Para cada estado $q \in Q^0$, criar o correspondente não-terminal $q' \in V_N^0$.

Passo 2: Os símbolos Σ do autômato são os símbolos terminais do conjunto V_T e também os símbolos de contexto do conjunto V_C da gramática.

Passo 3: Γ = são todos os elementos que aparecem na primeira posição da tripla que constitui o lado esquerdo ou o lado direito de qualquer transição

Transformação prévia das transições do autômato

Passo 4: Para toda transição do tipo

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

criar um estado adicional e", tal que $e'' \notin Q^0$ e transformar em

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

eliminando assim ações adaptativas executadas antes da transição de estado.

Passo 5: Acrescentar e" em Q^0 e acrescentar o correspondente não terminal ao conjunto V_N

Passo 6: Se a ação adaptativa `A` torna a transição que a contém tipo auto-eliminável, ou seja, se nela for executada alguma ação adaptativa que elimina a própria transição que a contém, então transformar a ação `A` em `A '` , onde `A '` deve eliminar tanto a transição

$$(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) \ , \texttt{A '}$$

como a transição

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

Para isto, incluir em A ' as seguintes ações elementares

$$-[(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) , A ']$$

$$-[(\gamma g, e'', s\alpha) : \rightarrow (\gamma g', e', s'\alpha) , B]$$

Passo 7: Analogamente, se a ação adaptativa B for do tipo auto-eliminável, então transformar a ação B em B ' , onde B ' deve eliminar tanto

$$(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) , A$$

como

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

incluindo em B ' as seguintes ações elementares

$$-[(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) , A]$$

$$-[(\gamma g, e'', s\alpha) : \rightarrow (\gamma g', e', s'\alpha) , B ']$$

Passo 8: Para toda transição do tipo

$$(\gamma g, e, s\alpha) : A , \rightarrow (\gamma g', e', s'\alpha)$$

criar um estado adicional e", tal que e" $\notin Q^0$ e transformar em

$$(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) , A$$

$$(\gamma g, e'', s\alpha) : \rightarrow (\gamma g', e', s'\alpha)$$

Passo 9: Acrescentar e" em Q^0 e acrescentar o correspondente não terminal ao conjunto V_N

Passo 10: Se a ação adaptativa A for do tipo auto-eliminável, então transformar a ação A em A ' , onde A ' deve eliminar tanto

$$(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) , A '$$

como

$$(\gamma g, e'', s\alpha) : \rightarrow (\gamma g', e', s'\alpha)$$

isto é, incluir em A ' as seguintes ações elementares

$$-[(\gamma, e, s\alpha) : \rightarrow (\gamma, e'', s\alpha) , A ']$$

$$-[(\gamma g, e'', s\alpha) : \rightarrow (\gamma g', e', s'\alpha)]$$

Correspondência das transições do autômato com as regras de produção da gramática

Passo 11: Para cada transição interna $(\gamma, a, \sigma\alpha) : \rightarrow (\gamma, b, \alpha)$, $A \in TR^0$, onde $\sigma \in \Sigma \cup \{\varepsilon\}$ e tomar os não-terminais a' e b' que são correspondentes aos estados a e b, respectivamente.

Criar uma regra de produção se $\sigma \notin V_C$,

$$a' \rightarrow \{B\} \sigma b'$$

Senão, se $\sigma \in V_C$, então criar uma regra de produção do tipo

$$\sigma a' \rightarrow \{B\} b'$$

Se $A = \varepsilon$, então $B = \varepsilon$, caso contrário a relação entre A e B está descrita nos passos 17 em diante deste algoritmo

Passo 12: Para cada transição interna $(\gamma, a, \sigma\alpha) : \rightarrow (\gamma, b, \sigma'\alpha)$, $A \in TR^0$, onde $\sigma \in \Sigma \cup \{\varepsilon\}$ e $\sigma' \in \Sigma$, tomar os não-terminais a' e b' que são correspondentes aos estados a e b, respectivamente. Criar um não-terminal c auxiliar

Criar as duas seguintes regras de produção

$$a' \rightarrow \sigma c$$

e

$$c \leftarrow \{B\} \sigma' b'$$

Se $A = \varepsilon$, então $B = \varepsilon$, caso contrário a relação entre A e B está descrita nos passos 17 em diante deste algoritmo

Passo 13: Acrescentar o não-terminal auxiliar c no conjunto V_N.

Passo 14: Para cada chamada de submáquina $(\gamma, a, \alpha) : \rightarrow (\gamma b, N_0, \alpha) , \in TR^0$, onde N_0 é o estado inicial da submáquina N, tomar o não-terminal N_0' correspondentes ao estado N_0 da submáquina N. e os não-terminais a' e b' correspondentes aos estados a e b respectivamente.

Criar uma produção

$$a' \rightarrow N_0' b'$$

Passo 15: Para cada retorno de submáquina $(\gamma b, N_i, \alpha) : \rightarrow (\gamma, b, \alpha)$, onde N_i são estados finais da submáquina associada ao não terminal N, tomar o não-terminal N_i' correspondente e criar uma produção

$$N_i' \rightarrow \varepsilon$$

Passo 16: Para cada $a \in F^0$ e a' o não-terminal correspondente, criar uma produção

$$a' \rightarrow \varepsilon$$

Correspondência entre as ações adaptativas do autômato com as ações adaptativas da gramática. Nos passos seguintes o símbolo $\otimes$ representa ? ou + ou –. As ações adaptativas `A` e são `B` opcionais.

Passo 17: Para ações elementares do tipo $\otimes [(\gamma, a, \sigma\alpha) : \rightarrow (\gamma, b, \alpha),$ `A`]

tomar os não-terminais a' e b' correspondentes a a e b respectivamente e criar uma ação elementar correspondente na gramática

$$\otimes [a' \rightarrow \{ \texttt{B} \} \sigma b']$$

onde `A` e `B` são as ações adaptativas correspondentes opcionais

Passo 18: Para ações elementares do tipo $\otimes [(\gamma, a, \sigma\alpha) : \rightarrow (\gamma, b, \sigma'\alpha),$ `A`]

tomar os não-terminais a' e b' correspondentes a a e b respectivamente, tomar um não-terminal auxiliar c e criar duas ações elementares correspondentes na gramática

$$\otimes [a' \rightarrow \sigma c \ ]$$

e

$$\otimes [c \leftarrow \{ \texttt{B} \} \sigma b' \ ]$$

onde `A` e `B` são as ações adaptativas correspondentes opcionais

Passo 19: Para ações elementares do tipo $\otimes [(\gamma, a, \alpha) : \rightarrow (\gamma b, N_0, \alpha)$]

tomar os não-terminais a' e b' correspondentes a a e b respectivamente e criar uma ação elementar correspondente na gramática

$$\otimes [a' \rightarrow N_0 'b']$$

Passo 20: Para ações elementares do tipo $\otimes [(\gamma b, N_i, \alpha) : \rightarrow (\gamma, b, \alpha)]$

tomar os não-terminais a' e b' correspondentes a a e b respectivamente e criar uma ação

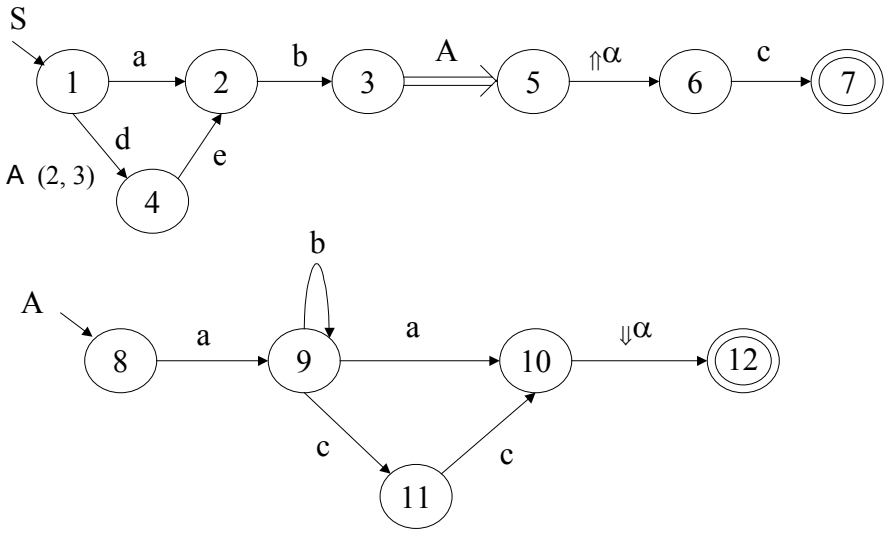
elementar correspondente na gramática

$$\otimes [N_i' \rightarrow \varepsilon]$$

Exemplo de conversão de autômato adaptativo para gramática adaptativa

Seja dado o seguinte autômato adaptativo $M = (E^0, A, \Phi^0)$, onde

$E^0 = (Q^0, SM^0, \Sigma, \Gamma^0, TR^0, Z^0, q^0, F^0)$, representado pela seguinte figura:



O autômato de pilha estruturado que implementa o autômato no instante inicial é

$E^0 = (Q^0, SM^0, \Sigma, \Gamma^0, TR^0, Z^0, q^0, F^0)$, onde:

$Q^0 = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$

$SM^0 = \{ A \}$

$\Sigma = \{ a, b, c, d, e, \alpha\}$

$F^0 = \{ 7 \}$

$TR^0 = \{ (\varepsilon, 1, a) :\rightarrow (\varepsilon, 2, \varepsilon)$

$(\varepsilon, 1, d) :\rightarrow (\varepsilon, 4, \varepsilon)$, A (2, 3)

$(\varepsilon, 4\ d) :\rightarrow (\varepsilon, 4, \varepsilon)$

(ε, 2 b) :→ (ε, 3, ε)

(ε, 3, ε) :→ (5, 8, ε)

(ε, 5, α) :→ (ε, 6, ε)

(ε, 6, ε) :→ (ε, 7, ε)

(HALT, 7, ε) :→ (ε, HALT, ε)

(ε, 8, a) :→ (ε, 9, ε)

(ε, 9, b) :→ (ε, 9, ε)

(ε, 11, c) :→ (ε, 11, ε)

(ε, 9, a) :→ (ε, 10, ε)

(ε, 10, ε) :→ (ε, 12, α)

(5, 12, ε) :→ (ε, 5, α)

}

O conjunto de ações adaptativas é dado por

A = { A (x, y) = { - [(ε, x, b) :→ (ε, y, ε)]

+ [(ε, x, c) :→ (ε, y, ε)]

}

}

A tabela seguinte apresenta as regras de transição do autômato e suas respectivas conversões para as regras de produção da gramática adaptativa equivalente.

(ε, 1, a) :→ (ε, 2, ε)	1' → a 2'
(ε, 1, d) :→ (ε, 4, ε), A (2, 3)	1' → {B (2', 3')} d 4'
(ε, 4 d) :→ (ε, 4, ε)	4' → e 2'
(ε, 2 b) :→ (ε, 3, ε)	2' → b 3'
(ε, 3, ε) :→ (5, 8, ε)	3' → 8' 5'
(ε, 5, α) :→ (ε, 6, ε)	α5' → 6'
(ε, 6, ε) :→ (ε, 7, ε)	6' → c 7'

(HALT, 7, ε) :→ (ε, HALT, ε)	7’ → ε
(ε, 8, a) :→ (ε, 9, ε)	8’ → a 9’
(ε, 9, b) :→ (ε, 9, ε)	9’ → b 9’
(ε, 9, c) :→ (ε, 11, ε)	9’ → c 11’
(ε, 11, c) :→ (ε, 11, ε)	11’ → c 10’
(ε, 9, a) :→ (ε, 10, ε)	9’ → a 10’
(ε, 10, ε) :→ (ε, 12, α)	10’ → ε 100 100 ← α 12’
(5, 12, ε) :→ (ε, 5, α)	12’ → ε

A ação adaptativa `A` do autômato

`A` (x, y) = { - [(ε, x, b) :→ (ε, y, ε)]

+ [(ε, x, c) :→ (ε, y, ε)]

}

é transformada na ação adaptativa `B` da gramática

`B` (x’, y’) = { - [x’ → by’]

+[x’ → cy’]

}

A gramática adaptativa resultante a partir do autômato adaptativo M é $G = (G^0, T, R^0)$, onde

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$

$S = 1’$

$V_N^0 = \{ 1’, 2’, 3’, 4’, 5’, 6’, 7’, 8’, 9’, 10’, 11’, 12’, 100 \}$

$V_T = \{ a, b, c, d, e \}$

$V_C = \{ \alpha \}$

$P_L^0 \cup P_D^0 = \{$ 1’ → a 2’

1’ → {`B` (2’, 3’)} d 4’

4’ → e 2’

2’ → c 3’

3’ → 8’ 5’

$\alpha 5' \rightarrow 6'$

$6' \rightarrow c\ 7'$

$7' \rightarrow \varepsilon$

$8' \rightarrow a\ 9'$

$9' \rightarrow b\ 9'$

$9' \rightarrow c\ 11'$

$11' \rightarrow c\ 10'$

$9' \rightarrow a\ 10'$

$10' \rightarrow \varepsilon\ 100$

$100 \leftarrow \alpha\ 12'$

$12' \rightarrow \varepsilon$

}

O conjunto de ações adaptativas da gramática que é equivalente às ações adaptativas do autômato é dado por:

$T = \{ B\ (x', y') = \{ - [x' \rightarrow by']$

$+[x' \rightarrow cy']$

}

}

4.3. Algoritmo eficiente para a obtenção de um analisador sintático a partir de gramáticas adaptativas

No capítulo 3 de [Jos93] foi apresentado um algoritmo que transforma uma gramática livre de contexto em um analisador sintático e um outro algoritmo que transforma este analisador sintático em um autômato de pilha estruturado.
Neste trabalho extraimos e adaptamos estes métodos para que operem também no caso das gramáticas adaptativas e autômatos adaptativos. Para isto, o algoritmo é dividido em partes.

O primeiro algoritmo prepara a gramática adaptativa para a construção do transdutor

1. Numerar as regras de produção, para que se possa identificá-las de modo único.
2. Para cada não-terminal A, identificar as produções cujos lados direitos possuam recursões à direita, recursões à esquerda, recursões centrais e demais alternativas, isto é:

$A \rightarrow \{ \; A \; \} \; A \; \alpha$

$A \rightarrow \{ \; A \; \} \; \alpha \; A$

$A \rightarrow \{ \; A \; \} \; \alpha A \beta$

$A \rightarrow \{ \; A \; \} \; \alpha$

$A \rightarrow \phi$

onde $\alpha, \beta \in (V_N \cup V_T)^*$ e $A \notin V_T$. Notar que a ação adaptativa é opcional e vem denotada entre chaves quando existir.

3. Rotular as produções da seguinte maneira.
Se A for uma regra de produção do tipo padrão:
$A \rightarrow \{ \; A \; \} \; \mu_1 \; \mu_2 \; \mu_3 \; ..... \; \mu_n$
onde $\mu_i \in (V_N \cup V_T) \cup \{\varepsilon\}$, $1 \leq i \leq n$, então os rótulos serão do seguinte tipo
$\mu'_i = \varepsilon$ se μ_i for não-terminal e
$\mu'_i = \mu_i$ caso contrário
A extremidade esquerda do lado direito da produção recebe o símbolo [e a extremidade direita recebe o símbolo]. Assim, para cada tipo de regra de produção, temos o seguinte:

a) Para auto-recursões à esquerda

$A \rightarrow \{\ A\ \}\ A\ \mu_1\ \mu_2\ \mu_3\ .....\ \mu_n$

ficam:

$[\{\ A\ \}\qquad \mu'_1\quad \mu'_2\qquad \mu'_n A_i^e\ ]$

p[i]: $A \rightarrow \{\ A\ \} \uparrow A \uparrow \mu_1 \uparrow \mu_2 \uparrow \mu_3 \uparrow ..... \uparrow \mu_n \uparrow$

Observar que o símbolo utilizado nos rótulos para denotar que a regra é do tipo recursiva a esquerda é A_i^e.

b) Para auto-recursões à direita

$A \rightarrow \{\ A\ \}\ \mu_1\ \mu_2\ \mu_3\ .....\ \mu_n\ A$

ficam:

$[\{\ A\ \}\qquad \mu'_1\quad \mu'_2\qquad \mu'_n\quad A_i^d]$

p[i]: $A \rightarrow \{\ A\ \} \uparrow \mu_1 \uparrow \mu_2 \uparrow \mu_3 \uparrow ..... \uparrow \mu_n \uparrow\ A \uparrow$

Observar que o símbolo utilizado nos rótulos para denotar que a regra é do tipo recursiva a direita é A_i^d.

c) As produções

$A \rightarrow \{\ A\ \}\ \mu_1\ \mu_2\ \mu_3\ .....\ \mu_n$

ficam:

$[\{\ A\ \}\qquad \mu'_1\quad \mu'_2\qquad \mu'_n A_i]$

p[i]: $A \rightarrow \{\ A\ \} \uparrow \mu_1 \uparrow \mu_2 \uparrow \mu_3 \uparrow ..... \uparrow \mu_n \uparrow$

Observar que o símbolo utilizado nos rótulos para denotar que a regra é do tipo recursiva a direita é A_i.

Para as produções do tipo

$A \rightarrow \{\ A\ \}\ \varepsilon$

temos:

$[\{\ A\ \}\quad \varepsilon A_i]$

p[i]: $A \rightarrow \{\ A\ \}\uparrow \varepsilon \uparrow$

Para as produções do tipo

$A \rightarrow \phi$

temos:

$[\quad]$

p[i]: $A \rightarrow \uparrow \phi \uparrow$

Obs: Regras de produção com ϕ não possuem ação adaptativa associada

Para expressões do tipo:

$A \rightarrow \{\ A\ \}\ \alpha B$

onde $\alpha \in V_C$

a produção fica:

$[\{\ A\ \}\quad \Downarrow\alpha\quad A_i]$

p[i]: $A \rightarrow \{\ A\ \}\uparrow \Downarrow\alpha \uparrow\ B\ \uparrow$

Para expressões do tipo:

$\alpha A \rightarrow \{\ A\ \}\ B$

onde $\alpha \in V_C$

a produção se torna:

$[\{\ A\ \}\quad \Uparrow\alpha\quad A_i]$

p[i]: $\alpha A \rightarrow \{\ A\ \}\uparrow \Uparrow\alpha\ \uparrow B \uparrow$

Para expressões do tipo:

$\alpha A \rightarrow \{\ A\ \}\ \beta B$

onde $\alpha, \beta \in V_C$

temos:

$[\{\ A\ \}\quad \Uparrow\alpha\quad \Downarrow\beta\quad A_i]$

p[i]: $\alpha A \rightarrow \{\ A\ \} \Uparrow\alpha\ \uparrow\ \Downarrow\beta\ \uparrow\ B \uparrow$

Para expressões do tipo:

$A \leftarrow \{\ A\ \} \alpha B$

onde $\alpha \in V_C$

a produção se torna:

$[\{\ A\ \}\quad \Downarrow\alpha\quad A_i]$

p[i]: $A \leftarrow \{\ A\ \}\uparrow\ \Downarrow\alpha\ \uparrow\ B \uparrow$

4. Agrupar entre parênteses as produções que definem um não-terminal que contém mais de uma alternativa, rearranjando os rótulos

Abreviamos o lado direito da regra de produção através da expressão μ''_j como sendo:

$\mu'_1\quad \mu'_2\quad \mu'_n$

$\mu''_j = \uparrow \mu_1\ \uparrow\ \mu_2\ \uparrow \mu_3 \uparrow ..... \uparrow \mu_n \uparrow$

onde $\mu_i = \mu'_i$ se $\mu_i \in V_T$, com $1 \leq i \leq n$

μ'_i é omitida se $\mu_i \in V_N$

a) auto-recursão à esquerda

$[\quad \{Ação_1\}\quad A^e_1\quad \{Ação_2\}\quad A^e_2\quad \{Ação_k\}\quad A^e_k\quad]$

$A \rightarrow\ \uparrow\ A\ (\{Ação_1\}\ \mu''_1 \uparrow\ |\{Ação_2\}\ \mu''_2 \uparrow\ |\ ....\ |\{Ação_k\}\ \mu''_k \uparrow\quad)\uparrow$

b) auto-recursão à direita

$[\ \{Ação_1\}\quad A^d_1\quad \{Ação_2\}\quad A^d_2\quad \{Ação_k\}\quad A^d_k\quad]$

$A \rightarrow\ \uparrow(\{Ação_1\}\ \mu''_1 \uparrow\ |\{Ação_2\}\ \mu''_2 \uparrow\ |\ ....\ |\{Ação_k\}\ \mu''_k\ \uparrow\ )\uparrow\ A \uparrow$

c) demais produções

$[\ \{Ação_1\}\quad A_1\ \{Ação_2\}\quad A_2\quad \{Ação_k\}\quad A_k\ ]$

$A \rightarrow \uparrow (\{Ação_1\}\ \mu''_1 \uparrow \ | \{Ação_2\}\ \mu''_2 \uparrow | \ \ldots \ | \{Ação_k\}\ \mu''_k \uparrow) \uparrow$

d.) Regras de produção do tipo dependente de contexto (P_D)

Neste caso, definimos μ''_n como sendo

$$\mu''_n = \uparrow \mu_1 \uparrow \mu_2 \uparrow \mu_3 \uparrow \ldots \uparrow \mu_n \uparrow$$

onde $\mu'_1 = \Downarrow\mu$, quando $\mu \in V_C$

1° caso:

Se existirem regras do tipo

$\alpha_1 A \rightarrow \mu''_1$

$\alpha_2 A \rightarrow \mu''_2$

.....

$\alpha_k A \rightarrow \mu''_k$

onde $\alpha_i \in V_C$ para $1 \leq i \leq n$

agrupar estas expressões da seguinte forma

$$[\ \{Ação_1\} \Uparrow\alpha_1\ A_1 \ \{Ação_2\} \Uparrow\alpha_2\ A_2 \ \ldots \ \{Ação_k\} \Uparrow\alpha_k\ A_k\ ]$$

$$A \rightarrow \uparrow (\{Ação_1\} \uparrow \Uparrow\alpha_1 \uparrow \mu''_1 \uparrow \ | \{Ação_2\} \uparrow \Uparrow\alpha_2 \uparrow \mu''_2 \uparrow \ | \ \ldots \ | \{Ação_k\} \uparrow \Uparrow\alpha_k \uparrow \mu''_k \uparrow) \uparrow$$

2° Caso:

Se existirem regras do tipo

$A \leftarrow \mu''_1$

$A \leftarrow \mu''_2$

.....

$A \leftarrow \mu''_k$

agrupar estas expressões da seguinte forma

$$[\ \{Ação_1\}\ A_1 \ \{Ação_2\}\ A_2 \ \ldots \ \{Ação_k\}\ A_k\ ]$$

$$A \leftarrow \uparrow (\{Ação_1\}\ \mu''_1 \uparrow \ | \{Ação_2\}\ \mu''_2 \uparrow \ | \ \ldots \ | \{Ação_k\}\ \mu''_k \uparrow\ ) \uparrow$$

Embora esta expressão possua a seta apontada para a esquerda (←), a sua substituição em outras expressões ocorre normalmente, como é feito com todas as outras expressões.

5. Podemos eliminar auto-recursões da seguinte maneira:

Se N é uma expressão da forma

$$N = a \mid b\ N \mid N\ c \mid N\ d\ N$$

onde a, b, c, d $\in (V_N \cup V_T)^*$, N $\in V_N$ então, segundo [Jos93] pode-se aplicar a seguinte identidade:

$$N = ((\varepsilon \setminus b)\ a\ (\varepsilon \setminus c) \setminus d)$$

Chamemos

$$\{Ação_1'\}\ A_1\ \{Ação_2'\}\ A_2\ \{Ação_k'\}\ A_k$$

$$a = (\{Ação_1'\} \uparrow \alpha''_1 \uparrow \mid \{Ação_2'\} \uparrow \alpha''_2 \uparrow \mid \ldots \mid \{Ação_k'\} \uparrow \alpha''_k \uparrow)$$

$$\alpha'_{i\,1}\quad \alpha'_{i\,2}\quad \alpha'_{in}$$

onde cada $\alpha''_i = \uparrow \alpha_{i\,1} \uparrow \alpha_{i\,2} \uparrow \ldots\ldots \uparrow \alpha_{in} \uparrow$, para $1 \le i \le k$

$$\{Ação_1\}\ A_1^d\ \{Ação_2\}\ A_2^d\ \{Ação_m\}\ A_m^d$$

$$b = (\{Ação_1\} \uparrow \beta''_1 \uparrow \mid \{Ação_2\} \uparrow \beta''_2 \uparrow \mid \ldots \mid \{Ação_m\} \uparrow \beta''_m \uparrow)$$

$$\beta'_{i\,1}\quad \beta'_{i\,2}\quad \beta'_{in}$$

onde cada $\beta''_i = \uparrow \beta_{i\,1} \uparrow \beta_{i\,2} \uparrow \ldots\ldots \uparrow \beta_{in} \uparrow$, para $1 \le i \le m$

$$\{Ação_1''\}\ A_1^e\ \{Ação_2''\}\ A_2^e\ \{Ação_p''\}\ A_p^e$$

$$c = (\{Ação_1''\} \uparrow \gamma''_1 \uparrow \mid \{Ação_2''\} \uparrow \gamma''_2 \uparrow \mid \ldots \mid \{Ação_p''\} \uparrow \gamma''_p \uparrow)$$

$$\gamma'_{i\,1}\quad \gamma'_{i\,2}\quad \gamma'_{in}$$

onde cada $\gamma''_i = \uparrow \gamma_{i\,1} \uparrow \gamma_{i\,2} \uparrow \ldots\ldots \uparrow \gamma_{in} \uparrow$, para $1 \le i \le p$

$$\{Ação_1'''\}\quad \{Ação_2'''\}\quad \{Ação_q'''\}$$

$d = (\{Ação_1'''\}\uparrow \delta''_1 \uparrow \mid \{Ação_2'''\} \uparrow \quad \delta''_2 \uparrow \quad \mid ... \mid \{Ação_q'''\}\ \delta''_q \uparrow)$

Segundo esta fórmula, pode-se fatorar em uma única expressão todas as alternativas para o não-terminal A, eliminando as auto-recursões.

$$A \rightarrow [\ \{Ação_1\}\ A_1^d\ \{Ação_2\}\ A_2^d\ \{Ação_m\}\ A_m^d$$
$$(\uparrow(\uparrow \varepsilon \uparrow \setminus \{Ação_1\}\uparrow \beta''_1 \uparrow \mid \{Ação_2\}\uparrow \quad \beta''_2 \uparrow \mid ... \mid \{Ação_m\}\uparrow \quad \beta''_m \uparrow)$$

$$\{Ação_1'\}\ A_1\ \{Ação_2'\}\ A_2\ \{Ação_k'\}\ A_k$$
$$\uparrow(\{Ação_1'\}\uparrow \alpha''_1 \uparrow \mid \{Ação_2'\} \uparrow \alpha''_2 \uparrow \mid ... \mid \{Ação_k'\} \uparrow \alpha''_k \uparrow) \uparrow$$

$$\{Ação_1''\}\ A_1^e\ \{Ação_2''\}\ A_2^e\ \{Ação_p''\}\ A_p^e$$
$$(\uparrow \varepsilon \uparrow \setminus \{Ação_1''\}\uparrow \gamma''_1 \uparrow \mid \{Ação_2''\}\uparrow \quad \gamma''_2 \uparrow \quad \mid ... \mid \{Ação_p''\}\gamma''_p\uparrow) \uparrow$$

$$Ação_1'''\}\ \{Ação_2'''\}\ \{Ação_q'''\}\ ]$$
$$\setminus\ (\{Ação_1'''\} \uparrow \delta''_1 \uparrow \mid \{Ação_2'''\} \quad \uparrow \quad \delta''_2 \uparrow \mid ... \mid \{Ação_q'''\}\delta''_q \uparrow) \uparrow$$

6. Eliminar não-determinismos

Consideram-se os seguintes casos:

a) Quando os prefixos de duas ou mais expressões são comuns, colocam-se em evidência os prefixos comuns mais longos possível e repete-se esta operação enquanto existirem agrupamentos com esta característica. Por exemplo: Dadas duas opções rotuladas, com prefixos comuns $\alpha_1, \alpha_2, \ldots. \alpha_n$, tem-se:

$R_0\ R_1\ R_2\ R_{n-1}\ R_n\ T_1\ T_2\ T_{p-1}\ T_p$

$\uparrow \alpha_1 \uparrow \alpha_2 \uparrow .... \uparrow \alpha_n \uparrow \beta_1 \uparrow \beta_2 \uparrow ... \uparrow \beta_p \uparrow$

$S_0\ S_1\ S_2\ S_{n-1}\ S_n\ U_1\ U_2\ U_{q-1}\ U_q$

$\uparrow \alpha_1 \uparrow \alpha_2 \uparrow .... \uparrow \alpha_n \uparrow \gamma_1 \uparrow \gamma_2 \uparrow ... \uparrow \gamma_p \uparrow$

Nesta expressão, temos:

$R_0 \quad R_1 \quad R_2 \ldots R_m = S_0 \quad S_1 \quad S_2 \quad \ldots \quad S_m \quad , 0 < m \leq n$

Obtemos, então a expressão fatorada seguinte:

$R_0 \quad R_1 \quad R_2 \quad R_{m-1} \quad R_m$

$(\uparrow \alpha_1 \uparrow \alpha_2 \uparrow \ldots. \uparrow \alpha_m \uparrow \alpha_{m+1} \uparrow \ldots.. \uparrow \alpha_n \uparrow)$

$R_{m+1} \ldots. R_n \; T_1 \quad T_2 \quad T_{p-1} \quad T_p$

$(\uparrow \beta_1 \uparrow \beta_2 \uparrow \ldots \uparrow \beta_p \uparrow \; | \;$

$R_{m+1} \ldots. R_n \; U_1 \quad U_2 \quad U_{q-1} \quad U_q$

$\uparrow \gamma_1 \uparrow \gamma_2 \uparrow \ldots \uparrow \gamma_p \uparrow \;) \uparrow$

b) Para evitar não-determinismos, quando surgir um não-teminal na extremidade esquerda da expressão, substitui-se este símbolo pela expressão que o define. Por exemplo, dada a expressão seguinte:

$R \quad R_0 \quad R_1 \quad R_2 \quad R_{n-1} \quad R_n$

$\uparrow N \uparrow \mu_1 \uparrow \mu_2 \uparrow \ldots\ldots\ldots \uparrow \mu_n \uparrow$

onde $N = \delta$.

Substituindo N por δ, tem-se:

$R \quad R_0 \quad R_1 \quad R_2 \quad R_{n-1} \quad R_n$

$\uparrow (\delta) \uparrow \mu_1 \uparrow \mu_2 \uparrow \ldots\ldots\ldots \uparrow \mu_n \uparrow$

c) Quando existir um ε (cadeia vazia) como uma das alternativas para o agrupamento. Por exemplo, em uma expressão deste tipo:

$R_0 \quad R_1 \; R_2 \quad R_3 \quad R_4 \quad R_5 \; R_6 \quad R_7$

$\uparrow \alpha \uparrow (\uparrow \varepsilon \uparrow \; | \; \uparrow \mu \uparrow) \uparrow \beta \uparrow$

Neste caso, a expressão se transforma em algo equivalente a:

$R_0 \quad R_1 R_2 R_3 R_6 \quad R_7 \; R_0 \; R_1 R_4 \quad R_5 R_6 \quad R_7$

$\uparrow \alpha \uparrow \beta \uparrow \; | \uparrow \alpha \uparrow \mu \uparrow \beta \uparrow$

d) Quando existe uma construção cíclica na extremidade esquerda de uma das alternativas. Por exemplo, dada a expressão cíclica seguinte:

$R_0 \quad R_1 \quad R_2 \quad R_3 \quad R_4 \quad R_5$

$\uparrow (\quad \uparrow \varepsilon \quad \uparrow \quad \backslash \quad \uparrow \quad \alpha \quad \uparrow) \uparrow$

pode-se substituir pela expressão:

$R_0 \; R_1 \quad R_2 R_5 \quad R_0 R_1 R_2 R_3 \quad R_4 \quad R_3 \quad R_4 \quad R_5$

$\uparrow \quad \varepsilon \quad \uparrow \quad | \quad \uparrow \quad \alpha \quad \uparrow \quad (\uparrow \varepsilon \uparrow \backslash \uparrow \quad \alpha \uparrow \quad) \uparrow$

7. Dar continuidade à eliminação de todos os não-determinismos, até não existir mais nenhum caso, ou, caso isto não seja possível, repetir o processo um número suficiente de vezes, até garantir determinismo para todas as situações de interesse.

4.4.Algoritmo para a construção do autômato

Este algoritmo, desenvolvido em sua versão livre de contexto em [Jos93], transforma a expressão obtida com o algoritmo anterior em um autômato determinístico. Apresentamos neste tópico uma nova versão deste algoritmo, aplicado ao caso da gramática adaptativa. Para não sobrecarregar demais a notação, serão omitidos os rótulos. As ações adaptativas apresentadas nas fórmulas são todas opcionais e podem ser omitidas.

a. Identificar os agrupamentos entre parênteses da expressão.

b. Tratar um agrupamento isoladamente. Começando pelos níveis mais externos, atribuir novos números aos pontos extremos das expressões que compõem cada uma das suas opções.

c. Se existirem números r e/ou s já atribuídos aos estados correspondentes aos pontos imediatamente à esquerda ou à direita do agrupamento entre parênteses, de uma das formas:

$$\uparrow(\{\text{Ação\_}a_1\}\uparrow a_1\uparrow|\ldots.|\{\text{Ação\_}a_m\}\uparrow a_m\uparrow)\uparrow$$
$$r \qquad\qquad\qquad\qquad\qquad\qquad s$$

ou $\uparrow(\{\text{Ação\_}a_1\}\uparrow a_1\uparrow|\ldots.|\{\text{Ação\_}a_m\}\uparrow a_m\uparrow \setminus \{\text{Ação\_}b_1\}\uparrow b_1\uparrow|\ldots..|\{\text{Ação\_}b_n\}\uparrow b_n\uparrow)\uparrow$
$r \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad s$

consideram-se os seguintes casos:

Se existir r, fazer x = r, senão, associa-se a x um número correspondente a um novo estado.

Atribuir x aos pontos extremos esquerdos de todas as opções internas ao agrupamento.

Se existir s, fazer y = s, senão, associa-se a y um número correspondente a um novo estado.

Atribuir y aos pontos extremos direitos de todas as opções internas ao agrupamento.

d. Para agrupamentos do seguinte tipo:

$$\uparrow(\{\text{Ação\_}\varepsilon\}\uparrow\varepsilon\uparrow \setminus \{\text{Ação\_}b_1\}\uparrow b_1\uparrow|\ldots..|\{\text{Ação\_}b_n\}\uparrow b_n\uparrow)\uparrow$$
$$r \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad s$$

para cada extremidade de uma expressão b_i, criar um único estado x = y , que será associado a cada uma das extremidades de tais opções.

fica

$$\uparrow(\{\text{Ação\_}\varepsilon\}\uparrow\varepsilon\uparrow \setminus \{\text{Ação\_}b_1\}\uparrow b_1\uparrow|\ldots..|\{\text{Ação\_}b_n\}\uparrow b_n\uparrow)\uparrow$$
$$r \qquad\quad x\ \ y \qquad\qquad y \qquad x \qquad\qquad\qquad y \quad x\ \ s$$

e. Para os demais casos, sempre que for preciso criar novos estados, gerar sempre estados x e y distintos.

f. Numerar os estados extremos de cada uma das epressões que compõem o agrupamento, obtendo-se uma das seguintes configurações. Conforme o caso:

$$\uparrow (\{Ação\_a_1\}\uparrow a_1 \uparrow | \ldots . | \{Ação\_a_m\}\uparrow a_m \uparrow) \uparrow$$
$$r \quad x \quad y \quad x \quad y \quad s$$

$$\uparrow (\{Ação\_a_1\}\uparrow a_1\uparrow | \ldots . | \{Ação\_a_m\}\uparrow a_m \uparrow \setminus \{Ação\_b_1\}\uparrow \ b_1 \ \uparrow | \ldots .. | \{Ação\_b_n\}\uparrow \ b_n \uparrow) \uparrow$$
$$r \quad x \quad y \quad x \quad y \quad y \quad x \quad y \quad x \quad s$$

ou com x = y

$$\uparrow (\{Ação\_\varepsilon\} \uparrow \varepsilon \uparrow \setminus \{Ação\_b_1\}\uparrow \ b_1 \ \uparrow | \ldots .. | \{Ação\_b_n\}\uparrow \ b_n \uparrow) \uparrow$$
$$r \quad x \quad y \quad y \quad x \quad y \quad x \quad s$$

g. Reproduzir o estado x para o ponto externo imediatamente à esquerda do agrupamento entre parênteses caso ainda não tenha sido nomeado nenhum estado neste ponto da expressão:

$$\uparrow (\{Ação\_a_1\}\uparrow a_1 \uparrow | \ldots . | \{Ação\_a_m\}\uparrow a_m \uparrow \setminus \{Ação\_b_1\} \uparrow \ b_1 \ \uparrow | \ldots .. | \{Ação\_b_n\}\uparrow \ b_n \uparrow) \uparrow$$
$$x \quad x \quad y \quad x \quad y \quad y \quad x \quad y \quad x$$

h. Reproduzir y para o ponto externo imediatamente à direita do agrupamento entre parênteses caso ainda não tenha sido nomeado nenhum estado neste ponto da expressão:

$$\uparrow (\{Ação\_a_1\}\uparrow a_1 \uparrow | \ldots . | \{Ação\_a_m\}\uparrow a_m \uparrow \setminus \{Ação\_b_1\}\uparrow \ b_1 \ \uparrow | \ldots .. | \{Ação\_b_n\}\uparrow \ b_n \uparrow) \uparrow$$
$$x \quad y \quad x \quad y \quad y \quad x \quad y \quad x \quad y$$

i. Se em ambos os pontos externos à direita e à esquerda do agrupamento ainda não tenha sido atribuído nenhum estado, então fazer as duas atribuições citadas anteriormente. Assim, obtém-se:

$$\uparrow (\uparrow a_1 \{Ação\_a_1\}\uparrow | \ldots . | \{Ação\_a_m\}\uparrow a_m \uparrow \setminus \{Ação\_b_1\}\uparrow \ b_1 \ \uparrow | \ldots .. | \{Ação\_b_n\}\uparrow \ b_n \uparrow) \uparrow$$

x x y x y y x y x y

j. Se existir um caso particular, em que a_1 tenha somente uma opção:

↑ ({Ação_a_1}↑ a_1 ↑ \ {Ação_b_1}↑ b_1 ↑ | .....| {Ação_b_n}↑ b_n ↑) ↑
x x y y x y x y

e a expressão a_1 tenha um agrupamento cíclico, na sua extremidade esquerda, do seguinte tipo:

↑ ({Ação_c_1}↑ c_1 ↑ | .... | {Ação_c_p}↑ c_p ↑ \ {Ação_d_1} ↑ d_1 ↑ | .....| {Ação_d_q}↑ d_q ↑) ..
x

então pode-se reproduzir o estado x para o interior deste agrupamento cíclico:

↑ ({Ação_c_1}↑ c_1 ↑ | .... | {Ação_c_p}↑ c_p ↑ \ {Ação_d_1}↑ d_1 ↑ | .....| {Ação_d_q}↑ d_q ↑) ...
x x x x x

Se a expressão a_1 tiver um agrupamento cíclico na sua extremidade direita:

↑ ({Ação_c_1}↑ c_1 ↑ | .... | {Ação_c_p}↑ c_p ↑ \{Ação_d_1} v↑ d_1 ↑ | .....| {Ação_d_q}↑ d_q ↑) ↑
y

então pode-se reproduzir o estado y para o interior deste agrupamento cíclico:

↑ ({Ação_c_1}↑ c_1 ↑ | .... | {Ação_c_p}↑ c_p ↑ \ {Ação_d_1}↑ d_1 ↑ | .....| {Ação_d_q}↑ d_q ↑) ↑
y y y y y

Se estas condições forem verdadeiras em uma mesma expressão:

↑ ({Ação_c_1} ↑ c_1 ↑ | .... | {Ação_c_p}↑ c_p ↑ \ {Ação_d_1}↑ d_1 ↑ | .....| {Ação_d_q}↑ d_q ↑) ↑
x

então pode-se reproduzir os estados x e y para o interior deste agrupamento cíclico, obtendo-se:

↑ ({Ação_c_1}↑ c_1 ↑ | .... | {Ação_c_p}↑ c_p ↑ \ {Ação_d_1}↑ d_1 ↑ | .....| {Ação_d_q}↑ d_q ↑) ↑
x x y x y y x y x y

k. De forma semelhante ao que foi descrito anteriormente, reproduzir para o interior de todos os agrupamentos (não cíclicos) que ainda não foram tratados, os estados previamente associados às extremidades direita e esquerda dos agrupamentos.

l. Completar a nomeação dos estados faltantes atribuindo diferentes novos estados, de forma arbitrária, àqueles que ainda não foram tratados.

Obtenção das regras de transição do autômato

Uma vez que as expressões foram tratadas para receberem as nomeações dos estados, é preciso interpretá-las e transformá-las em regras de transição para o autômato. Para isso, são dados a seguir os procedimentos necessários para executar estas operações.

1. Para cada ocorrência do símbolo que representa a cadeia vazia ε:

$$\uparrow \varepsilon \uparrow$$
$$x \quad y$$

criar uma transição em vazio de x para y:

$$(\gamma, x, \alpha): \rightarrow (\gamma, y, \alpha)$$

2. Para cada ocorrência de terminais t:

$$\uparrow \quad t \uparrow$$
$$x \quad\quad y$$

criar uma transição interna do estado x para y com o consumo do terminal t:

$$(\gamma, x, t\,\alpha) : \rightarrow (\gamma, y, \alpha)$$

3. Para cada ocorrência de não-terminais N:

$$\uparrow \quad N \uparrow$$
$$x \quad\quad y$$

criar uma transição em vazio de x para o estado inicial n_0 da submáquina associada ao não-terminal N, empilhando y na pilha do autômato para ser utilizado na execução da transição de retorno

$$(\gamma, x, \alpha): \rightarrow (\gamma y, n_0, \alpha)$$

5. Para cada agrupamento cíclico do tipo:

$$\uparrow(\uparrow \varepsilon \uparrow \ \backslash \uparrow \mu \uparrow)\uparrow$$

x y z t u v

criar as produções seguintes:

$(\gamma, x, \alpha): \rightarrow (\gamma, y, \alpha)$

$(\gamma, z, \alpha): \rightarrow (\gamma, v, \alpha)$

$(\gamma, z, \alpha): \rightarrow (\gamma, t, \alpha)$

$(\gamma, t, \alpha): \rightarrow (\gamma, v, \alpha)$

$(\gamma, u, \alpha): \rightarrow (\gamma, t, \alpha)$

6. Para cada estado n_f que está associado ao final de alguma das alternativas que definem o não-terminal N, criar uma transição em vazio que execute o retorno ao estado y da sub-maquina chamadora, que tinha sido empilhado anteriormente no item 3.

$(\gamma\ y, n_f, \alpha): \rightarrow (\gamma, y, \alpha)$

7. O estado inicial da sob-máquina associada ao não-terminal N é o estado que está na extremidade esquerda da expressão que define N.

8. Os estados associados as início e ao final da expressão que define a raiz da gramática correspondem aos estados inicial e final do autômato, respectivamente.

Construção das regras de mapeamento

Uma vez construídas as expressões que representam a gramática, e feita a construção do autômato, é possível passar à interpretação dos rótulos definidos no algoritmo do item 4.3. Aplica–se a tabela abaixo quando um rótulo estiver sendo utilizado.

Esta tabela é composta por quatro colunas. A primeira que indica o rótulo que está sendo utilizado, a segunda coluna apresenta a saída que deve ser empregada, na terceira tem-se os movimentos da pilha do transdutor e, por último, encontra-se a cadeia gerada.

A_i^e corresponde aos não-terminais A_i aos quais se associam as regras auto-recursivas à esquerda

A_i^d corresponde aos não-terminais A_i aos quais se associam as regras auto-recursivas à direita

A_i corresponde aos não-terminais A_i aos quais se associam as demais produções

Os símbolos $\uparrow$ e $\downarrow$ indicam desempilhamento e empilhamento dos símbolos na pilha de transudção

Os símbolos $\Downarrow\alpha$ e $\Uparrow\alpha$ indicam a inserção e a remoção de informações de contexto na geração da sentença

O símbolo π denota todos os símbolos da pilha de transdução, compreendidos entre o topo e o primeiro delimitador], exclusive.

Símbolos não-terminais são denotados por A

Símbolos terminais são denotados por σ

A cadeia vazia é denotada por ε

O índice i indica a i-ésima produção da gramática original.

Rótulo	Saída	Pilha
ε	( )	
σ	(σ)	
A_i^e	A_i^e)	
A_i	A_i)π	$\uparrow \pi$
A_i^d	(	$\downarrow$)A_i^d
[	[(	$\downarrow$]
]	π]	$\uparrow \pi$]
$\Downarrow\alpha$	$\Downarrow\alpha$	
$\Uparrow\alpha$	$\Uparrow\alpha$	
{Ação$_i$}	executa a ação adaptativa	

A aplicação desta tabela é feita da seguinte maneira:

Seja dada uma regra de produção que apresenta um rótulo $R = r_1\ r_2....r_n$ associado ao seu estado destino. Para cada símbolo r_i do rótulo, toma-se a linha correspondente na tabela acima e verifica-se a existência da indicação da alteração da saída (s_i) e da pilha de

transdução (p_i).

A interpretação completa do rótulo gera uma alteração na saida S, obtida pela concatenação dos símbolos s_i e uma alteração na pilha de transdução P, obtida pela concatenação dos símbolos p_i.

A presença de uma ação adaptativa nos rótulos implica na alteração da gramática e, portanto, na modificação das regras de produção e conseqüentemente das expressões rotuladas. O método mais simples e canonico para este caso é aplicar novamente o algoritmo 4.3 para obter a expressão correspondente da gramática alterada.

Finalizado o processo, obtém-se a regra de mapeamento:

$(\gamma, \lambda) \rightarrow (\gamma P, \lambda S)$

O lado esquerdo da regra de mapeamento, (γ, λ), indica a situação da pilha do transdutor e da cadeia de saida, antes da aplicação da regra.

O lado direito da regra de mapeamento, (γP, λS), indica a situação da pilha do transdutor e da cadeia de saida, após a aplicação da regra, onde P e S foram concatenadas a direita de γ e λ, respectivamente.

A seguir é apresentado um exemplo de aplicação dos algoritmos 4.3 e 4.4.

Exemplo:

L= {ω | ω é alguma permutação de { a, b, c}}

A gramática inicial que gera a linguagem é dada por

$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0 , S)$

$V_N^0 = \{ S, L, M, N\}$

$V_T = \{ a, b, c \}$

$V_C = P_D = \phi$

P_L^0 é formada pelas regras de produção:

1^0. $S \rightarrow LMN$

2^0. $L \rightarrow \{A\ (a, M, N)\}\ a$

3^0. $L \rightarrow \{A\ (b, M, N)\}\ b$

4^0. $L \rightarrow \{A\ (c, M, N)\}\ c$

5^0. $M \rightarrow \{B\ (a, N)\}\ a$

6^0. $M \rightarrow \{B\ (b, N)\}\ b$

7^0. $M \rightarrow \{B\ (c, N)\}\ c$

8^0. $N \rightarrow a$

9^0. $N \rightarrow b$

10^0. $N \rightarrow c$

O conjunto das ações adaptativas é o seguinte:

T = {

A (α, x, y) = {

- $[x \rightarrow v \{B\ (\alpha, N)\}\ \alpha]$
- $[y \rightarrow \alpha]$

}

B (α, x) = { - $[x \rightarrow \alpha]$ }

}

Podemos ilustrar o funcionamento desta gramática adaptativa através da geração de uma sentença qualquer.

Ao se iniciar a geração da sentença, a gramática é formada por 3 tipos de regras que definem os não terminais L, M e N.

O primeiro tipo de regra que define o não-terminal L possui, no seu lado direito, um símbolo terminal a ou b ou c, e chama a ação adaptativa `A` .

O segundo tipo define o não-terminal M e possui, no seu lado direito, um símbolo terminal a ou b ou c, e chama a ação adaptativa `B` .

O terceiro tipo define o não-terminal N e possui, no seu lado direito, um símbolo terminal a ou b ou c.

Quando a gramática inicia a geração de uma sentença, por exemplo

$$\omega = cba,$$

ela parte do símbolo inicial S, aplicando a regra $S \rightarrow LMN$. Em seguida, a derivação da sentença busca a substituição do primeiro símbolo não-terminal mais à esquerda L.

A regra escolhida para gerar a sentença é $L \rightarrow \{A\ (c, M, N)\}\ c$. Esta regra possui uma ação adaptativa associada `A` (c, M, N) que, quando executada, transforma a gramática G^0 em G^1, alterando o conjunto de regras de produção pela eliminação das regras que possuem um padrão específico (as que possuem como lado direito o símbolo c), e, desta forma, a gramática impede que estas regras sejam utilizadas em derivações futuras.

A ação adaptativa `A` executada é assim instanciada da seguinte maneira:

`A` (c, M, N) = { - [$M \rightarrow \{$`B` (c, N)$\}$ c]

- [$N \rightarrow c$]

}

Desta forma, a nova gramática G^1 apresenta as seguintes regras de produção:

1^0. $S \rightarrow LMN$

2^0. $L \rightarrow \{$`A` (a, M, N)$\}$ a

3^0. $L \rightarrow \{$`A` (b, M, N)$\}$ b

4^0. $L \rightarrow \{$`A` (c, M, N)$\}$ c

5^0. $M \rightarrow \{$`B` (a, N)$\}$ a

6^0. $M \rightarrow \{$`B` (b, N)$\}$ b

8^0. $N \rightarrow a$

9^0. N → b

O próximo passo consiste na geração do símbolo b. Para isto é preciso substituir o símbolo não-terminal M. Neste caso, existem somente duas opções: as regras 5^0 e 6^0.
Escolhe-se a regra 6^0. A ação adaptativa `B` (b, N) é executada, eliminando a regra N → b, restringindo ainda mais as opções para o próximo passo de derivação.
Na nova gramática G^2 restam portanto as seguintes regras de produção:

1^0. S → LMN

2^0. L → {`A` (a, M, N)} a

3^0. L → {`A` (b, M, N)} b

4^0. L → {`A` (c, M, N)} c

5^0. M → {`B` (a, N)} a

6^0. M → {`B` (b, N)} b

8^0. N → a

O último passo de derivação consiste na substituição do símbolo não-terminal N, e neste caso, resta somente uma opção, que é a regra 8^0, que não possui nenhuma ação adaptativa associada. A regra 8^0, definida como N → a, gera portanto o último símbolo da cadeia, e assim, termina a geração desta sentença.

Exemplo de obtenção do analisador sintático para esta linguagem

	[S_1^0]
1^0. S → LMN	S → ↑ L ↑ M ↑ N ↑
	[{`A` (a, M, N)} a L_2^0]
2^0. L → {`A` (a, M, N)} a	L → {`A` (a, M, N)}↑ a ↑
	[{`A` (b, M, N)} b L_3^0]
3^0. L → {`A` (b, M, N)} b	L → {`A` (b, M, N)}↑ b ↑
	[{`A` (c, M, N)} c L_4^0]
4^0. L → {`A` (c, M, N)} c	L → { `A` (c, M, N)}↑ c ↑

5^0. M → {B (a, N)} a	[{B (a, N)} a M_5^0] M → {B (a, N)}↑ a ↑
6^0. M → {B (b, N)} b	[{B (b, N)} b M_6^0] M → {B (b, N)}↑ b ↑
7^0. M → {B (c, N)} c	[{B (c, N)} c M_7^0] M → {B (c, N)}↑ c ↑
8^0. N → a	[a N_8^0] N → ↑ a ↑
9^0. N → b	[b N_9^0] N → ↑ b ↑
10^0. N → c	[c N_{10}^0] N → ↑ c ↑

Agrupando as regras 2^0. 3^0. 4^0., temos

[{A (a, M, N)} a L_2^0] [{A (b, M, N)} b L_3^0] [{A (c, M, N)} c L_4^0]

L → {A (a, M, N)}↑ a ↑ | {A (b, M, N)} ↑ b ↑ | {A (c, M, N)} ↑ c ↑

Agrupando as regras 5^0. 6^0. 7^0., temos

[{B (a, N)} a M_5^0] [{B (b, N)} b M_6^0] [{B (c, N)} c M_7^0]

M → {B (a, N)}↑ a ↑ | {B (b, N)} ↑ b ↑ | {B (c, N)} ↑ c ↑

Agrupando as regras 8^0. 9^0. 10^0., temos

[a N_8^0] [b N_9^0] [c N_{10}^0]

N → ↑ a ↑ | ↑ b ↑ | ↑ c ↑

Substituindo as três expressões obtidas em S, temos:

$$
\begin{array}{l}
\quad [\quad [\{\mathtt{A}\ (a, M, N)\}\ \ a\ L_2^{\,0}\ ]\ [\{\mathtt{A}\ (b, M, N)\}\ \ b\ L_3^{\,0}\ ]\ [\{\mathtt{A}\ (c, M, N)\}\ \ c\ L_4^{\,0}\ ] \\
S \rightarrow \uparrow\ (\{\mathtt{A}\ (a, M, N)\} \uparrow a \uparrow \quad | \ \{\mathtt{A}\ (b, M, N)\} \uparrow b \uparrow \quad | \ \{\mathtt{A}\ (c, M, N)\} \uparrow c \uparrow \quad) \\
\quad [\{\mathtt{B}\ (a, N)\}\ \ a\ M_5^{\,0}\ ]\ [\{\mathtt{B}\ (b, N)\}\ \ b\ M_6^{\,0}\ ]\ [\quad \{\mathtt{B}\ (c, N)\}\ \ c\ M_7^{\,0}\ ] \\
\quad (\{\mathtt{B}\ (a, N)\} \uparrow\ a \uparrow \quad | \ \{\mathtt{B}\ (b, N)\} \uparrow b \uparrow \quad | \ \{\mathtt{B}\ (c, N)\} \quad \uparrow c \uparrow) \\
\quad [\ a\ N_8^{\,0}\ ] \quad [\ b\ N_9^{\,0}\ ] \quad [\ c\ N_{10}^{\,0}\ ] \quad S_1^{\,0}\ ] \\
\quad (\ \ \uparrow a \uparrow \quad | \ \uparrow b \uparrow \quad | \ \uparrow c \uparrow \quad)\uparrow
\end{array}
$$

A expressão obtida para S define a gramática adaptativa de modo compacto e contém todas as informações importantes sobre a origem dos símbolos na gramática.

O esquema a seguir faz uma simulação da varredura da expressão para a sentença cba

Rótulo	Saída	Pilha	Entrada
[	[(	]	
[	[(	]]	
c	(c)	]]	c
$L_4^{\,0}$	$L_4^{\,0}$)	]]	
]	]	]	
A (c, M, N)			

Neste momento é encontrada uma ação adaptativa e a gramática é alterada, sem a presença das regras 7^0. e 10^0..

Novamente a expressão deve ser obtida através das operações acima indicadas.

G^1 :

1^0. $S \rightarrow LMN$	[$S_1^{\,0}$] $S \rightarrow \uparrow L \uparrow M \uparrow N \uparrow$
2^0. $L \rightarrow \{\mathtt{A}\ (a, M, N)\}\ a$	[{A (a, M, N)} a $L_2^{\,0}$] $L \rightarrow \{\mathtt{A}\ (a, M, N)\} \uparrow a \uparrow$
3^0. $L \rightarrow \{\mathtt{A}\ (b, M, N)\}\ b$	[{A (b, M, N)} b $L_3^{\,0}$] $L \rightarrow \{\mathtt{A}\ (b, M, N)\} \uparrow\ b \uparrow$
	[{A (c, M, N)} c $L_4^{\,0}$]

4^0. L → {A (c, M, N)} c	L → {A (c, M, N)}↑ c ↑
5^0. M → {B (a, N) } a	[{B (a, N)} a M_5^0] M → {B (a, N)}↑ a ↑
6^0. M → {B (b, N)} b	[{B (b, N)} b M_6^0] M → {B (b, N)}↑ b ↑
8^0. N → a	[a N_8^0] N → ↑ a ↑
9^0. N → b	[b N_9^0] N → ↑ b ↑

Agrupando as regras 2^0. 3^0. 4^0., temos

[{A (a, M, N)} a L_2^0] [{A (b, M, N)} b L_3^0] [{A (c, M, N)} c L_4^0]

L → {A (a, M, N)}↑ a ↑ | {A (b, M, N)} ↑ b ↑ | {A (c, M, N)} ↑ c ↑

Agrupando as regras 5^0. 6^0., temos

[{B (a, N)} a M_5^0] [{B (b, N) } b M_6^0]

M →{B (a, N)} ↑ a ↑ | {B (b, N) } ↑ b ↑

Agrupando as regras 8^0. 9^0., temos

[a N_8^0] [b N_9^0]

N → ↑ a ↑ | ↑ b ↑

Substituindo as três expressões obtidas em S, temos:

[[{A (a, M, N)} a L_2^0] [{A (b, M, N)} b L_3^0] [{A (c, M, N)} c L_4^0]

S → ↑ ({A (a, M, N)}↑ a ↑ | {A (b, M, N)} ↑ b ↑ | {A (c, M, N)} ↑ c ↑)

[{B (a, N)} a M_5^0] [{B (b, N)} b M_6^0]

({B (a, N)}↑ a ↑ | {B (b, N)} ↑ b ↑)

[a N_8^0] [b N_9^0] S_1^0]

(↑ a ↑ | ↑ b ↑) ↑

Continuando a geração da sentença temos novamente a simulação dos passos do transdutor

Rótulo	Saída	Pilha	Entrada
[	[(	]]	
b	(b)	]]	b
M_6^0	M_6^0)	]]	
]	]	]	
B (b, N)			

Encontra-se novamente uma ação adaptativa, que altera o conjunto de regras de produção.

Esta ação remove a regra 9^0.

A expressão deve ser refeita novamente.

G^2:

	[S_1^0]
1^0. S → LMN	S → ↑ L ↑ M ↑ N ↑
	[{A (a, M, N)} a L_2^0]
2^0. L → {A (a, M, N)} a	L → {A (a, M, N)}↑ a ↑
	[{A (b, M, N)} b L_3^0]
3^0. L → {A (b, M, N)} b	L → {A (b, M, N)}↑ b ↑
	[{A (c, M, N)} c L_4^0]
4^0. L → {A (c, M, N)} c	L → {A (c, M, N)}↑ c ↑
	[{B (a, N)} a M_5^0]
5^0. M → {B (a, N)} a	M →{B (a, N)} ↑ a ↑
	[{B (b, N)} b M_6^0]
6^0. M → {B (b, N)} b	M → {B (b, N)}↑ b ↑
	[a N_8^0]
8^0. N → a	N → ↑ a ↑

[[{A (a, M, N)} a L_2^0] [{A (b, M, N)} b L_3^0] [{A (c, M, N)} c L_4^0]

S $\rightarrow$ ↑ ({A (a, M, N)}↑ a ↑ | {A (b, M, N)} ↑ b ↑ | {A (c, M, N)} ↑ c ↑)

[{B (a, N) } a M_5^0] [{B (b, N)} b M_6^0]

({B (a, N) }↑ a ↑ | {B (b, N)} ↑ b ↑)

[a N_8^0] S_1^0]

(↑ a ↑) ↑

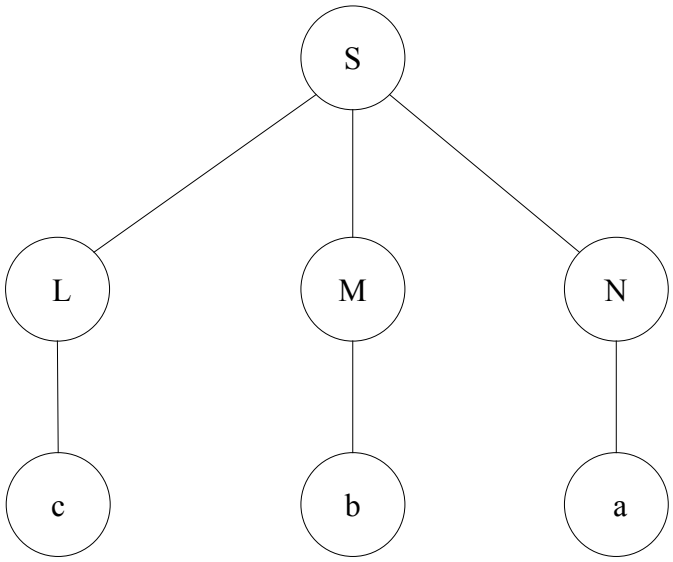
Rótulo	Saída	Pilha	Entrada
[	[(	]]	
a	(a)	]]	a
N_8^0	N_8^0)	]]	
]	]	]	
S_1^0	S_1^0)	]	
]	]		

A saída completa é portanto:

[([((c) L_4^0)] [((b) M_6^0)] [((a) N_8^0)] S_1^0)]

Esta saída devidamente diagramada é apresentada a seguir:

[(
  [((c) L_4^0)
  ]
    [((b) M_6^0)
    ]
     [((a) N_8^0)
     ] S_1^0)
]



Exemplo de utilização do transdutor sintático para gramáticas adaptativas

O objetivo deste exemplo é o de apresentar o funcionamento de algoritmo de transdução quando a gramática utiliza informações de contexto.

A linguagem gerada por esta gramática é formada por sentenças do tipo z, xzt, xxzty e $x^n xzty^n$, onde n>0

Seja G uma gramática adaptativa constituída por uma tripla ordenada (G^0, T, R^0), onde

$T = \phi$,

$G^0 =$ ($V_N{}^0$, V_T, V_C, $P_L{}^0$, $P_D{}^0$, S) é a gramática inicial, onde

$V_N{}^0 = \{ S, A, B, C, D, E, F, G, H, I, J, K, L\}$

$V_T = \{ x, z, t, y \}$

$V_C = \{ \alpha, \beta \}$

$P_L{}^0 \cup P_D{}^0 = \{$

$S \rightarrow AB$	$A \rightarrow xE$
$\alpha B \rightarrow C$	$E \rightarrow AF$
$\beta B \rightarrow D$	$\alpha F \rightarrow G$
$C \rightarrow \varepsilon$	$\beta F \rightarrow H$
$D \rightarrow \varepsilon$	$G \rightarrow yI$
	$H \rightarrow t\ I$
	$I \leftarrow \alpha\ J$
	$A \rightarrow zK$
	$K \leftarrow \beta L$
	$J \rightarrow \varepsilon$
	$L \rightarrow \varepsilon$

}

Exemplos de derivações de sentenças:

$S \Rightarrow AB \Rightarrow xEB \Rightarrow xAFB \Rightarrow xzKFB \Rightarrow xz\beta LFB \Rightarrow xz\beta\varepsilon FB \Rightarrow xzHB \Rightarrow xztIB \Rightarrow xzt\alpha JB \Rightarrow xzt\alpha\varepsilon B \Rightarrow xztC \Rightarrow xzt\varepsilon$

$S \Rightarrow AB \Rightarrow zKB \Rightarrow z\ \beta LB \Rightarrow z\ \beta\varepsilon B \Rightarrow z\ D \Rightarrow z\ \varepsilon$

Designação dos rótulos para cada regra enumerada univocamente.

1. $S \rightarrow AB$	$[\quad S_1]$ $S \rightarrow \uparrow A \uparrow B \uparrow$
2. $\alpha B \rightarrow C$	$[\quad \Uparrow\alpha \quad B_2]$ $\alpha B \rightarrow \uparrow \Uparrow\alpha \uparrow C \uparrow$
3. $\beta B \rightarrow D$	$[\quad \Uparrow\beta \quad B_3]$ $\beta B \rightarrow \uparrow \Uparrow\beta \uparrow D \uparrow$
4. $C \rightarrow \varepsilon$	$[\quad \varepsilon C_4]$ $C \rightarrow \uparrow \varepsilon \uparrow$
5. $D \rightarrow \varepsilon$	$[\quad \varepsilon D_5]$ $D \rightarrow \uparrow \varepsilon \uparrow$
6. $A \rightarrow xE$	$[\quad x \quad A_6]$ $A \rightarrow \uparrow x \uparrow E \uparrow$
7. $E \rightarrow AF$	$[\quad E_7]$ $E \rightarrow \uparrow A \uparrow F \uparrow$
8. $\alpha F \rightarrow G$	$[\quad \Uparrow\alpha \quad F_8]$ $\alpha F \rightarrow \uparrow \Uparrow\alpha \uparrow G \uparrow$
9. $\beta F \rightarrow H$	$[\quad \Uparrow\beta \quad F_9]$ $\beta F \rightarrow \uparrow \Uparrow\beta \uparrow H \uparrow$
10. $G \rightarrow yI$	$[\quad y \quad G_{10}]$ $G \rightarrow \uparrow y \uparrow I \uparrow$
11. $H \rightarrow t\ I$	$[\quad t \quad H_{11}]$ $H \rightarrow \uparrow t \uparrow I \uparrow$
12. $I \leftarrow \alpha\ J$	$[\quad \Downarrow\alpha \quad I_{12}]$ $I \leftarrow \uparrow \Downarrow\alpha \uparrow \quad J \uparrow$
13. $A \rightarrow zK$	$[\quad z \quad A_{13}]$ $A \rightarrow \uparrow z \uparrow \quad K \uparrow$
14. $K \leftarrow \beta L$	$[\quad \Downarrow\beta \quad K_{14}]$ $K \leftarrow \uparrow \Downarrow\beta \uparrow \quad L \uparrow$

15. $J \rightarrow \varepsilon$	$[\ \varepsilon J_{15}]$ $J \rightarrow \uparrow\ \varepsilon \uparrow$
16. $L \rightarrow \varepsilon$	$[\ \varepsilon L_{16}]$ $L \rightarrow \uparrow \varepsilon\ \uparrow$

Substituindo 4 em 2, temos

$[\ \Uparrow\alpha\ [\ \varepsilon C_4]\ B_2]$

$\alpha B \rightarrow \uparrow\ \Uparrow\alpha\ \uparrow\ \varepsilon\ \uparrow$

Substituindo 5 em 3, temos

$[\ \Uparrow\beta\ [\ \varepsilon D_5]\ B_3]$

$\beta B \rightarrow \uparrow\ \Uparrow\beta\ \uparrow\ \varepsilon\ \uparrow$

Agrupando αB com βB, temos

$[\ [\ \Uparrow\alpha\ [\ \varepsilon C_4]\ B_2]\ [\ \Uparrow\beta\ [\ \varepsilon D_5]\ B_3]\ ]$

$B \rightarrow \uparrow (\uparrow\ \Uparrow\alpha \uparrow\ \varepsilon\ \uparrow\ |\ \uparrow \Uparrow\beta\ \uparrow\ \varepsilon\ \uparrow\ ) \uparrow$

Substituindo 16 em 14, temos

$[\ \Downarrow\beta\ [\ \varepsilon L_{16}]\ K_{14}]$

$K \leftarrow\ \uparrow \Downarrow\beta \uparrow\ \varepsilon\ \uparrow$

Substituindo 15 em 12, temos

$[\ \Downarrow\alpha\ [\ \varepsilon J_{15}]\ I_{12}]$

$I \leftarrow\ \uparrow \Downarrow\alpha \uparrow\ \varepsilon\ \uparrow$

Substituindo K em 13

$[\ z\ [\ \Downarrow\beta\ [\ \varepsilon L_{16}]\ K_{14}]\ A_{13}]$

$A \rightarrow \uparrow z \uparrow\ \Downarrow\beta \uparrow\ \varepsilon\ \uparrow$

Substituindo I em 11

$$[\quad t\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\quad I_{12}]\quad H_{11}]$$

$$H \rightarrow \uparrow\ t \uparrow\quad \Downarrow\alpha \uparrow\ \uparrow\ \varepsilon\quad \uparrow$$

Substituindo I em 10

$$[\quad y\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\ \ I_{12}]\ \ G_{10}]$$

$$G \rightarrow \uparrow\ y \uparrow\quad \Downarrow\alpha \uparrow\quad \varepsilon\quad \uparrow$$

Substituindo H em 9

$$[\quad \Uparrow\beta\,[\quad t\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\ \ I_{12}]\ H_{11}]\ F_9]$$

$$\beta F \rightarrow\ \uparrow\ \Uparrow\beta\ \uparrow\quad t \uparrow\quad \Downarrow\alpha \uparrow\quad \varepsilon\quad \uparrow$$

Substituindo G em 8

$$[\quad \Uparrow\alpha\,[\quad y\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\ I_{12}]\ G_{10}]\ F_8]$$

$$\alpha F \rightarrow \uparrow \Uparrow\alpha\ \uparrow\quad y \uparrow\quad \Downarrow\alpha \uparrow\quad \varepsilon\quad \uparrow$$

Agrupando βF e αF, temos

$$[\ [\quad \Uparrow\alpha\ [\quad y\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\ I_{12}]\ G_{10}]\ F_8]$$

$$F \rightarrow\ \uparrow(\uparrow \Uparrow\alpha\ \uparrow\quad y \uparrow\quad \Downarrow\alpha \uparrow\quad \varepsilon\quad \uparrow\quad |$$

$$[\quad \Uparrow\beta\,[\quad t\,[\quad \Downarrow\alpha[\quad \varepsilon J_{15}]\ I_{12}]\ H_{11}]\ F_9]\quad]$$

$$\uparrow \Uparrow\beta\quad \uparrow\quad t \uparrow\ \Downarrow\alpha\ \uparrow\quad \varepsilon\quad \uparrow\quad)\uparrow$$

Substituindo F em E, temos

$$[\quad [\ [\quad \Uparrow\alpha\ [\quad y\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\ \ I_{12}]\ G_{10}]\ F_8]$$

$$E \rightarrow\uparrow\ A \uparrow(\uparrow \Uparrow\alpha\ \uparrow\quad y \uparrow\ \Downarrow\alpha \uparrow\quad \varepsilon\ \uparrow\quad |\uparrow$$

$$[\quad \Uparrow\beta\,[\quad t\,[\quad \Downarrow\alpha\,[\quad \varepsilon J_{15}]\ \ I_{12}]\ H_{11}]\ F_9]\quad]\ E_7]$$

$$\uparrow \Uparrow\beta\ \uparrow\quad t\ \uparrow\ \Downarrow\alpha\ \uparrow\quad \varepsilon\quad \uparrow\quad)\uparrow$$

Agrupando as expressões que definem A, temos

$$[\ [\ x\ A_6]\ [\ z\,[\ \Downarrow\beta\ [\ \varepsilon L_{16}]\ K_{14}]\ A_{13}]\ ]$$
$$A \rightarrow \uparrow(\uparrow x \uparrow E \uparrow \ \mid\ \uparrow z \uparrow\ \Downarrow\beta \uparrow\ \uparrow \varepsilon\ \uparrow\ )\uparrow$$

Substituindo E em A, temos

$$[\ [\ x\,[\ [\ [\uparrow\alpha\ [\ y\,[\ \Downarrow\alpha\,[\ \varepsilon J_{15}]\ I_{12}]\,G_{10}]\,F_8]$$
$$A \rightarrow \uparrow(\uparrow x \uparrow\ A \uparrow(\uparrow\ \uparrow y \uparrow\ \Downarrow\alpha \uparrow\ \varepsilon\ \uparrow\ \mid\uparrow$$

$$[\ \Uparrow\beta\,[\ t\,[\ \Downarrow\alpha\,[\ \varepsilon J_{15}]\ I_{12}]\,H_{11}]\,F_9]\ ]\,E_7]\,A_6]$$
$$\uparrow\Uparrow\beta\ \uparrow\ t\ \uparrow\ \Downarrow\alpha\ \uparrow\ \varepsilon\ \uparrow\ )\uparrow$$

$$[\ z\ [\ \Downarrow\beta\ [\ \varepsilon L_{16}]\ K_{14}]\ A_{13}]\ ]$$
$$\mid\ \uparrow z \uparrow\ \Downarrow\beta \uparrow\ \uparrow \varepsilon\ \uparrow\ )\uparrow$$

Substituindo B em S, temos

$$[\ [\ [\ \Uparrow\alpha\ [\ \varepsilon C_4]\ B_2]\ [\ \Uparrow\beta\,[\ \varepsilon D_5]\,B_3]\ ]\,S_1]$$
$$S \rightarrow \uparrow A\ \uparrow(\uparrow\Uparrow\alpha\ \uparrow\ \varepsilon\ \uparrow\ \mid\ \uparrow\Uparrow\beta\ \uparrow\ \varepsilon\ \uparrow\ )\uparrow$$

Substituindo A em S, temos

$$S \rightarrow \uparrow (\uparrow (\uparrow x \uparrow \; A \uparrow (\uparrow \Uparrow\alpha \uparrow \; y \uparrow \; \Downarrow\alpha \uparrow \; \varepsilon \uparrow \; | \uparrow$$

$$\uparrow \Uparrow\beta \uparrow \; t \uparrow \; \Downarrow\alpha \uparrow \; \varepsilon \uparrow \;) \uparrow$$

$$| \uparrow z \uparrow \; \Downarrow\beta \uparrow \uparrow \varepsilon \uparrow \;) \uparrow) \uparrow$$

$$\uparrow (\uparrow \Uparrow\alpha \uparrow \; \varepsilon \uparrow \; | \uparrow \Uparrow b \uparrow \; \varepsilon \uparrow \;) \uparrow$$

Após as operações algébricas envolvendo as regras de produção da gramática, com seus respectivos rótulos, chegamos assim a duas expressões importantes. A primeira expressão define o não-terminal S, que é a raiz da gramática, e a segunda define o não-terminal essencial A encarregado do aninhamento da expressão xzt.

A seguir será apresentada a simulação da interpretação da expressão rotulada para uma sentença xzt, seguindo a tabela do algoritmo 4.4.

Símbolo	Rótulo	saída	pilha
	[	[(	]
(	[	[(	]]
(	[	[(	]]]
x	x	(x)	
[	[	[(	]]]]
A			
	[	[(	]]]]]
(	[	[(	]]]]]]

z	z	(z)		
	[	[(	]]]]]]]	
⇓β	⇓β			
	[	[(	]]]]]]]	
ε	ε	( )		
	L_{16}	L_{16})		
	]	]	]]]]]]	
	K_{14}	K_{14})		
	]	]	]]]]]	
	A_{13}	A_{13})		
	]	]	]]]]	
)	]	]	]]]	
	[	[(	]]]]	
(	[	[	]]]]]	
⇑β	⇑β			
	[	[(	]]]]]]	
t	t	(t)		t
	[`	[(	]]]]]]]	
⇓α	⇓α			
	[	[(	]]]]]]]]	
ε	ε	( )		
	J_{15}	J_{15})		
	]	]	]]]]]]]	
	I_{12}	I_{12})		
	]	]	]]]]]]	
	H_{11}	H_{11})		
	]	]	]]]]]	
	F_9	F_9)		
	]	]	]]]]	
)	]	]	]]]]	
	E_7	E_7)		

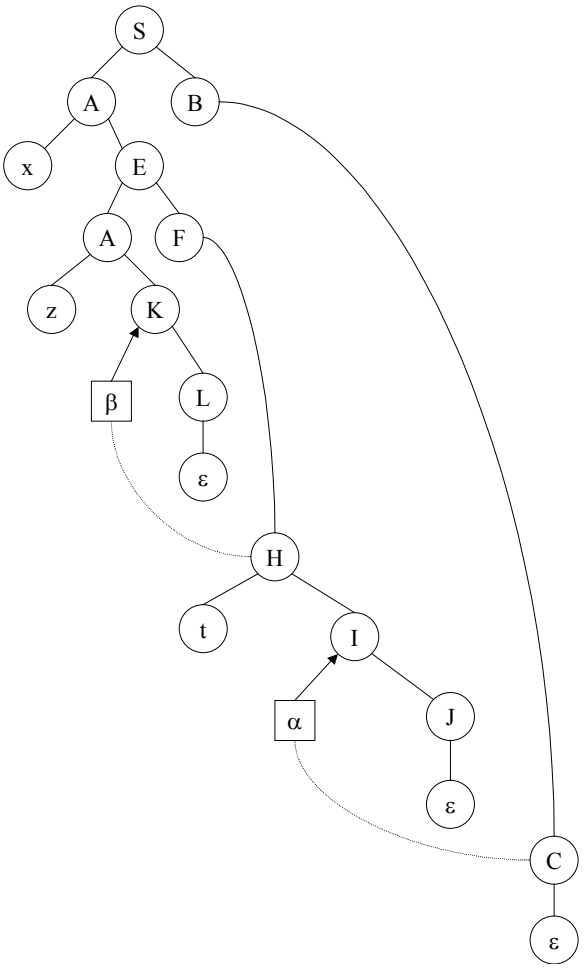
	]	]	]]]
	A_6	A_6)	
	]	]	]]
	]	]	]
	[	[(	]]
(	[	[(	]]]
⇓α	⇓α		
	[	[(	]]]]
ε	ε	( )	
	C_4	C_4)	
	]	]	]]]
	B_2	B_2)	
	]	]	]]
)	]	]	]
	S_1	S_1)	
	]	]	

A saida é:

[([([((x) [([([((z) [(⇓β [(( ) L_{16})] K_{14})] A_{13})]] [([(⇑β [((t) [(⇓α [(( ) J_{15})] I_{12})] H_{11})] F_9)]] E_7)]A_6)]][([(⇓α [(( ) C_4)] B_2)]]S_1)]

Esta saída devidamente diagramada para realçar os aninhamentos é apresentada a seguir:

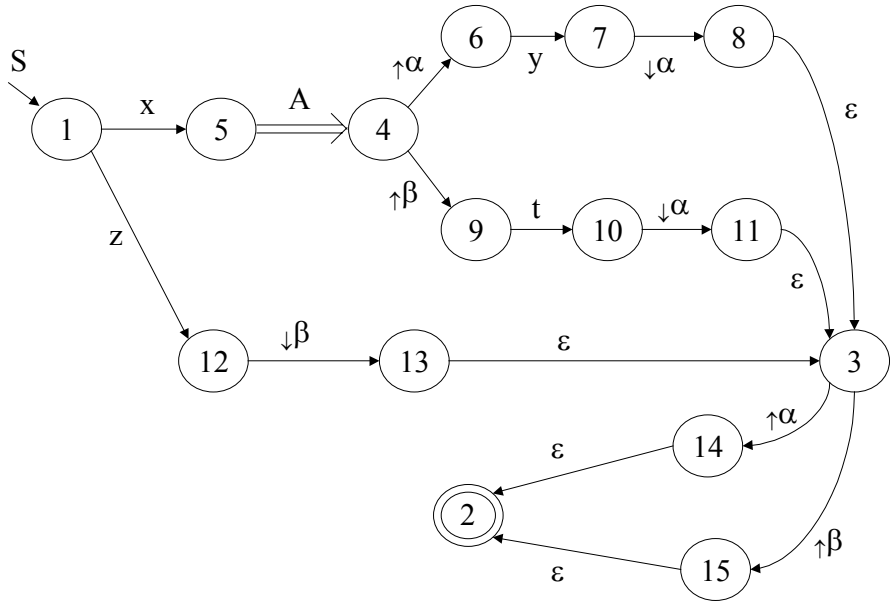
```
[(
    [(
        [( (x)
            [(
                [(
                    [( (z)
                        [(⇓β
                            [( ( ) L16)
                            ] K14)
                        ] A13)
                    ]
                ]
                  [(
                     [
                        [(⇑β
                           ( t )
                              [(⇓α
                                  [( ( ) J15)
                                  ] I12)
                              ] H11)
                        ] F9)
                     ]
                  ] E7)
            ]A6)
        ]
    ]
  [(
      [(⇓α
          [( ( ) C4)
          ]  B2)
      ]
  ]  S1)
]
```

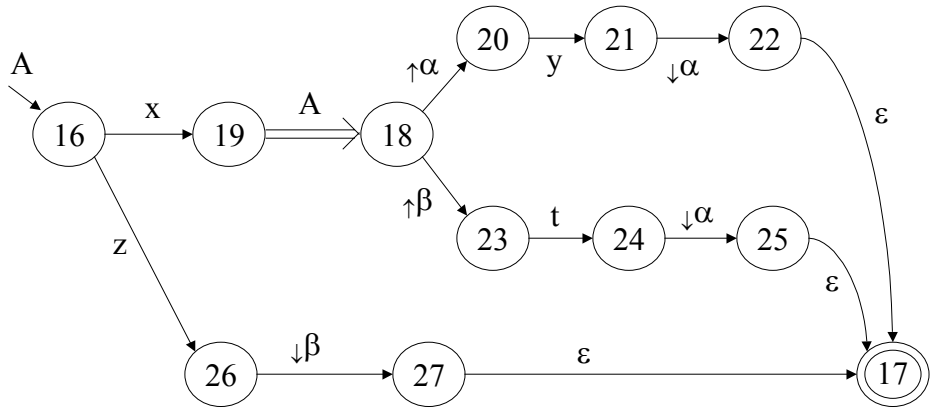


Aplicando o algoritmo para a construção do autômatos a partir da gramática, vamos designar estados para as expressões que definem os não-terminais S e A.

```
      [  [   [    x [     [  [⇑α  [    y [  ⇓α [    εJ15]  I12] G10] F8]
S → ↑ ( ↑ ( ↑  x ↑    A ↑ ( ↑⇑α  ↑  y ↑  ⇓α ↑    ε  ↑                |↑
      1  1   1    5      4   4   6   7    8       3                  4
        [⇑β  [     t [   ⇓α [     εJ15]  I12] H11] F9]  ] E7] A6]
        ↑ ⇑β  ↑  t ↑  ⇓α ↑    ε   ↑                ) ↑
        4     9   10     11       3                  3

              [   z [  ⇓β   [     εL16]   K14]  A13]   ]
           |  ↑ z  ↑ ⇓β     ↑ ε   ↑                ) ↑ ) ↑
              1   12        13     3                  3  3

           [  [⇑α  [    εC4]  B2]     [⇑b  [    εD5]  B3]   ] S1]
           ↑ ( ↑ ⇑α  ↑ ε  ↑          |   ↑   ↑  ε  ↑        ) ↑
           3   3   14   2              3   15   2           2

      [   [    x [     [  [⇑α  [    y [  ⇓α[    εJ15]  I12] G10] F8]
A → ↑ ( ↑  x ↑    A ↑ ( ↑    ↑  y ↑  α  ↑    ε  ↑                |↑
      16 16  19     18  18   20   21   22       17                 18

        [⇑β  [     t [   ⇓α [     εJ15]  I12] H11] F9]  ] E7] A6]
        ↑    ↑   t  ↑ ⇓α ↑    ε   ↑                ) ↑
        18   23     24    25       17                 17

              [   z [  ⇓β [     εL16]   K14]  A13]     ]
           |  ↑ z ↑  ⇓β ↑    ε   ↑                ) ↑
              16  26    27       17                 17
```

O autômato resultante pode ser representado pelos seguintes diagramas de estados:





Exemplo:

Este exemplo apresenta uma linguagem simples que compreende programas que possuem somente um bloco, delimitado pelos símbolos { e }, que indicam o início e o fim do texto, respectivamente.

Este bloco é constituído por uma ou mais listas de declarações de variáveis, que podem ser do tipo I (inteiro) ou R (real), seguido por uma seqüência de um ou mais comandos de expressões aritméticas do tipo adição de números e/ou variáveis.

Esta linguagem faz uma verificação do tipo das variáveis no momento em que são encontrados os comandos de atribuição de valores.

As variáveis são separadas por vírgulas (,). As listas de variáveis são separadas por ponto e vírgula (;), exceto a última lista, que é finalizada por ponto (**.**).

Os comandos são separados por ponto e vírgula, exceto o último comando. Cada comando é composto por uma variável, seguida por um sinal de igual (=), seguido por uma seqüência de variáveis ou números separados pelo operador mais (+). Em cada comando, as variáveis utilizadas devem ser do mesmo tipo (inteiro ou real).

Esta linguagem-exemplo será representada de duas formas, a primeira através de uma gramática adaptativa, a segunda, através de um autômato adaptativo.

Parte 1 – Especificação da linguagem-exemplo através de uma gramática adaptativa

Nesta seção será apresentada uma gramática adaptativa que formaliza a linguagem mencionada acima.

Na declaração das variáveis, esta gramática utiliza um técnica que permite que seja armazenada a informação do tipo da variável que esta sendo declarado.

Se por exemplo a variável for do tipo inteiro, indicada pela palavra reservada I, então será aplicada uma ação adaptativa que utiliza um não-terminal especial, chamado Tipo’ que vai criar um vínculo entre o não-terminal var_int com as variáveis que serão declaradas como sendo inteiras.

A gramática desta linguagem é especificada da seguinte maneira: $G = (G^0, T, R^0)$, onde

$G^0 = (V_N{}^0, V_T, V_C, P_L{}^0, P_D{}^0 , P_1)$

$V_N{}^0 = \{ P_1$, dec, com, var_int, var_real, id, num, var , Tipo’$\}$

V_T = { {, } ; , . , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p ,q , r, s, t, u, v, x, y, w, z, 1, 2, 3, 4, 5, 6, 7,8, 9, 0, +, , , = , I, R}

$V_C = \phi$

$P_D^0 = \phi$

P_L^0 = {

$P_1 \rightarrow$ "{" (dec \ ";") "." (com \ ";") "}"

dec $\rightarrow$ ({C (var_int)} "I" | {C (var_real)} "R") (id \ ",")

id $\rightarrow$ "a" S ("a") | "b" S ("b") | ..... | "z" S ("z")

com $\rightarrow$ ({F (var_int)} var_int | {F (var_real)} var_real) "=" ((var |num) \ "+") {R (var)}

num $\rightarrow$ "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

var $\rightarrow \phi$

Tipo' $\rightarrow \phi$

}

T = {

C (t) = { x :

- [Tipo' $\rightarrow$ x]

+ [Tipo' $\rightarrow$ t]

}

F (x) = { +[var $\rightarrow$ x] }

R (x) = { y :

-[x $\rightarrow$ y]

}

S (α) = { t:

? [Tipo' $\rightarrow$ t]

B (t , α)

}

B (t, α) = {

+ [t → α]

E (t, α)

}

E (t, α) = { - [id → {S (α)} α] }

}

Observe-se que a ação adaptativa S (α) é composta pela chamada de outra ação adaptativa B (t, α) que por sua vez chama a ação E (t, α).

Isto acontece devido a necessidade da aplicação das ações elementares em uma ordem seqüencial. Não existe uma forma de se impor uma seqüência de execução às ações elementares. Existe apenas uma ordem de prioridade, por isso, deve-se utilizar este artifício para que a seqüencialização seja garantida.

Normalizando a gramática G, temos a nova gramática G'= ($G^{0'}$, T', $R^{0'}$)

$G^{0'}$ = ($V_N^{0'}$, V_T', V_C', $P_L^{0'}$, $P_D^{0'}$, P_1)

$V_N^{0'} = V_N^0 \cup \{ P_2, P_3, P_4, P_5, P_6, P_7, P_8, D_2, D_3, D_4, A_2, A_3, A_4, A_5\}$

$V_T' = V_T$

$V_C'= \{\chi , \delta\}$

O conjunto $P_L^{0'} \cup P_D^{0'}$ é formado pelas regras que serão apresentadas abaixo:

Normalizando $P_1 \rightarrow$ "{" (dec \ ";") "." (com \ ";") "}" , temos

1^0. $P_1 \rightarrow$ "{" P_2

2^0. $P_2 \rightarrow$ dec P_3

3^0. δ $P_3 \rightarrow P_4$

4^0. $P_4 \rightarrow$ ";" P_2

5^0. $P_4 \rightarrow$ "." P_5

6^0. $P_5 \rightarrow$ com P_6

7^0. χ $P_6 \rightarrow P_7$

8^0. $P_7 \rightarrow$ ";" P_5

9^0. $P_7 \rightarrow$ "}" P_8

10^0. $P_8 \rightarrow \varepsilon$

Notar os símbolos de contexto δ e χ associados respectivamente aos não-terminais dec e com da gramática original.

Notar também que os não-terminais introduzidos pela normalização não utilizam símbolos de contexto.

Normalizando a produção dec $\rightarrow$ ({C (var_int)} "I" | {C (var_real)} "R") (id \ ","), temos:

11^0. dec $\rightarrow$ {C (var_int) } "I" D_2

12^0. dec $\rightarrow$ {C (var_real)} "R" D_2

13^0. $D_2 \rightarrow$ id D_3

14^0. $D_3 \rightarrow$ "," D_2

15^0. $D_3 \leftarrow \delta$ D_4

16^0. $D_4 \rightarrow \varepsilon$

Normalizando a produção

com $\rightarrow$ ({F (var_int)}var_int | {F (var_real)} var_real) "=" ((var |num) \ "+")
{R (var)},

temos

17^0. com → {F (var_int)} var_int A_2

18^0. com → {F (var_real) }var_real A_2

19^0. A_2 → "=" A_3

20^0. A_3 → var A_4

21^0. A_3 → num A_4

22^0. A_4 → "+" A_3

23^0. A_4 ←{R (var)} χ A_5

24^0. A_5 → ε

As seguintes regras de produção são mantidas

id → {S ("a")} "a" | {S ("b")} "b" | ..... | {S ("z")} "z"

num → "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

var → ϕ

Tipo' → ϕ

As ações adaptativas também não necessitam ser normalizadas, pois o seu formato já se mostra satisfatório.

Para ilustrar como são geradas as sentenças da linguagem, vamos tomar como exemplo um programa escrito nesta linguagem e mostrar o seu processo de derivação passo a passo.

Tomaremos o seguinte programa

```
{     R k , m ;        // declaração de duas variáveis do tipo real
      I  a .           // declaração de uma variável do tipo inteiro
      k = 4 + m ;      // expressão aritmética de adição envolvendo somente variáveis reais e
                       // números
```

a = 1 + a // expressão aritmética de adição envolvendo somente variáveis inteiras

// e números

}

Obs: Os textos escritos após os símbolos // são comentários do programa.

A seguir será apresentado o processo de derivação do programa-texto escrito na linguagem deste exemplo.

1. (regra 1^0.) $P_1 \rightarrow$ "{" P_2	$\underline{P_1} \Rightarrow_{G^0}$ "{" $\underline{P_2}$
2. (regra 2^0.) $P_2 \rightarrow$ dec P_3	$\Rightarrow_{G^0}$ "{" $\underline{\text{dec}}$ P_3
3. (regra 12^0.) dec $\rightarrow$ {C (var_real)} "R" D_2	$\Rightarrow_{G^0}$ "{"{C (var_real)} "R" $\underline{D_2}$ P_3

Neste momento, encontra-se uma regra de produção adaptativa. Antes de prosseguir a derivação, é preciso que a ação adaptativa C (var_real) seja executada da seguinte forma:

C (var_real) = { x :

- [Tipo' $\rightarrow$ x]

+ [Tipo' $\rightarrow$ var_real]

}

Assim, temos uma alteração na gramática, cujas regras de produção são as seguintes:

G^1:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec $\rightarrow$ {C (var_int)} "I" D_2	17^0. com $\rightarrow$ {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec $\rightarrow$ {C (var_real)}"R" D_2	18^0.com $\rightarrow$ {F (var_real)}var_real A_2
3^0. δ $P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow \delta$ D_4	21^0. $A_3 \rightarrow$ num A_4

6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. χ $P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}χ A_5
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

25^0. id $\rightarrow$ {S ("a") } "a"	34^0. id $\rightarrow$ {S ("j")} "j"	43^0. id $\rightarrow$ {S ("s")} "s"
26^0. id $\rightarrow$ {S ("b") } "b"	35^0. id $\rightarrow$ {S ("k")} "k"	44^0. id $\rightarrow$ {S ("t")} "t"
27. id $\rightarrow$ {S ("c")} "c"	36^0. id $\rightarrow$ {S ("l")} "l"	45^0. id $\rightarrow$ {S ("u")} "u"
28^0. id $\rightarrow$ {S ("d")} "d"	37^0. id $\rightarrow$ {S ("m")} "m"	46^0. id $\rightarrow$ {S ("v")} "v"
29^0. id $\rightarrow$ {S ("e")} "e"	38^0. id $\rightarrow$ {S ("n")} "n"	47^0. id $\rightarrow$ {S ("x")} "x"
30^0. id $\rightarrow$ {S ("f")} "f "	39^0. id $\rightarrow$ {S ("o")} "o"	48^0. id $\rightarrow$ {S ("y")} "y"
31^0. id $\rightarrow$ {S ("g")} "g "	40^0. id $\rightarrow$ {S ("p")} "p"	49^0. id $\rightarrow$ {S ("w")} "w"
32^0. id $\rightarrow$ {S ("h")} "h"	41^0. id $\rightarrow$ {S ("q")} "q"	50^0. id $\rightarrow$ {S ("z")} "z"
33^0. id $\rightarrow$ {S ("i")} "i"	42^0. id $\rightarrow$ {S ("r")} "r"	

51^0. num $\rightarrow$ "0"	54^0. num $\rightarrow$ "3"	57^0. num $\rightarrow$ "6"	60^0. num $\rightarrow$ "9"
52^0. num $\rightarrow$ "1"	55^0. num $\rightarrow$ "4"	58^0. num $\rightarrow$ "7"	
53^0. num $\rightarrow$ "2"	56^0. num $\rightarrow$ "5"	59^0. num $\rightarrow$ "8"	

61^0. var $\rightarrow \phi$

62^1. Tipo' $\rightarrow$ var_real

Retomando o processo de derivação, seguem os seguintes passos

4. (regra 13^0.) $D_2 \rightarrow$ id D_3	$\Rightarrow_{G^1}$ { R id D_3 P_3
5. (regra 34^0.) id $\rightarrow$ {S ("k")} "k"	$\Rightarrow_{G^1}$ { R {S ("k")}k D_3 P_3

Encontramos outra regra adaptativa. A aplicação de S ("k") é feita da seguinte forma:

S ("k") = { t:

? [Tipo' $\rightarrow$ t]

B (var_real , "k")

}

B (var_real, "k") = {

+ [var_real $\rightarrow$ "k"]

E (var_real, "k")

}

E (var_real, "k") = { - [id $\rightarrow$ {S ("k")} "k"] }

}

Neste ponto, a ação adaptativa consulta qual é o não-terminal indicado por Tipo'. Esta informação é responsável pela identificação do tipo das variáveis que serão declaradas neste comando. Observe-se que a regra que define o identificador desta variável k é removida do conjunto, o que significa que existe somente uma variável com este nome, e é do tipo real.

O novo conjunto de regras de produção é o seguinte:

G^2:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec $\rightarrow$ {C (var_int)} "I" D_2	17^0. com $\rightarrow$ {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec $\rightarrow$ {C (var_real)}"R" D_2	18^0.com $\rightarrow$ {F (var_real)}var_real A_2

3^0. $\delta\ P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow \delta\ D_4$	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. $\chi\ P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}$\chi\ A_5$
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

25^0. id $\rightarrow$ {S ("a") } "a"	34^0. id $\rightarrow$ {S ("j")} "j"	43^0. id $\rightarrow$ {S ("s")} "s"
26^0. id $\rightarrow$ {S ("b") } "b"		44^0. id $\rightarrow$ {S ("t")} "t"
27. id $\rightarrow$ {S ("c")} "c"	36^0. id $\rightarrow$ {S ("l")} "l"	45^0. id $\rightarrow$ {S ("u")} "u"
28^0. id $\rightarrow$ {S ("d")} "d"	37^0. id $\rightarrow$ {S ("m")} "m"	46^0. id $\rightarrow$ {S ("v")} "v"
29^0. id $\rightarrow$ {S ("e")} "e"	38^0. id $\rightarrow$ {S ("n")} "n"	47^0. id $\rightarrow$ {S ("x")} "x"
30^0. id $\rightarrow$ {S ("f")} "f "	39^0. id $\rightarrow$ {S ("o")} "o"	48^0. id $\rightarrow$ {S ("y")} "y"
31^0. id $\rightarrow$ {S ("g")} "g "	40^0. id $\rightarrow$ {S ("p")} "p"	49^0. id $\rightarrow$ {S ("w")} "w"
32^0. id $\rightarrow$ {S ("h")} "h"	41^0. id $\rightarrow$ {S ("q")} "q"	50^0. id $\rightarrow$ {S ("z")} "z"
33^0. id $\rightarrow$ {S ("i")} "i"	42^0. id $\rightarrow$ {S ("r")} "r"	

51^0. num $\rightarrow$ "0"	54^0. num $\rightarrow$ "3"	57^0. num $\rightarrow$ "6"	60^0. num $\rightarrow$ "9"
52^0. num $\rightarrow$ "1"	55^0. num $\rightarrow$ "4"	58^0. num $\rightarrow$ "7"	
53^0. num $\rightarrow$ "2"	56^0. num $\rightarrow$ "5"	59^0. num $\rightarrow$ "8"	

61^0. var $\rightarrow \phi$

62^1. Tipo' → var_real

63^2. var_real → "k"

6. (regra 14^0.) $D_3 \rightarrow$ "," D_2	$\Rightarrow_{G^2}$ { R k , $\underline{D_2}$ P_3
7. (regra 13^0.) $D_2 \rightarrow$ id D_3	$\Rightarrow_{G^2}$ { R k , $\underline{id}$ D_3 P_3
8. (regra 37^0.) id →{S ("m")} "m"	$\Rightarrow_{G^2}$ { R k , {S ("m")}m D_3 P_3

Novamente, a ação adaptativa S foi acionada, pois se declarou mais uma variável m que é do tipo real. Esta variável vai ser acrescentada à lista da variáveis do tipo real e vai ser removida da lista dos identificadores gerais, que ainda não foram utilizados.

S ("m") = { t:

? [Tipo' → t]

B (var_real , "m")

}

B (var_real, "m") = {

+ [var_real → "m"]

E (var_real, "m")

}

E (var_real, "m") = { - [id → {S ("m")} "m"] }

}

O novo conjunto de regras de produção passa a ser a seguinte:

G^3:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec $\rightarrow$ {C (var_int)} "I" D_2	17^0. com $\rightarrow$ {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec $\rightarrow$ {C (var_real)}"R" D_2	18^0.com$\rightarrow$ {F (var_real)}var_real A_2
3^0. δ $P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow \delta$ D_4	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. χ $P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}χ A_5
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

25^0. id $\rightarrow$ {S ("a") } "a"	34^0. id $\rightarrow$ {S ("j")} "j"	43^0. id $\rightarrow$ {S ("s")} "s"
26^0. id $\rightarrow$ {S ("b") } "b"		44^0. id $\rightarrow$ {S ("t")} "t"
27. id $\rightarrow$ {S ("c")} "c"	36^0. id $\rightarrow$ {S ("l")} "l"	45^0. id $\rightarrow$ {S ("u")} "u"
28^0. id $\rightarrow$ {S ("d")} "d"		46^0. id $\rightarrow$ {S ("v")} "v"
29^0. id $\rightarrow$ {S ("e")} "e"	38^0. id $\rightarrow$ {S ("n")} "n"	47^0. id $\rightarrow$ {S ("x")} "x"
30^0. id $\rightarrow$ {S ("f")} "f "	39^0. id $\rightarrow$ {S ("o")} "o"	48^0. id $\rightarrow$ {S ("y")} "y"
31^0. id $\rightarrow$ {S ("g")} "g "	40^0. id $\rightarrow$ {S ("p")} "p"	49^0. id $\rightarrow$ {S ("w")} "w"
32^0. id $\rightarrow$ {S ("h")} "h"	41^0. id $\rightarrow$ {S ("q")} "q"	50^0. id $\rightarrow$ {S ("z")} "z"
33^0. id $\rightarrow$ {S ("i")} "i"	42^0. id $\rightarrow$ {S ("r")} "r"	

51^0. num $\rightarrow$ "0"	54^0. num $\rightarrow$ "3"	57^0. num $\rightarrow$ "6"	60^0. num $\rightarrow$ "9"

52^0. num → "1"	55^0. num → "4"	58^0. num → "7"	
53^0. num → "2"	56^0. num → "5"	59^0. num → "8"	

61^0. var → ϕ

62^1. Tipo' → var_real

63^2. var_real → "k"

64^3. var_real → "m"

Retomando a derivação do texto-fonte, temos os seguintes passos:

9. (regra 15^0.) $D_3 \leftarrow \delta\ D_4$	$\Rightarrow_{G^3}$ { R k , m δ $\underline{D_4}$ P_3
10.(regra 16^0.) $D_4 \rightarrow \varepsilon$	$\Rightarrow_{G^3}$ { R k , m $\underline{\delta\ \varepsilon\ P_3}$
11.(regra 3^0.) $\delta\ P_3 \rightarrow P_4$	$\Rightarrow_{G^3}$ { R k , m $\underline{P_4}$
12.(regra 4^0.) $P_4 \rightarrow$ ";" P_2	$\Rightarrow_{G^3}$ { R k , m ; $\underline{P_2}$
13.(regra 2^0.) $P_2 \rightarrow$ dec P_3	$\Rightarrow_{G^3}$ { R k , m ; $\underline{\text{dec}}$ P_3
14.(regra 11^0.) dec → {C (var_int)} "I" D_2	$\Rightarrow_{G^3}$ { R k , m ; {C (var_int)}I $\underline{D_2}$ P_3

Aqui, foi encontrada outra regra de produção adaptativa e desta vez ela aciona a ação C , que é utilizada para iniciar a designação das próximas variáveis que forem geradas como sendo do tipo inteiro.

C (var_int) = { x :

- [Tipo' → x]

+ [Tipo' → var_int]

}

Novamente a gramática se adaptou a esta nova situação e produziu um novo conjunto de regras de produção que é o seguinte:

G^4:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec $\rightarrow$ {C (var_int)} "I" D_2	17^0. com $\rightarrow$ {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec $\rightarrow$ {C (var_real)}"R" D_2	18^0.com$\rightarrow$ {F (var_real)}var_real A_2
3^0. δ $P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow \delta$ D_4	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. χ $P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}χ A_5
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

25^0. id $\rightarrow$ {S ("a") } "a"	34^0. id $\rightarrow$ {S ("j")} "j"	43^0. id $\rightarrow$ {S ("s")} "s"
26^0. id $\rightarrow$ {S ("b") } "b"		44^0. id $\rightarrow$ {S ("t")} "t"
27. id $\rightarrow$ {S ("c")} "c"	36^0. id $\rightarrow$ {S ("l")} "l"	45^0. id $\rightarrow$ {S ("u")} "u"
28^0. id $\rightarrow$ {S ("d")} "d"		46^0. id $\rightarrow$ {S ("v")} "v"
29^0. id $\rightarrow$ {S ("e")} "e"	38^0. id $\rightarrow$ {S ("n")} "n"	47^0. id $\rightarrow$ {S ("x")} "x"
30^0. id $\rightarrow$ {S ("f")} "f "	39^0. id $\rightarrow$ {S ("o")} "o"	48^0. id $\rightarrow$ {S ("y")} "y"
31^0. id $\rightarrow$ {S ("g")} "g "	40^0. id $\rightarrow$ {S ("p")} "p"	49^0. id $\rightarrow$ {S ("w")} "w"

32^0. id → {S (“h”)} “h”	41^0. id → {S (“q”)} “q”	50^0. id → {S (“z”)} “z”
33^0. id → {S (“i”)} “i”	42^0. id → {S (“r”)} “r”	

51^0. num → “0”	54^0. num → “3”	57^0. num → “6”	60^0. num → “9”
52^0. num → “1”	55^0. num → “4”	58^0. num → “7”	
53^0. num → “2”	56^0. num → “5”	59^0. num → “8”	

61^0. var → ϕ

63^2. var_real → “k”

64^3. var_real → “m”

65^4. Tipo’→ var_int

O processo de derivação continua com os passos seguintes:

15. (regra 13^0.) D_2 → id D_3	$\Rightarrow_{G^4}$ { R k , m ; I id D_3 P_3
16. (regra 25^0.) id → S (“a”) “a”	$\Rightarrow_{G^4}$ { R k , m ; I {S (a)}a D_3 P_3

Nesta ocasião, para a geração da variável inteira a, será acionada a ação adaptativa S (“a”) que acrescenta a variável a à lista de variáveis inteiras e a remove da lista de identificadores não utilizados.

S (“a”) = { t:

? [Tipo’ → t]

B (var_int , “a”) }

B (var_int, “a”) = {

+ [var_int → “a”]

E (var_int, “a”)

}

E (var_int, “a”) = { - [id → {S (“a”)} “a”] }

}

É apresentado a seguir o novo conjunto de regras de produção:

G^5:

1^0. $P_1 \rightarrow$ “{” P_2	11^0. dec → {C (var_int)} “I” D_2	17^0. com → {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec →{C (var_real)}“R” D_2	18^0.com→ {F (var_real)}var_real A_2
3^0. δ $P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ “=” A_3
4^0. $P_4 \rightarrow$ “;” P_2	14^0. $D_3 \rightarrow$ “,” D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ “.” P_5	15^0. $D_3 \leftarrow \delta D_4$	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ “+” A_3
7^0. χ $P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}χ A_5
8^0. $P_7 \rightarrow$ “;” P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ “}” P_8		
10^0. $P_8 \rightarrow \varepsilon$		

	34^0. id → {S (“j”)} “j”	43^0. id →{S (“s”)} “s”
26^0. id → {S (“b”) } “b”		44^0. id → {S (“t”)} “t”
27. id → {S (“c”)} “c”	36^0. id → {S (“l”)} “l”	45^0. id → {S (“u”)} “u”
28^0. id → {S (“d”)} “d”		46^0. id → {S (“v”)} “v”
29^0. id → {S (“e”)} “e”	38^0. id → {S (“n”)} “n”	47^0. id → {S (“x”)} “x”
30^0. id → {S (“f”)} “f ”	39^0. id → {S (“o”)} “o”	48^0. id → {S (“y”)} “y”

31^0. id → {S (“g”)} “g ”	40^0. id → {S (“p”)} “p”	49^0. id → {S (“w”)} “w”
32^0. id → {S (“h”)} “h”	41^0. id → {S (“q”)} “q”	50^0. id → {S (“z”)} “z”
33^0. id → {S (“i”)} “i”	42^0. id → {S (“r”)} “r”	

51^0. num → “0”	54^0. num → “3”	57^0. num → “6”	60^0. num → “9”
52^0. num → “1”	55^0. num → “4”	58^0. num → “7”	
53^0. num → “2”	56^0. num → “5”	59^0. num → “8”	

61^0. var → ϕ

63^2. var_real → “k”

64^3. var_real → “m”

65^4. Tipo’→ var_int

66^5. var_int → “a”

Retomando os passos da derivação, temos:

17. (regra 15^0.) $D_3 \leftarrow \delta\ D_4$	$\Rightarrow_{G^5}$ { R k , m ; I a δ $\underline{D_4}$ P_3
18. (regra 16^0.) $D_4 \rightarrow \varepsilon$	$\Rightarrow_{G^5}$ { R k , m ; I a $\underline{\delta\ \varepsilon\ P_3}$
19. (regra 3^0.) $\delta\ P_3 \rightarrow P_4$	$\Rightarrow_{G^5}$ { R k , m ; I a $\underline{P_4}$
20. (regra 5^0.) $P_4 \rightarrow$ “.” P_5	$\Rightarrow_{G^5}$ { R k , m ; I a . $\underline{P_5}$
21. (regra 6^0.) $P_5 \rightarrow$ com P_6	$\Rightarrow_{G^5}$ { R k , m ; I a . $\underline{\text{com}}$ P_6
22. (regra 18^0.) com→{F (var_real)} var_real A_2 P_6	$\Rightarrow_{G^5}$ { R k , m ; I a . {F (var_real)}var_real A_2 P_6

Neste ponto, a geração das declarações das variáveis foi concluída e se inicia o processo de

derivação dos comandos de atribuição.

Foi aplicada uma regra de produção adaptativa cuja ação adaptativa associada permite que sejam utilizados no comando somente variáveis do tipo real.

F (var_real) = { + [var → var_real] }

O conjunto de regras de produção para esta nova gramática passa a ser o seguinte:

G^6:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec → {C (var_int)} "I" D_2	17^0. com → {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec →{C (var_real)}"R" D_2	18^0.com→ {F (var_real)}var_real A_2
3^0. δ $P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow \delta$ D_4	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. χ $P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}χ A_5
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

	34^0. id → {S ("j")} "j"	43^0. id →{S ("s")} "s"
26^0. id → {S ("b") } "b"		44^0. id → {S ("t")} "t"
27. id → {S ("c")} "c"	36^0. id → {S ("l")} "l"	45^0. id → {S ("u")} "u"
28^0. id → {S ("d")} "d"		46^0. id → {S ("v")} "v"

29^0. id → {S (“e”)} “e”	38^0. id → {S (“n”)} “n”	47^0. id → {S (“x”)} “x”
30^0. id → {S (“f”)} “f”	39^0. id → {S (“o”)} “o”	48^0. id → {S (“y”)} “y”
31^0. id → {S (“g”)} “g”	40^0. id → {S (“p”)} “p”	49^0. id → {S (“w”)} “w”
32^0. id → {S (“h”)} “h”	41^0. id → {S (“q”)} “q”	50^0. id → {S (“z”)} “z”
33^0. id → {S (“i”)} “i”	42^0. id → {S (“r”)} “r”	

51^0. num → “0”	54^0. num → “3”	57^0. num → “6”	60^0. num → “9”
52^0. num → “1”	55^0. num → “4”	58^0. num → “7”	
53^0. num → “2”	56^0. num → “5”	59^0. num → “8”	

61^0. var → ϕ

63^2. var_real → “k”

64^3. var_real → “m”

65^4. Tipo’→ var_int

66^5. var_int → “a”

67^6. var → var_real

O processo de derivação continua com os seguintes passos:

23. (regra 63^2.) var_real → k	$\Rightarrow_{G^6}$ { R k , m ; I a . k $\underline{A_2}$ P_6
24. (regra 19^0.) A_2 → “=” A_3	$\Rightarrow_{G^6}$ { R k , m ; I a . k = $\underline{A_3}$ P_6
25. (regra 21^0.) A_3 → num A_4	$\Rightarrow_{G^6}$ { R k , m ; I a . k = $\underline{\text{num}}$ A_4 P_6
26. (regra 55^0.) num → 4	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 $\underline{A_4}$ P_6

27. (regra 22^0.) $A_4 \rightarrow$ "+" A_3	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + $\underline{A_3}$ P_6
28. (regra 20^0.) $A_3 \rightarrow$ var A_4	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + $\underline{\text{var}}$ A_4 P_6
29. (regra 67^6.) var $\rightarrow$ var_real	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + $\underline{\text{var\_real}}$ A_4 P_6
30. (regra 64^3.) var_real $\rightarrow$ "m"	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + m $\underline{A_4}$ P_6
31. (regra 23^0.) $A_4 \leftarrow$ {R (var)}χ A_5	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + m {R (var)}χ A_5 P_6

Neste momento, o primeiro comando aritmético envolvendo somente variáveis inteiras foi gerado e, para continuar o processo de derivação é removida a regra que especificava o tipo de variáveis que deveriam ser utilizadas no comando anterior.

R (var) = { y :

-[var $\rightarrow$ var_real]

}

O novo conjunto de regras de produção passa a ser o seguinte: (repare que este conjunto é o mesmo que o conjunto da gramática G^5)

G^7:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec $\rightarrow$ {C (var_int)} "I" D_2	17^0. com $\rightarrow$ {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec $\rightarrow$ {C (var_real)}"R" D_2	18^0.com $\rightarrow$ {F (var_real)}var_real A_2
3^0. δ $P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow$ δ D_4	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow$ ε	22^0. $A_4 \rightarrow$ "+" A_3
7^0. χ $P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}χ A_5
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow$ ε

9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

	34^0. id $\rightarrow$ {S ("j")} "j"	43^0. id $\rightarrow$ {S ("s")} "s"
26^0. id $\rightarrow$ {S ("b") } "b"		44^0. id $\rightarrow$ {S ("t")} "t"
27. id $\rightarrow$ {S ("c")} "c"	36^0. id $\rightarrow$ {S ("l")} "l"	45^0. id $\rightarrow$ {S ("u")} "u"
28^0. id $\rightarrow$ {S ("d")} "d"		46^0. id $\rightarrow$ {S ("v")} "v"
29^0. id $\rightarrow$ {S ("e")} "e"	38^0. id $\rightarrow$ {S ("n")} "n"	47^0. id $\rightarrow$ {S ("x")} "x"
30^0. id $\rightarrow$ {S ("f")} "f "	39^0. id $\rightarrow$ {S ("o")} "o"	48^0. id $\rightarrow$ {S ("y")} "y"
31^0. id $\rightarrow$ {S ("g")} "g "	40^0. id $\rightarrow$ {S ("p")} "p"	49^0. id $\rightarrow$ {S ("w")} "w"
32^0. id $\rightarrow$ {S ("h")} "h"	41^0. id $\rightarrow$ {S ("q")} "q"	50^0. id $\rightarrow$ {S ("z")} "z"
33^0. id $\rightarrow$ {S ("i")} "i"	42^0. id $\rightarrow$ {S ("r")} "r"	

51^0. num $\rightarrow$ "0"	54^0. num $\rightarrow$ "3"	57^0. num $\rightarrow$ "6"	60^0. num $\rightarrow$ "9"
52^0. num $\rightarrow$ "1"	55^0. num $\rightarrow$ "4"	58^0. num $\rightarrow$ "7"	
53^0. num $\rightarrow$ "2"	56^0. num $\rightarrow$ "5"	59^0. num $\rightarrow$ "8"	

61^0. var $\rightarrow \phi$

63^2. var_real $\rightarrow$ "k"

64^3. var_real $\rightarrow$ "m"

65^4. Tipo' $\rightarrow$ var_int

66^5. var_int $\rightarrow$ "a"

Continuando o processo de derivação do programa, temos os seguintes passos:

32. (regra 24^0.) $A_5 \rightarrow \varepsilon$	$\Rightarrow_{G^7}$ { R k , m ; I a **.** k = 4 + m $\underline{\chi\ \varepsilon\ P_6}$
33. (regra 7^0.) $\chi\ P_6 \rightarrow P_7$	$\Rightarrow_{G^7}$ { R k , m ; I a **.** k = 4 + m $\underline{P_7}$
34. (regra 8^0.) $P_7 \rightarrow$ "**;**" P_5	$\Rightarrow_{G^7}$ { R k , m ; I a **.** k = 4 + m **;** $\underline{P_5}$
35. (regra 6^0.) $P_5 \rightarrow$ com P_6	$\Rightarrow_{G^7}$ { R k , m ; I a **.** k = 4 + m **;** $\underline{\text{com}}$ P_6
36. (regra 17^0.) com $\rightarrow$ {F (var_int)} var_int A_2	$\Rightarrow_{G^7}$ { R k , m ; I a **.** k = 4 + m **;** {F (var_int)} var_int A_2 P_6

Foi encontrada, neste momento, uma regra de produção que possui a ação adaptativa F (var_int) associada. Esta ação permite que sejam utilizadas somente variáveis do tipo inteiro no comando que será gerado.

F (var_int) = { +[var $\rightarrow$ var_int] }

O conjunto de regras de produção é o seguinte:

G^8:

1^0. $P_1 \rightarrow$ "{" P_2	11^0. dec $\rightarrow$ {C (var_int)} "I" D_2	17^0. com $\rightarrow$ {F (var_int)}var_int A_2
2^0. $P_2 \rightarrow$ dec P_3	12^0. dec $\rightarrow$ {C (var_real)}"R" D_2	18^0.com$\rightarrow$ {F (var_real)}var_real A_2
3^0. $\delta\ P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "**.**" P_5	15^0. $D_3 \leftarrow \delta\ D_4$	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. $\chi\ P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}$\chi\ A_5$
8^0. $P_7 \rightarrow$ "**;**" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		

10^0. $P_8 \rightarrow \varepsilon$		

	34^0. id → {S (“j”)} “j”	43^0. id →{S (“s”)} “s”
26^0. id → {S (“b”) } “b”		44^0. id → {S (“t”)} “t”
27. id → {S (“c”)} “c”	36^0. id → {S (“l”)} “l”	45^0. id → {S (“u”)} “u”
28^0. id → {S (“d”)} “d”		46^0. id → {S (“v”)} “v”
29^0. id → {S (“e”)} “e”	38^0. id → {S (“n”)} “n”	47^0. id → {S (“x”)} “x”
30^0. id → {S (“f”)} “f ”	39^0. id → {S (“o”)} “o”	48^0. id → {S (“y”)} “y”
31^0. id → {S (“g”)} “g ”	40^0. id → {S (“p”)} “p”	49^0. id → {S (“w”)} “w”
32^0. id → {S (“h”)} “h”	41^0. id → {S (“q”)} “q”	50^0. id → {S (“z”)} “z”
33^0. id → {S (“i”)} “i”	42^0. id → {S (“r”)} “r”	

51^0. num → “0”	54^0. num → “3”	57^0. num → “6”	60^0. num → “9”
52^0. num → “1”	55^0. num → “4”	58^0. num → “7”	
53^0. num → “2”	56^0. num → “5”	59^0. num → “8”	

61^0. var → ϕ

63^2. var_real → “k”

64^3. var_real → “m”

65^4. Tipo’→ var_int

66^5. var_int → “a”

68^8. var → var_int

Continuando o processo de derivação, temos os seguinte passos:

37. (regra 66^5.) var_int → a	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a $\underline{A_2}$ P_6
38. (regra 19^0.) A_2 → "=" A_3	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = $\underline{A_3}$ P_6
39. (regra 21^0.) A_3 → num A_4	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = $\underline{num}$ A_4 P_6
40. (regra 52^0.) num → "1"	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = 1 $\underline{A_4}$ P_6
41. (regra 22^0.) A_4 → "+" A_3	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = 1 + $\underline{A_3}$ P_6
42. (regra 20^0.) A_3 → var A_4	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = 1 + $\underline{var}$ A_4 P_6
43. (regra 68^8.) var → var_int	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = 1 + $\underline{var\_int}$ A_4 P_6
44. (regra 66^5.) var_int → "a"	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = 1 + a $\underline{A_4}$ P_6
45. (regra 23^0.) $A_4 \leftarrow \{$R (var)$\}\chi$ A_5	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m ; a = 1 + a {R (var)} χ A_5 P_6

Novamente, é aplicada a regra de produção que possui a ação R (var) associada. Esta ação remove a regra que definia as variáveis que poderiam ser utilizadas no comando de atribuição.

R (var) = { y :

-[var → var_int]

}

O conjunto de regras de produção é agora o seguinte:

G^9:

1^0. P_1 → "{" P_2	11^0. dec → {C (var_int)} "I" D_2	17^0. com → {F (var_int)}var_int A_2
2^0. P_2 → dec P_3	12^0. dec →{C (var_real)}"R" D_2	18^0.com→ {F (var_real)}var_real A_2

3^0. $\delta\ P_3 \rightarrow P_4$	13^0. $D_2 \rightarrow$ id D_3	19^0. $A_2 \rightarrow$ "=" A_3
4^0. $P_4 \rightarrow$ ";" P_2	14^0. $D_3 \rightarrow$ "," D_2	20^0. $A_3 \rightarrow$ var A_4
5^0. $P_4 \rightarrow$ "." P_5	15^0. $D_3 \leftarrow \delta\ D_4$	21^0. $A_3 \rightarrow$ num A_4
6^0. $P_5 \rightarrow$ com P_6	16^0. $D_4 \rightarrow \varepsilon$	22^0. $A_4 \rightarrow$ "+" A_3
7^0. $\chi\ P_6 \rightarrow P_7$		23^0. $A_4 \leftarrow$ {R (var)}$\chi\ A_5$
8^0. $P_7 \rightarrow$ ";" P_5		24^0. $A_5 \rightarrow \varepsilon$
9^0. $P_7 \rightarrow$ "}" P_8		
10^0. $P_8 \rightarrow \varepsilon$		

	34^0. id $\rightarrow$ {S ("j")} "j"	43^0. id $\rightarrow$ {S ("s")} "s"
26^0. id $\rightarrow$ {S ("b") } "b"		44^0. id $\rightarrow$ {S ("t")} "t"
27. id $\rightarrow$ {S ("c")} "c"	36^0. id $\rightarrow$ {S ("l")} "l"	45^0. id $\rightarrow$ {S ("u")} "u"
28^0. id $\rightarrow$ {S ("d")} "d"		46^0. id $\rightarrow$ {S ("v")} "v"
29^0. id $\rightarrow$ {S ("e")} "e"	38^0. id $\rightarrow$ {S ("n")} "n"	47^0. id $\rightarrow$ {S ("x")} "x"
30^0. id $\rightarrow$ {S ("f")} "f "	39^0. id $\rightarrow$ {S ("o")} "o"	48^0. id $\rightarrow$ {S ("y")} "y"
31^0. id $\rightarrow$ {S ("g")} "g "	40^0. id $\rightarrow$ {S ("p")} "p"	49^0. id $\rightarrow$ {S ("w")} "w"
32^0. id $\rightarrow$ {S ("h")} "h"	41^0. id $\rightarrow$ {S ("q")} "q"	50^0. id $\rightarrow$ {S ("z")} "z"
33^0. id $\rightarrow$ {S ("i")} "i"	42^0. id $\rightarrow$ {S ("r")} "r"	

51^0. num $\rightarrow$ "0"	54^0. num $\rightarrow$ "3"	57^0. num $\rightarrow$ "6"	60^0. num $\rightarrow$ "9"
52^0. num $\rightarrow$ "1"	55^0. num $\rightarrow$ "4"	58^0. num $\rightarrow$ "7"	
53^0. num $\rightarrow$ "2"	56^0. num $\rightarrow$ "5"	59^0. num $\rightarrow$ "8"	

61^0. var $\rightarrow \phi$

63^{2}. var_real → "k"

64^{3}. var_real → "m"

65^{4}. Tipo' → var_int

66^{5}. var_int → "a"

Retomando o processo de derivação, temos, finalmente:

46. (regra 24^{0}.) $A_5 \rightarrow \varepsilon$	$\Rightarrow_{G}^{9}$ { R k , m ; I a **.** k = 4 + m **;** a = 1 + a $\underline{\chi\ \varepsilon\ P_6}$
47. (regra 7^{0}.) $\chi\ P_6 \rightarrow P_7$	$\Rightarrow_{G}^{9}$ { R k , m ; I a **.** k = 4 + m **;** a = 1 + a $\underline{P_7}$
48. (regra 9^{0}.) $P_7 \rightarrow$ "}" P_8	$\Rightarrow_{G}^{9}$ { R k , m ; I a **.** k = 4 + m **;** a = 1 + a } $\underline{P_8}$
49. (regra 10^{0}.) $P_8 \rightarrow \varepsilon$	$\Rightarrow_{G}^{9}$ { R k , m ; I a **.** k = 4 + m **;** a = 1 + a } ε

Logo o programa gerado pela linguagem através do processo descrito acima é:

{ R k , m ; I a **.** k = 4 + m **;** a = 1 + a }

Parte 2 – Especificação da linguagem-exemplo através de um autômato adaptativo

Nesta seção, a representação da linguagem-exemplo será feita através de um autômato adaptativo, composto por quatro máquinas de estados: uma máquina principal, que faz o tratamento de um programa completo nesta linguagem-exemplo, uma submáquina chamada *dec*, que é utilizada para reconhecer a declaração das variáveis que podem ser do tipo inteiro ou real, uma submáquina chamada *com* que é utilizada para reconhecer comandos de atribuição aritmética envolvendo a adição de variáveis inteiras ou reais e números. E uma quarta submáquina que faz o tratamento léxico da linguagem.

A submáquina que faz o tratamento léxico do texto-fonte segue o modelo convencional e pode ser construída de forma similar aos modelos clássicos, por exemplo, o apresentado em [Jos93].

O analisador léxico extrai do programa-fonte e classifica cadeias de caracteres correspondentes a identificadores, algumas palavras reservadas e números inteiros de um só dígito sem sinal.

Os identificadores podem ser cadeias de caracteres compostas por apenas uma letra minúscula. O analisador léxico classifica os identificadores de acordo com as seguintes categorias:

palavras reservadas: existem somente duas palavras reservadas que são I, que significa inteiro, e R, que significa real

identificadores novos: qualquer identificador que surge pela primeira vez é classificado como sendo novo, sendo substituído pelo meta-símbolo id. Mais tarde, após a extração de outros identificadores novos e nas próximas ocorrências deste identificador, ele deixa de ser classificado como novo.

identificador já conhecido – este identificador já foi extraído anteriormente e após algumas operações do autômato, passa a ser reconhecido pelo meta-símbolo var.

número inteiro sem sinal – cadeias formadas por um único dígito

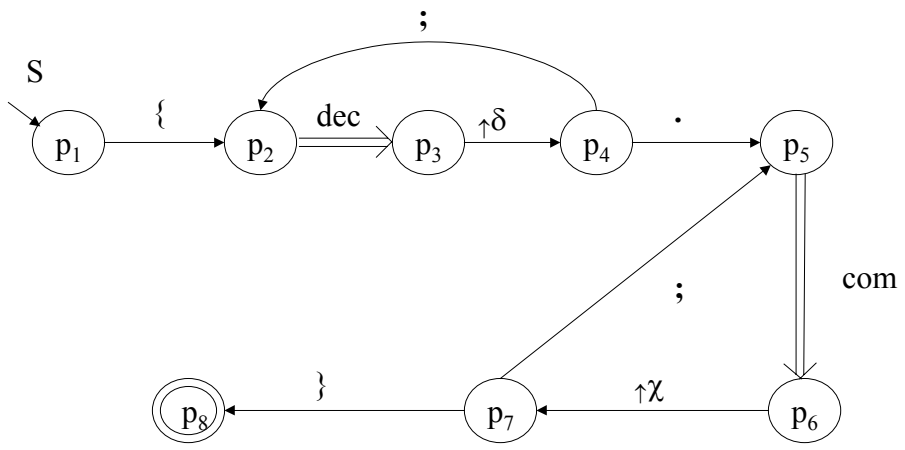
A saída do analisador léxico são os átomos que identificam cada um dos elementos extraídos do texto-fonte. São eles: var (variável não-definida), var_int (variável definida como sendo do tipo inteiro), var_real (variável definida como sendo do tipo real), número inteiro, □x,. onde x é o caractere ASCII extraído do texto-fonte (são os símbolos especiais), e as saídas χ e δ que são os símbolos que indicam o término dos reconhecimentos das submáquinas com e dec, respectivamente.

Convencionalmente, o analisador léxico é acionado sempre antes que qualquer transição seja aplicada. De forma a simplificar a diagramação das submáquinas, essas chamadas serão feitas implicitamente, para não sobrecarregar as figuras.

Máquina pricipal

A máquina principal é representada sem o uso de ações adaptativas. Ela possui um funcionamento bastante simples, e faz somente o controle das chamadas das submáquinas que reconhecem as declarações e os comandos.

A máquina principal é representada pela seguinte figura.



As transições iniciais da máquina principal são as seguintes:

$(\varepsilon, p_1, \{\ ) :\rightarrow (\varepsilon, p_2, \varepsilon)$

$(\varepsilon, p_2, \text{dec}) :\rightarrow (p_3, d_1, \varepsilon)$

$(\varepsilon, p_3, \delta) :\rightarrow (\varepsilon, p_4, \varepsilon)$

$(\varepsilon, p_4, ;) :\rightarrow (\varepsilon, p_2, \varepsilon)$

$(\varepsilon, p_4, .) :\rightarrow (\varepsilon, p_5, \varepsilon)$

$(\varepsilon, p_5, \text{com}) :\rightarrow (p_6, a_1, \varepsilon)$

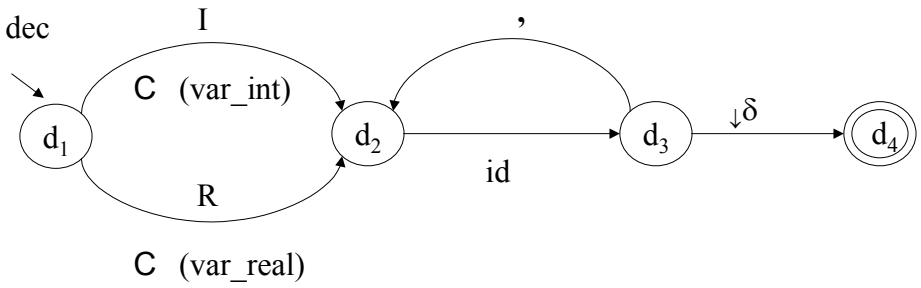
$(\varepsilon, p_6, \chi) :\rightarrow (\varepsilon, p_7, \varepsilon)$

$(\varepsilon, p_7, ;) :\rightarrow (\varepsilon, p_5, \varepsilon)$

$(\varepsilon, p_7, \}) :\rightarrow (\varepsilon, p_8, \varepsilon)$

Submáquina dec (declaração)

A submáquina dec faz o reconhecimento das declarações dos identificadores. Inicialmente, o primeiro símbolo da cadeia de entrada pode ser I (inteiro) ou R (real). Qualquer uma das transições contém a ação adaptativa `C` associada a elas. Esta ação prepara o estado especial chamado Tipo que irá apontar para o estado que designa o tipo que serão classificados os identificadores que serão reconhecidos nesta declaração. O analisador léxico faz a manipulação desta informação, através da consulta deste estado especial Tipo para classificar os identificadores que forem sendo reconhecidos nesta lista de declarações.



As transições iniciais da submáquina dec são as seguintes:

$(\varepsilon, d_1, \text{int}) :\rightarrow (\varepsilon, d_2, \varepsilon)$, C (20)

$(\varepsilon, d_1, \text{real}) :\rightarrow (\varepsilon, d_2, \varepsilon)$, C (17)

$(\varepsilon, d_2, \text{id}) :\rightarrow (\varepsilon, d_3, \varepsilon)$

$(\varepsilon, d_3, ,) :\rightarrow (\varepsilon, d_2, \varepsilon)$

$(\varepsilon, d_3, \varepsilon) :\rightarrow (\varepsilon, d_4, \delta)$, R (d_2, d_3)

C (t) = { x:

/* memoriza, apontado pelo estado especial Tipo, o tipo default para todas as variáveis definidas nesta declaração */

- [$(\varepsilon$, Tipo, $\varepsilon) :\rightarrow (\varepsilon, x, \varepsilon)$]

+[$(\varepsilon$, Tipo, $\varepsilon) :\rightarrow (\varepsilon, t, \varepsilon)$]

}

R (u, v) = {

- [(ε, u, int) :→ (ε, v, ε)]

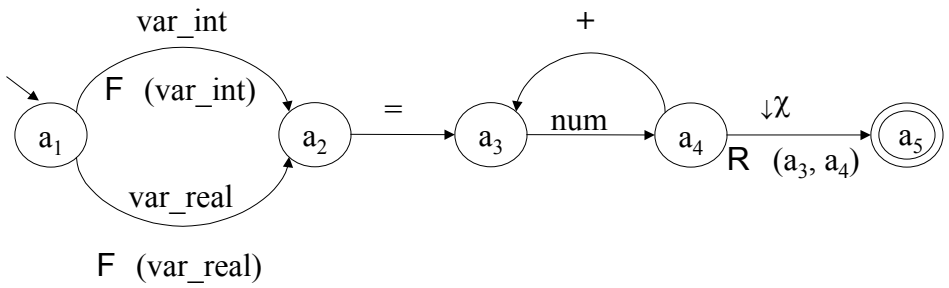
- [(ε, u, real) :→ (ε, v, ε)]

}

Submáquina com (comando)

Esta submáquina reconhece as expressões aritméticas que fazem uso de variáveis, números e operadores de adição.

A primeira transição desta submáquina define o tipo das variáveis que devem fazer parte do comando. Se a primeira variável reconhecida for, por exemplo, do tipo inteiro, a ação adaptativa F acrescenta uma nova transição que parte de estado a_3 e chega no estado a_4, e que obriga o autômato a reconhecer apenas variáveis compatíveis com a primeira variável reconhecida.



$(\varepsilon, a_1, \text{var\_int}) :\rightarrow (\varepsilon, a_2, \varepsilon)$, `F` (var_int)

$(\varepsilon, a_1, \text{var\_real}) :\rightarrow (\varepsilon, a_2, \varepsilon)$, `F` (var_real)

$(\varepsilon, a_2, =) :\rightarrow (\varepsilon, a_3, \varepsilon)$

$(\varepsilon, a_3, \text{num}) :\rightarrow (\varepsilon, a_4, \varepsilon)$

$(\varepsilon, a_4, +) :\rightarrow (\varepsilon, a_3, \varepsilon)$

$(\varepsilon, a_4, \varepsilon) :\rightarrow (\varepsilon, a_5, \chi)$, `R` (a_3, a_4)

`F` (x) = {

/* cria uma transição $a_3 \rightarrow a_4$ consumindo var_int ou var_real, conforme o tipo da variável encontrada na transição $a_1 \rightarrow a_2$ */

+ [$(\varepsilon, a_3, x) :\rightarrow (\varepsilon, a_4, \varepsilon)$

}

Nesta seção será omitido o funcionamento de um reconhecimento realizado com o autômato adaptativo desta linguagem-exemplo. por se tratar de assunto tratado em publicações prévias, por exemplo [Jos93].

Capítulo 5 – Considerações finais

Nesta tese, procurou-se elaborar um estudo de diversos aspectos ligados à especificação e implementação de linguagens sensíveis ao contexto.

Apresentou-se um levantamento de trabalhos correlatos, procurando dar uma visão geral dos formalismos dinâmicos para a especificação e reconhecimento de linguagens, bem como o embasamento teórico relacionado ao assunto.

Uma das partes mais importantes dos avanços obtidos nesta pesquisa foi a apresentação do formalismo gramátical adaptativo, bem como as argumentações que levaram à conclusão da sua equivalência com o seu formalismo dual, que são os Autômatos Adaptativos.

Quantos aos aspectos práticos deste trabalho, foram apresentadas propostas de alguns algoritmos que permitem efetuar a convesão de Gramáticas Adaptativas para a forma equivalente de Autômato Adaptativo e vice-versa. Visando à futura construção de uma ferramenta, apresentou-se também um segundo algoritmo, mais eficiente, de conversão da Gramática Adaptativa para o Autômato Adaptativo.

Este capítulo tem o objetivo de apresentar algumas conclusões obtidas decorrentes do processo desenvolvimento deste formalismo. Apresenta também as contribuições que esta pesquisa trouxe à área, e, por fim, relata uma série de trabalhos futuros que podem dar continuidade a este tema de pesquisa, provavelmente podendo ser utilizados como tema de futuras dissertações ou teses.

5.1 Conclusões e resultados

Mesmo em uma primeira análise deste trabalho pode-se facilmente constatar que, por razões históricas do desenvolvimento desta pesquisa, houve uma forte influência dos Autômatos Adaptativos, sobre os modelos e algoritmos desenvolvidos neste trabalho.

Existem situações em que o emprego do Autômato Adaptativo não é o mais usual ou mais simples, recomendando-se nesses casos que seja feita a utilização da Gramática Adaptativa. Um caso em que a utilização da Gramática Adaptativa é mais apropriada que a dos Autômatos Adaptativos é no caso de representação de linguagens por parte de seu projetista, antes de para ela haver sido construído seu reconhecedor.

Um outra conclusão obtida é que a Gramatica Adaptativa pode ser classificado como sendo do tipo imperativa

Assim como os Autômatos Adaptativos, as Gramática Adaptativas podem representar qualquer tipo de linguagem, desde as regulares, livres de contexto até as sensíveis ao contexto, bastando para isso o emprego ou não das regras de produção adaptativas.

5.2 Contribuições

Nesta seção serão apresentadas algumas contribuições que o presente trabalho trouxe para a área das linguagens formais e autômatos. A primeira ser refere a uma nova notação gramatical, que permite aos projetistas de linguagens especificar linguagens sensíveis ao contexto utilizando técnicas adaptativas.

Uma contribuição desta pesquisa foi fazer um compilação do material bibliográfico sobre a área, reunindo trabalhos relevantes que possibilitaram traçar um histórico sobre a evolução das gramáticas adaptáveis.

Uma das inovações desta notação é a introdução um novo meta-símbolo ϕ (que em teoria dos conjuntos representa o conjunto vazio) que é utilizado para definir o lado direito de uma regra de produção de um não-teminal que será posteriormente alterado dinamicamente pela aplicação de alguma ação adaptativa.

Uma outra característica desta notação em relação às das gramáticas tradicionais constitui-se na utilização de informações de contexto atribuídas nas regras de produção através da utilização de regras especiais, caracterizadas pela presença do símbolo $\leftarrow$ separando o seus lados esquerdo e direito e de símbolos não pertencentes ao conjunto normal de terminais e de não-terminais, que foram denominados símbolos de contexto.

Durante o desenvolvimento deste trabalho, foram também acumulados diversos conhecimentos acerca da aplicação desta teoria a problemas práticos, representados através de técnicas de utilização dos formalismos desenvolvidos, e que podem facilitar o uso da tecnologia adaptativa.

Algumas dessas técnicas foram ilustradas através do exemplo apresentado no capítulo 3 na linguagem $a^n b^n c^n$.

Neste exemplo muitas dessas técnicas foram utilizadas simultaneamente. Uma delas foi a de manter uma linearidade separada nas partes, o que intuitivamente significa que ao invés de gerar os três símbolos ao mesmo tempo, gera-se primeiro os a's, depois os b's e por fim, os c's.

O que se entende por linearidade é que as regras passam a ter o seguinte aspecto

$A \rightarrow \alpha A_1$

$A_1 \rightarrow \alpha A_2$

.....

$A_n \rightarrow \alpha$

onde α é um símbolo terminal e A, A_1,.... A_n são símbolos não-terminais

A técnica empregada para manter esta linearidade e ao mesmo tempo armazenar a informação da quantidade de elementos gerados foi a criação das regras de produção que geram os símbolos b's e c's à medida em que os a's são gerados, em outras palavras a partir de um estado especial, a gramática gera os elementos do primeiro termo e constroi os elementos dos outros termos através da adição dinâmica das próximas regras de produção que serão aplicadas no momento em que forem solicidados.

A cada passo da geração dos símbolos da sentença, a ação adaptativa constrói as regras de produção especiais que utilizam o meta-símbolo ϕ, para que novos símbolos não-terminais possam ser gerados. Tais não-terminais devem estar definidos para que possam ser corretamente utilizados, muito embora sem necessariamente ter nenhum significado na ocasião em que são gerados, pois são posteriormente definidos nos passos subseqüentes da geração da sentença.

Outra técnica interessante levantada nesta pesquisa se refere a enumeração das gramáticas e, conseqüentemente, das regras que compõem cada gramática, ao longo da sua evolução. Esta técnica foi criada para permitir a identificação da origem da regra quando esta evolução se encontra em estágio mais avançado. Assim, torna-se possível recuperar algumas informações importantes, tais como, por exemplo, a ocasião tenha sido efetuadas em que determinadas adaptações em alguma gramática intermediária específica.

Durante o desenvolvimento da gramática adaptativa foram determinadas ainda algumas restrições quanto ao seu uso. Uma delas é a posição que devem ocupar as ações adaptativas, sempre ao final da expressão, do lado direito da regra de produção. Sempre que

ocorrerem situações em que a ação adaptativa fique no meio de uma expressão, então esta expressão deverá ser decomposta em duas partes, de forma que a ação adaptativa fique ao final de uma das regras resultantes. Por exemplo: Se existir uma regra de produção da forma

$N \rightarrow \alpha M \{A\} \beta\delta$,

então deve-se transformá-la em

$N \rightarrow XY$

$X \rightarrow \alpha M \{A\}$

$Y \rightarrow \beta\delta$

As implicações desta técnica se devem à utilização do algoritmo dos rótulos, pois padronizou-se que todas as ações adaptativas devem ficar após o delimitador]. Isto implica que a execução desta ação ocorre posteriormente à aplicação da regra de produção, mantendo-se assim, a coerência com a derivação usualmente empregada.

Outras contribuições deste trabalho à área das linguagens formais foram o desenvolvimento dos algoritmos de conversão canônica entre os formalismos adaptativos mais empregados nesta tese, bem como os teoremas que estabelecem a equivalência entre eles.

Um algoritmo eficiente para a conversão de gramáticas adaptativas para a forma de autômato adaptativo mostrou-se também bastante interessante e útil, pois emprega técnicas simples, desenvolvidas inicialmente para linguagens livres de contexto, e que, com as devidas alterações, puderam ser adequadas às necessidades das linguagens sensíveis ao contexto. Neste caso, foram incorporadas as situações em que devem ser executadas ações adaptativas, e, principalmente, aquelas em que se identificam situações que exigem a incorporação no formalismo, de informações de contexto presentes nas sentenças da linguagem.

Para tanto, as informações de contexto foram representadas nos rótulos associadas às produções, os quais no caso são acompanhados dos símbolos $\downarrow$ e $\uparrow$, que indicam respectivamente as ações de empilhamento e de desempilhamento de informações de contexto que devem ocorrer durante a operação do algoritmo.

Neste processo de desenvolvimento do algoritmo de conversão canônica das gramáticas adaptativas para autômatos adaptativos descobrimos a necessidade de efetuar

transformações prévias na gramática, para que o mapeamento pudesse ser realizado com maior facilidade. A solução escolhida foi a da normalização das regras de produção. Dessa forma, o mapeamento de cada regra de produção para a forma de regras de transição do autômato poderia ser efetuado de maneira mais natural.

No processo inverso, ou seja, para a conversão dos autômatos adaptativos em gramáticas adaptativas equivalentes foi necessária a decomposição das regras de transições que possuíam ações adaptativas anterior e posterior. Neste caso, como a gramática adaptativa só possui ações posteriores, contornou-se a dificuldade substituindo cada regra nesta situação em duas regras de transição, ambas com ações posteriores apenas. Dessa forma, tornou-se possível executar facilmente o mapeamento do autômato para uma gramática equivalente.

Uma outra importante necessidade foi constatada durante o exercício de derivação de sentenças nas novas gramáticas. Constatou-se que as substituição dos não-terminais deveriam ser feitas sempre pelo não-terminal mais à esquerda, ou seja, o esquema de derivação deveria ser o da derivação mais à esquerda. Isto se deve à necessidade de estabelecer e fixar a seqüência de derivações, visto serem as gramáticas adaptativas sensíveis a ordem de aplicação das substituições, pois estas provocam efeitos colaterais.

Um outra contribuição foi a de um pequeno estudo da complexidade da gramática com a apresentação do pior caso e do melhor caso.

5.3 Trabalhos futuros

Pretende-se dar continuidade a essa pesquisa através do desenvolvimento de diversos aspectos não consideradas no escopo do presente trabalho. Uma das principais metas imediatas é a implementação dos algoritmos apresentados neste projeto que representam uma grande importância para a construção de ferramentas, que possam ser utilizadas para a especificação de linguagens sensíveis ao contexto. Podemos citar, como exemplo de ferramenta, um compilador de gramática adaptativa para autômato adaptativo. Este compilador seria construído a partir da implementação do algoritmo de enumeração dos estados na expressão que representa a gramática em questão. A partir dessa enumeração dos estados é possível construir automaticamente o autômato adaptativo.

Além disso, espera-se fazer a demonstração formal dos algoritmos apresentados neste presente trabalho, bem como aprofundar as pesquisas sobre as propriedades deste

formalismo gramatical, como por exemplo investigar a influência da ordem da substituição dos não-terminais na formas sentenciais.

Espera-se que no futuro possamos fazer um estudo mais aprofundado da complexidade deste formalismo.

Tratando-se de um formalismo dependente de contexto, torna-se natural desejar aplicá-lo à formalização de linguagens naturais. Por essa razão, torna-se esta aplicação uma meta importante a ser atingida em um futuro próximo.

5.4 Observações finais

Esta tese destinou-se a dar uma contribuição à área das linguagens formais e autômatos através da apresentação de um formalismo gramatical com características especiais que permitem a alteração da sua estrutura durante a geração de uma sentença da linguagem, e também contribuiu para acrescentar mais um formalismo às técnicas adaptativas iniciadas com a proposta do Autômato Adaptativo.

Esta Gramática Adaptativa procurou também levar em consideração seu importante potencial como ferramenta didática para uso em cursos de computação, com aspectos predominantemente científicos, em especial nas disciplinas que utilizem linguagens formais e autômatos e também em compiladores.

Esta pesquisa ainda está em seu desenvolvimento, e tem toda uma trajetória a ser percorrida.

Para o prosseguimento desta linha de pesquisa, diversas outras deverão se suceder, tais como: a elaboração de ferramentas centradas nos autômatos e nas gramáticas adaptativas, o desenvolvimento de máquinas virtuais baseadas neste modelo, a criação de linguagens de alto nível com paradigma adaptativo, o desenvolvimento dos teoremas que provem as propriedades dos modelos e sua inter-relação.

Estes e outros poderão ser utilizados como temas de pesquisa e como assuntos para o desenvolvimento de teses e dissertações de pós-graduação.

Muitas aplicações, as deversas áreas do conhecimento, também poderão ser desenvolvidos como base nos formalismos adaptativos.

Espera-se que em um futuro próximo diversas aplicações dos formalismos adaptativos sejam testadas e implementadas.

Referências Bibliográficas

[Aho86] AHO, A.V.; SEHTI, R.; ULLMAN, J.D. **Compilers:** Principles, techniques and tools. Reading, Addison-Wesley, 1986.

[Alb91] ALBLAS, H.; MELICHAR, B. **Attribute grammars, applications and systems**. Berlin, Springer-Verlag, 1991. (Lecture notes in computer science, 545)

[Alm95] ALMEIDA JUNIOR, J.R. **STAD - Uma ferramenta para representação e simulação de sistemas através de statecharts adaptativos.** São Paulo 1995, 202p. Tese (Doutorado). Escola Politécnica, Universidade de São Paulo.

[Bas99] BASSETO, B. A.; JOSÉ NETO, J. A stochastic musical composer based on adaptive algorithms. **Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação**. SBC-99 PUC-Rio, Vol 3, pp105-13, 19 a 23 de julho de 1999

[Bro90] BROWNE, D.; TOTTERDELL, P.; NORMAN, M. eds. **Adaptive User Interfaces.** London, Academic Press, (Computers and people series) , 227p., 1990

[Bur90a] BURSHTEYN, B. On the modification of the formal grammar at parse time. **SIGPLAN Notices**, v.25, n.5, p.117-23, 1990.

[Bur90b] BURSHTEYN, B. Generation and recognition of formal languages by modifiable grammars. **ACM SIGPLAN Notices**, v.25, n.12, p.45-53, 1990.

[Bur92] BURSHTEYN, B. USSA - Universal Syntax and Semantics Analyser. **ACM SIGPLAN Notices**, v.27, n.1, p.42-60, 1992.

[Cab92] CABASINO, S.; PAOLUCCI, P.S.; TODESCO, G.M. Dynamic parsers and evolving grammars. **ACM SIGPLAN Notices**, v.27, n.11, p.39-48, 1992.

[Chr69] CHRISTENSEN, C.; SHAW, C.J., eds., Proceedings of the extensible languages symposium, Boston, Massachusetts, May, 13, 1969 [**SIGPLAN Notices,** v.4, n.8, 1969]

[Chr85] CHRISTIANSEN, H. **Syntax, semantics, and implementation strategies for programming languages with powerful abstration mechanisms**. *Datalogiske skrifter*, n.1, Roskilde University Centre, 1985.

[Chr86a] CHRISTIANSEN, H. Recognition of generative languages. **Lectures notes in computer science**, 217, New York: Springer-Verlag, 1986, p.63-81.

[Chr86b] CHRISTIANSEN, H. **Parsing and compilation of generative languages**. *Datalogiske skrifter*, n.3, Roskilde University Centre, 1986.

[Chr88a] CHRISTIANSEN, H. **The syntax and semantics of extensible programming languages**. *Datalogiske skrifter*, n.14, Roskilde University Centre, 1988.

[Chr88b] CHRISTIANSEN, H. **Programming as language development**. *Datalogiske skrifter*, n.15, Roskilde University Centre, 1988.

[Chr90] CHRISTIANSEN, H. A survey of adaptable grammars. **SIGPLAN Notices**, vol.25, n.11, p.35-44, 1990.

[Col78] COLMERAUER, A. Metamorphosis grammars. In: BOLC, L. ed., Natural language communications with computers, **Lecture notes in computer science**, 63, New York, Springer-Verlag, p.133-89, 1978.

[Cor00] CORE-2000. II Workshop de computação reconfigurável. Fundação Eurípedes de Marília – Faculdade de Informática, Marília, SP, Brasil, Agosto 10-11 de 2000. http://www.fim.fundanet.br/core2000, data: 16/3/00

[Der88] DERANSART, P.; JOURDAN, M.; LORHO, B. Attribute grammars: Definitions, systems and bibliography. **Lectures Notes in Computer Science**, 323, New York, Springer-Verlag, 1988

[DeR76] DeRemer, F.L. Review of formalisms and notation. In: Compiler Construction: An advanced course. **Lecture Notes in Computer Science**, 21. Goos, G.; Hartmanis, J.(eds.), New York: Springer-Verlag, p. 37-56, 1976

[Fpl00] FPL-200 10th International conference on field programmable logic and applications. Villach, Austria 28-30 August 2000
URL: http://xputers.informatik.uni-kl.de/FPL/FPL2000/detailed_fpl.html,
data: 16/03/00

[For63] FORINO, A.C. Some remarks on the syntax of symbolic programming languages. **Communications of the ACM**, v.6, n.8, p.456-60, 1963.

[Gog79] GOGUEN, J.A.; THATCHER, J.W.; WAGNER, E.G. An initial algebra approach to the specification, correctness, and implementation of abstract data types. chapter 5 of Raymond T. Yeh, editor. **Current trends in programming methodology**, volume IV [Data Structuring], Englewood Cliffs: Prentice-Hall, p.80-149, 1979.

[Gri86] GRISHMAN, R. **Computational linguistics**: An introduction. (Studies in natural language processing) Cambridge University Press, 1986

[Han73] HANFORD, K.V.; JONES, C.B. Dynamic syntax: A concept for the definition of the syntax of programming languages, **Annual review in automatic programming**, 7, n.2, p.115-42, 1973.

[Har87] HAREL, D. Statecharts: a visual formalism for complex systems. Science of Computer Programming, v.8, n.3, p.231-74, August. 1987.

[Har78] HARRISON, M.A. **Introduction to formal language theory**. Addison-Wesley, 1978.

[Hop69] HOPCROFT, J.E.; ULLMAN, J.D. **Formal languages and their relation to automata**. Addison-Wesley, 1969.

[Jos87] JOSÉ NETO, J. **Introdução à compilação**. Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, 1987.

[Jos88] JOSÉ NETO, J. Uma solução adaptativa para reconhecedores sintáticos. **Anais da Escola Politécnica, Departamento de Engenharia de Eletricidade**, 1988.

[Jos93] JOSÉ NETO, J. **Contribuição à metodologia de construção de compiladores**. São Paulo, 1993, 272p. Tese (Livre-Docência) Escola Politécnica, Universidade de São Paulo.

[Jos94] JOSÉ NETO, J. Adaptive automata for context-dependent languages. **ACM SIGPLAN Notices**, v.29, n.9, p.115-24, 1994.

[Jos98] JOSÉ NETO, J.; IWAI, M.K. Adaptive automata for syntax learning. **XXIV Conferencia Latinoamericana de Informática CLEI'98**, Quito - Ecuador, Centro Latinoamericano de Estudios em Informatica, Pontificia Universidade Católica Del Ecuador, tomo 1, pp.135-146. 19 a 23 de Outubro de 1998

[Knu68] KNUTH, D.E. Semantics of context-free languages. **Mathematical System Theory**, 2, n.2, p.127-45, 1968

[Knu71] KNUTH, D.E. Semantics of context-free languages. **Mathematical System Theory**, 5, n.1, p.95-6, 1971(correction)

[Lew81] LEWIS, H.R.; PAPADIMITRIOU, C.H. **Elements of the theory of**

computation. Prentice-Hall, 1981.

[Lie96] LIEBERHERR, K.J. **Adaptive object-oriented software:** The Demeter Method wiht propagation patterns. PWS Publishing Company, Boston, 1996. URL: http://www.ccs.neu.edu/research/demeter/biblio/dem-book.html, Data:16/11/1999

[Mas87] MASON, K.P. Dynamic Template Translator - A new device for specifying programming languages. **International Journal of Computer Mathematics**, v.22, n.3+4, p.199-212, 1987.

[Paa95] PAAKKI, J. Attribute grammar paradigms - A high-level methodology in language implementation. **ACM Computing Surveys**, v. 27, n.2, p.196-255, 1995.

[Pag81] PAGAN, F.G. **Formal specification of programming languages:** A panoramic primer. New Jersey, Prentice-Hall, Inc., 1981

[Per80] PEREIRA, F.C.N.; WARREN, D.H.D. Definite clause grammars for language analysis - A survey of the formalism and a comparison with augmented transition networks. **Artificial Intelligence**, 13, p.231-78, 1980.

[Per97] PEREIRA, J.C.D; JOSÉ NETO, J. Um ambiente de desenvolvimento de reconhecedores sintáticos baseados em autômatos adaptativos. II Brazilian Symposium on Programming Languages (SBLP'97), 3-5 September 1997, Institute of Computing, State University of Campinas, Campinas, SP, Brazil, pp.139-50

[Per99] PEREIRA, J.C.D. **Ambiente integrado de desenvolvimento de reconhecedores sintáticos baseado em autômatos adaptativos.** São Paulo, 1999, 162p. Dissertação (Mestrado) Escola Politécnica, Universidade de São Paulo.

[Pet81] PETERSON, J.L. **Petri net theory and the modeling of systems**. Prentice-Hall,

Inc., Englewood Cliffs, NJ, 1981

[Rob91] ROBERTS, G.H. A note on modifiable grammars. **ACM SIGPLAN Notices**, v.26, n.3, p.18, 1991.

[Rub93] RUBINSTEIN, R.; SHUTT, J.N. **Self-modifying finite automata**. Technical Report WPI-CS-TR-93-11, Worcester Polytechnic Institute, Worcester, Massachusetts, December, 14p, 1993.

[Rub95a] RUBINSTEIN, R.; SHUTT, J.N. **Self-modifying finite automata** - Basic definitions and results. Technical Report WPI-CS-TR-95-2, Worcester Polytechnic Institute, Worcester, Massachusetts, August, 14p, 1995.

[Rub95b] RUBINSTEIN, R.S.; SHUTT. J.N. Self-modifying finite automata: An introduction, **Information processing letters**, v.56, n.4, 24, p.185-90, 1995.

[Rub95c] RUBINSTEIN, R.; SHUTT, J.N. Self-modifying finite automata. In: Pehrson, B.; Simon I., editors, Technology and Foundations: Information'94 Vol. I: Proc. 13th IFIP World Computer Congress, p. 493-498, Amsterdam, 1994. North-Holland.

[Sam69] SAMMET, J.E. **Programming languages:** History and fundamentals. Prentice-Hall, Englewood Cliffs, NJ, 1969.

[San97] SANTOS, J.M.N. **Um formalismo adaptativo com mecanismos de sincronização para aplicação concorrentes.** São Paulo, 1997, 98p. Dissertação (Mestrado) Escola Politécnica, Universidade de São Paulo.

[Sch71] SCHUMAN, S., ed., Proceedings of the international symposium on extensible languages, Grenoble, France, September 6-8, 1971 (**SIGPLAN Notices**, v.6, n.12, 1971)

[Sch81] SCHWFEL, H.-P. **Numerical optimization of computer models**. New York,

John Wiley & Sons, 1981.

[Shu93] SHUTT, J.N. **Recursive adaptable grammar**. M.S. Thesis, Computer Science Department, Worcester Polytecnic Institute, Worcester Massachusetts, 140p, 1993.

[Shu95] SHUTT, J.N. **Self-modifying finite automata** - Power and limitations. Technical Report WPI-CS-TR-95-4, Worcester Polytechnic Institute, Worcester, Massachussets, December, 32p, 1995.

[Sol73] SOLNTSEFF, N.; YEZERSKI, A. A survey of extensible programming languages, **Annual review in automatic programming**, 7, n.2, p. 267-307, 1973.

[Spe93] SPEARS, W.M.; DE JONG, K.A.; BÄCK, T.; FOGEL, D.B.; GARIS, H. An overview of evolutionary computation. **Lecture notes in artificial intelligence**, 667, pp.442-59, 1993.

[Sta71] STANDISH, T.A., PPL - An extensible language that failed. Proceedings of the international symposium on extensible languages, Grenoble, France, September 6-8, 1971 (**SIGPLAN Notices**, v.6, n.12, 1971), p.144-45, 1971

[Sta75] STANDISH, T.A.. Extensibility in programming language design. **SIGPLAN Notices**, v.10, n.7, p. 18-21, 1975.

[Weg80] WEGBREIT, B. **Studies in extensible programming languages.** [Outstanding dissertations in the Computer Science], New York: Garland Publishing, Inc. 1980.

[Wij75] WIJNGAARDEN, A.V., et al, Revised report on the Algorithmic Language Algol 68, **Acta Informatica**, v.5, n.1-3, p.1-236, 1975.

[Woo70] WOODS, W.A. Transition network grammars for natural language analysis.

Communications of ACM, v.13, n.10, p. 591-606, 1970.