Introdução às Gramáticas Adaptativas

Margarete Keiko Iwai
João José Neto
Escola Politécnica da USP – Av. Prof.Luciano Gualberto, trav.3, n.158
CEP 05508-900 – São Paulo – SP – Fone: (0xx11) 3818-5402
e-mail: mkiwai@pcs.usp.br
jjneto@pcs.usp.br

RESUMO

Este trabalho apresenta 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. Este artigo faz uma pequena apresentação dos trabalhos correlatos e também apresenta um exemplo de utilização da gramática.

ABSTRACT

The present work 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 work, we have made a compilation of the some related works on adaptable grammars and also presents an example of adaptive grammar

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 máquina 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, baseado em [Iwa00], representa uma continuidade à linha de pesquisa iniciada em [Jos93], em que são apresentados os autômatos adaptativos, formalismo cognitivo utilizado para o tratamento de linguagens sensíveis ao contexto. Este modelo constitui 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 trabalho 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 trabalho 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 servirá 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.

2. Conceitos

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 [Alm95] e das Redes de Markov Adaptativos [Bas99].

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.

3.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*.
$V^0 = V_N{}^0 \cup V_T \cup V_C$, onde $V_N{}^0$, 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 $i \in \natural$:

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

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 $i \in \natural$:

$$N \rightarrow \phi$$

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 $i \in \natural$:

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

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

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

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 $i \in \mathbb{N}$, 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:

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

Derivação (seqüência de derivação)

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 α_k, α_{k+1}, α_{k+2},.... , α_{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 ($\Leftrightarrow$):
a.) Define-se uma derivação da forma { A } $\alpha \Rightarrow_{G^i} \beta$ como sendo $\alpha \Leftrightarrow_{(A,\, i)} \beta$
b.) Define-se uma derivação da forma $\alpha \Rightarrow_{G^i} \beta$ como sendo $\alpha \Leftrightarrow_{(\varepsilon,\, i)} \beta$
c.) Define-se derivação da forma { A } $\alpha \Rightarrow_{G^i} \beta$ ou $\alpha \Rightarrow_{G^i} \beta$ como sendo $\alpha \Leftrightarrow \beta$
d.) Uma seqüência de zero ou mais derivações da forma $\alpha \Leftrightarrow \alpha_1 \Leftrightarrow \alpha_2 \Leftrightarrow$ ..... $\Leftrightarrow \alpha_n$
como sendo $\alpha \Leftrightarrow^* \alpha_n$
Uma linguagem gerada por uma gramática adaptativa G (denota-se L(G)) é definida como sendo { $\omega \mid \omega \in V_T^*$ e $S \Leftrightarrow^* \omega$ }. 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, $S \Leftrightarrow^* \alpha$.
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:

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

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

A esta transição corresponde o seguinte par de produções gramaticais:

$$E' \rightarrow \sigma E'' \text{ e}$$
$$E'' \leftarrow \{B\} \sigma' E'$$

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

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

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 $S' \rightarrow S$, garantindo que todas as raizes da gramática não sejam auto-recursivas.

Tomemos uma regra de produção com a seguinte forma:

$$A \rightarrow \{A\} a_1 a_2 .... a_n$$

a.) quando n = 0 ou seja $A \rightarrow \varepsilon$ a forma normal correspondente é $A \rightarrow \{A\} \varepsilon$

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

1. $A \rightarrow \{A'\} a_1 A_1$	n. $\alpha_{n-1} A_{n-1} \rightarrow a_n A_n$
2. $\alpha_1 A_1 \rightarrow a_2 A_2$	n+1. $\alpha_n A_n \leftarrow \alpha A_{n+1}$
.....	n+2. $A_{n+1} \rightarrow \varepsilon$

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 $A = \varepsilon$, então $A' = \varepsilon$

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:

$X \rightarrow \{A_1'\} A_1 X_1$	$X_2 \rightarrow \{A_2'\} A_2 X_3$	$X_2 \rightarrow \varepsilon$
$\alpha_1 X_1 \rightarrow X_2$	$\alpha_2 X_3 \rightarrow X$	

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 $X \rightarrow (\{A_1\} A_1 | \{A_2\} A_2 | .... | \{A_n\} A_n)$ é feita da seguinte forma:

$X \rightarrow \{A_1'\} A_1 X_1$	$X \rightarrow \{A_2'\} A_2 X_1$	.....	$X \rightarrow \{A_n'\} A_n X_1$	$X_2 \rightarrow \varepsilon$
$\alpha_1 X_1 \rightarrow X_2$	$\alpha_2 X_1 \rightarrow X_2$		$\alpha_n X_1 \rightarrow X_2$	

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 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_4 → ε]	- [v_3 → ε]
? [u_1 → b u_2]	- [y → d v_1]	+[z → f t_1]
? [u_2 → c u_3]	- [v_1 → e v_2]	+[t_1 ← z' t_2]
? [u_3 ← x' u_4]	- [v_2 ← y' v_3]	+[t_2 → ε]

}

As produções de contexto, da forma

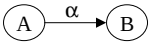
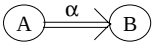
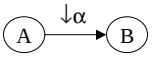
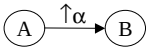
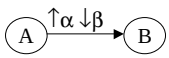
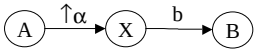
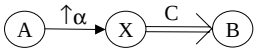
A ←αB	αA←βB
αA→B	αA→ bB

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

Propriedades

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

Prova: A demostração deste teorema é bastante simples e consiste na simulação de cada regra da gramática adaptativa em uma regra de transição do autômato adaptativo e vice-versa, partindo sempre da forma normalizada da gramática para simplificar a demonstração. A seguir é apresentada uma tabela que representa a conversão da regra de produção da gramática na regra de transição do autômato.

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

4. 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 uma 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 $\rightarrow \alpha$]
E (t, α) }
E (t, α) = { - [id $\rightarrow$ {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	5^0. $P_4 \rightarrow$ "." P_5	9^0. $P_7 \rightarrow$ "}" P_8
2^0. $P_2 \rightarrow$ dec P_3	6^0. $P_5 \rightarrow$ com P_6	10^0. $P_8 \rightarrow \varepsilon$

3^0. $\delta\ P_3 \rightarrow P_4$	7^0. $\chi\ P_6 \rightarrow P_7$	
4^0. $P_4 \rightarrow$ ";" P_2	8^0. $P_7 \rightarrow$ "**;**" P_5	

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 → ({C (var_int)} "I" | {C (var_real)} "R") (id \ ","), temos:

11^0. dec → {C (var_int) } "I" D_2	14^0. $D_3 \rightarrow$ "," D_2
12^0. dec → {C (var_real)} "R" D_2	15^0. $D_3 \leftarrow \delta\ D_4$
13^0. $D_2 \rightarrow$ id D_3	16^0. $D_4 \rightarrow \varepsilon$

Normalizando a produção

com → ({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	21^0. $A_3 \rightarrow$ num A_4
18^0. com → {F (var_real) }var_real A_2	22^0. $A_4 \rightarrow$ "+" A_3
19^0. $A_2 \rightarrow$ "=" A_3	23^0. $A_4 \leftarrow$ {R (var)} $\chi\ A_5$
20^0. $A_3 \rightarrow$ var A_4	24^0. $A_5 \rightarrow \varepsilon$

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.

Devido ao pouco espaço disponível serão apresentadas a seguir as regras de produção pertencentes às gramáticas inicial, intermediárias e final que foram adaptadas para obter a derivação do programa escrito na linguagem descrita anteriormente.

As regras foram numeradas de acordo com a gramática a qual ela pertence e serão utilizadas no momento adequado.

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. $\delta\ P_3 \rightarrow P_4$	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 → {S ("a") } "a"	34^0. id → {S ("j")} "j"	43^0. id →{S ("s")} "s"
26^0. id → {S ("b") } "b"	35^0. id → {S ("k")} "k"	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"	37^0. id → {S ("m")} "m"	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”	66^5. var_int → “a”
62^0. Tipo’ → ϕ	64^3. var_real → “m”	67^6. var → var_real
62^1. Tipo’→ var_real	65^4. Tipo’→ var_int	68^8. var → var_int

A seguir é apresentado o processo de derivação do programa-texto:

1. (regra 1^0.) P_1 → “{” P_2	$\underline{P_1} \Rightarrow_{G}^{0}$ “{” $\underline{P_2}$
2. (regra 2^0.) P_2 → dec P_3	$\Rightarrow_{G}^{0}$ “{” $\underline{\text{dec}}$ P_3
3. (regra 12^0.) dec → {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’ → x]
+ [Tipo’→ var_real] }

Esta ação adaptativa remove a regra 62^0. e acrescenta a regra 62^1.

A nova gramática G^1 contém as seguintes regras de produção: 1^0. até 61^0. e a regra 62^1.

Continuando a derivação do texto de acordo com as regras pertencentes com a gramática G^1, temos os seguintes passos:

4. (regra 13^0.) D_2 → id D_3	$\Rightarrow_{G}^{1}$ { R $\underline{\text{id}}$ D_3 P_3
5. (regra 34^0.) id → {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’ → t]
B (var_real , “k”) }

B (var_real, “k”) = { + [var_real → “k”]
E (var_real, “k”) }

E (var_real, “k”) = { - [id → {S (“k”)} “k”] }

Esta ação adaptativa acrescenta a regra 63^2. e remove a regra 35^0.

A nova gramática G^2 contém as seguintes regras de produção:1^0. até 34^0., 36^0. até 61^0 , 62^1. e 63^2.

Continuando a derivação do texto de acordo com as regras pertencentes com a gramática G^2, temos os seguintes passos

6. (regra 14^0.) D_3 → “,” D_2	$\Rightarrow_{G}^{2}$ { R k , $\underline{D_2}$ P_3
7. (regra 13^0.) D_2 → id D_3	$\Rightarrow_{G}^{2}$ { R k , $\underline{\text{id}}$ 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”] }

Esta ação adaptativa acrescenta a regra 64^3. e remove a regra 37^0.

A nova gramática G^3 contém as seguintes regras de produção:1^0. até 34^0., 36^0., 38^0. até 61^0., 62^1. , 63^2. e 64^3.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^3, 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 $\rightarrow$ {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' $\rightarrow$ x]
+ [Tipo'$\rightarrow$ var_int] }

Esta ação adaptativa acrescenta a regra 65^4. e remove a regra 62^1.

A nova gramática G^4 contém as seguintes regras de produção:1^0. até 34^0., 36^0., 38^0. até 61^0., 63^2. , 64^3. e 65^4.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^4, temos os seguintes passos:

15. (regra 13^0.) $D_2 \rightarrow$ id D_3	$\Rightarrow_{G^4}$ { R k , m ; I $\underline{\text{id}}$ D_3 P_3
16. (regra 25^0.) id $\rightarrow$ 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' $\rightarrow$ t]
B (var_int , "a") }

B (var_int, "a") = { + [var_int $\rightarrow$ "a"]
E (var_int, "a") }

E (var_int, "a") = { - [id $\rightarrow$ {S ("a")} "a"] }

Esta ação adaptativa acrescenta a regra 66^5. e remove a regra 25^0.

A nova gramática G^5 contém as seguintes regras de produção:1^0. até 24^0., 26^0. até 34^0., 36^0., 38^0. até 61^0., 63^2., 64^3. , 65^4. e 66^5.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^5, temos os seguintes passos:

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$\rightarrow${F (var_real)} var_real A_2	$\Rightarrow_{G^5}$ { 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 $\rightarrow$ var_real] }

Esta ação adaptativa acrescenta a regra 67^6.

A nova gramática G^6 contém as seguintes regras de produção:1^0. até 24^0., 26^0. até 34^0., 36^0., 38^0. até 61^0., 63^2., 64^3. , 65^4. , 66^5. e 67^6.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^6, temos 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 → "+" A_3	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + $\underline{A_3}$ P_6
28. (regra 20^0.) A_3 → 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 → 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 → "m"	$\Rightarrow_{G^6}$ { R k , m ; I a . k = 4 + m $\underline{A_4}$ P_6
31. (regra 23^0.) A_4 ← {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. A seguinte ação adaptativa é aplicada:

R (var) = { y : -[var → var_real] }

Esta ação adaptativa remove a regra 67^6.

A nova gramática G^7 contém as seguintes regras de produção:1^0. até 24^0., 26^0. até 34^0., 36^0., 38^0. até 61^0., 63^2., 64^3. , 65^4. e 66^5.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^7, temos os seguintes passos:

32. (regra 24^0.) A_5 → ε	$\Rightarrow_{G^7}$ { R k , m ; I a . k = 4 + m $\underline{\chi\ \varepsilon}$ P_6
33. (regra 7^0.) χ P_6 → P_7	$\Rightarrow_{G^7}$ { R k , m ; I a . k = 4 + m $\underline{P_7}$
34. (regra 8^0.) P_7 → "**;**" P_5	$\Rightarrow_{G^7}$ { R k , m ; I a . k = 4 + m **;** $\underline{P_5}$
35. (regra 6^0.) P_5 → com P_6	$\Rightarrow_{G^7}$ { R k , m ; I a . k = 4 + m **;** $\underline{\text{com}}$ P_6
36. (regra 17^0.) com → {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 → var_int] }

Esta ação adaptativa acrescenta a regra 68^8.

A nova gramática G^8 contém as seguintes regras de produção:1^0. até 24^0., 26^0. até 34^0., 36^0., 38^0. até 61^0., 63^2., 64^3. , 65^4. , 66^5. e 68^8.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^8, temos os seguintes 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 = $\underline{\text{num}}$ 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 + $\underline{\text{var}}$ A_4 P_6
43. (regra 68^8.) var → var_int	$\Rightarrow_{G^8}$ { R k , m ; I a . k = 4 + m **;** a = 1 + $\underline{\text{var\_int}}$ 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)}χ 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] }

Esta ação adaptativa acrescenta a regra 68^8.

A nova gramática G^9 contém as seguintes regras de produção:1^0. até 24^0., 26^0. até 34^0., 36^0., 38^0. até 61^0., 63^2., 64^3. , 65^4. e 66^5.

Retomando a derivação do texto de acordo com as regras pertencentes com a gramática G^9, temos os seguintes passos:

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.) χ $P_6 \rightarrow P_7$	$\Rightarrow_{G^9}$ { 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 }

5. Avaliação

As gramáticas adaptativas podem ser empregadas em algumas aplicações práticas que são comuns na área de linguagens formais. Algumas das aplicações mais usuais é a representacão de linguagens de programação e de linguagens naturais.

Este formalismo também pode ser utilizado para representar sistemas de software a partir da sua especificação, que pode ser transformado em uma gramática adaptativa. Com isso, utilizando um algoritmo de conversão de gramática para autômato, pode-se criar o autômato adaptativo equivalente. Em seguida, pode-se acrescentar rotinas semânticas e assim, obter o esquema do sistema. Um exemplo de aplicação pode ser em sistema de controle de processos ou então na criação de páginas Web.

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, 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 pior caso, que todas as ações adaptativas tenham o mesmo comportamento e sejam repetidas 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 comportamento 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.

As gramáticas adaptativas apresentam uma boa capacidade de expressar com clareza os aspectos léxico, sintático e semântico de uma linguagem.
Os fenômenos envolvendo os aspectos semânticos são descritos através da utilização das ações adaptativas que conseguem descrever o fenômeno de forma considerável, embora de maneira um pouco camuflada.
A exemplo de outros formalismos auto-modificáveis, as gramáticas adaptativas não explicitam suficientemente os efeitos e as ocasiões exatas em que determinadas modificações ocorrem, exigindo do usuário uma previsão muito detalhada das alterações esperadas em cada situação. Isto também ocorre com as gramáticas de atributos, com as suas regras de cálculo de atributos, e com as gramáticas W, com as suas meta-regras. O que acaba exigindo o redesenho do cenário cada vez que se aplica algum técnica de modificação.
Uma vantagem da gramática adaptativa é a sua capacidade para descrever, com apenas esta notação, linguagens do tipo 3, 2 e 1.
Outra vantagem diz respeito a facilidade de implementação. Comparando com gramáticas W, a gramática adaptativa é mais fácil de ser implementada. Ao contrário do que ocorre com a gramática de atributos que apresenta mais facilidade de implementação.
Uma outra vantagem da gramática adaptativa foi apresentada através do estudo de complexidade que demonstrou que a gramática adaptativa é essencialmente linear em relação ao espaço.

Uma observação que se pode fazer sobre a gramática adaptativa é que ela é algébrica, mas pode ser utilizada como inspiração para outra meta-linguagem equivalente, em um formato visual, que poderá ser automatizada através de uma ferramenta, a exemplo do que foi feito com o sistema RSW [Per99], baseado nos autômatos adaptativos.

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 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.

Estas técnicas foram ilustradas através do exemplo da linguagem $a^n b^n c^n$, desenvolvido no capítulo 3 de [Iwa00]

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.

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 integração dos algoritmos apresentados em [Iwai00] 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.

Observações finais

Este trabalho 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.
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 diversas 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.

6. Referências

[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

[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.

[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.

[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.

[Iwa00] IWAI, M.K. **Um formalismo gramatical adaptativo para linguagens dependentes de contexto**. São Paulo, 2000, 191p. Tese (Doutorado). Escola Politécnica, Universidade de São Paulo.

[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)

[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.

[Pag81] PAGAN, F.G. **Formal specification of programming languages:** A panoramic primer. New Jersey, Prentice-Hall, Inc., 1981

[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.

[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.

[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.

[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.

[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.