UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

CÉSAR GARCIA DAUDT

GABRIEL BARUFI VERAS

Ambigüidade em gramáticas livre de contexto: Um problema não-solucionável

Trabalho de conclusão de Disciplina.

Prof. Dr. Tiarajú Asmuz Diverio

Porto Alegre, junho de 2010.

SUMÁRIO

LISTA DE FIGURAS....4

RESUMO....5

1 INTRODUÇÃO....6

1.1 Abordagem prática do problema....6

2 AMBIGUIDADE EM GRAMÁTICAS....7

2.1 Ambiguidade inerente a linguagem....7

2.2 Ambiguidade em linguagens de programação....8

3 NÃO-AMBIGUIDADE VERTICAL E HORIZONTAL....9

4 DEMONSTRAÇÃO DA NÃO-SOLUCIONABILIDADE....9

4.1 Definições importantes....10

4.2 Formalização....11

4.3 Demonstração....11

4.4 Observações relevantes....12

5 CONCLUSÃO....13

REFERÊNCIAS....14

LISTA DE FIGURAS

Figura 2.1: Exemplo de linguagem inerentemente ambígua 7

Figura 2.2: Árvore de derivação de dois se-então encadeados 8

Figura 3.1: Definições de não-ambiguidade 9

RESUMO

Propõe-se a definição e exploração breve de tópicos relevantes no que toca as linguagens livre de contexto, sendo esses usados como base para a prova da não-solucionabilidade do problema da ambigüidade em gramáticas livre de contexto, o qual é caracterizado propriamente neste trabalho. Por fim, dá-se uma noção de caminhos utilizados em demonstrações deste mesmo gênero e indica-se referências relevantes nas questões levantadas.

Palavras-Chave: ambigüidade, gramáticas livres de contexto, linguagens livres de contexto, linguagens formais.

1 INTRODUÇÃO

A ambiguidade sintática permite que uma senteça tenha mais de uma interpretação sintática. Um exemplo clássico é a frase "ela viu o homem com um telescópio", nela a frase "com um telescópio" pode estar associada com "viu" ou com "o homem". A presença de ambiguidade numa gramática livre de contexto pode ameaçar a confiabilidade ou a performance de ferramentas que se utilizam dela. Alguns campos onde gramáticas livres de contexto são usadas incluem análises de RNA, linguagens naturais controladas ou mesmo linguagens de programação. (SCHMITZ, 2006)

1.1 Abordagem prática do problema

Na bioinformática, linguagens livres de contexto tem diversas aplicações importantes como por exemplo na comparação de sequências, análise de padrões e análise da estrutura secundária do RNA. Recentemente, a ambiguidade tem ganhado atenção, pois muitos algorítmos importantes tem mostrado resultados incorretos na presença de ambiguidade. O problema da ambiguidade aumenta na análise de biosequência por causa da necessidade de checar propriedades estáticas de algorítmos de programação dinâmica empregados. Pode parecer surpreendente que a análise estática da propriedade de programas pode ser abordada como uma questão de ambiguidade da linguagem num nível de linguagens formais.. (SCHMITZ, 2006)

2 DA AMBIGUIDADE DAS GRAMÁTICAS

Seja *G* uma gramática e *w* uma frase gerada por *G*, *w* será ambígua se há pelo menos duas árvores de derivação diferentes com suas folhas formando a frase *w*. Uma gramática será ambígua se *G* gerar pelo menos uma frase ambígua.

Com isso é definido ambiguidade de gramáticas, mas não de linguagens. Existe muitos casos em que é possível eliminar a ambiguidade de frases alterando regrasn da gramática sem alterar a linguagem. Porém em alguns casos não é possível eliminar a geração de frases ambíguas de uma gramática sem alterar a linguagem, nesses caso a linguagem é dita como uma linguagem inerentemente ambígua. Um exemplo é a linguagem $a^n b^m c^m d^n$.(ČEŠKA, 2007)

2.1 Ambiguidade inerente a linguagem

Para exemplificar um caso comum de ambiguidade inerente à linguagem é possível tomar uma linguagem que forma um conjunto qualquer de somas e subtrações $(x|x\text{-}x|x+x)^+$ e a sua gramatica (G=({S}, {x,+,-}, P, S) tal que P={ S→x ; S→S+S ; S→S-S }). É fácil notar que para qualquer frase com mais do que três símbolos "x" a produção será ambígua, pois não há ordem de precedência entre as somas e subtrações. Por exeplo, caso seja gerada a árvore de derivação para "x+x-x", seria obtido duas árvores. Uma delas gerando E+E e depois E-E com o segundo E, e outra gerando E-E e depois E+E com o primeiro E. Assim é visto que, claramente, a linguagem para $(x|x\text{-}x|x+x)^+$ é inerentemente ambígua.

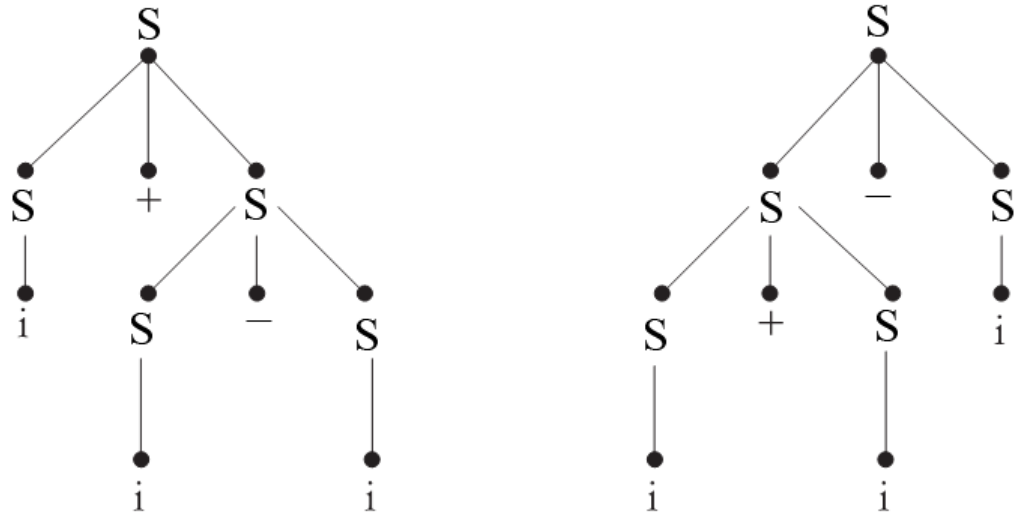

Figura 2.1: Exemplo de linguagem inerentemente ambígua (ČEŠKA, 2007).

A ambiguidade de uma gramática geralmente é uma propriedade negativa para certas aplicações, pois apesar de tornar a gramática mais simples ela pode levar a interpretações indesejáveis, como na construição de compiladores. Neste caso, muitas

vezes são adicionadas regras semânticas adicionais para evitar interpretações errôneas de frases ambíguas. (ČEŠKA, 2007)

2.2 Ambiguidade em linguagens de programação

Um exemplo comum de ambiguidade em linguagens de programação são os condicionais se-então-senão.

S → se *b* então S senão S

S → se *b* então S

S → *p*

Onde *b* denota uma expressão booleana e *p* algo que não seja uma expressão condicional. Este pequeno exemplo pode levar a uma interpretação ambígua do condicional Por exemplo, há duas árvores de derivações diferentes para a sentença:

se *b* então se *b* então *p* senão *p*

Algumas linguagens como Pascal solucionam a ambiguidade com a regra semântica: "um dado senão está relacionado com o então mais próximo". Assim, é possível capturar essa regra sintaticamente:

S_1 → se *b* então S_1 | se *b* então S_2 senão S_1 | *p*

S_2 → se *b* então S_2 senão S_1 | *p*

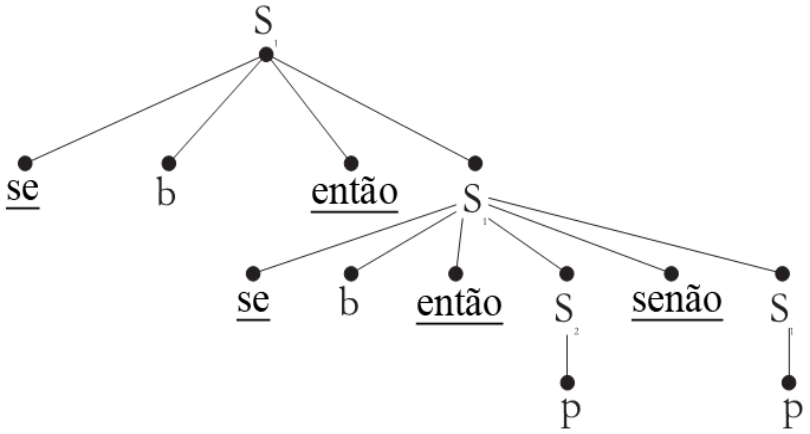

Figura 2.2: Árvore de derivação de dois se-então encadeados (ČEŠKA, 2007).

3 NÃO-AMBIGUIDADE VERTICAL E HORIZONTAL

Uma gramática G é verticalmente não-ambígua sse

$$\forall n \in \mathcal{N}, \alpha, \alpha' \in \pi(n), \alpha \neq \alpha' : \ \mathcal{L}_G(\alpha) \cap \mathcal{L}_G(\alpha') = \emptyset$$

Uma gramática G é horizontalmente não-ambígua sse

$$\forall n \in \mathcal{N}, \alpha \in \pi(n), i \in \{1, \ldots, |\alpha|-1\} : \ \mathcal{L}_G(\alpha_0 \cdots \alpha_{i-1}) \between \mathcal{L}_G(\alpha_i \cdots \alpha_{|\alpha|-1}) = \emptyset$$

aonde $\between$ *é o operador de sobreposição definido por*

$$X \between Y = \{\ xay \mid x, y \in \Sigma^* \wedge a \in \Sigma^+ \wedge x, xa \in X \wedge y, ay \in Y\}$$

Figura 3.1: Definições de não-ambiguidade (BRABAND, 2008).

Intuitivamente, não-ambiguidade vertical significa que, durante o *parsing* de uma *string*, haverá a escolha entre duas produções diferentes de um não-terminal. A sobreposição é o conjunto de *strings* em XY que podem ser divididas não unicamente em uma parte X e uma parte Y. Por exemplo, se X = {x, xa} e Y = {a, ay} então XY = {xay}.

Não-ambiguidade horizontal significa então que, quando é feito o *parsing* de uma *string* de acordo com a produção, nunca há chance de escolher como dividir a *string*\ em *substrings* correspondentes as entidades na produção. (BRABAND, 2008)

G é verticalmente e horizontalmente não-ambígua $\Leftrightarrow$ *G é is não ambígua*

4 DEMONSTRAÇÃO DA NÃO-SOLUCIONABILIDADE

4.1 Definições importantes:

Gramática Livre de contexto: Uma gramática livre de contexto G é definida por:

$$G = (V, T, P, S)$$

onde:

1. V → conjunto de variáveis (símbolos não-terminais);
2. T → conjunto de terminais;
3. P → conjunto de todas as produções de G;
4. S → Símbolo especial, o qual é chamado de "Símbolo Inicial".

Há ainda uma restrição importante a ser feita sobre as produções de G. Em uma gramática livre de contexto essas produções sempre são da forma:

$$V \rightarrow \alpha$$

onde $\alpha \in (V \cup T)^*$. Em suma, restringe-se que as produções de G sejam da seguinte forma: O lado esquerdo das regras tem comprimento unitário e é formado apenas por variáveis. Já o lado direito não possui qualquer restrição.

Gramática Ambígua: Tipo de gramática na qual existe pelo menos uma sentença formada por seus teminais na qual é possível obter-se, no mínimo, 2 árvores de derivação.

Gramática Simplificada: Uma gramática G simplificada é da seguinte forma:

1. Não possui produções vazias (a não ser no símbolo incial, caso necessário);
2. Não possui símbolos inúteis (aqueles não podem ser atingidos a partir da variável inicial);
3. Nem produções simples (ou produções que geram variáveis, que são da forma P → V, onde P e V são variáveis).

Uma referência completa dos procedimentos necessários para se chegar nesta forma estão descritos com detalhes no livro *Linguagens Formais e Autômatos* de Paulo Blauth Menezes.

Forma normal de Chomsky: Uma gramática livre de contexto é dita estar na *Forma Normal de Chomsky (*CNF) se suas produções são do seguinte tipo:

$$V \rightarrow XZ$$

$$V \rightarrow t$$

$$S \rightarrow \varepsilon$$

onde X,Z $\in$ V, t $\in$ T, S é o símbolo inicial e ε representa a palavra vazia. A essência desta representação é limitar o lado direito das produções de forma a ter duas variáveis ou um terminal. Caso a linguagem gere a palavra vazia, acrescenta-se nas produções do símbolo inicial este símbolo.

Sistema de Post: Um sistema de Post é definido sobre um alfabeto A e é um conjunto finito e não-vazio de pares ordenados de palavras sobre A. Exemplificando, um Sistema de Post é da forma:

$$S = \{ (x_1, y_1), ..., (x_n, y_n) \}$$

onde x_i, $y_i \in A$ para i = {1, 2, …, n}. Uma solução para este sistema é uma sequência não-vazia de números naturais $\{i_1, \ldots, i_k\}$ com valores em {1, …, n} tal que:

$$x_1 x_2 ... x_{ik} = y_1 y_2 ... y_{ik}$$

4.2 Formalização:

Dada uma gramática qualquer livre de contexto G, existe um algoritmo capaz de identificar se G é ambígua ou não?

4.3 Demonstração

A demonstração que segue é feita usando como base o Problema da Correspondência de Post. De fato, é mostrado que o problema citado acima pode ser considerado como um sistema de Post, o que é o centro de uma demonstração para a resolução do Problema da Correspondência de Post. Além disso, supõe-se que a gramática livre de contexto G está na Forma Normal de Chomsky.

O Problema da Correspondência de Post é reduzido ao Problema da Ambigüidade. Portanto, este problema preserva propriedades daquele (ou seja, é não-solucionável).

Suponha um Sistema de Post o qual possui cada um de seus elementos – que são pares ordenados – construídos da seguinte forma:

1. Toma-se o lado direito de todas as produções de G;

2. Faz-se todo elemento (x_j, y_j) ser da forma $x_j = \Delta$ e $y_j = \Theta$, onde Δ, Θ correspondem ao lado direito de alguma das produções tomadas em 1 e, além disso $\Delta \neq \Theta$.

Feito isto, e considerando que para obter alguma sentença pertencente à linguagem gerada por G basta aplicarmos uma seqüência válida de passos de derivação,

podemos indagar se esta seqüencia de passos é única. Isto pode ser respondido se soubermos se o Sistema de Post construído acima possui ou não ao menos uma solução.

Isto decorre do fato que a igualdade necessária para a resolução do sistema é, na verdade, uma igualdade entre duas formas de derivação diferentes. Existindo a igualdade – e, como conseqüencia, a solução – sabemos que a gramática G é ambígua.

Veja que a restrição $\Delta \neq \Theta$ é muito importante, pois com ela sabemos que a seqüencia de passos (lado direito das fproduções) foi diferente.

A exigência que G esteja na CNF ajuda a vermos a validade da equivalência. Como o processo para chegar nesta forma requer que a gramática esteja simplificada e as produções possuem o lado direito da forma AB (onde AB são variáveis) ou na forma t (onde t é um terminal), sabemos que a seqüencia obtida para a solução será uma seqüência de passos de derivação valida.

4.4 Observações relevantes

Mais relevante que esta demonstração, é a percepção da relação forte entre o Problema da correspondência de Post e a ambigüidade em uma gramática livre de contexto. No livro *Teoria da Computação: Máquinas Universais e Computabilidade* de Tiarajú Asmuz Divério e Paulo Blauth Menezes no 5º capítulo, o qual é dedicado exclusivamente à solucionabilidade de problemas, mostra-se que um dos problemas base da Ciência da computação, o da Auto aplicação, é Parcialmente Solucionável e Não-solucionável.

A partir disso, é possível usar a técnica de redução para provar as classes nas quais se encaixam outros problemas, como: Parada e Correspondência de Post. Além disso, através da noção de complemento de um problema – bem como alguns teoremas fundamentais de linguagens formais – mostra-se em que classe está contido, por exemplo, o problema da "não Parada" ou seja, da existência de Loops.

Não é o intuito deste trabalho entrar em questões específicas de demonstrações de solucionabilidade de problemas. No entanto, a publicação anteriormente citada pode ser uma boa fonte para o leitor interessado nos tópicos relacionados às demonstrações aqui citadas.

5 CONCLUSÃO

Apesar de ser um problema não solucionável, determinar a ambigüidade de uma linguagem é de veras importante para processamento de certos problemas, não só na área da computação, mas como também na química e na biologia. Apesar de ser comprovado a redução do problema à Correspondência de Post, e com isso provando ser um problema não solucionável, existem diversas abordagem ao problema na tentativa de se chegar a uma aproximação da identificação de possíveis ambigüidades numa linguagem qualquer.

REFERÊNCIAS

BRABRAND, C. ; GIEGERICH, R. ; MØLLER, A. Analyzing Ambiguity of Context-Free Grammars. **Science of Computer Programming**, [S.l.], v.75, n.3, p. 176-191, mar. 2010

SCHMITZ, S. Conservative Ambiguity Detection in Context-Free Grammars. **Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP'07)**, [S.l.], v.4596, p. 692-703, jul. 2007

CUDIA, D. F. The degree hierarchy of undecidable problems of formal grammars **Annual ACM Symposium on Theory of Computing**, Northampton, Massachusetts, United States, p. 10-21, 1970

ČEŠKA, M. **Theoretical Computer Science Text Book**. 2007 – Faculty of Information Technology, BUT, Brno, República Tcheca.