Edson Campos Maia

Uma abordagem da lógica gramatical e da anáfora pronominal

Dissertação de Mestrado apresentada ao Instituto de Filosofia e Ciências Humanas da Universidade Estadual de Campinas, sob a orintentação do Prof. Dr. Carlos A. Lungarzo.

Este exemplar corresponde à redação final da dissertação defendida e aprovada pela Comissão Julgadora em 29/2/1995.

UNICAMP

Fevereiro de 1995

Uma abordagem da lógica gramatical e da anáfora pronominal

Edson Campos Maia

Agradecimentos

Ao Prof. Dr. Carlos A. Lungarzo pelas sugestões e críticas que me auxiliaram na elaboração deste trabalho.

Ao Prof. Dr. Elias H. Alves pelo estímulo, apoio e confiança depositados, que foram tão importantes para levar a cabo esta dissertação.

Ao colega e amigo Dr. Carlos G. González pelo assessoramento na confecção e impressão desta dissertação.

Eu gostaria ainda de agradecer muito especiamente à Profa. Dra. Ariadne M. B. R. Carvalho e ao Prof. Dr. Carlos Franchi, pelas diretrizes que muito me guiaram e por suas críticas que foram fundamentais para o aprimoramento deste trabalho.

Sumário

Este dissertação trata da lingüística computacional do ponto de vista da programação lógica, sendo uma contribuição à utilização da lógica gramatical como uma ferramenta para descrever a sintaxe de subconjuntos de linguagem natural, em particular do fenômeno lingüístico chamado referência pronominal, onde em seu trabalho de tese, CARVALHO [1989], desenvolve desde formalismos gramaticais mais simples, nos quais as restrições têm de ser explicitamente estabelecidas nas regras gramaticais, até formalismos mais complexos, onde o(a) gramático(a) não tem que se preocupar com as restrições, pois tudo é compilado automaticamente pelo compilador Prolog. Em seu trabalho de tese ela utiliza o conceito de categoria mínima de regência de um nódulo A como sendo a minimal S ou SN dominando A.

Porém, com o desenvolvimento da teoria lingüística, o conceito de categoria mínima de regência de um nódulo A passa a ser a primeira categoria maior SN ou S que domina A, um regente de A e um SUJEITO accessível a A.

Nesta dissertação será usado este último conceito de categoria mínima de regência e serão analisadas algumas das possíveis implicações que decorrem deste fato.

Conteúdo

Introdução

Esta dissertação trata da lingüística computacional do ponto de vista da programação lógica, cuja conexão tem aspectos tanto formal quanto utilitário. Na parte formal, explora a restrita linguagem lógica das cláusulas de Horn como um método de expressar e representar as análises lingüísticas. Do lado utilitário, a linguagem de programação lógica Prolog, cujo suporte teórico é o formalismo das cláusulas de Horn, é utilizado como uma ferramenta dos sistemas de processamento de linguagem natural.

No desenvolvimento deste estudo há a confluência de duas vertentes teóricas: a gramática gerativa de Chomsky, e os formalismos computacionais desenvolvidos em Prolog.

Na primeira vertente temos que com as publicações de Noam Chomsky (1957 e anos seguintes), surgiu uma nova corrente lingüística — o gerativismo, também chamado gramática transformacional, ou gramática gerativa, ou gramática gerativo-transformacional — que constitui uma tentativa de formalização (tratamento matemático) dos fatos lingüísticos, ou seja, um tratamento preciso e explícito. Embasado nos sistemas formais da lógica e na teoria matemática da recursão, Chomsky elaborou um sistema de regras e princípios formalizado ou explícito, significando que essas regras e princípios só podem ser operados sob condições específicas, sendo automaticamente aplicados desde que satisfeitas essas condições, tendo como efeito a

caracterização da classe potencialmente infinita das sentenças de uma língua natural, com conseqüente atribuição, a cada sentença, de uma descrição estrutural que represente suas propriedades fonéticas, sintáticas e semânticas.

Na segunda vertente temos que um dos alvos da lógica simbólica é o de capturar a noção de conseqüência lógica como um método formal (mecânico). Se as condições para uma certa classe de problemas podem ser formalizadas dentro de uma lógica adequada como um conjunto de premissas, e se um problema para ser resolvido pode ser determinado como uma sentença na lógica, então a solução pode ser encontrada pela construção de uma prova formal deste problema através das premissas.

No caso lingüístico, as premissas deveriam fornecer restrições na classe de declarações gramaticais, e os problemas a serem resolvidos teriam a forma geral: "existe algum a tal que a é uma análise da declaração gramatical u". Uma prova construtiva desta declaração não mostraria somente que existe uma análise a, mas também encontraria os valores para a.

Um procedimento de prova construtiva que não gere somente provas, mas que também construa valores para as incógnitas no problema pode ser visto como um dispositivo computacional para determinar estas incógnitas. Nesta perspectiva, as premissas podem ser vistas como um programa, o problema como uma invocação do programa tendo certos valores de entrada e tendo como saída incógnitas, e uma prova como uma computação do programa. Esta é a intuição básica que está por detrás da programação lógica, cujo desenvolvimento originou da exploração de um compromisso razoável entre as cláusulas de Horn e sua implementação parcial em Prolog.

O desenvolvimento da programação lógica tem investigado formalismos computacional para expressar análises semântica e sintática de sentenças

de linguagem natural, criando uma linguagem em que as regras de estrutura de frase e de interpretação semântica para um sistema de pergunta-resposta de linguagem natural seriam facilmente expressas.

Uma sentença numa linguagem tal como o Português, o Inglês, etc. é muito mais que uma seqüência arbitrária de palavras, isto é, não podemos juntar qualquer conjunto de palavras e com isso obtermos uma sentença que seja gramatical, ou seja, o resultado deve estar em conformidade com o que é considerado gramatical nesta linguagem. Temos que uma gramática para uma linguagem é um conjunto de regras que especifica que seqüências de palavras são aceitáveis como sentenças dessa linguagem. Ela estabelece como as palavras devem ser agrupadas em frases e qual a ordem das palavras dentro das frases. Dada uma gramática para um linguagem, podemos olhar para qualquer seqüência de palavras e ver se nela encontramos o critério para que esta seja uma sentença aceitável desta linguagem, com isso estabelecemos algo como a estrutura fundamental da sentença.

As regras numa lógica gramatical são apenas instâncias de fórmulas lógicas, por exemplo a regra

$$S \longrightarrow SN, SV$$

pode ser considerada como uma fórmula:

$$\forall x, y, z \quad (SN(x) \wedge SV(y) \wedge \text{Concat}(x, y, z) \supset S(z))$$

onde as variáveis x, y, e z são referências a seqüência de palavras na linguagem, os predicados SN (sintagma nominal) e SV (sintagma verbal) são verdadeiros para quaisquer seqüências que sejam bem formadas SN e SV, respectivamente, o predicado Concat é verdadeiro se seu terceiro argumento é a concatenação de seus primeiros argumentos. Esta fórmula, portanto, es-

tabelece que qualquer seqüência consistindo de um SN seguido por um SV é uma S (sentença).

Esta expressão em lógica de primeira ordem estabelece que qualquer sintagma nominal x, seguido de um sintagma verbal y, pode ser concatenado para formar uma sentença $z = xy$. O termo lógica gramatical é uma referência a tal uso da lógica para formalizar regras gramaticais.

A fórmula acima é um exemplo de uma cláusula de Horn, e a maioria das classes de regras lingüísticas e das classes de restrições podem ser colocadas nesta forma geral, onde estabelece que qualquer objeto satisfazendo certas restrições (propriedades ou relações) também satisfaz outras restrições (propriedades ou relações).

O fato das regras lingüísticas poderem ser representadas desta forma é a base da utilidade das cláusulas de Horn em análise de linguagem, tendo com isso não somente importância teórica mas também prática, pois as regras lingüísticas codificadas como cláusulas de Horn podem ser computadas diretamente em Prolog, fornecendo uma verificação computacional direta e eficiente das gramáticas e das regras de interpretação.

Como utilização destes fatos, em CARVALHO [1989] há o desenvolvimento de uma série de formalismos gramaticais que contribuem para a utilização da lógica gramatical como uma ferramenta para descrever a sintaxe e a semântica de subconjuntos da linguagem natural, em particular do fenômeno lingüístico chamado referência pronominal. Em seu trabalho, as restrições da teoria lingüística foram implementadas em três métodos diferentes:

- manual-explícita, onde as restrições têm de ser explicitamente estabelecidas nas regras gramaticais, ficando a responsabilidade de definir as restrições a cargo do(a) gramático(a);

- manual-implícita, onde as características das restrições não são óbvias, pois são especificadas para que implicitamente implementem as restrições, ficando a responsabilidade de definir as restrições também a cargo do(a) gramático(a); e

- automática-explícita, onde o(a) gramático(a) não tem que se preocupar com as restrições, pois as mesmas são automaticamente implementadas pelo formalismo, porque são explicitamente estabelecidas nas condições inseridas no analisador pelo compilador para este formalismo.

CARVALHO [1989] desenvolve desde formalismos gramaticais mais simples, onde as restrições têm de ser explicitamente estabelecidas nas regras gramaticais, até formalismos mais complexos, onde o(a) gramático(a) não tem que se preocupar com as restrições, pois tudo é compilado automaticamente pelo compilador Prolog.

Esses formalismos gramaticais tencionam apoiar gramáticas que admitem sentenças onde os *SN*s são substantivos próprios e quantificados existencialmente através de dois tipos de formalismos: procedimental[1] e declarativo[2]. Os formalismos procedimentais são similares aos procedimentos semânticos do Prolog, e podem ser implementados usando um processo estratégico descendente, da esquerda para a direita[3]. Os formalismos declarativos têm existência em si próprio, independente de qualquer esquema de implementação

[1] Em Inglês Procedural.
[2] Em Inglês Declarative.
[3] Em Inglês top-down left-to-right.

particular. Neste caso, os formalismos procedimentais esclarecem como os formalismos declarativos são implementados.

Aqui CARVALHO [1989] trata do fenômeno lingüístico chamado referência pronominal, que é encontrado em sentenças do tipo:

(0.1) Rodrigo chamou Larissa e deu um presente para ela.

onde "ela" é uma anáfora que tem como antecedente "Larissa"; ou em

(0.2) Ele estudava com Tatiana porque Leonardo gosta de ajudar.

onde "ele" é uma catáfora que é o antecedente de "Leonardo"; ou em

(0.3) Ele ajudou Marcela porque Felipe gosta dela.

onde "ele" é uma catáfora que é o antecedente de "Felipe", e "dela" é uma anáfora que tem como antecedente "Marcela". Expressões onde ocorrem anáfora e catáfora são denominadas endófora, que é o termo genérico usado para se referir tanto às anáforas quanto às catáforas.

Com isso temos que os termos anáforas e catáforas (endóforas) são itens que, em algum sentido, precisam ter sua referência em algum outro item na sentença, ou seja, eles dizem respeito às relações de dependência, em termos de referência, entre dois itens na sentença.

Em seu trabalho, CARVALHO [1989], utiliza várias estruturas que auxiliam a manipulação dos dados:Mound, Pile, e Bag, e a pilha Stack, que é um tipo de estrutura pertencente à subclasse de listas lineares que por sua vez faz parte das estruturas lineares de dados.

Uma lista é um conjunto ordenado que consiste em um número variável de elementos aos quais inserções e eliminações podem ser feitas em qualquer

posição. Uma lista que mostra o relacionamento de adjacência física é conhecida como lista linear. Se restringirmos a ocorrência de inseções e eliminações a uma extremidade da lista linear, então temos uma pilha.

Portanto temos que, a estrutura de dados que permite representar um conjunto de dados de forma a preservar a relação de ordem linear (ou total) entre eles é uma lista linear, e se as inserções e eliminações são feitas em apenas uma extremidade da lista linear, então temos uma pilha.

Capítulo 1

Gramática gerativa

Segundo a teoria gerativa, os falantes/ouvintes de uma língua, sabem que as seqüências de sua língua se estruturam sintaticamente em hierarquias, ou seja, grupos sucessivamente maiores denominados constituintes. Essas seqüências gramaticais são denominadas sentenças, termo que também se aplica geralmente às frases e às orações : em "Rodrigo falou que Tatiana não virá", o todo (isto é, a frase) é uma sentença, e suas partes (ou seja, as orações) também são sentenças.

Com a sentença "O garoto assustou a menina." vamos exemplificar o conhecimento intuitivo dos falantes/ouvintes de uma língua sobre hierarquização estrutural.

Confrontados com ela, os falantes/ouvintes concordarão em que se divide em duas partes principais: "O garoto", de um lado, e "assustou a menina", de outro. Qualquer outra divisão, diferente desta, é rejeitada intuitivamente pelos falantes/ouvintes como não sendo natural. Subdividirão ainda "O garoto" e "assustou a menina" em suas partes constituintes. A cada uma dessas partes em que se divide uma sentença, dá se o nome de *constituinte*.

Pode-se fazer uso de colchetes, por exemplo, para representar essa hierarquização estrutural, como na figura 1.1.

a. [O garoto assustou a menina.]

b. [[O garoto] [assustou a menina]]

c. [[O garoto] [[assustou] [a menina]]]

d. [[[O] [garoto]] [[assustou] [[a] [menina]]]]

Figura 1.1

Os falantes/ouvintes também possuem intuição sobre categorias ou classes de palavras, sabendo identificar constituintes do mesmo tipo. Entretanto, com isso não se quer dizer que eles possuam, intuitivamente, conhecimento sobre os rótulos dados às diferentes partes do discurso — substantivo, verbo, etc. As intuições em questão dizem respeito exclusivamente às classes em si mesmas, e não aos convencionais nomes das classes.

Com base nessa intuição sobre classes de palavras, pode-se classificar a sentença em análise, com os rótulos que a tradição gramatical já lhes atribuiu, como na figura 1.2,

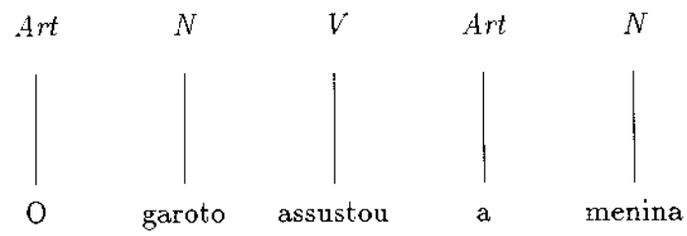

Figura 1.2

na qual, Art = artigo, N = substantivo e pronomes pessoais, e V = verbo.

A associação existente entre o artigo "a" e o nome "menina" ficou evidenciada, como resultado do julgamento dos falantes/ouvintes sobre os agrupamentos naturais formados pelas palavras de uma sentença (os constituintes). Assim, sintagmas são os constituintes que desempenham uma função sintática na frase.

O sintagma nominal (SN) é o agrupamento formado pelo artigo e pelo nome (rotulado dessa forma porque seu núcleo é o nome), podendo também ser formado só pelo nome, como em "Larissa viu Leonardo.". O sintagma verbal (SV) é o agrupamento formado pelo verbo e pelo seu complemento, se existir, (assim rotulado porque seu núcleo é o verbo), como em "Tatiana dormiu.".

Pode-se usar a representação como nas figuras 1.3 e 1.4 para exemplificar os sintagmas nominal e verbal, respectivamente.

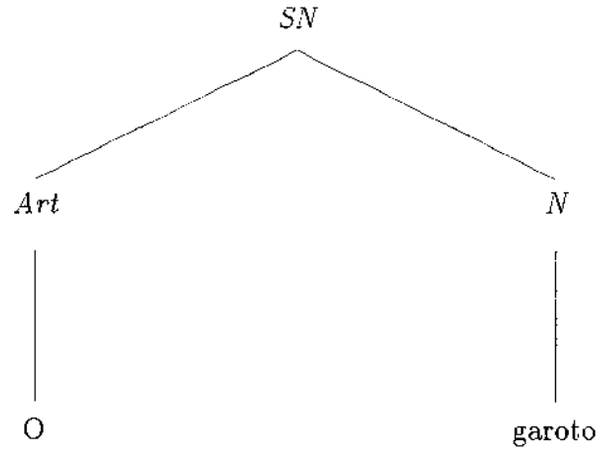

Figura 1.3

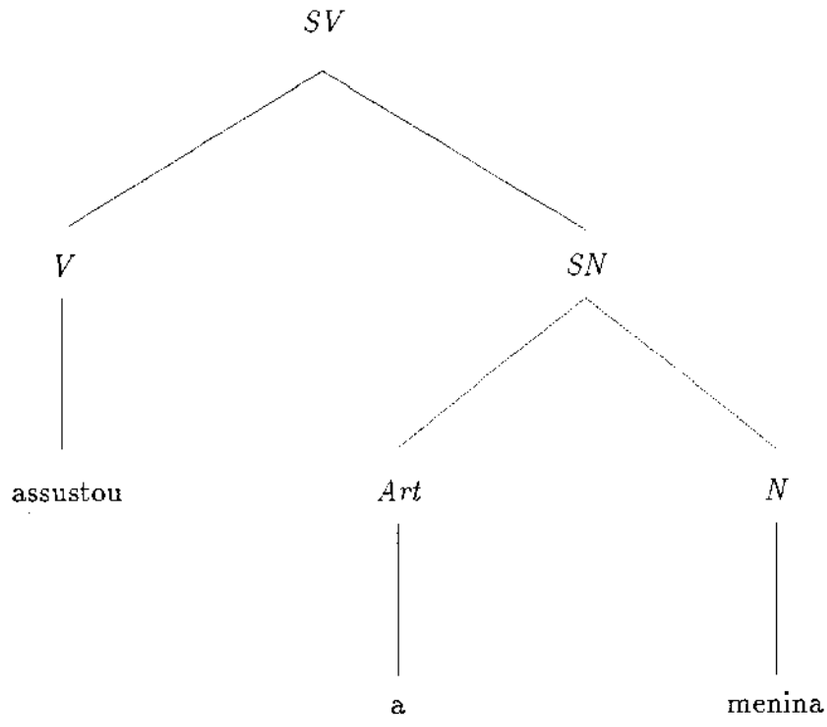

Figura 1.4

A figura 1.5 é a combinação das figuras 1.3 e 1.4, ou seja, é a representação da sentença (S), "O garoto assustou a menina" como um todo, cuja denominação é árvore rotulada; pois é uma representação arbórea com nódulos rotulados (S, SN, SV, N, V, Art).

Uma representação é enraizada se houver um, e só um, símbolo mais alto, pelo qual começa toda a ramificação, onde S é a raiz (considerando-se sempre o símbolo S mais alto, se houver mais de uma oração).

Os nódulos ou expressam categorias de palavras (as partes do discurso como Art, N, V, etc, que são categorias lexicais, pois seus representantes são enumerados no léxico), ou se referem aos agrupamentos em constituintes dessas categorias.

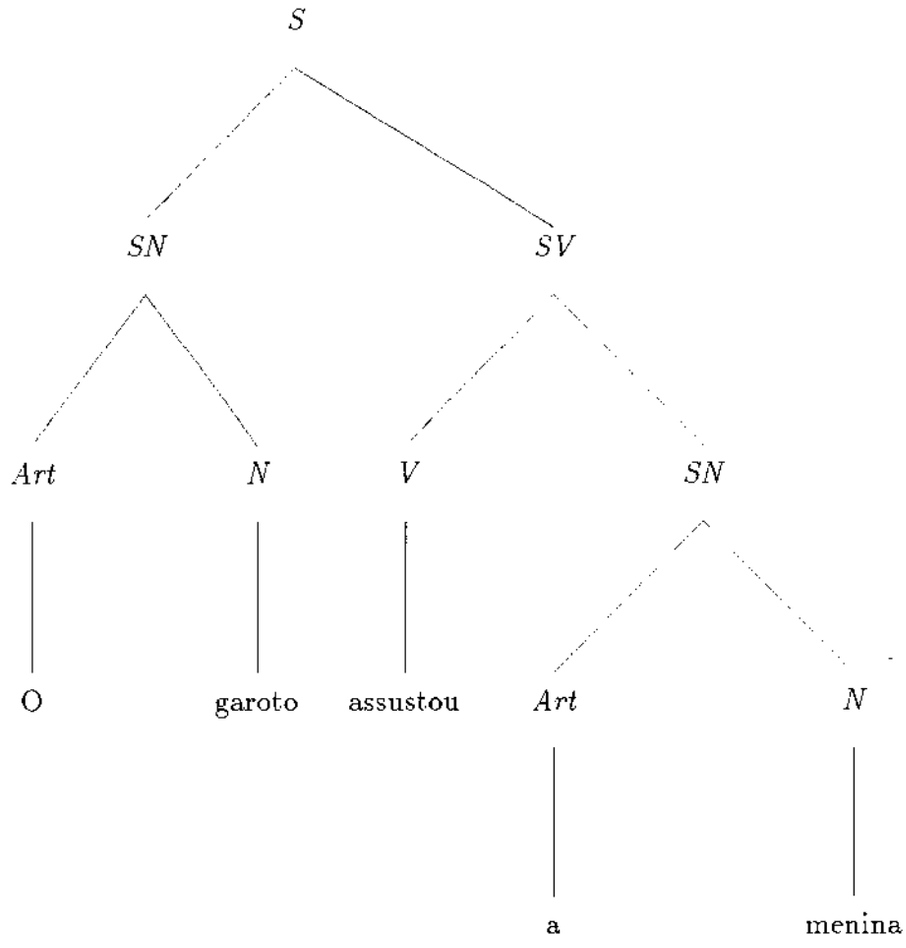

Figura 1.5

As categorias sintagmáticas são os agrupamentos funcionais, pois são combinações sintagmáticas de elementos. As ramificações das árvores são as ligações entre as categorias sintagmáticas e os símbolos em que elas se expandem, ou seja, os nódulos que se ramificam são os nódulos de categorias sintagmáticas, e não os nódulos de categorias lexicais.

Qualquer par de nódulos contidos numa mesma estrutura arbórea estará relacionado por uma das duas relações:

i. **DOMINÂNCIA**: Um nódulo X domina um outro nódulo Y, se X ocorre mais alto na árvore que Y, e é conectado a Y por um conjunto de ramos (linhas). Um nódulo domina imediatamente outro nódulo, se ele é o próximo nódulo mais alto na árvore e está conectado ao outro por um único ramo (linha).

ii. **PRECEDÊNCIA**: Um nódulo X precede um outro nódulo Y, se X ocorre à esquerda do nódulo Y. Um nódulo precede imediatamente outro se ele ocorre imediatamente à esquerda do outro nódulo.

Observe que os nódulos podem estar relacionados ou por dominância ou por precedência, mas não por ambas.

Pode-se definir o constituinte de uma sentença através da definição dominância, como na definição 1.1:

DEFINIÇÃO 1.1

a. *Um conjunto de nódulos forma um* constituinte, *de alguma sentença, se, e somente se, eles estão exaustivamente dominados por um nódulo comum (isto é, se, e somente se, eles são ramos de um único nódulo, e se não houver outros nódulos ramificados deste nódulo único).*

b. X é um constituinte de Y *se, e somente se,* X *é dominado por* Y.

c. X *é um* constituinte imediato *de* Y *se, e somente se,* X *é imediatamente dominado por* Y.

Por exemplo, na figura 1.6 temos que D e E não formam um constituinte, mas D, E e F formam um constituinte, a saber o constituinte C. Os constituintes de A são B, C, D, E e F; os constituintes imediatos de A são B e C.

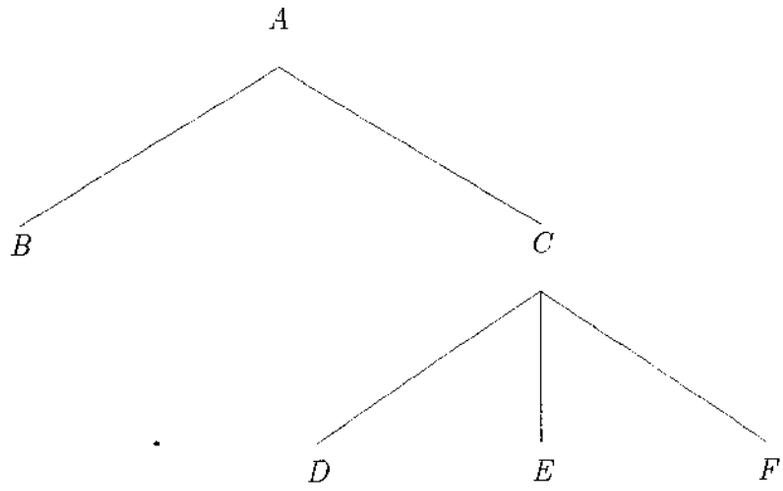

Figura 1.6

Convém salientar que o uso de colchetes ou de árvore rotulada para representar a estrutura sintagmática de uma sentença são totalmente equivalentes, ou seja, não importa se representamos a estrutura sintagmática de uma sentença através de colchetes ou através de árvore rotulada, como na figura 1.5, pois se tivéssemos escolhido o uso de colchetes para representar a estrutura sintagmática da sentença " O garoto assustou a menina ", teríamos de acrescentar na figura 1.1 os rótulos de categorias lexicais, obtendo a figura 1.7 **a** e a este acrescentar os rótulos de categorias sintagmáticas, obtendo com isto a figura 1.7 **b**:

a. $[_{Art}$ O$]$ $[_{N}$ garoto$]$ $[_{V}$ assustou$]$ $[_{Art}$ a$]$ $[_{N}$ menina$]$

b. $[_{S}$ $[_{SN}$ $[_{Art}$ O$]$ $[_{N}$ garoto$]]$ $[_{SV}$ $[_{V}$ assustou$]$ $[_{SN}$ $[_{Art}$ a$]$ $[_{N}$ menina$]]]]$

Figura 1.7

Em suma, temos que a figura 1.5 é uma árvore enraizada rotulada que representa a estrutura sintagmática da sentença "O garoto assustou a menina", e que a figura 1.5 é totalmente equivalente à estrutura sintagmática

da mesma sentença, representada por colchetes na figura 1.7; temos também que de " O garoto " sobe-se na árvore e se chega a um nódulo comum, conclui-se que " O garoto " é um constituinte sintagmático da sentença (um *SN*), mas como a partir de " garoto assustou") não se chega a nenhum ponto de origem comum, conclui-se que "garoto assustou" não é um constituinte da sentença em questão.

Vamos passar agora à noção de constituinte comando (c-comando), versão simplificada, que leva em conta não o que está abaixo do primeiro nódulo S que domina o nódulo em questão, mas sim o que está abaixo da primeira categoria ramificante que domina o nódulo em questão:

DEFINIÇÃO 1.2 *Para* X *e* Y *nódulos numa mesma árvore, diz-se que* X c-comanda Y *se, e somente se*:

a. *o primeiro nódulo ramificante que domina* X *domina* Y;

b. *nem* X *domina* Y *e nem* Y *domina* X.

Ramificante nesta definição quer dizer "que se bifurca", ou seja, é um nódulo com ramos em dois ou mais constituintes imediatos.

Vamos exemplificar a noção de c-comando na figura 1.8. Nessa figura temos:

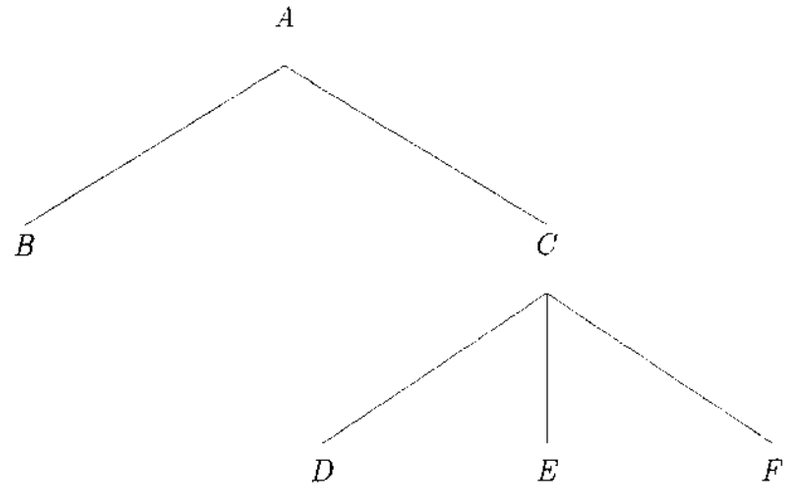

Figura 1.8

A c-comanda nada

B c-comanda C, D, E e F

C c-comanda B

D c-comanda E e F

E c-comanda D e F

F c-comanda D e E

Numa perspectiva derivacional a gramática núcleo, segundo Chomsky [1981], teria a organização do esquema da figura 1.9.

Vamos dar uma idéia geral do que vem a ser o esquema da figura 1.9. As regras do léxico e do componente categorial (regras de base) geram estruturas profundas (estruturas P), que são as entradas do componente transformacional, que as converterá em estruturas superficiais (estruturas S), caracterizadas por conter vestígios.

Os vestígios podem ser resumidos da seguinte forma : qualquer constituinte X^n quando é deslocado, por uma regra de Deslocamento de α, deixa

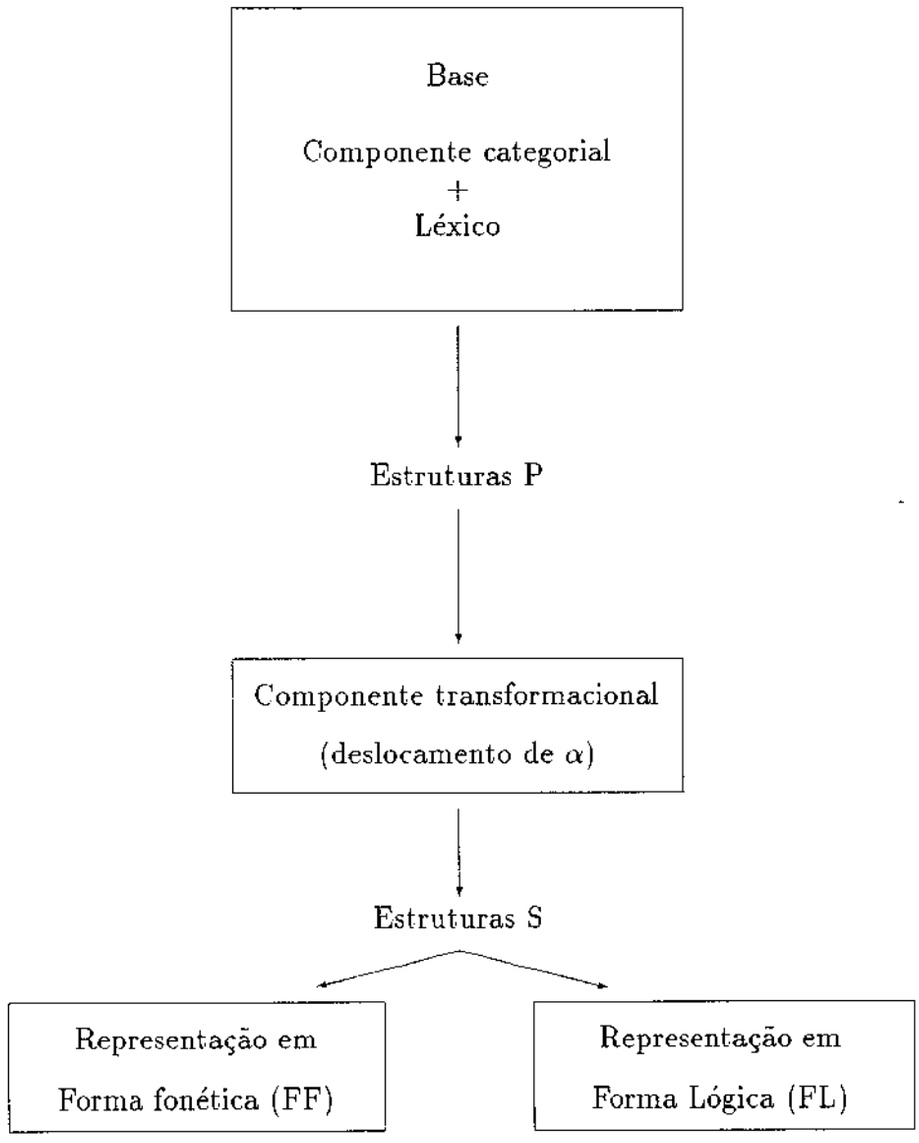
Base
Componente categorial
+
Léxico
Estruturas P
Componente transformacional
(deslocamento de α)
Estruturas S
Representação em
Forma fonética (FF)
Representação em
Forma Lógica (FL)

Figura 1.9

no seu lugar de origem uma cópia idêntica ao elemento deslocado (isto é, todas as característica sintáticas, faltando apenas, a esta cópia, uma matriz fonética), chamada categoria vazia. Esta categoria vazia é conhecida como vestígio e o constituinte X^n movido é dito ser o antecedente do vestígio.

Por exemplo, em (1) temos :

(1) Rodrigo$_i$ foi promovido V_i.

Onde V_i é o vestígio e Rodrigo é o antecedente de V_i.

Em (2) temos :

(2) Quem$_i$ você acha [que [V_i leu o livro]]

Onde V_i é o vestígio e Quem é o antecedente de V_i.

Essas estruturas S resultam em:

a. representações em Forma Fonética (FF) - pela aplicação de filtros e regras fonológicas, e

b. representações em Forma Lógica (FL) - pela aplicação, por exemplo, da regra do deslocamento do quantificador.

Essa concepção gramatical é conhecida como Teoria "T", por ser basicamente uma organização tripartida conforme figura 1.10.

A perspectiva derivacional no esquema da figura 1.9, tem sido substituída por uma perspectiva representacional, em que se propõe que as propriedades das estruturas sintagmáticas geralmente atribuídas à ação das regras da base sejam derivadas dos subsistemas de princípios da gramática.

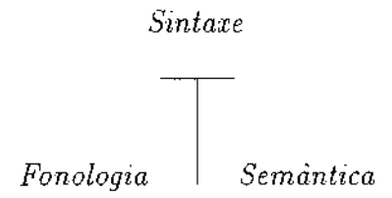

Figura 1.10

Nesse novo enfoque fica o componente categorial das gramáticas das diferentes línguas reduzido à indicação dos valores paramétricos escolhidos pela língua, e limitando-se o componente categorial da Gramática Universal a indicar que as categorias sintagmáticas são projetadas a partir de X ($\overline{X} \longrightarrow \ldots X \ldots$, isto é, X se projeta em $\overline{X}$), sendo X uma variável que representa uma das quatro categorias lexicais básicas (N, V, A e P).

As estruturas P vão ser caracterizadas como:

a. um nível derivado das estruturas S pela abstração de todos os efeitos de Deslocamento de α, e

b. uma representação pura das funções gramaticais relevantes para a atribuição de funções semânticas (tais como as funções temáticas), pois as estruturas P são os resultados de projeções diretas do léxico que discrimina as propriedades abstratas fonológicas, morfológicas, sintáticas e semânticas dos itens da língua.

A classe de estruturas P que uma dada língua admite é determinada pelos princípios dos subsistemas da gramática, mais os valores paramétricos

fixados pela língua para cada um desses subsistemas e as informações contidas nos verbetes lexicais.

Com isso, "ser gerado pela base" significa: ser projetado do léxico, a partir de X, de acordo com os princípios da Gramática Universal e os parâmetros que a língua fixou.

Essa nova perspectiva permite então que se considere que uma estrutura S seja gerada pela base, sendo o Deslocamento de α uma propriedade das estruturas S, e não uma regra que converterá estruturas P em estruturas S.

Do mesmo modo, qualquer outro nível de representação pode ser considerado como "derivado pela base", uma vez que qualquer nível de representação é determinado pela fixação dos parâmetros da Gramática Universal.

Essa expressão Gramática Universal é freqüentemente usada para se referir à faculdade da linguagem como uma estrutura cognitiva inata, que faz parte da herança genética de cada ser humano, se bem que Gramática Universal seja também usada para se referir à teoria proposta por Chomsky para refletir esse estado inicial.

Chomsky pressupõe a existência de parâmetros variáveis no interior da Gramática Universal para tornar sua hipótese (de a Gramática Universal ser inata) compatível com a diversidade entre as línguas existentes. Um desses parâmetros se relaciona com a ordem dos elementos Sujeito - Verbo - Objeto. Fixando os valores desses parâmetros abertos chegamos à Gramática de Núcleo (segundo Chomsky) que é uma gramática particular idealizada, pois faz abstração de heterogeneidade existente no âmbito de cada comunidade lingüística.

Temos então que num processo idealizado de aquisição da linguagem a Gramática Universal seria uma caracterização do estado inicial pré-lingüístico da criança. A experiência serviria para fixar os parâmetros variáveis da Gramática Universal levando à Gramática Núcleo.

O que está representado na mente de um indivíduo, mesmo sob a idealização de uma comunidade lingüística homogênea, é uma Gramática Núcleo acrescida de uma periferia de construções e elementos marcados (periferia marcada), pela heterogeneidade da experiência concreta nas comunidades lingüísticas reais, e pela incorporação de fatores periféricos como empréstimos, resíduos históricos, criações, etc.

O componente transformacional contém a regra de Deslocamento de α (regras de adjunção e substituição).

A regra de adjunção é a regra de Deslocamento de QU. Uma dessas regras é aquela que desloca um elemento Y e o insere imediatamente à direita ou à esquerda de um nódulo A já existente criando, com isso, outra ramificação na árvore. Por exemplo na adjunção chomskiana, se na figura 1.11, adjungimos Y à esquerda de X, obtemos a figura 1.12 **(i)** em p. 25, se a adjunção se faz à direita, obtemos a figura 1.12 **(ii)**; já na adjunção fraterna à esquerda obteríamos a figura 1.13 **(i)** em p. 26 e na adjunção fraterna à direita obteríamos a figura 1.13 **(ii)**.

A regra de substituição é a regra de Deslocamento de *SN* aplicada na derivação de sentenças passivas, e de sentenças com verbo da classe de parecer. Ela é uma regra de substituição porque substitui um elemento não terminal da derivação, ou seja, um símbolo vazio, por um elemento lexical, isto é, um *SN*.

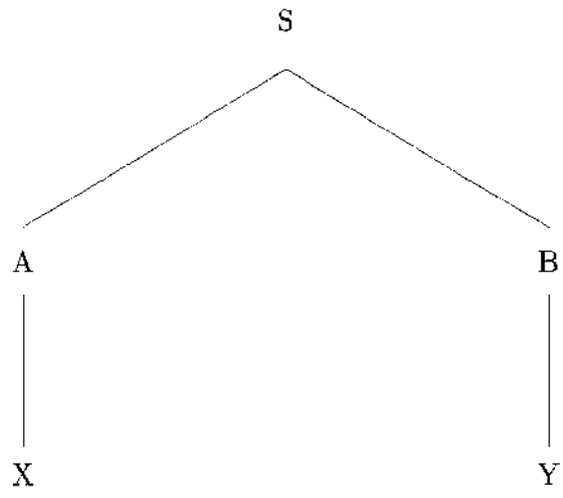

Figura 1.11

Por exemplo, em (3) temos que o *SN* " O carro " originou na posição pós-verbal de *SN* objeto, marcado pelo - , e foi subseqüentemente movido para a posição pré-verbal de *SN* sujeito.

(3) O carro será colocado - na garagem.

Podemos indicar isso esquematicamente como em (4)

(4) $[_S\ [_{SN}\ V]$ será $[_V$ colocado $[_{SN}$ o carro] na garagem.]

Quando um verbo se flexiona, ele concorda com seu sujeito em pessoa e número, portanto os traços de pessoa e número, em Conc (concordância) são traços do sujeito, fato esse expresso pela coindexação entre o sujeito e Conc, que está representado em (5):

(5) $[SN_i\ [Flex\ [\pm Tempo]\ Conc_i]SV]$

A palavra traço é usada tanto para denotar elementos como Humano, Masculino, Singular, quanto para denotar o conjunto desses elementos. Por

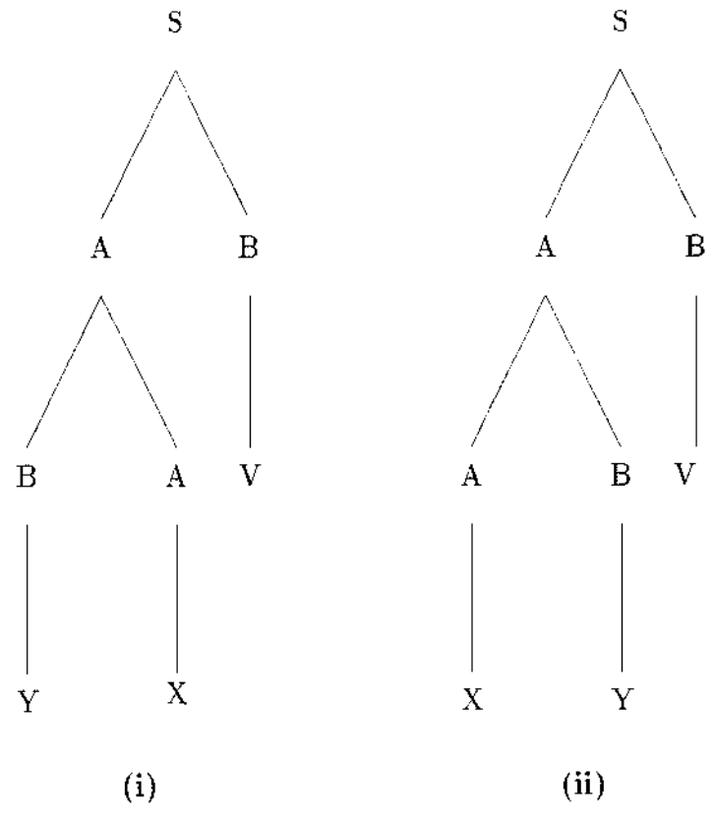

Figura 1.12

exemplo: é Humano é um traço de [+ Humano], e não é Humano é um traço de [− Humano].

Flex é o elemento flexionado abstrato que pode ser ou [+ *Flex*] (temporal) ou [− *Flex*] (infinitivo) e que pode conter o elemento de concordância (*Conc*) e presumivelmente os modais. [+ *Tempo*] ocorre na estrutura de frases com verbo flexionado. [− *Tempo*] ocorre na estrutura de frases com verbo não flexionado.

Só se tem Conc quando se tem o traço [+ *Tempo*], como mostra a figura 1.14 na p. 27.

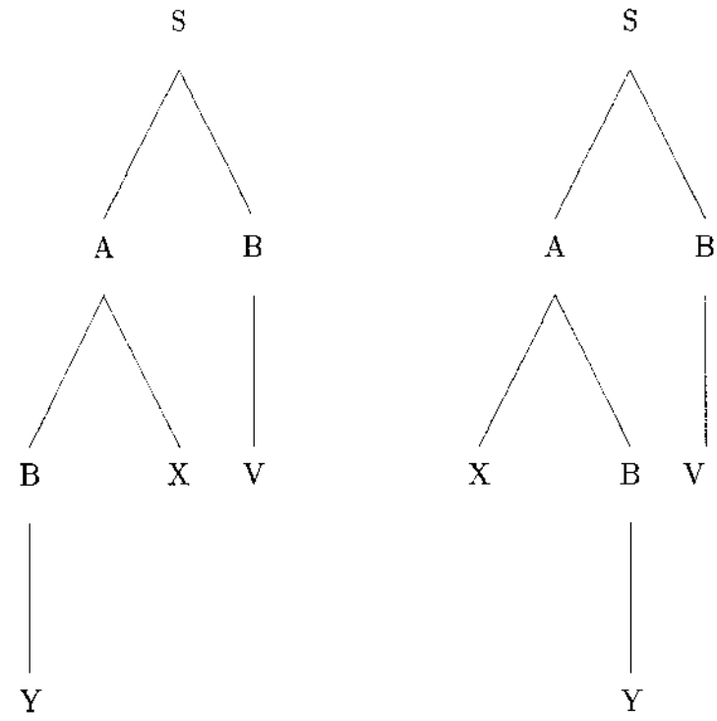

Figura 1.13

Como Conc tem características nominais e pode ser coindexado com sintagmas nominais, temos que Conc é o sujeito de uma oração assim como um *SN* o é.

Tendo *SN* e Conc numa oração, com um sujeito descontínuo, Conc é o sujeito mais proeminente dos dois, e quando Conc não está presente, *SN* é o sujeito mais proeminente

É essa noção de "sujeito mais proeminente" que se traduz com o termo SUJEITO:

SUJEITO = **(i)** Conc nas orações com [+ *Tempo*]

(ii) [*SN*, *S*] ou [*SN*, *SN*] nas orações sem [+ *Tempo*],

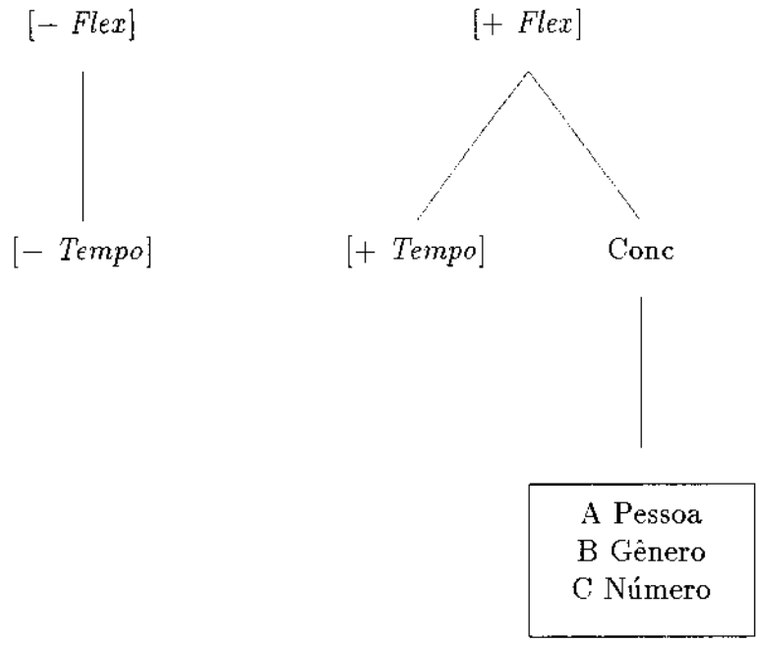

Figura 1.14

onde [*SN*, *S*] lê-se "*SN* imediatamente dominado por *S*", e [*SN*, *SN*] lê-se "*SN* imediatamente dominado por *SN*".

As noções de anáfora, pronominal e expressão referencial (expressão R), são de que elas são expressões que não podem ter referência independentes, elas têm de pegar suas referências de alguma outra expressão, expressão esta que é conhecida como seu antecedente.

As anáforas são itens que, em algum sentido, precisam ter sua referência em algum outro item na sentença, ou seja, ela diz respeito às relações de dependência, em termos de referência, entre dois itens na sentença. Essa referência se faz através da indexação dos elementos que se dá ou pela regra de Deslocamento de α, ou pela regra de coindexação entre sujeito e Conc, ou livremente, e consiste na atribuição de índices referenciais (números ou letras) a todos os sintagmas nominais das seqüências.

Os anafóricos incluem os reflexivos, os vestígios e Pro (sujeito oculto), além do recíproco do Inglês (each other). Os pronominais compreendem os pronomes e Pro. As expressões R são os nomes como Leonardo, garoto, etc. e as variáveis ligadas a um operador (os quantificadores e os conectivos lógicos).

Os indefinidos e as palavras Qu são tratados na Fórmula Lógica (FL) como quantificadores (operadores) que ligam uma variável. Os indefinidos seriam os quantificadores existenciais.

Em toda oração interrogativa com Qu, existe uma outra oração sendo pressuposta como verdadeira. Em cada oração abaixo, **a** pressupõe **b** como verdadeira:

(6) **a.** Quem abriu a janela?

b. Alguém abriu a janela.

(7) **a.** Onde você colocou a mão?

b. Você colocou a mão em algum lugar.

(8) **a.** O que você comeu?

b. Você comeu algo.

As seqüências **b** são pressupostas como verdadeiras quando se enunciam as seqüências em **a**. As seqüências **b** correspondem semanticamente, respectivamente, a: "existe um alguém, tal que alguém abriu a janela"; "existe

um lugar, tal que você colocou a mão nesse lugar" e "existe algo, tal que você comeu algo".

Com isso os morfemas interrogativos se assemelham a quantificadores existenciais. O que se quer com as perguntas em **a**, é obter uma resposta que se identifique precisamente esse "alguém", esse "algum lugar" e esse "algo", e o que se pede em **b** é o valor da variável (alguém, algum lugar e algo).

A teoria $\overline{X}$ (X barra) é uma teoria sobre as categorias gramaticais, onde $\overline{X}$ (equivalentemente, X' ou X^n) é uma notação para referência ao nódulo que domina imediatamente X, onde X é uma das categorias básicas: N = substantivo, V = verbo, A = adjetivo e P = preposição.

Essa convenção notacional permite explicar que entre as categorias lexicais $X(N, V, A, P)$ e as categorias sintagmáticas $\overline{\overline{X}}(SN, SV, SA, SP)$ existem categorias intermediárias ($\overline{X}$), e ao mesmo tempo restringe as regras sintagmáticas possíveis numa língua natural, pois ela exige que o núcleo de uma dada categoria sintagmática tenha a categoria lexical correspondente como elemento obrigatório de reescrita.

A versão Chomskiana da teoria $\overline{X}$ sustenta que as quatro categorias lexicais básicas são o resultado de diferentes combinações de dois traços distintivos lexicais fundamentais: nominal [N] e verbal [V]:

(9) **a.** $N = \begin{bmatrix} +N \\ -V \end{bmatrix}$

b. $V = \begin{bmatrix} -N \\ +V \end{bmatrix}$

c. $A = \begin{bmatrix} +N \\ +V \end{bmatrix}$

d. $P = \begin{bmatrix} -N \\ -V \end{bmatrix}$

Na sintaxe $\overline{X}$, cada categoria-barra é formada por projeções das categorias lexicais:

$$\overline{X} \text{ é uma projeção de } X,$$
$$\overline{\overline{X}} \text{ é uma projeção de } \overline{X},$$
$$\vdots$$

A sintaxe $\overline{X}$ é dada pela seguinte regra:

(10) $X^n \longrightarrow \ldots X^m \ldots$, onde $m = n$ ou $m = n - 1$

Entenda-se X^i como i pertencente ao conjunto dos números naturais e i indicando o número de barras; as (...) na regra como podendo inserir nestas ou modificadores ou complementos (determinantes de X^i), antes ou depois de X^i.

Temos a seguinte equivalência:

(11) $X^0 = X$

$X^1 = X' = \overline{X}$

$X^2 = X'' = \overline{\overline{X}}$

Nessa análise o nível X^n indica que se chegou ao topo da construção do sintagma; o nível X^0 designa o nível lexical em que ainda não se tem construção sintática; e o nível X^m indica que se está em um nível intermediário de construção, entre X^n e X^0.

Uma posição argumental (Posição A) é uma posição que desempenha uma função gramatical (tal como sujeito, objeto).

Diz-se de um elemento que ele está A ligado se o elemento ao qual se liga ocupa uma posição A, e diz-se que um elemento está livre num domínio se ele não está ligado nesse domínio.

A noção de ligação entre nódulos de uma árvore, consiste na noção de c-comando e na noção de indexação:

DEFINIÇÃO 1.3 *Para A e B nódulos numa mesma árvore, A* liga *B se, e somente se:*

a. *A c-comanda B, e*

b. *A e B estão coindexados,*

onde coindexados significa ter o mesmo índice

A noção de correferência está embasada na noção de indexação:

(12) **a.** Se dois *SN*s têm o mesmo índice, então eles são correferênciais.

b. Se dois *SN*s têm índices diferentes, então eles são disjuntos em referência.

Com isso, temos que ser correferencial significa designar a mesma coisa, então considere os conjuntos em (13), onde os elementos do conjunto são os indivíduos e os conjuntos A e B são os *SN*s:

(13) **a.** $A = \{a, b, c\}$ $B = \{d, e, f\}$

b. $A = \{a, b, c\}$ $B = \{a, b, c\}$

c. $A = \{a, b, c\}$ $B = \{c, d, e\}$

Em (13) **a**, os conjuntos A e B são disjuntos, e portanto não designam a mesma coisa. Em (13) **b**, os conjuntos A e B incluem os mesmos indivíduos, com isso A e B são dois SNs correferenciais designando a mesma coisa. Entretanto em (13) **c**, os conjuntos A e B não são correferenciais, pois alguns indivíduos que pertencem ao conjunto A não pertencem ao conjunto B, nem são disjuntos, pois existe ao menos um elemento que pertence a ambos, a saber c.

Se a linguagem faz a opção pela teoria dos conjuntos, então (12) está inadequada: pois esta tem somente um índice (número ou letra), e se dois SNs ou têm o mesmo índice, ou têm índices diferentes, não há meio de descrever as três possibilidades encontradas em (13).

Entretanto, se estivermos trabalhando, exclusivamente, com SNs no singular não haverá problemas, pois para o singular, não correferência e disjuntos em referência são equivalentes. As dificuldades surgem com os SNs no plural. Considere (14):

(14) **a.** Nós$_1$ pensamos que eu$_1$ venceria.

b. Nós$_2$ pensamos que eu$_1$ venceria.

Como (14) é perfeitamente bem formada, as regras sintáticas e semânticas precisam permitir essa sentença. Considere (14) **a**, pela teoria da ligação, esta representação está boa, pois o ligador do pronome está fora de sua categoria de regência. Mas a regra semântica (12) **a** rejeita esta indexação, porque ela contêm dois SNs com o mesmo índice, e portanto precisam ser correferenciais, mas "nós" e "eu", pelos seus significados lexicais, não podem ser correferenciais.

Considere (14) **b**, a teoria da ligação nada diz sobre esse caso, pois nada está ligando coisa alguma. Porém a regra semântica (12) **b** rejeita essa indexação, porque os *SN*s precisam ser disjuntos em referência. Ou seja, se olharmos para o significado de "nós" (que inclui eu e mais alguém, ou alguns) e para o significado de "eu" (que inclui somente eu), vemos que estes dois *SN*s não são disjuntos em referência. O problema é que (14) é perfeitamente gramatical.

Este problema será resolvido postulando as seguintes regras semânticas 1.1 **a** e **b**:

REGRA 1.1

a. *Se dois SNs têm o mesmo índice, então o conjunto denotado por um dos SNs inclui o conjunto denotado pelo outro SN.*

b. *Se dois SNs têm índices diferentes, então eles são disjuntos em referência.*

Adicionamos uma regra morfológica para garantir que os anafóricos concordem em características sintáticas com seus antecedentes:

REGRA 1.2 *Uma anáfora precisa concordar em caraterísticas sintáticas com seus antecedentes.*

O filtro do i-sobre-i ($* i/i$) diz que é agramatical uma seqüência com um constituinte de índice i encaixado em outro constituinte de mesmo índice i, $*[_A \ldots B \ldots]$, onde A e B levam o mesmo índice, ou seja, o filtro simplesmente diz que seja atribuído a A o índice de B, onde A é a anáfora e B é o SUJEITO, o que ocorre então?

Por exemplo, em (15) temos:

(15) **a.** $*$ Ele desenhou $[SN_i$ um retrato de $[SN_i$ si mesmo.]]

b. $*$ $[SN_i$ Os amigos de $[SN_i$ [seus pais.]]

Em (15) **a**, a seqüência só é gramatical com "si mesmo" coindexado com "ele", não podendo ser coindexado com "um retrato de si mesmo". Em (15) **b**, o possessivo "seus" não pode ser coindexado com "os amigos de seus pais".

O filtro do $(* i/i)$ torna-se necessário porque a indexação é livre, sendo a única exceção a coindexação que se dá automaticamente por ocasião do Deslocamento de α (sendo α e seu vestígio coindexados automaticamente).

A teoria da regência e ligação está embasada no que se convencionou chamar de Condição A, Condição B e Condição C, que por sua vez estão embasadas nas noções de ligação e Categoria de Regência.

DEFINIÇÃO 1.4

Condição A: *Uma anáfora precisa estar ligada em sua categoria de regência.*

Condição B: *Um pronominal precisa estar livre em sua categoria de regência.*

Condição C: *Uma expressão R precisa estar livre.*

DEFINIÇÃO 1.5 *A* categoria mínima de regência de *A é a primeira categoria maior SN ou S contendo A, um regente de A e um SUJEITO accessível a A.*

Entenda "contendo" na definição 1.5 como dominando.

A noção de categoria de regência está embasada na noção de dominância, de acessibilidade e de regência.

DEFINIÇÃO 1.6 *A é* acessível a *B se:*

a. *A c-comanda B; e*

b. *a atribuição a B do índice de A não viola o filtro do (* i/i).*

DEFINIÇÃO 1.7 *A rege B se, e somente se:*

a. $A = X^0$ *ou* [+ *Tempo*]; *ou*

b. *A c-comanda B e B não está protegida de A por uma projeção máxima.*

Na definição 1.7, $X^0 = N$, V, A ou P, e as projeções máximas de X são X''.

DEFINIÇÃO 1.8 B está protegido de A por uma projeção máxima, *se esta inclui B mas não inclui A.*

Para facilitar a leitura, resumiremos agora, a teoria da regência e ligação.

Glossário dos conceitos mencionados neste capítulo

I. **DOMINÂNCIA**: Um nódulo X domina um outro nódulo Y, se X ocorre mais alto na árvore que Y, e é conectado a Y por um conjunto de ramos (linhas). Um nódulo domina imediatamente outro nódulo, se ele é o próximo nódulo mais alto na árvore e está conectado ao outro por um único ramo (linha).

II. **PRECEDÊNCIA**: Um nódulo X precede um outro nódulo Y, se X ocorre à esquerda do nódulo Y. Um nódulo precede imediatamente outro se ele ocorre imediatamente à esquerda do outro nódulo.

Para A e B nódulos numa mesma árvore:

III. A c-comanda B se, e somente se:

a. o primeiro nódulo ramificante que domina A domina B; e

b. nem A domina B e nem B domina A.

IV. A e B estão coindexados se eles têm o mesmo índice.

V. A liga B se, e somente se:

a. A c-comanda B; e

b. A e B estão coindexados.

VI. **Condição A**: Uma anáfora precisa estar ligada em sua categoria de regência.

VII. **Condição B**: Um pronominal precisa estar livre em sua categoria de regência.

VIII. **Condição C**: Uma expressão R precisa estar livre.

IX. SUJEITO = **(i)** Conc nas orações com [+ *Tempo*]

(ii) [*SN*, *S*] ou [*SN*, *SN*] nas orações sem [+ *Tempo*],

onde [*SN*, *S*] lê-se "*SN* imediatamente dominado por *S*", e [*SN*, *SN*] lê-se "*SN* imediatamente dominado por *SN*".

X. *A rege B* se, e somente se:

a. $A = X^0$ ou [+ *Tempo*]; ou

b. *A* c-comanda *B* e *B* não está protegida de *A* por uma projeção máxima.

Onde $X^0 = N,\ V,\ A$ ou P, e as projeções máximas de X são X''.

XI. *B está protegido de A por uma projeção máxima*, se esta inclui *B* mas não inclui *A*.

XII. A categoria mínima de regência de A é a primeira categoria maior *SN* ou S dominando A, um regente de A e um SUJEITO accessível a A.

XIII. A sentença raiz é uma categoria de regência para um elemento regido.

XIV. A é accessível a B se:

a. A c-comanda B; e

b. a atribuição a B do índice de A não viola o filtro $(* \; i/i)$.

XV. O filtro do i-sobre-i $(* \; i/i)$ diz que é agramatical uma seqüência com um constituinte de índice i encaixado em outro constituinte de mesmo índice i, $*[_A \ldots B \ldots]$, onde A e B levam o mesmo índice.

XVI. Se dois *SN*s têm o mesmo índice, então o conjunto denotado por um dos *SN*s inclui o conjunto denotado pelo outro *SN*.

XVII. Se dois *SN*s têm índices diferentes, então eles são disjuntos em referência.

XVIII. Uma anáfora precisa concordar em características sintáticas com seus antecedentes.

Capítulo 2

Lógica gramatical e anáfora pronominal

Em sua tese de doutorado (CARVALHO [1989]) apresenta uma série de formalismos gramaticais que contribuem para a utilização da lógica gramatical como uma ferramenta para descrever a sintaxe e a semântica de subconjuntos da linguagem natural, em particular do fenômeno lingüístico chamado referência pronominal. Neste trabalho, as restrições da teoria lingüística foram implementadas em três métodos diferentes:

a. manual-explícita, onde as restrições têm de ser explicitamente estabelecidas nas regras gramaticais, ficando a responsabilidade de definir as restrições a cargo do(a) gramático(a);

b. manual-implícita, onde as características das restrições não são óbvias, pois são especificadas para que implicitamente implementem as restrições, ficando a responsabilidade de definir as restrições também a cargo do(a) gramático(a); e

c. automática-explícita, onde o(a) gramático(a) não tem que se preocupar com as restrições, pois as mesmas são automaticamente imple-

mentadas pelo formalismo, porque são explicitamente estabelecidas nas condições inseridas no analisador pelo compilador para este formalismo.

CARVALHO desenvolve desde formalismos gramaticais mais simples, onde as restrições têm de ser explicitamente estabelecidas nas regras gramaticais, até formalismos mais complexos, onde o(a) gramático(a) não tem que se preocupar com as restrições, pois tudo é compilado automaticamente pelo compilador Prolog.

Antes de apresentarmos esses formalismos, vamos definir alguns termos para que possamos entendê-los.

A lógica gramatical foi idealizada no contexto de sistemas de dedução computacional, como um método para expressar gramáticas em lógica. A idéia é separar a especificação lógica do problema a ser resolvido do procedimento de prova que interpreta esta especificação lógica para resolver o problema.

Como vimos no capítulo anterior, uma sentença numa linguagem tal como o Português, o Inglês, etc. é muito mais que uma seqüência arbitrária de palavras, isto é, não podemos juntar qualquer conjunto de palavras e com isso obter uma sentença que seja gramatical, ou seja, o resultado deve estar em conformidade com o que é considerado gramatical nesta linguagem. Com isso uma gramática para uma linguagem é um conjunto de regras que especifica que seqüências de palavras são aceitáveis como sentenças dessa linguagem. Ela estabelece como as palavras devem ser agrupadas em frases e qual a ordem das palavras dentro das frases. Dada uma gramática para uma linguagem, podemos olhar para qualquer seqüência de palavras e ver se nela encontramos o critério para que esta seja uma sentença aceitável desta

linguagem, com isso estabelecemos algo como a estrutura fundamental da sentença.

As regras numa lógica gramatical são apenas instâncias de fórmulas lógicas, por exemplo a regra

$$S \longrightarrow SN, SV$$

pode ser considerada como uma fórmula:

$$\forall x, y, z \qquad (SN(x) \ \wedge \ SV(y) \ \wedge \ \text{Concat}(x, y, z) \ \supset \ S(z))$$

onde as variáveis x, y, e z são referências a seqüência de palavras na linguagem, os predicados SN e SV são verdadeiros para quaisquer seqüências que sejam bem formados SN e SV, respectivamente, o predicado Concat é verdadeiro se seu terceiro argumento é a concatenação de seus primeiros argumentos. Esta fórmula, portanto, estabelece que qualquer seqüência consistindo de um SN seguido por um SV é uma S.

COLMERAUER [1978], usando os conceitos de programação lógica desenvolvidos por KOWALSKY [1974], idealizou um método particular para expressar regras da gramática livre de contexto como declarações lógicas de um tipo restrito conhecidas como cláusulas de Horn[1]. Uma cláusula de Horn é composta de uma cabeça e um corpo. A cabeça consiste de uma única meta (goal), ou é vazia; o corpo consiste de uma seqüência de zero ou mais metas. Se nem a cabeça e nem o corpo de uma cláusula são vazios, nós a chamamos de uma cláusula não-unitária[2], e a escrevemos na forma:

$$P :- Q, R, S.$$

onde P é a meta da cabeça e Q, R e S são as metas que compõem o corpo. Podemos ler tal cláusula como "P é verdadeira se Q, R e S são verdadeiras".

[1]Em inglês: *definite clauses.*
[2]Em inglês: *non-unit clause.*

Se o corpo de uma cláusula é vazio, nós a chamamos de cláusula unitária[3], e a escrevemos na forma:

$$P.$$

onde P é a meta da cabeça. Tal cláusula é lida como "P é verdadeira". Se a cabeça da cláusula é vazia, nós a chamamos de uma questão e a escrevemos na forma:

$$:- P, Q.$$

onde P e Q são as metas do corpo. Tal questão é lida como "P e Q são verdadeiras?".

A semântica declarativa das cláusulas de Horn nos informa que metas podem ser consideradas verdadeiras, de acordo com um dado programa, e são definidas recursivamente como: uma meta é verdadeira se ela é a cabeça de alguma instância de cláusula e cada uma das metas no corpo da cláusula instanciada (se existirem) forem verdadeiras, onde uma instância de uma cláusula é obtida ao satisfazer para cada zero, ou mais de suas variáveis, um novo termo, para todas as ocorrências da variável. A semântica declarativa não faz referência à seqüência de metas dentro do corpo de uma cláusula, nem às seqüências de cláusulas dentro de um programa.

A lógica gramatical tem existência em si própria, independente de qualquer esquema de implementação particular. Existem várias formas de implementar lógica gramatical, porém a maioria das implementações são feitas em Prolog e, portanto, herdam a semântica dos procedimentos que Prolog fornece às cláusulas de Horn, isto é, definem exatamente como o sistema Prolog executará uma meta e o seqüenciamento de informações é o meio pelo qual o programador Prolog ordena o sistema para executar seu programa.

[3]Em inglês: *unit clause*.

Os vários formalismos que têm sido publicados na literatura, tais como: Definite Clause Grammars (DCGs) (PEREIRA and WARREN [1980]), Restriction Grammars (RGs) (HIRSCHMAN and PUDER [1986]), Extraposition Grammars (XGs) (PEREIRA [1981]), Definite Clause Translation Grammars (DCTGs) (ABRAMSON [1984], Restricted Logic Grammars (RLGs) (STABLER [1986-7]), Modifier Structure Grammars (MSGs) (DAHL and MCCORD [1983]), Gapping Grammars (GGs) (DAHL [1984], DAHL and ABRAMSON [1984]), Constrained Discontinuous Grammars (DGs) (DAHL and SAINT-DIZIER [1986]) e Prolog with Atributes Grammars (Pr Att bs) (JOHNSON and KLEIN [1986]) são apresentados em CARVALHO, Capítulo 2.

Ela desenvolve uma seqüência de formalismos gramaticais que tencionam apoiar gramáticas que admitem sentenças onde os *SN*s são substantivos próprios e quantificados existencialmente através de dois tipos de formalismos: procedimental e declarativo. Os formalismos procedimentais são similares aos procedimentos semânticos do Prolog, e podem somente ser implementados usando um processo estratégico descendente, da esquerda para a direita[4]. Os formalismos declarativos têm existência em si próprios, independentes de qualquer esquema de implementação particular.

Apresentaremos esses formalismos na ordem em que se apresentam em CARVALHO, Capítulo 4, contudo não mostraremos como ela desenvolve a gramática no formalismo em virtude de não ser o objetivo dos capítulos 2 e 3 deste trabalho.

[4]Em ingês top-down left-to-right.

2.1 Virtual Mound Grammar *VMG*

2.1.1 O formalismo

VMG é um formalismo procedimental que usa um mound como sua estrutura de armazenagem virtual.

Um mound é uma estrutura com as seguintes características:

a. Os itens novos são adicionados no topo do mound;

b. Os itens nunca são removidos do mound;

c. Sempre que for necessário determinar se um item tendo certas características está presente no mound; e, se ele está presente, resgata todas suas características, a busca começa no topo do mound: isto é, os itens mais novos no mound são tratados como sendo mais relevantes que os outros;

d. Quando um item com as características requisitadas é encontrado, ele é movido para o topo do mound.

As características desta estrutura foram motivadas para serem usadas a: armazenar antecedentes que podem ser referenciados por pronomes anafóricos (isto é, não catafóricos); o armazenamento de um item novo no topo do mound é pelo fato de que o último item a ser referenciado é aquele que está em foco e, conseqüentemente, o mais provável a ser referenciado novamente; e não remover itens do mound é pelo fato de que os itens podem ser referidos mais de uma vez.

As regras *VMG* têm a forma:

$$\alpha \longrightarrow \beta_1 \ldots, \beta_n$$

com α variando sobre V_N e β_i variando sobre

$$(V_N \cup V_T \cup \{\langle \text{read\_mound}(v)\rangle, \langle \text{write\_mound(v)}\rangle\} \cup V_p),$$

com v variando sobre V_N, onde V_N é a classe de símbolos não-terminais da linguagem, V_T é a classe de símbolos terminais da linguagem, e V_P é o conjunto de chamadas Prolog.

VMG usa os operadores "write_mound" e "read_mound" para armazenar e resgatar tokens no mound.

Considere a seguinte regra *VMG*:

$$a \longrightarrow b, \langle \text{write\_mound}(c)\rangle$$

Esta regra pode ser lida como:

Temos um símbolo não-terminal de classe a, desde que encontremos um símbolo não-terminal de classe b, mas precisamos armazenar no topo do mound um símbolo virtual não-terminal de classe c, que pode ser acessado mais tarde, sempre que for necessário.

Considere a seguinte *VMG*:

$$d \longrightarrow e, \langle \text{read\_mound}(f)\rangle.$$

Esta regra pode ser lida como:

Temos um símbolo não-terminal de classe d, desde que encontremos um símbolo não-terminal de classe c, e desde que estejamos aptos a ler no mound um símbolo virtual não-terminal de classe f.

```
sentence          -->  clause.                                   (1)
sentence          -->  clause, s-conj, sentence                  (2)
clause            -->  nounph(nom), verbph.                      (3)
nounph(Case)      -->  proper_noun(P,G),
                       ⟨write_mound(ref(X,X=P,Gen)) ⟩.           (4)
proper_noun(P,Gen) --> [P],{proper( P,Gen)}.                     (5)
nounph(Case)      -->  determiner, noun(X,P, Gen),
                       ⟨write_mound(ref(X,P,Gen)) ⟩.             (6)
determiner        -->  [D], {det(D)}.                            (7)
noun(X, P, Gen)   -->  [Noun], {n(Noun,X,P,Gen)}.                (8)
nounph(Case)      -->  pronoun(Case, Gen),
                       ⟨read_mound(ref(_,_,Gen))⟩.               (9)
pronoun(Case, Gen) --> [Pr],{prop(Pr,Case,Gen)}.                 (10)
verbph            -->  verb, nounph(acc).                        (11)
verb              -->  [V], {v(V)}.                              (12)
s-conj            -->  [C], {s-conj(C)}.                         (13)

/* Lexicon */
s-conj(and).
det(the).
n(pear,X,pear(X),neuter).
proper(fred,masc).
proper(mary,fem).
v(saw).
v(ate).
v(kissed).
pro(he,nom,masc).
pro(she,nom,fem).
pro(it,nom,neuter).
pro(it,acc,neuter).
pro(her,acc,fem).
```

Quadro 2.1

2.1.2 Uma gramática no formalismo

Considere como um *VMG*, como no quadro 2.1, pode ser usado para analisar a seguinte sentença:

(I) Fred saw the pear and he ate it.

Quando o substantivo próprio "Fred" é encontrado, conforme a regra (4), um token que representa "Fred" é escrito no mound, especificando o gênero apropriado. Quando o *SN* "the pear" é encontrado, conforme a regra (6), outro token que representa "pear" é escrito no mound.

Quando o pronome "he" é encontrado, conforme a regra (9), um referente masculino é lido no mound. Portanto, o pronome "he" é dito referir ao token que representa "Fred". Quando o pronome "it" é encontrado, conforme a regra (9), um referente neutro é lido do mound e o pronome "it" é referido ao token que representa "pear". Com isso, a análise obteve sucesso e a sentença **(I)** é considerada gramatical.

2.2 O formalismo $AG1$ (Anaphora Grammar)

2.2.1 O formalismo

$AG1$ é um formalismo gramatical declarativo. As regras $AG1$ têm uma das duas formas seguintes:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

com α e δ variando sobre V_N, β_i variando sobre $(V_T \cup V_N \cup V_p)$ e γ_i variando sobre $(V_T \cup V_N \cup \{\Theta \lll \phi\} \cup V_p)$ e onde Θ varia sobre V_N e ϕ varia sobre $(V_T \cup V_N)$. A regra $AG1$:

$$a \,//\, b \longrightarrow c.$$

pode ser lida como:

Um a que introduz uma marca de referência b pode ser reescrito como um c.

A seguinte regra $AG1$:

$$a \longrightarrow b \lll c.$$

pode ser lida como:

Um a pode ser reescrito como uma anáfora c que correfere com alguma categoria à sua esquerda que introduz uma marca de referência b.

2.2.2 Uma gramática no formalismo

Considere como a sentença **(I)** é analisada usando o $AG1$ do quadro 2.2.

Quando o substantivo próprio "Fred" é encontrado, conforme a regra (4), um token que representa "Fred" é escrito no mound, especificando o gênero apropriado. Quando o SN "the pear" é encontrado, conforme a regra (6), um token que representa "pear" é escrito no mound com o gênero apropriado. Quando o pronome "he" é encontrado, conforme a regra (9), um referente cujo gênero é o mesmo que o do pronome deve ser lido no mound. Portanto, o token que representa "Fred" é lido e fica sendo o antecedente

```
sentence                                --> clause.                                  (1)
sentence                                --> clause, s-conj, sentence.                (2)
clause                                  --> nounph(nom), verbph.                     (3)
nounph(Case)// ref(X,X=Name,Gen)        --> proper_noun(Name,Gen).                   (4)
proper_noun(Name,Gen)                   --> [Name], {proper(Name, Gen)}.             (5)
nounph(Case)// ref(X,P,Gen)             --> determiner,noun(X,P,Gen)                 (6)
determiner                              --> [D], {det(D)}.                           (7)
noun(X,P,Gen)                           --> [Noun], {n(Noun,X,P,Gen)}.               (8)
nounph(Case)                            --> ref(_,_,Gen)<<<pronoun(Case,Gen).        (9)
pronoun(Case,Gen)                       --> [Pr], {pro(Pr,Case,Gen)}.                (10)
verbph                                  --> verb,nounph(acc).                        (11)
verb                                    --> [V], {v(V)}.                             (12)
s-conj                                  --> [C],{s-conj(C)}.                         (13)

/* Lexicon */
s-conj(and).
det(the).
n(pear,X,pear(X),neuter).
proper(fred,masc).
proper(mary,fem).
v(saw).
v(ate).
v(kissed).
v(killed).
pro(him,acc,masc).
pro(he,nom,masc).
pro(she,nom,fem).
pro(it,nom,neuter).
pro(it,acc,neuter).
pro(her,acc,fem).
```

Quadro 2.2

do pronome "he". Quando o pronome "it" é encontrado, conforme a regra (9), um referente cujo gênero é o mesmo que o do pronome deve ser lido no mound. Com isso, o token representando "pear" é lido e estabelecido como o referente do pronome "it". Como não há mais palavras na string, a análise obteve sucesso e a sentença (I) é considerada gramatical.

Suponha como uma sentença "agramatical"[5]:

(II) Fred kissed Mary and she killed her.

seria analisada usando $AG1$ do quadro 2.2.

Quando os substantivos "Fred" e "Mary" são encontrados, conforme a regra (4), tokens são escritos no mound com seus gêneros. Quando o pronome "she" é encontrado, conforme a regra (9), um referente cujo gênero é o mesmo que o do pronome deve ser lido do mound. Portanto, o token representando "Mary" é lido e estabelecido como o antecedente do pronome "she". Quando o pronome "her" é encontrado, conforme a regra (9), um referente cujo gênero é o mesmo que o do pronome deve ser lido do mound. Com isso, o token representando "Mary" é mais uma vez lido do mound e erroneamente estabelecido como o antecedente do pronome "her".

Com isso, vimos que uma restrição em gênero não é suficiente para evitar a análise de sentenças agramaticais como (II).

O quadro 2.2 mostra uma versão melhorada do $AG1$ do quadro 2.3 com a seguinte restrição em reflexivização implementada explicitamente:

Um objeto pode correferir com o sujeito da mesma sentença se, e somente se, o objeto é um pronome reflexivo.

[5]O termo "agramatical" é usado em CARVALHO quando um referente para um pronome não pode ser encontrado dentro da própria sentença.

```
sentence                                  -->  clause.                                          (1)
sentence                                  -->  clause,s-conj,sentence                           (2)
clause                                    -->  nounph(X,nom), verbph(Y,Case),
                                               {satisfied_reflex_constraint(X,Y,Case)}.       (3)
nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(P,Gen),
                                               {Case\==reflex}.                               (4)
proper_noun(P,Gen)                        -->  [P],{proper(P,Gen)}.                             (5)
nounph(X,Case)                            -->  ref(X,_,Gen)<<<pronoun(Case,Gen).                (6)
pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                        (7)
verbph(Y,acc)                             -->  verb,nounph(Y,acc).                              (8)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                           (9)
verb                                      -->  [V], {v(V)}.                                     (10)
s-conj                                    -->  [C],{s-conj(C)}.                                 (11)
/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                       (12)
satisfied_reflex_constraint(X,Y,Case)     :-   X== Y.                                           (13)

/* Lexicon */
s-conj(and).
s-conj(but).
s-conj(so).
proper(fred,masc).
proper(tom,masc).
proper(mary,fem).
v(saw).
v(kissed).
v(loved).
v(killed).
pro(he,nom,masc).
pro(him,acc,masc).
pro(she,nom,fem).
pro(her,acc,fem).
pro(herself,reflex,fem).
```

Quadro 2.3

Na gramática do quadro 2.3, durante a análise da sentença **(II)**, quando o substantivo "Fred" é encontrado, conforme a regra (4), um token é escrito no mound com seu gênero e uma variável X'. Esta variável representa "Fred"; ela é passada como um argumento da primeira sub-meta na regra (3) (nounph), e é usada quando for checada a reflexividade. Quando o substantivo "Mary" é encontrado, conforme a regra (4), outro token é escrito no mound com seu gênero e uma variável Y'. Esta variável representa "Mary"; ela é passada como um argumento da segunda sub-meta na regra (3) (verbph), e é também usada quando for checada a reflexividade.

As regras verbph passam, na variável "case", a descrição do tipo de *SN* embutido nelas. Esta variável contém o valor "reflex" se o *SN* é um pronome reflexivo; caso contrário, contém o valor "acc". Após as duas primeiras sub-metas na regra (3) serem satisfeitas, a terceira sub-meta, que é um teste Prolog, deve ser satisfeita. Este teste casa (bate) com a regra Prolog (12), onde estabelece que X e Y devem ser diferentes quando o caso não é "reflex", como este é o caso, a meta é satisfeita.

A análise continua normalmente. A primeira sub-meta na regra (2) foi satisfeita. Agora a conjunção "and" é consumida e as regras (1) e (3) são novamente chamadas para satisfazer a terceira sub-meta da regra (2) (sentence). O pronome "she" é o próximo elemento. Conforme a regra (6), devemos ler no mound um referente cujo gênero é o mesmo que o do pronome. O token representando "Mary" é lido e estabelecido como o referente do pronome "she". A variável X'' (a encarnação de X produzida na segunda chamada da regra (3)), é então compartilhada com a variável Y' que foi associada com o substantivo "Mary".

Quando o pronome "her" é encontrado, conforme a regra (6), um referente cujo gênero é o mesmo que o do pronome deve ser lido do mound. O token representando "Mary" é novamente lido do mound e a variável Y'' (a encarnação de Y produzida na segunda chamada da regra (3)), é também compartilhada com a variável Y', que foi associada com o substantivo "Mary" e que é, portanto, igual à variável X''. Agora, a terceira sub-meta da regra (3) deve ser satisfeita. A regra Prolog (12) é chamada e falha, pois X'' e Y'' são iguais. A regra (13) também falha porque, embora X'' e Y'' sejam iguais, o pronome não é "reflex". Com isso, a análise falha e a sentença **(II)** é considerada agramatical.

Note que se a sentença:

(III) Fred kissed Mary and she killed herself.

fosse analisada, seria bem sucedida, pois quando o pronome reflexivo fosse encontrado "reflex" seria atribuído à variável "case" e, conseqüentemente, a regra Prolog (13) teria êxito e a sentença **(II)** seria considerada gramatical.

Sentenças como:

(IV) Fred loved Mary but she loved Tom so Fred killed him

teriam êxito e sentenças como:

(V) Fred killed Fred

não obteriam êxito, pois transgridem a restrição de reflexividade.

2.3 Virtual Bag Grammar (*VBG*)

2.3.1 O Formalismo

VBG é um formalismo gramatical procedimental que usa uma bag como sua estrutura de armazenagem virtual. Uma bag é uma estrutura com as seguintes características de armazenagem e resgate:

a. Os itens novos são adicionados no topo da bag;

b. Quando procuramos por um item, a busca começa no topo da bag;

c. Os itens podem ser removidos da bag somente uma vez;

d. A bag precisa estar vazia ao fim da análise.

As regras *VBG* são da forma:

$$\alpha \longrightarrow \beta_1, \ldots, \beta_n$$

com α variando sobre V_N e β_i variando sobre ($V_N \cup V_T \cup \{\langle$remove_bag$\rangle$, $\langle$write_bag$(v)\rangle, \langle$empty_bag$\rangle\} \cup V_p$) e com v variando sobre V_N.

VBG usa os operadores "write_bag" e "remove_bag" para armazenar e resgatar tokens, respectivamente, na bag. O operador "empty_bag" é um predicado que retorna verdadeiro se a bag estiver vazia, isto é, sem tokens.

Considere a seguinte regra *VBG*:

$$a \longrightarrow c, \langle \text{write\_bag}(b) \rangle.$$

Esta regra pode ser lida como:

Temos um símbolo não-terminal de classe a, desde que encontremos um símbolo não-terminal de classe c, mas devemos armazenar no topo da

bag um símbolo não-terminal virtual de classe b, que precisa ser consumido antes do fim da análise.

Considere a seguinte regra *VBG*:

$$d \longrightarrow e, \langle \text{remove\_bag}(f) \rangle.$$

Esta regra pode ser lida como:

Temos um símbolo não-terminal de classe d, desde que encontremos um símbolo não-terminal de classe e, e desde que estejamos aptos a consumir da bag um símbolo não-terminal virtual de classe f.

Uma regra *VBG* da forma:

$$g \longrightarrow \langle \text{empty\_bag} \rangle, h$$

pode ser lida como:

Temos um símbolo não-terminal de classe g, desde que a bag esteja vazia ao findar a procura por um símbolo não-terminal de classe h.

*VBG*s podem manipular catáforas e o formalismo *CG*1 a seguir será implementado em *VBG* para fazê-lo.

2.4 O Formalismo *CG*1 (Cataphora Grammar)

2.4.1 O Formalismo

O formalismo *CG*1 é um formalismo gramatical declarativo, as regras de *CG*1 têm uma das duas formas:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

Com α e δ variando sobre V_N, β_i variando sobre $(V_T \cup V_N \cup V_p)$ e γ_i variando sobre $(V_T \cup V_N \cup \{\Theta \ggg \emptyset, \#\Theta\} \cup V_p)$, onde $\emptyset$ varia sobre V_N e Θ sobre $(V_T \cup V_N)$.

A regra $CG1$:

$$a \,//\, b \longrightarrow C.$$

pode ser lida como:

Um a que introduz uma marca de referência b pode ser rescrito como c.

A regra $CG1$:

$$a \longrightarrow c \ggg b.$$

pode ser lida como:

Um a pode ser reescrito como uma catáfora c que correfere com alguma categoria à sua direita que introduz uma marca de referência b.

Uma regra $CG1$ da forma:

$$a \longrightarrow \#b.$$

pode ser lida como:

Um a pode ser reescrito como um b, desde que todas catáforas à esquerda de a correfiram com marcas cujas realizações superficiais estejam também à esquerda de a.

```
sentence                                  -->  clause.                                          (1)
sentence                                  -->  sub_clause,clause.                               (2)
sentence                                  -->  clause,sub_clause.                               (3)
sub_clause                                -->  s_conj, c_clause.                                (4)
clause                                    -->  nounph(X,nom), verbph(Y,Case),
                                               {satisfied_reflex_constraint(X,Y,Case)}.         (5)
c_clause                                  -->  clause.                                          (6)
c_clause                                  -->  clause,c_conj,clause                             (7)
nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(X,P,Gen),
                                               {Case\==reflex}.                                 (8)
proper_noun(X,P,Gen)                      -->  [P],{proper(P,Gen)}.                             (9)
nounph(X,Case)                            -->  pronoun(Case, Gen) >>>ref(X,_,Gen).              (10)
pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                        (11)
verbph(Y,acc)                             -->  verb,nounph(Y,acc).                              (12)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                           (13)
verb                                      -->  [V], {v(V)}.                                     (14)
s-conj                                    -->  [S],{s-conj(S)}.                                 (15)
s-conj                                    -->  [C],{s-conj(C)}.                                 (16)
/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                       (17)
satisfied_reflex_constraint(X,Y,reflex)   :-   X == Y                                           (18)

/* Lexicon */
s-conj(because).
s-conj(and).
proper(fred,masc).
proper(mary,fem).
proper(tom,masc).
v(saw).
v(kissed).
v(loved).
v(liked).
v(killed).
v(hated).
pro(he,nom,masc).
pro(him,acc,masc).
pro(she,nom,fem).
pro(her,acc,fem).
pro(herself,reflex,fem).
```

Quadro 2.4

2.4.2 Uma gramática no formalismo

Considere como a gramática do quadro 2.4 analisa a sentença:

(VI) Because he liked her, Fred kissed Mary

que envolve duas catáforas: "he" e "her".

Quando o pronome "he" é encontrado, conforme a regra (10), um pedido catafórico para um referente masculino é escrito na bag. Quando o pronome "her" é encontrado, conforme a regra (10), um pedido para um referente feminino é escrito na bag.

Quando o substantivo "Fred" é encontrado, conforme a regra (8), o pedido para um referente masculino que foi gerado por "he" é consumido da bag. O pronome "he" é tomado como referindo a "Fred". Similarmente, quando o substantivo "Mary" é encontrado, conforme a regra (8), o pedido para um referente feminino que foi gerado por "her" é consumido da bag e o pronome "her" é tomado como referindo a "Mary". A sentença **(VI)** é considerada gramatical, pois todos itens lexicais foram analisados e não há catáforas sem resolução na bag.

A gramática $CG1$ do quadro 2.4 rejeita sentenças agramaticais como:

(VII) Because he liked her, Fred kissed Tom.

Entretanto, uma sentença gramatical como:

(VIII) Because he loved her and she loved him, Fred kissed Mary

é rejeitada pela implementação[6] da gramática $CG1$ do quadro 2.4.

CARVALHO apresenta uma implementação melhor da gramática $CG1$, porém, nesta nova implementação, sentenças gramaticais como:

(IX) Because he hated him, Tom killed Fred.

serão rejeitadas, erroneamente, pela gramática $CG1$ do quadro 2.4.

Ela apresenta uma nova implementação da gramática $CG1$ que contorna o problema apresentado em sentenças como **(IX)**, mas passa a aceitar sentenças agramaticais como:

(X) He killed him because Fred hated Tom.

CARVALHO apresenta uma versão melhor da gramática $CG1$, quadro 2.5, que implementa, além da restrição de reflexividade, a restrição de precede – comando:

Um pronome, para ser interpretado como anaforicamente relacionado a um SN antecedente, deve ser precedido ou comandado por este SN.

Na gramática do quadro 2.5, durante a análise da sentença **(X)**, quando o pronome "he" é encontrado, conforme a regra (10), um pedido para um referente masculino, junto com uma variável X', que representa "he", é escrito na bag. Quando o pronome "him" é encontrado, nenhum pedido para um referente masculino é escrito na bag, pois já tem um na bag. Com isso, a variável Y' (a encarnação da variável Y) é compartilhada com a variável X'. Por causa da restrição de reflexividade, a análise falha. Agora o sistema retrocede ao ponto onde a bag foi checada para ver se existia um

[6] A implementação da gramática não foi mostrada, por não ser objetivo deste capítulo.

```
sentence                                  -->  clause.                                          (1)
sentence                                  -->  sub_clause,clause.                               (2)
sentence                                  -->  clause,sub_clause.                               (3)
sub_clause                                -->  # s_conj, c_clause.                              (4)
clause                                    -->  nounph(X,nom), verbph(Y,Case),
                                               {satisfied_reflex_constraint(X,Y,Case)}.         (5)
c_clause                                  -->  clause.                                          (6)
c_clause                                  -->  clause,c_conj,clause                             (7)
nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(X,P,Gen),
                                               {Case\==reflex}.                                 (8)
proper_noun(X,P,Gen)                      -->  [P],{proper(P,Gen)}.                             (9)
nounph(X,Case)                            -->  pronoun(Case, Gen) >>>ref(X,_,Gen).              (10)
pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                        (11)
verbph(Y,acc)                             -->  verb,nounph(Y,acc).                              (12)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                           (13)
verb                                      -->  [V], {v(V)}.                                     (14)
s-conj                                    -->  [S],{s-conj(S)}.                                 (15)
s-conj                                    -->  [C],{s-conj(C)}.                                 (16)
/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                       (17)
satisfied_reflex_constraint(X,Y,reflex)   :-   X == Y                                           (18)

/* Lexicon */
s-conj(because).
s-conj(and).
proper(fred,masc).
proper(mary,fem).
proper(tom,masc).
v(saw).
v(kissed).
v(loved).
v(liked).
v(killed).
v(hated).
pro(he,nom,masc).
pro(him,acc,masc).
pro(she,nom,fem).
pro(her,acc,fem).
pro(herself,reflex,fem).
```

Quadro 2.5

pedido prévio para um referente masculino. Neste momento, outro pedido para um referente masculino é escrito na bag, junto com a variável Y', que representa "him"; agora Y' não compartilha com X'.

Quando a regra (4) é chamada para satisfazer a segunda sub-meta na regra (3) (sub-clause), o operador "#" obriga que a bag seja checada a um pedido catafórico para um referente. Como há dois pedidos sem resolução na bag, a análise falha e a sentença **(X)** é considerada agramatical.

2.5 Virtual Mound and Bag Grammar (*VMBG*)

2.5.1 O formalismo

O formalismo *VMBG* é um formalismo procedimental com estrutura de armazenagem múltipla, que usa um mound e uma bag.

VMBG usa os operadores "read_mound", "write_mound", "remove_bag", "empty_bag" e "write_bag" para armazenar e resgatar tokens em sua estrutura de armazenagem virtual. As regras *VMBG* são da forma:

$$\alpha \longrightarrow \beta_1, \ldots, \beta_n$$

com α variando sobre V_N e β variando sobre ($V_N \cup V_T \cup$ {⟨read_mound(v)⟩, ⟨write_mound(v)⟩, ⟨remove_bag(v)⟩ ⟨write_bag(v)⟩, ⟨empty_bag⟩} $\cup$ V_p) e com v variando sobre V_N.

Considere a regra *VMBG*:

$$a \longrightarrow d, \langle \text{read\_mound}(b) \rangle, \langle \text{write\_bag}(c) \rangle.$$

Esta regra pode ser lida como:

Temos um símbolo não-terminal de classe a, desde que encontremos um símbolo não-terminal de classe d, mas precisamos ler no mound um símbolo não-terminal virtual de classe b, e precisamos escrever também um símbolo não-terminal virtual de classe c na bag.

VMBG pode manipular endófora (isto é, ambos anáfora e catáfora) e os dois formalismos que se seguem serão implementados em *VMBG* para fazê-lo.

2.6 O formalismo $ACG1$ (Anaphora and Cataphora Grammar)

2.6.1 O formalismo

As regras $ACG1$ têm uma das duas formas:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

com α e δ variando sobre V_N, β_i variando sobre $(V_N \cup V_T \cup V_p)$ e γ_i sobre $(V_N \cup V_T \cup \{\Theta \ggg \emptyset, \emptyset \lll \Theta, \#\Theta\} \cup V_p)$ e onde $\emptyset$ varia sobre V_N e Θ sobre $(V_T \cup V_N)$.

Considere a seguinte regra $ACG1$:

$$a \,//\, b \longrightarrow c$$

Esta regra pode ser lida como:

Um a que introduz uma marca de referência b pode ser reescrito como um c.

A regra $ACG1$:

$$a \longrightarrow b \lll c$$

pode ser lida como:

Um a pode ser reescrito como uma anáfora c que correfere com alguma categoria à sua esquerda que introduz uma marca de referência b.

A regra $ACG1$:

$$a \longrightarrow c \ggg b$$

pode ser lida como:

Um a pode ser reescrito como uma catáfora c que correfere com alguma categoria à sua direita que introduz uma marca de referência b.

Considere como a gramática do quadro 2.6 analisa a sentença:

(XI) Because he liked Mary, Fred kissed her.

Quando o pronome "he" é encontrado, conforme a regra (9), um pedido para um referente masculino é escrito na bag. Quando o substantivo "Mary" é encontrado, conforme a regra (6), um token com o gênero apropriado é escrito no mound e a bag é checada para um pedido catafórico feminino. O único pedido na bag é para um referente masculino. Com isso, o pedido fica na bag esperando pelo referente correto. Quando o substantivo "Fred" é encontrado, conforme a regra (6), um token com o gênero apropriado é escrito no mound e o pedido para um referente catafórico masculino é consumido da bag. O pronome "he" é tomado como referindo a "Fred". Quando o pronome "her" é encontrado, conforme a regra (8), o mound é checado para um referente

```
sentence                                  -->  clause.                                          (1)
sentence                                  -->  sub_clause,clause.                               (2)
sub_clause                                -->  # s_conj, c_clause.                              (3)
s-conj                                    -->  [S],{s-conj(S)}.                                 (4)
clause                                    -->  nounph(X,nom), verbph(Y,Case),
                                               {satisfied_reflex_constraint(X,Y,Case)}.       (5)
nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(X,P,Gen),{Case\==reflex}.            (6)
proper_noun(X,P,Gen)                      -->  [P], {proper(P,Gen)}.                            (7)
nounph(X,Case)                            -->  ref(X,_,Gen)<<<pronoun(Case,Gen).                (8)
nounph(X,Case)                            -->  pronoun(Case,Gen) >>>ref(X,_,Gen).               (9)
pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                        (10)
verbph(Y,acc)                             -->  verb,nounph(Y,acc).                              (11)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                           (12)
verb                                      -->  [V], {v(V)}.                                     (13)

/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                       (14)
satisfied_reflex_constraint(X,Y,reflex)   :-   X == Y.                                          (15)

/* Lexicon */
s-conj(because).
proper(fred,masc).
proper(mary,fem).
proper(jane,fem).
v(saw).
v(kissed).
v(liked).
pro(he,nom,masc).
pro(she,nom,fem).
pro(her,acc,fem).
pro(herself,reflex,fem).
```

Quadro 2.6

com o mesmo gênero que o do pronome. O token que representa "Mary" é lido e estabelecido como o antecedente do pronome "her". A sentença **(XI)** é considerada gramatical, pois não há catáforas sem resolução na bag.

A gramática do quadro 2.6 rejeita sentenças agramaticais como:

(XII) Because he liked Mary, Jane kissed her.

2.7 O formalismo $EG1$ (Endophora Grammar)

2.7.1 O formalismo

A motivação para o desenvolvimento do formalismo $EG1$ é habilitar construções endofóricas mais elegantes, eliminando a distinção entre anáfora e catáfora.

As regras $EG1$ têm uma das duas formas:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots \gamma_n$$

onde α e δ variam sobre V_N, β_i varia sobre $(V_N \cup V_T \cup V_p)$ e γ_i sobre $(V_N \cup V_T \cup \{\emptyset \,//\, \Theta, \#\Theta\} \cup V_p)$, e onde Θ varia sobre V_N e $\emptyset$ sobre $(V_T \cup V_N)$.

A diferença entre $ACG1$ e $EG1$ é que, enquanto na $ACG1$ temos regras da forma:

$$a \longrightarrow b \lll c,$$

$$a \longrightarrow c \ggg b.$$

na $EG1$ elas são substituídas por uma única regra da forma:

$$a \longrightarrow c \,//\, b.$$

```
sentence                                  -->  clause.                                   (1)
sentence                                  -->  sub_clause,clause.                        (2)
sub_clause                                -->  # s_conj, c_clause.                       (3)
s-conj                                    -->  [S],{s_conj(S)}.                          (4)
clause                                    -->  nounph(X,nom), verbph(Y,Case),
                                               {satisfied_reflex_constraint(X,Y,Case)}. (5)
nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(X,P,Gen),
                                               {Case\==reflex}.                         (6)
proper_noun(X,P,Gen)                      -->  [P], {proper(P,Gen)}.                     (7)
nounph(X,Case)                            -->  pronoun(Case,Gen)// ref(X,_,Gen).         (8)
pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                 (9)
verbph(Y,acc)                             -->  verb,nounph(Y,acc).                       (10)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                    (11)
verb                                      -->  [V], {v(V)}.                              (12)
/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                (13)
satisfied_reflex_constraint(X,Y,reflex)   :-   X == Y.                                   (14)

/* Lexicon */
s-conj(because).
proper(fred,masc).
proper(mary,fem).
v(saw).
v(kissed).
v(liked).
pro(he,nom,masc).
pro(she,nom,fem).
pro(her,acc,fem).
pro(herself,reflex,fem).
```

Quadro 2.7

Esta regra pode ser lida como:

Um *a* pode ser reescrito como um *c* que correfere com alguma categoria (que pode estar à sua direita ou à sua esquerda) que introduz uma marca de referência *b*.

2.7.2 Uma gramática no formalismo

Considere como a gramática do quadro 2.7 analisa a sentença:

(XIII) Because he liked Mary, Fred kissed her.

Quando o pronome "he" é encontrado, conforme a regra (8), um pedido para um referente masculino é escrito na bag. Quando o substantivo "Mary" é encontrado, conforme a regra (6), um token com o gênero apropriado é escrito no mound, e a bag é checada para um pedido feminino. Como neste momento o único pedido na bag é para um referente masculino, ele fica na bag e a análise continua. Quando o substantivo "Fred" é encontrado, conforme a regra (6), um token com o gênero apropriado é escrito no mound, e o pedido para um referente masculino é removido da bag. O pronome "he" é tomado como referindo a "Fred". Quando o pronome "her" é encontrado, conforme a regra (8), um referente feminino é lido do mound. O pronome "her" é tomado como referindo ao token que representa "Mary". Como não há pedido sem solução na bag, a análise obteve sucesso e a sentença **(XIII)** é considerada gramatical.

2.8 Virtual Mound and Stack Grammar (*VMSG*)

Tendo considerado endófora, vamos considerar anáfora e suas interações com extraposição.

2.8.1 O formalismo

VMSG é outro formalismo procedimental com estrutura de armazenagem múltipla, que usa um mound e a pilha stack para sua armazenagem virtual.

A pilha stack é um estrutura com as seguintes características de armazenagem e resgate:

a. Os itens novos são armazenados no topo da stack.

b. A pilha stack usa o critério LIFO (last-in, first-out), ou seja, as operações de armazenagem e resgate são executadas no topo da stack e com isso os elementos podem somente ser removidos na ordem oposta a que foram inseridos;

c. A operação de armazenagem é a "push" e a de resgate é a "pop".

*VMSG*s usam os operadores "read_mound", "write_mound", "push" e "pop" para armazenar e resgatar tokens em sua estrutura de armazenagem virtual.

As regras *VMSG* são da forma:

$$\alpha \longrightarrow \beta_1, \ldots, \beta_n$$

com α variando sobre V_N e β_i variando sobre $(V_N \cup V_T \cup \{\langle \text{read\_mound}(v)\rangle, \langle \text{write\_mound}(v), \langle \text{push}(v)\rangle, \langle \text{pop}(v)\rangle\} \cup V_p\}$, e onde v varia sobre V_N.

A regra *VMSG*:

$$a \longrightarrow b, \langle \text{push}(c)\rangle, \langle \text{read\_mound(d)}\rangle$$

pode ser lida como:

Temos um símbolo não terminal de classe a, se encontrarmos um símbolo não-terminal de classe b, mas precisamos armazenar no topo da stack um símbolo não-terminal de classe c e precisamos também resgatar um símbolo não-terminal virtual de classe d no mound.

VMSG pode manipular anáfora e extraposição dos constituintes das cláusulas relativas, e o próximo formalismo declarativo será implementado em *VMSG* para fazê-lo.

2.9 O formalismo $AXG1$ (Anaphora and Extraposition Grammar)

2.9.1 O formalismo

$AXG1$ é um formalismo gramatical declarativo. Existem três formas das regras $AXG1$:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

$$\alpha \ldots \varepsilon \longrightarrow \beta_1, \ldots, \beta_n$$

onde α, δ e ε variam sobre V_N, β_i varia sobre $(V_N \cup V_T \cup V_p)$ e γ_i sobre $(V_N \cup V_T \cup \{\Theta \lll\} \cup V_p)$, e onde Θ varia sobre V_N e $\emptyset$ sobre $(V_T \cup V_N)$.

Uma regra $AXG1$ da forma:

$$a \ldots b \longrightarrow c, d.$$

pode ser lida como:

Uma seqüência de categorias consistindo de um a e um "trace" b que está à direita de a pode ser reescrito como um c seguido por um d, desde que

nem a e nem b interrompam um conjunto de categorias que são reescritas por uma regra similar a esta, exceto quando a está acompanhado de b.

Uma regra $AXG1$ da forma:

$$a \,//\, b \longrightarrow c.$$

pode ser lida como:

Um a que introduz uma marca de referência b pode ser rescrito como um c.

Uma regra $AXG1$ da forma:

$$a \longrightarrow b \lll c.$$

pode ser lida como:

Um a pode ser reescrito como uma anáfora c que correfere com alguma categoria à sua esquerda que introduz uma marca de referência b.

2.9.2 Uma gramática no formalismo

Vejamos como a gramática $AXG1$ do quadro 2.8 analisa a sentença:

(XIV) The man who saw Mary kissed her.

Quando o pronome relativo "who" é encontrado, conforme a regra (12), um "trace" é armazenado na stack, mas, antes disto, conforme a regra (13), um "close" é armazenado na stack para impedir que o "trace" seja consumido fora da cláusula relativa. Após "who" ser consumido, conforme a regra (10), uma sentença deve ser encontrada.

```
sentence                                --> nounph(X,nom),verbph(Y,Case),                    (1)
                                            {satisfied_reflex_constraint(X,Y,Case)}.

nounph(X,Case)// ref(X,X=P,Gen)         --> proper_noun(X,P,Gen),
                                            {Case\==reflex}.                                 (2)

nounph(X,Case)// ref(X,P,Gen)           --> determiner, noun(X,P,Gen),                       (3)
                                            relative(X), {Case\== reflex}.

nounph(X,Case)                          --> ref(X,_,Gen)<<<pronoun(Case,Gen).                (4)

nounph(X,_)                             --> trace(X), {Case\== reflex}.                      (5)

proper_noun(X,P,Gen)                    --> [P], {proper(P,Gen)}.                            (6)

determiner                              --> [D], {det(D)}.                                   (7)

noun(X,P,Gen)                           --> [N], {n(N,X,P,Gen)}.                             (8)

pronoun(Case,Gen)                       --> [Pr], {pro(Pr,Case,Gen)}.                        (9)

relative(X)                             --> open, rel_mk(X), sentence, close.                (10)

relative(X)                             --> [ ]                                              (11)

rel_mk(X)...trace(X)                    --> [R], {rel_pro(R)}.                               (12)

open...close                            --> [ ]                                              (13)

verbph(Y,acc)                           --> verb,nounph(Y,acc).                              (14)
verbph(Y,reflex)                        --> verb,nounph(Y,reflex).                           (15)

verb                                    --> [V], {v(V)}.                                     (16)

/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)   :-  X\== Y, Case \== reflex,!.                       (17)

satisfied_reflex_constraint(X,Y,reflex) :-  X == Y                                           (18)

/* Lexicon */
det(the).
n(man,X,man(X),masc).
proper(fred,masc).
proper(mary,fem).
v(saw).
v(kissed).
v(liked).
v(fancied)
pro(her,acc,dem).
pro(him,acc,masc).
pro(himself,reflex,masc).
rel_pro(who)
```

Quadro 2.8

Não há *SN* dentro da cláusula relativa para ser tomada como sujeito da sentença, porém, conforme a regra (5), o "trace" introduzido pelo pronome relativo é consumido como um *SN*. Quando o substantivo "Mary" é encontrado, conforme a regra (2), um token com o gênero apropriado é escrito no mound, e, conforme a regra (10), um "close" é consumido da stack, portanto completando a cláusula relativa.

Agora, e somente agora, um token que representa "man", com o gênero apropriado é escrito no mound. Quando o pronome "her" é encontrado, conforme a regra (4), um antecedente feminino é lido no mound. O pronome "her" é tomado como referindo a "Mary". Com isso terminamos a análise com sucesso e a sentença **(XIV)** é considerada gramatical.

A gramática *AXG*1 do quadro 2.8 rejeita sentenças agramaticais como:

(XV) The man who saw Mary kissed him.

Porém rejeita, também, sentenças gramaticais como:

(XVI) The man who fancied himself kissed Mary.

Para contornar esse problema, CARVALHO apresenta um novo tipo de estrutura de armazenagem, uma pile, com características diferentes do mound.

2.10 Virtual Pile and Stack Grammar (*VPSG*)

VPSG é um formalismo procedimental com estrutura de armazenagem múltipla, que usa uma pile e uma stack como sua estrutura de armazenagem virtual.

Uma pile é uma estrutura com as seguintes características de armazenagem e resgate:

a. Os itens novos são adicionados no topo da pile;

b. Sempre que for necessário determinar se um item tendo certas características está presente na pile e, se ele está presente, resgata todas suas características, a busca começa no topo do pile: isto é, os itens mais novos na pile são tratados como sendo mais relevantes que os outros;

c. Quando um item com as características requisitadas é encontrado, ele é movido para o topo da pile;

d. Existem dois mecanismos de acesso: remove_pile e absent_pile. O operador remove_pile remove um item particular da pile, e absent_pile verifica se um item particular já foi retirado da pile.

As regras *VPSG* têm a forma:

$$\alpha \longrightarrow \beta_1, \ldots \beta_n$$

onde α varia sobre V_N e β_i varia sobre ($V_N \cup V_T \cup \{\langle \text{read\_pile}(v)\rangle$, $\langle \text{write\_pile}(v)\rangle$, $\langle \text{remove\_pile}(v)\rangle$, $\langle \text{absent\_pile}(v)\rangle$, $\langle \text{push}(v)\rangle, \langle \text{pop}(v)\rangle\} \cup V_p$) e onde v varia sobre V_N.

VPSG usa os operadores "read_pile", "write_pile", "absent_pile", "remove_pile", "push" e "pop" para armazenar e resgatar informações de suas estruturas de armazenagem virtual.

A regra *VPSG*:

$$a \longrightarrow b, \langle \text{push}(c) \rangle, \langle \text{read\_pile}(d) \rangle$$

pode ser lida como:

Temos um símbolo não-terminal de classe a se encontrarmos um símbolo não-terminal de classe b, mas precisamos armazenar no topo da stack um símbolo não-terminal de classe c, e precisamos também ler um símbolo não-terminal virtual de classe d da pile.

Uma regra *VPSG* da forma:

$$a \longrightarrow b, \langle \text{absent\_pile}(c) \rangle, \langle \text{pop(d)} \rangle$$

pode ser lida como segue.

Temos um símbolo não-terminal de classe a, se encontrarmos um símbolo não-terminal de classe b, mas um símbolo não-terminal de classe c precisa ter sido retirado da pile e um símbolo não-terminal de classe d precisa ser resgatado da stack.

A sentença:

(XVII) The man who fancied himself kissed Mary

é analisada pela gramática *AXG*1 do quadro 2.8 da página 71 da seguinte forma: quando a regra (3) é chamada para resolver a primeira sub-meta da regra (1) (nounph), um token é escrito na pile. Neste ponto do processo, os três argumentos da estrutura ref(X', P', Gen') não estão instanciados. Quando o

SN "the man" é analisado, a variável P' fica estanciada com "man(X')" e a variável Gen' com "masc". Quando o pronome relativo "who" é encontrado, conforme a regra (12), um "trace", cujo argumento é X', é armazenado na stack. Mas antes disto, conforme a regra (13), um close é armazenado na stack para impedir que o "trace" seja consumido fora da cláusula relativa. Agora uma sentença deve ser analisada, conforme a regra (10).

Como não há *SN* dentro da cláusula relativa para ser tomada como o sujeito da sentença embutida, conforme a regra (5), o "*trace*(X')" é consumido como um *SN* e a variável X'' é compartilhada com a variável X'. Quando o pronome "himself" é encontrado, conforme a regra (4), um referente masculino é lido da pile e o pronome "himself" é tomado como referindo a "man". A variável Y'' (a encarnação de Y que é passada como um argumento da segunda sub-meta na regra (1), (verbph), é compartilhada com a variável X'. Para satisfazer à terceira submeta da regra (1) a regra Prolog (17) é chamada, e falha, pois X'' e Y'' são iguais, então, a regra Prolog (18) é chamada e obtém sucesso, pois as variáveis X'' e Y'' são iguais e o caso é "reflex", como a regra exige. Agora, conforme a regra (10), um "close" é resgatado da stack para completar a cláusula relativa.

Após a cláusula relativa ter sido analisada, conforme a regra (3), o token que está no topo da pile, que é o token que representa "man", é removido da pile. Com isso, o sistema tenta ler o mesmo token (o token que representa "man") na pile. A motivação para remover o token da pile e tentar lê-lo novamente é pelo fato de que não queremos que haja duplicação na pile. Neste caso, após removermos o token que representa "man" da pile, não podemos lê-lo novamente, porque este é o único "man" que há na sentença. Portanto, escrevemos o token que representa "man" com o gênero apropriado de volta na pile. Quando a regra (2) é chamada para satisfazer à segunda submeta

na regra (14) (nounph), um token é escrito na pile com três variáveis não-instanciadas (Y', P'' e Gen''). Após o substantivo "Mary" ter sido analisado, a variável P'' fica instanciada com "Mary" e a variável Gen'' com "fem". Agora o token representando "Mary" é removido da pile e o sistema tenta ler o mesmo token na pile. Como este token não está presente, escrevemos o token representando "Mary" de volta na pile. Com isso a análise obteve sucesso e a sentença **(XVII)** é considerada gramatical.

2.11 Virtual Bag and Stack Grammar ($VBSG$)

Tendo considerado anáfora e extraposição, vamos considerar catáfora e suas interações com *extraposição*.

2.11.1 O formalismo

VBSG é um formalismo procedimental com estrutura de armazenagem múltipla, que usa uma bag e uma stack como sua estrutura de armazenagem virtual.

*VBSG*s usam os operadores "remove_bag", "write_bag", "empty_bag", "push" e "pop" para armazenar e resgatar tokens em sua estrutura de armazenagem virtual.

As regras *VBSG* são da forma:

$$\alpha \longrightarrow \beta_1 \ldots \beta_n$$

onde α varia sobre V_N e β_i varia sobre $(V_N \cup V_T \cup \{\langle \text{write\_bag}(v)\rangle, \langle \text{remove\_bag}(v)\rangle, \langle \text{empty\_bag}\rangle, \langle \text{write\_pile}(v)\rangle, \langle \text{read\_pile}(v)\rangle, \langle \text{remove\_pile}(v)\rangle, \langle \text{push}(v)\rangle, \langle \text{pop}(v)\rangle\}, \cup V_P)$, onde v varia sobre V_N.

A regra *VBSG* da forma:

$$a \longrightarrow b, \langle \text{remove\_bag}(c) \rangle, \langle \text{push}(d) \rangle.$$

pode ser lida como:

Temos um símbolo não-terminal de classe a, desde que encontremos um símbolo não-terminal de classe b, mas precisamos remover da bag um símbolo não-terminal virtual de classe c, e precisamos também armazenar um símbolo não-terminal de classe d na stack.

*VBSG*s podem manipular catáfora e extraposição, e o próximo formalismo declarativo será implementado em *VBSG*, para fazê-lo.

2.12 O formalismo $CXG1$ (Cataphora and Extraposition Grammar)

2.12.1 O formalismo

$CXG1$ é um formalismo gramatical declarativo, as regras de $CXG1$ têm uma das três formas:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

$$\alpha \ldots \xi \longrightarrow \gamma_1, \ldots, \gamma_n$$

onde α, δ e ξ variam sobre V_N, β_i varia sobre $(V_N \cup V_T \cup V_P)$ e γ_i sobre $(V_N \cup V_T \cup \{\phi \ggg \Theta, \#\phi\} \cup V_P)$, com Θ variando sobre V_N e ϕ sobre $(V_T \cup V_N)$.

Uma regra $CXG1$ da forma:

$$a \,//\, b \longrightarrow c.$$

pode ser lida como:

Um a que introduz uma marca de referência b, pode ser reescrito como um c.

A regra $CXG1$ da forma.

$$a \longrightarrow c \ggg b.$$

pode ser lida como:

Um a pode ser reescrito como uma catáfora c que correfere com alguma categoria à sua direita que introduz uma marca de referência b.

Uma regra $CXG1$ da forma:

$$a \ldots b \longrightarrow c, d.$$

pode ser lida como:

Uma seqüência consistindo de um a e um vestígio b, que está em algum lugar à direita de a, pode ser reescrito como um c seguido por um d, desde que nem a nem b interrompam um conjunto de categorias que são reescritos por uma regra similar a esta, exceto quando a está acompanhado por b.

Vamos considerar como a gramática do quadro 2.9, analisa a sentença:

(XVIII) The man who she saw kissed Mary.

Quando o pronome relativo "who" é encontrado, conforme a regra (12), um "trace" é armazenado na stack. Mas antes disto, conforme a regra (13), um "close" é armazenado na stack para retroceder até a cláusula relativa, e impedir que o vestígio seja consumido fora da cláusula relativa. Quando o pronome "she" é encontrado conforme a regra (4), um pedido para um

```
sentence                                  -->  nounph(X,nom),verbph(Y,Case),                    (1)
                                               {satisfied_reflex_constraint(X,Y,Case)}.

nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(X,P,Gen),
                                               {Case\==reflex}.                                 (2)

nounph(X,Case)// ref(X,P,Gen)             -->  determiner, noun(X,P,Gen),                       (3)
                                               relative(X), {Case\== reflex}.

nounph(X,Case)                            -->  pronoun(Case, Gen)>>>ref(X,_,Gen).               (4)

nounph(X,_)                               -->  trace(X), {Case\== reflex}.                      (5)

proper_noun(X,P,Gen)                      -->  [P], {proper(P,Gen)}.                            (6)

determiner                                -->  [D], {det(D)}.                                   (7)

noun(X,P,Gen)                             -->  [N], {n(N,X,P,Gen)}.                             (8)

pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                        (9)

relative(X)                               -->  open, rel_mk(X), sentence, close.                (10)

relative(X)                               -->  [ ]                                              (11)

rel_mk(X)...trace(X)                      -->  [R], {rel_pro(R)}.                               (12)

open...close                              -->  [ ]                                              (13)

verbph(Y,acc)                             -->  verb,nounph(Y,acc).                              (14)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                           (15)

verb                                      -->  [V], {v(V)}.                                     (16)

/*Reflexivity Constraint */
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                       (17)

satisfied_reflex_constraint(X,Y,reflex)   :-   X == Y                                           (18)

/* Lexicon */
det(the).
n(man,X,man(X),masc).
proper(fred,masc).
proper(mary,fem).
v(saw).
v(kissed).
v(liked).
pro(her,acc,fem).
pro(she,nom,fem).
rel_pro(who).
```

Quadro 2.9

referente feminino é escrito na bag. Após o verbo "saw" ter sido analisado, conforme a regra (14), um *SN* deve ser encontrado. Não há *SN* dentro da cláusula relativa; entretanto, conforme a regra (5), o "trace" é consumido como um *SN*. Agora, conforme a regra (10), o "close" virtual é consumido da stack, completando a cláusula relativa.

Agora que a cláusula relativa está completa, conforme a regra (3), a bag é checada para um pedido de referente masculino. Como não há um tal pedido na bag, a análise continua. Quando o substantivo "Mary" é encontrado, conforme a regra (2), um pedido para um referente feminino é lido da bag, e o pronome "she" é tomado como referente de "Mary". Como não há pedidos sem resolução na bag a análise obteve sucesso e a sentença **(XVIII)** é considerada gramatical.

Convém salientar que sentenças como

(XIX) The man who saw her liked Mary.

e

(XX) The man who saw him killed fred.

são analisadas pela gramática do quadro 2.9 com sucesso.

2.13 Virtual Pile, Bag and Stack Grammar (*VPBSG*)

Tendo considerado catáfora e sua interação com extraposição, vamos considerar endófora e sua interação com extraposição.

2.13.1 O formalismo

VPBSG é um formalismo procedimental com estrutura de armazenagem múltipla, que usa uma pile, uma bag e uma stack como sua estrutura de armazenagem virtual.

*VPBSG*s usam os operadores "write_pile", "read_pile", "absent_pile", "remove_pile", "write_bag", "empty_bag", "remove_bag", "push" e "pop" para armazenar e resgatar informação virtual.

As regras *VPBSG* são da forma:

$$\alpha \longrightarrow \beta_1, \ldots, \beta_n$$

onde α varia sobre V_N e β_i varia sobre $(V_N \cup V_T \cup \{\langle \text{write\_bag}(v)\rangle, \langle \text{remove\_bag}(v)\rangle, \langle \text{empty\_bag}\rangle, \langle \text{write\_pile}(v)\rangle, \langle \text{read\_pile}(v)\rangle, \langle \text{absent\_pile}(v)\rangle, \langle \text{remove\_pile}(v)\rangle, \langle \text{push}(v)\rangle, \langle \text{pop}(v)\rangle\} \cup V_P\}$, onde v varia sobre V_N.

A seguinte regra *VPBSG*:

$$a \longrightarrow b, \langle \text{write\_pile}(c)\rangle, \langle \text{remove\_bag}(d)\rangle, \langle \text{push}(e)\rangle.$$

pode ser lida como:

Temos um símbolo não-terminal de classe a, desde que encontremos um símbolo não-terminal de classe b, mas precisamos escrever um símbolo não-terminal de classe c na pile, precisamos remover um símbolo não-terminal de classe d da bag, e precisamos armazenar um símbolo não-terminal de classe e na stack.

VPBSG pode manipular endófora e extraposição, e o formalismo que se segue será implementado em *VPBSG* para fazê-lo.

2.14 O formalismo $EXG1$ (Endofora and Extraposição Grammar).

2.14.1 O formalismo

$EXG1$ é um formalismo gramatical declarativo, as regras $EXG1$ têm uma das três formas:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

$$\alpha \ldots \xi \longrightarrow \gamma_1, \ldots, \gamma_n$$

onde α, δ e ξ variam sobre V_N, β_i varia sobre $(V_N \cup V_T \cup V_P)$, e γ_i sobre $(V_N \cup V_T \cup V_P \cup \{\phi \,//\, \Theta, \# \Theta\})$, e onde Θ varia sobre V_N e ϕ varia sobre $(V_T \cup V_N)$.

Vejamos como a gramática do quadro 2.10 analisa a seguinte sentença:

(XXI) Because he liked her, the man who saw Mary kissed her.

Quando o pronome "he" é encontrado, conforme a regra (8), um pedido para um referente masculino junto com uma variável X' (a encarnação da variável X que é passada como um argumento da primeira sub-meta na regra (5) (nounph)), é escrito na bag. Quando o pronome "her" é encontrado, conforme a regra (8), um pedido para um referente feminino junto com uma variável Y' (a encarnação da variável Y que é passada como um argumento da segunda sub-meta na regra (5) (verbph)), é escrito na bag. Para satisfazer a terceira sub-meta da regra (5), a regra Prolog (21) é chamada, e obtém sucesso, pois X' e Y' são diferentes e case não é "reflex", como exige a regra.

```
sentence                                  -->  clause.                                          (1)
sentence                                  -->  sub_clause, sentence.                            (2)
sub_clause                                -->  # s_conj, clause.                                (3)
s-conj                                    -->  [S],{s_conj(S)}.                                 (4)
clause                                    -->  nounph(X,nom), verbph(Y,Case),
                                               {satisfied_reflex_constraint(X,Y,Case)}.       (5)
nounph(X,Case)// ref(X,X=P,Gen)           -->  proper_noun(X,P,Gen),
                                               {Case\==reflex}.                               (6)
nounph(X,Case)// ref(X,P,Gen)             -->  determiner, noun(X,P,Gen),
                                               relative(X), {Case \== reflex}.                (7)
nounph(X,Case)                            -->  pronoun(Case,Gen)// ref(X,_,Gen).             (8)
nounph(X,_)                               -->  trace(X), {Case \== reflex}.                  (9)
proper_noun(X,P,Gen)                      -->  [P], {proper(P,Gen)}.                         (10)
determiner                                -->  [D], {det(D)}.                                (11)
noun(X,P,Gen)                             -->  [N], {n(N,X,P,Gen)}.                          (12)
pronoun(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                     (13)
relative(X)                               -->  open, rel_mk(X), sentence, close.             (14)
relative(X)                               -->  [ ]                                           (15)
rel_mk(X)...trace(X)                      -->  [R], {rel_pro(R)}.                            (16)
open...close                              -->  [ ]                                           (17)
verbph(Y,acc)                             -->  verb,nounph(Y,acc).                           (18)
verbph(Y,reflex)                          -->  verb,nounph(Y,reflex).                        (19)
verb                                      -->  [V], {v(V)}.                                  (20)
satisfied_reflex_constraint(X,Y,Case)     :-   X\== Y, Case \== reflex,!.                    (21)
satisfied_reflex_constraint(X,Y,reflex)   :-   X == Y.                                       (22)

s_conj(because)
det(the).
n(man,X,man(X),masc).
proper(fred,masc).
proper(mary,fem).
v(saw).
v(kissed).
v(liked).
pro(he,nom,masc).
pro(her,acc,fem).
rel_pro(who).
```

Quadro 2.10

Quando a regra (7) é chamada para satisfazer a primeira sub-meta da regra (5) (nounph), um token é escrito na pile com três variáveis não-instanciadas (X'', P' e Gen'). Quando o SN "the man" é analisado, a variável P' fica instanciada com "man(X'')" e a variável Gen' com "masc". Quando o pronome relativo "who" é encontrado, conforme a regra (17), um "close" é armazenado na stack e conforme a regra (16), um "trace", cujo argumento é X'' é armazenado no topo da stack. Agora conforme a regra (14), uma sentença deve ser analisada.

Não há SN dentro da cláusula relativa para ser tomado como o sujeito da sentença, porém, conforme a regra (9), o "$trace(X'')$" é consumido como um SN e a variável X'' é compartilhada com a variável X''. Quando a regra (6) é chamada para satisfazer a segunda sub-meta da regra (18) (nounph), um token é escrito na pile com três variáveis não-instanciadas (Y''', P'' e Gen''). Após o substantivo "Mary" ter sido analisado, a variável P'' fica instanciada com "Mary" e a variável Gen'' com "fem".

Agora, de acordo com a regra (6), o token que representa "Mary" é removido da pile e como não há outro token representando "Mary" lá, ele é escrito de volta na pile. O pedido para um referente feminino é removido da bag e o pronome "her" é tomado como referente de "Mary" e compartilhado com as variáveis Y' e Y'''.

Para satisfazer a terceira sub-meta da regra (5), a regra Prolog (21) é chamada, e obtém sucesso, pois as variáveis Y''' e X''' são diferentes e o caso não é "reflex". De acordo com a regra (14), o "close" é consumido da stack completando a cláusula relativa.

Agora, conforme a regra (7), o token que representa "man" é removido da pile e como não há outro token representando "man" na pile, ele é escrito

de volta na pile. Além disso, conforme a regra (7), o pedido para um referente masculino é removido da bag e o pronome "he" é tomado como referente de "man" compartilhando com as variáveis X' e X''.

Quando o pronome "her" é encontrado, conforme a regra (8), um referente masculino é lido da pile e o pronome "her" é tomado como referente do token que representa "Mary"; a variável Y'', que representa o pronome "her" é compartilhada com a variável Y''' que representa "Mary". Agora, para satisfazer a terceira sub-meta da regra (5), a regra Prolog (21) é chamada e obtém sucesso, pois as variáveis Y'' e X'' são diferentes e o caso é diferente de "reflex". Como não há pedidos sem resolução na bag, a sentença **(XXI)** é considerada gramatical.

Resumo

Neste capítulo, Carvalho apresentou vários formalismos gramaticais, desde os mais simples até os mais sofisticados. Iniciou com o formalismo gramatical $AG1$ para manipular anáfora, logo após desenvolveu o formalismo gramatical $CG1$ para manipular catáfora, e então desenvolveu o formalismo gramatical $ACG1$ capaz de manipular a interação entre anáfora e catáfora. Com o desenvolvimento do formalismo gramatical $EG1$ (que é uma melhoria do formalismo gramatical $ACG1$) desaparece a distinção entre anáfora e catáfora nas regras da gramática. No formalismo gramatical $AXG1$ há manipulação das anáforas e suas interações com extraposição, assim como no formalismo gramatical $CXG1$ há a manipulação das catáforas e suas interações com extraposição, e finalmente, no formalismo gramatical $EXG1$ há a manipulação das endóforas e suas interações com extraposição.

Duas restrições em correferência foram implementadas por dois métodos diferentes, a saber, a restrição em reflexivização foi implementada pelo método manual_explícito, e a restrição em precede comando pelo método manual_implícito.

No próximo capítulo, serão apresentados formalismos ainda mais sofisticados que implementam restrições em correferência automaticamente.

Capítulo 3

Formalismos gramaticais que implementam restrições em endófora embasados na noção de c-comando

CARVALHO nos apresenta uma série de formalismos gramaticais que implementam algumas restrições em correferência que estão embasadas na noção de $C-$comando, tais como:

(Ia) Um *SN* pleno deve ser interpretado como não-correferencial com outro *SN* pleno que ele $C-$comande.

(Ib) Um pronome deve ser interpretado como não-correferencial com outro *SN* pleno que ele $C-$comande.

(II) Um pronome reflexivo deve ser interpretado como correferencial com, e somente com, um *SN* $C-$comandante dentro de um domínio sintático especificado.

(III) Um pronome não-reflexivo deve ser interpretado como não-correferencial com outro *SN* $C-$comandante no domínio sintático que é especificado para **(II)**.

(IV) Um constituinte movido deve $C-$comandar seu vestígio[1].

3.1 Anáfora: Uma gramática $AG1$

Considere a gramática do quadro3.1, que está embasada no formalismo $AG1$, descrito anteriormente no capítulo 2, e que explicitamente implementa as restrições (Ib), (II) e (III). Esta gramática pode ser dividida em três partes: Regras de estrutura de frase, Léxico e restrições em correferência. As regras (12) e (13) lidam com as restrições. A primeira sub-meta da regra (12) checa-se o caso é diferente de reflexivo; a segunda sub-meta implementa a restrição (Ib) ao certificar que o pronome não $C-$comande seu antecedente; a terceira sub-meta encontra a categoria minimal de governo do pronome; e a quarta sub-meta implementa a restrição (III) ao certificar que o antecedente não $C-$comande o pronome e não está dentro de sua categoria minima de governo.

A primeira sub-meta da regra (13) checa se o caso do pronome é reflexivo; a segunda sub-meta certifica que o antecedente $C-$comande o pronome, de acordo com a restrição (II); a terceira sub-meta assegura que o pronome não $C-$comande seu antecedente, de acordo com a restrição (Ib); a quarta sub-meta encontra a categoria minimal de governo do pronome; e a quinta

[1]A restrição **(IV)** corresponde à restrição **(VI)** que se encontra na página 97 de CARVALHO.

```
sentence(N)                              -->  nounph(N-1,nom),verbph(N-2).                    (1)

nounph(N,Case)// ref(N,X,X=P,Gen)        -->  proper_noun(N~1,X,P,Gen)
                                              {Case=nom;Case=acc}.                            (2)

nounph(N,Case)// ref(N,X,P,Gen)          -->  determiner(N-1),nom(N-2,X,P,Gen),               (3)
                                              {Case=nom;Case=acc}.

nounph(N,Case)                           -->  ref(M_,_,Gen)<<<pronoun(N~1, Case,Gen),         (4)
                                              {satisfy_constraints(Case,M,N)}.

proper_noun(X,P,Gen)                     -->  [P], {proper(P,Gen)}.                           (5)

determiner(N)                            -->  [D], {det(D)}.                                  (6)

noun(N,X,P,Gen)                          -->  [W], {n(W,X,P,Gen)}.                            (7)

pronoun(N, Case,Gen)                     -->  [Pr], {pro(Pr,Case,Gen)}.                       (8)

verbph(N)                                -->  verb(N-1),nounph(N-2,acc).                      (9)

verbph(N)                                -->  verb(N-1),nounph(N-2,reflex).                   (10)

verb(N)                                  -->  [V], {v(V)}.                                    (11)

satisfy_constraint(Case,A,P)             -->  \+(Case = reflex, \+(c_commands(P,A)),          (12)
                                              min_gov_cat(P,MGC),
                                              \+((c_commands(A,P),dominates(MGC,A))).

satisfy_constraint(Case,A,P)             -->  Case = reflex, (c_commands(A,P)),
                                              \+(c_commands(P,A)),                            (13)
                                              min_gov_cat(P,MGC), dominates(MGC,A))).

dominates(X,X-_).                                                                            (14)
dominates(X,X~_).                                                                            (15)
dominates(X,A-_).                        -->  \+(X =A),dominates(X,A).                       (16)
dominates(X,A~_)                         -->  \+(X =A),dominates(X,A).                       (17)

c_commands(A,B)                          -->  first_branching_ancestor(A,C),
                                              dominates(C,B).                                 (18)

first_branching_ancestor(X-N,X).                                                             (19)

first_branching_ancestor(X~N,Y)          -->  first_branching_ancestor(X,Y).                 (20)

min_gov_cat(g(1)-N,g(1))                 -->  \+(N=(_-_)),\+(N=(_~_)).                       (21)
min_gov_cat(g(1)~N,g(1))                 -->  \+(N=(_~_)),\+(N=(_:-_)).                      (22)
min_gov_cat(K-g(L)-N,K-g(L))             -->  \+(N=(_-_)),\+(N=(_-_)).                       (23)
min_gov_cat(K-g(L)~N,K-g(L))             -->  \+(N=(_-_)),\+(N=(_-_)).                       (24)
min_gov_cat(K~g(L)N,K~g(L))              -->  \+(N=(_-_)),\+(N=(_~_)).                       (25)
min_gov_cat(K~g(L)~N,K~g(L))             -->  \+(N=(_~_)),\+(N=(_-_)).                       (26)
min_gov_cat(K-N,MGC)                     -->  \+(K=J-g(L), \+(K=J~g(L)),\+(K=g(1)),          (27)
                                              min_gov_cat(K,MGC).

min_gov_cat(K~N,MGC)                     -->  \+(K=J~g(L)), \+(K=J-g(L)),\+(K=g(1)),         (28)
                                              min_gov_cat(K,MGC).

proper(fred,masc).
v(fancied).
prop(him,acc,masc).
pro(himself,reflex,masc).
```

Quadro 3.1

sub-meta assegura que a categora minimal de governo do pronome domine o antecedente, conforme exige a restrição (II).

Vamos considerar como esta gramática analisa a sentença:

(XXII) Fred fancied himself.

Quando a regra (2) é chamada para satisfazer a primeira sub-meta da regra (1) (nounph), o seguinte token é escrito no mound:

$$ref(g(1)-1, X, X = P, Gen).$$

onde $g(1)-1$, rotula a posição do SN na análise de árvore.

Após o substantivo "fred" ser analisado, a variável P fica enstanciada com "fred" e Gen com "masc", e o token no mound fica:

$$ref(g(1)-1, X, X = \text{fred}, \text{masc}).$$

Quando o pronome "himself" é encontrado, conforme a regra (4), um referente masculino é lido do mound e o pronome "himself" é tomado como referente de "fred". Agora, para satisfazer o teste Prolog na regra (4), a regra (12) é chamada com a seguinte meta:

$$\text{satisfy\_constraints}(\text{reflex}, g(1)-1, g(1)-2-2).$$

onde $g(1)-2-2$, é a posição na análise de árvore do SN pronominal. A primeira sub-meta na regra (12) falha, pois caso é igual a "reflex". Com isso a regra (13) é chamada e a primeira sub-meta obtém sucesso, pois caso é igual a reflex, a segunda sub-meta também pois $g(1)-1$, $C-$Comanda $g(1)-2-2$. A terceira sub-meta também obtém sucesso, pois $g(1)-2-2$, não $C-$Comanda $g(1)-1$. A quarta sub-meta determina que $g(1)$ é a categoria minimal de governo de $g(1)-2-2$, e finalmente a quinta sub-meta obtém sucesso pois

$g(1)$ domina $g(1)-1$. Com isso, a análise obtém sucesso e a sentença **(XXII)** é considerada gramatical.

A sentença:

(XXIII) Fred fancied him.

é rejeitada por esta gramática.

3.2 O formalismo $AG2$ (Anaphora Grammar)

3.2.1 O formalismo

$AG2$ é um formalismo gramatical declarativo. Existem duas classes de entrada numa gramática $AG2$: regras gramaticais e declarações de domínio de correferência.

As regras $AG2$ têm uma das duas formas:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

onde α e δ variam sobre V_N, β_i varia sobre $(V_T \cup V_N \cup V_P)$ e γ_i sobre $(V_T \cup V_N \cup \{\Theta < -\mu < -\phi\} \cup V_p)$, e onde Θ varia sobre V_N, μ sobre (nom, acc, reflex) e ϕ sobre $(V_T \cup V_N)$.

As declarações de domínio de correferência têm a forma:

$$@\alpha,$$

onde α varia sobre V_N.

A regra $AG2$:

$$a \,//\, b \longrightarrow c.$$

pode ser lida como:

Um a que introduz uma marca de referência b pode ser reescrito como um c.

A regra $AG2$ da forma:

$$a \longrightarrow b < -d- < c.$$

pode ser lida como:

Um a pode ser lido como uma anáfora c, cujo caso é d, que correfere com alguma categoria à sua esquerda que introduz uma marca de referência b.

Neste formalismo correferência está sujeita às seguintes restrições:

a. Uma categoria anafórica C_1 pode somente correferir com outra categoria C_2 que introduz uma marca de correferência se C_1 não $C-$comanda C_2.

b. Uma categoria anafórica C_1 cujo caso não é "reflex" deve ser interpretada como correferencial com, e somente com, uma categoria C_2, onde C_2 $C-$comanda C_1, e C_2 está dentro do domínio minimal de correferência de C_1.

c. Uma categoria anafórica C_1 cujo caso não é "reflex" deve ser interpretada como não-correferencial com uma categoria $C-$comandante C_2 em seu domínio minimal de correferência.

Estas restrições são formulações abstratas das restrições **(Ib)**, **(II)** e **(III)**:

(Ib) Um pronome deve ser interpretado como não-correferencial com qualquer SN pleno que ele $C-$comande.

(II) Um pronome reflexivo deve ser interpretado como correferencial com, e somente com, um SN $C-$comandante dentro de um domínio sintático especificado.

(III) Um pronome não reflexivo deve ser interpretado como não-correferencial com qualquer SN $C-$comandante no domínio sintático que é especificado para **(II)**.

Com isso a regra $AG2$ anterior deveria ser lida como:

Um a pode ser reescrito como uma anáfora c cujo caso é d, se existir uma categoria à sua esquerda que introduz uma marca de referência b, e se as seguintes restrições forem satisfeitas:

a. a não $C-$comanda a categoria que introduz a marca de referência b.

b. se d é "reflex", então a categoria que introduz a marca de referência b deve $C-$comandar a e estar dentro de seu domínio minimal de correferência.

c. se d não é "reflex", então a categoria que introduz a marca de referência b não pode simultaneamente $C-$comandar a e estar dentro de seu domínio minimal de correferência.

3.2.2 Uma gramática no formalismo

Considere como a gramática do quadro 3.2, analisa a sentença:

```
@ sentence.
@ sub_clause.

sentence                              -->  nounph(nom), verbph.                          (1)
sentence                              -->  nounph(nom), verbph, sub_clause.              (2)
sub_clause                            -->  s_conj, nounph(nom), verbph.                  (3)
s_conj                                -->  [because].                                    (4)
nounph(Case)// ref(X,X=P,Gen)         -->  proper_noun(X,P,Gen),{Case=nom;Case=acc).     (5)
nounph(Case)                          -->  ref(_,_,Gen)<-- Case --< pronoun(Case,Gen)    (6)
proper_noun(X,P,Gen)                  -->  [P], {proper(P,Gen)}.                         (7)
pronoun(Case,Gen)                     -->  [Pr], {pro(Pr,Case,Gen)}.                     (8)
verbph                                -->  verb,nounph(acc).                             (9)
verbph                                -->  verb,nounph(reflex).                          (10)
verb                                  -->  [V], {v(V)}.                                  (11)

/* Lexicon */
proper(fred,masc).
proper(mary,fem).
v(loved).
v(kissed).
pro(he,nom,masc).
pro(him,acc,masc).
pro(himself,reflex,masc).
pro(her,acc,fem).
```

Quadro 3.2

(XXIV) Fred kissed Mary because he loved her.

Quando a regra (5) é chamada para satisfazer a primeira sub-meta na regra (2) (nounph), o token é escrito na pile:

$$(g(1) - 1, ref(X', X' = P', Gen')).$$

Após o substantivo "fred" ser analisado, a variável P' fica instanciada com "fred" e *Gen* com "masc". O token na pile fica:

$$(g(1) - 1, ref(X', X' = \text{fred}, \text{masc})).$$

Quando a regra (5) é chamada novamente para satisfazer a segunda sub-meta da regra (9) (nounph), um segundo token é escrito na pile:

$$(g(1) - 2 - 2, ref(X'', X'' = P'', Gen'')).$$

Após o substantivo "Mary" ser analisado, a variável P'' fica instanciada com "Mary" e Gen'' com "fem", e a pile fica com dois tokens:

$$(g(1) - 2 - 2, ref(X'', X'' = \text{mary}, \text{fem})).$$

$$(g(1) - 1, ref(X', X' = \text{fred}, \text{masc})).$$

Quando o pronome "he" é encontrado, conforme a regra (6), um referente masculino é lido da pile e o pronome "he" é tomado como referente de "fred". O analisador checa automaticamente as restrições em correferência. Quando o pronome "her" é encontrado, conforme a regra (6), um referente feminino é lido da pile e o pronome "her" é tomado como referente de "Mary", e o analisador checa mais uma vez, automaticamente, as restrições. Com isso a análise obteve sucesso e a sentença **(XXIV)** é considerada gramatical.

3.3 Anáfora e Extraposição: Uma gramática $AXG1$

A gramática $AXG1$ do quadro 3.3, implementa as restrições (Ib), (II), (III) e (IV). Esta gramática pode ser dividida em três partes: regras da estrutura de frase, léxico e restrições em correferência. A regra (9), implementa a restrição (IV) ao assegurar que sempre que um "trace" é consumido como um SN, o SN em cujo lugar o "trace" é consumido é $C-$comandado pela categoria que introduziu o "trace".

```
sentence(N)                          -->  nounph(N-1,nom),verb(N-2).                          (1)
sentence(N)                          -->  sub_clause(N-g(1)),nounph(N-2,nom),
                                          verbph(N-3)                                         (2)
sentence(N)                          -->  nounph(N-1,nom),verbph(N-2),
                                          sub_clause(N-g(3)).                                 (3)
sub_clause(N)                        -->  s_conj,nounph(N-1,nom),verbph(N-2).                 (4)
s_conj                               -->  [because]                                           (5)
nounph(N,Case)// ref(N,X,X=P,Gen)    -->  proper_noun(N~1,X,P,Gen,
                                          {Case=nom;Case=acc}.                                (6)
nounph(N,Case)// ref(N,X,P,Gen)      -->  determiner(N-1),noun(N-2,X,P,Gen),
                                          rel_clause(N-3),
                                          {Case=nom;Case=acc}.                                (7)
nounph(N,Case)                       -->  ref(M_,_,Gen)<<<pronoun(N~, Case,Gen),              (8)
                                          {satisfy_constraints(Case,M,N)}.
nounph(N,Case)                       -->  trace(X), {c_commands(M,N)},
                                          {Case\== reflex}.                                   (9)
proper_noun(X,P,Gen)                 -->  [P], {proper(P,Gen)}.                               (10)
determiner(N)                        -->  [D], {det(D)}.                                      (11)
noun(N,X,P,Gen)                      -->  [W], {n(W,X,P,Gen)}.                                (12)
rel_clause(N)                        -->  [ ]                                                 (13)
rel_clause(N)                        -->  rel_pro(N-1),sentence(N-g(2)).                      (14)
pronoun(N, Case,Gen)                 -->  [Pr], {pro(Pr,Case,Gen)}.                           (16)
verbph(N)                            -->  verb(N-1),nounph(N-2,acc).                          (17)
verbph(N)                            -->  verb(N-1),nounph(N-2,reflex).                       (18)
verb(N)                              -->  [V], {v(V)}.                                        (19)
/*Constraints */
satisfy_constraint(Case,A,P)         :-   \+(Case = reflex, \+(c_commands(P,A)),               (20)
                                          min_gov_cat(P,MGC),
                                          \+((c_commands(A,P),dominates(MGC,A))).
satisfy_constraint(Case,A,P)         :-   Case = reflex, (c_commands(A,P)),
                                          \+(c_commands(P,A)),                                (21)
                                          min_gov_cat(P,MGC), dominates(MGC,A))).
dominates(X,X-_).                                                                            (22)
dominates(X,X~_).                                                                            (23)
dominates(X,A-_).                    -->  \+(X=A), dominates(X,A).                           (24)
dominates(X,A~_)                     -->  \+(X=A), dominates(X,A).                           (25)
```

Quadro 3.3

```
c_commands(A,B)                    -->  first_branching_ancestor(A,C),
                                        dominates(C,B).                          (26)

first_branching_ancestor(X-N,X).                                                 (27)

first_branching_ancestor(X^N,Y)    -->  first_branching_ancestor(X,Y).           (28)

min_gov_cat(g(1)-N,g(1))           -->  \+(N=(_-_)),\+(N=(_^_)).                 (29)
min_gov_cat(g(1)^N,g(1))           -->  \+(N=(_^_)),\+(N=(_:-_)).                (30)
min_gov_cat(K-g(L)-N,K-g(L))       -->  \+(N=(_-_)),\+(N=(_-_)).                 (31)
min_gov_cat(K-g(L)^N,K-g(L))       -->  \+(N=(_-_)),\+(N=(_-_)).                 (32)
min_gov_cat(K^g(L)N,K^g(L))        -->  \+(N=(_-_)),\+(N=(_^_)).                 (33)
min_gov_cat(K^g(L)^N,K^g(L))       -->  \+(N=(_^_)),\+(N=(_-_)).                 (34)
min_gov_cat(K-N,MGC)               -->  \+(K=J-g(L), \+(K=J^g(L)),\+(K=g(1)),    (35)
                                        min_gov_cat(K,MGC).

min_gov_cat(K^N,MGC)               -->  \+(K=J^g(L)), \+(K=J-g(L)),\+(K=g(1)),   (36)
                                        min_gov_cat(K,MGC).
```

Quadro 3.3 (Continuação)

Vamos considerar como esta gramática analisa a sentença:

(XXV) The woman who loved Fred kissed him.

Quando a regra (7) é chamada para satisfazer a primeira sub-meta da regra (1) (nounph), o seguinte token é escrito na pile:

$$ref(g(1)-1, X', P', Gen').$$

Após o *SN* "the woman" ser analisado, a variável P' fica instanciada com woman(X') e Gen' com "fem", e o token na pile fica:

$$ref(g(1)-1, X', \text{woman}(X'), \text{fem}).$$

Quando o pronome relativo "who" é encontrado, conforme a regra (15), o vestígio é armazenado na stack:

$$trace(g(1)-1-3-1).$$

Conforme a regra (14), uma sentença deve ser analisada e como não há *SN* dentro da cláusula relativa que pode ser tomado como o sujeito da sentença embutida, conforme a regra (9), este vestígio é consumido. Agora, o teste Prolog da regra (9), chama a regra (26) com a seguinte meta:

$$C - \text{comands}(g(1) - 1 - 3 - 1, g(1) - 1 - 3 - g(2) - 1).$$

que obtém sucesso.

Quando a regra (6) é chamada para satisfazer a segunda sub-meta da regra (17) (nounph), um segundo token é escrito na pile:

$$ref(g(1) - 1 - 3 - g(2) - 2 - 2, X'', X'' = P'', Gen'').$$

Após o substantivo "fred" ser analisado, a variável P'' fica instanciada "fred" e Gen'' com "masc". Com isso a pile contém agora dois tokens:

$$ref(g(1) - 1 - 3 - g(2) - 2 - 2, X'', X'' = \text{fred}, \text{masc})$$

$$ref(g(1) - 1, X', \text{woman}(X'), \text{fem}).$$

Quando o pronome "him" é encontrado, conforme a regra (8), um referente masculino é lido da pile e o pronome "him" é tomado como referente de "fred". Agora, para satisfazer o teste Prolog na regra (8), a regra (20) é chamada com a seguinte meta:

$$\text{satisfy\_constaints}(\text{acc}, g(1) - 1 - 3 - g(2) - 2 - 2, g(1) - 2 - 2).$$

A primeira sub-meta obtém sucesso, pois o caso é "acc"; o segundo também, pois $g(1) - 2 - 2$ não $C-$comando $g(1) - 1 - 3 - g(2) - 2 - 2$; e a terceira sub-meta também obtém sucesso e encontra a categoria minimal de governo de $g(1) - 2 - 2$, que é $g(1)$; e a quarta sub-meta também, pois $g(1) - 1 -$

$3 - g(2) - 2 - 2$, não $C-$comanda $g(1) - 2 - 2$. Com isso a análise obtém sucesso e a sentença **(XXV)** é considerada gramatical.

Porém esta gramática tem suas deficiências.

Ela analisaria a sentença:

(XXVI) The man who fancied himself kissed Mary.

que envolve um pronome reflexivo dentro de uma cláusula relativa e que é perfeitamente gramatical erroneamente.

Isso ocorre pelo fato de que o token que representa "the man" na pile deve conter não somente o nódulo etiqueta que refere à posição em que "the man" foi primeiramente encontrado durante a análise mas também o nódulo etiqueta de todas as posições em que ele foi referido na sentença. Portanto durante a análise, quando um vestígio é consumido, o nódulo etiqueta da *SN* em cujo lugar o vestígio foi consumido, deve também ser armazenado, com o vestígio que representa o *SN*, na pile.

Mas, pronomes reflexivos dentro de cláusulas relativas, não é a única razão para ter uma lista de etiqueta de referência associada com cada marca na pile. A sentença:

(XXVII) Fred called Mary because he hurt himself.

seria erroneamento rejeitada por esta gramática.

Isto seria impedido se o nódulo etiqueta para o "he" anafórico tivesse sido associado com a marca de referência para "fred", pois a correferência entre o "he" anafórico e "himself" satisfaz a restrição (II).

A gramática $AXG1$ apresenta outra deficiência. A sentença:

(XXVIII) Fred fancied Fred.

seria erroneamente aceita como gramatical.

Isto não ocorreria se o tipo de referência, neste caso é plena, tivesse sido associado com o nódulo etiqueta, a gramática estaria apta a rejeitar a sentença **(XXVIII)**, conforme restrição (Ia).

3.4 O formalismo AXG_2 (Anaphora and Extraposition Grammar)

3.4.1 O formalismo

AXG_2 é um formalismo gramatical declarativo. Existem duas classes de entrada numa gramática AXG_2: regras gramaticais e declarações de domínio de correferência.

As regras AXG_2 têm três formas.

A primeira forma é:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots, \beta_n$$

onde α e δ variam sobre V_N e β_i varia sobre $(V_N \cup V_T \cup V_P)$.

A segunda forma é:

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

onde α varia sobre V_N e γ_i varia sobre $(V_T \cup V_N \cup V_P\{\Theta < -\mu- < \phi\})$, onde Θ varia sobre V_N, μ sobre $\{\mathrm{nom}, \mathrm{acc}, \mathrm{reflex}\}$ e ϕ varia sobre $(V_N \cup V_T)$.

A terceira forma é:

$$\alpha \ldots \xi \longrightarrow \beta_1, \ldots, \beta_n$$

onde α varia sobre V_N, β_i varia sobre $(V_N \cup V_T \cup V_P)$ e ξ varia sobre V_N.

A regra $AXG2$:

$$a \,//\, b \longrightarrow c.$$

pode ser lida como:

Um a que introduz uma marca de referência b pode ser reescrito como um c.

A regra $AXG2$:

$$a \longrightarrow b < -d- < c.$$

pode ser lida como:

Um a pode ser reescrito como uma anáfora c, cujo caso é d, que correfere com alguma categoria a sua esquerda que introduz uma marca de referência b.

Uma regra $AXG2$da forma:

$$a \ldots b \longrightarrow c, d.$$

pode ser lida como:

Uma seqüência de categorias consistindo de um a e um vestígio b que está à direita de a, com ambas, correferindo com a categoria minimal dominante que introduz uma marca de referência, caso seja introduzida esta marca da categoria dominante, pode ser reescrita como um c seguido por um d, desde que nem a e nem b interrompam um conjunto de categorias que são reescrito por uma regra similar a esta, exceto quando a está acompanhada por b.

Nesse formalismo, correferência está sujeita às seguintes restrições:

a. Uma categoria C_1 que introduz uma marca de referência deve ser interpretada como não-correferencial com uma categoria C_2 que também introduz uma marca de referência se C_1 $C-comanda$ C_2.

b. Uma categoria anafórica C_1 pode somente correferir com uma categoria C_2 que introduz uma marca de referência se C_1 não C-comanda C_2.

c. Uma categoria anafórica C_1 cujo caso é "reflex" deve ser interpretada como correferencial com e somente com uma categoria $C-$comandante C_2 que está dentro do domínio minimal de correferência de C_1.

d. Uma categoria anafórica C_1 cujo caso não é "reflex" deve ser interpretada como não correferencial com uma categoria $C-$comandante C_2 em seu domínio minimal de correferência.

e. Uma categoria C_1 movida pode somente correferir com uma categoria vazia C_2 se C_1 $C-$comanda C_2.

Como pode ser observado essas restrições são formulações abstratas das seguintes restrições: (Ia), (Ib), (II), (III) e (IV).

3.4.2 Uma gramática no formalismo

Vejamos como a gramática do quadro 3.4 analisa a sentença:

(XXIX) The man who fancied himself kissed mary.

```
@ sentence.
@ sub_clause.

sentence                              -->  nounph(nom), verbph.                          (1)
sentence                              -->  sub_clause, nounph(nom), verbph.              (2)
sentence                              -->  nounph(nom), verbph, sub_clause.              (3)

sub_clause                            -->  s_conj, nounph(nom), verbph.                  (4)

s_conj                                -->  [because]                                     (5)

nounph(Case)// ref(X,X=P,Gen)         -->  proper_noun(X,P,Gen),{Case\=reflex).          (6)
nounph(Case)// ref(X,P,Gen)           -->  determiner,noun(X,P,Gen),rel_clause,          (7)
                                           {Case == reflex}.
nounph(Case)                          -->  ref(_, _Gen) (:- Case:- ( pronoun(Case,Gen).  (8)
nounph(Case)                          -->  trace,{Case\==reflex}.                        (9)

proper_noun(X,P,Gen)                  -->  [P], {proper(P,Gen)}.                         (10)

determiner                            -->  [D], {det(D)}.                                (11)

noun(X,P,Gen)                         -->  [W], {n(W,X,P,Gen)}.                          (12)

rel_clause                            -->  [ ].                                          (13)
rel_clause                            -->  rel_pro, sentence.                            (14)

rel_pro... trace                      -->  [who]                                         (15)

pronoun(Case,Gen)                     -->  [Pr], {pro(Pr,Case,Gen)}.                     (16)

verbph                                -->  verb,nounph(acc).                             (17)
verbph                                -->  verb,nounph(reflex).                          (18)

verb                                  -->  [V], {v(V)}.                                  (19)

/* Lexicon */
det(the).
n(man,X,man(X),masc).
n(woman,X,woman(X),fem).
proper(fred,masc).
proper(mary,fem).
v(fancied).
v(loved).
v(kissed).
v(killed).
pro(he,nom,masc).
pro(him,acc,masc).
pro(himself,reflex,masc).
pro(her,acc,fem).
```

Quadro 3.4

Quando a regra gramatical (7) é chamada para satisfazer a primeira sub-meta na regra gramatical (1), o seguinte token é escrito na pile:

$$\Big([(g(1)-1,f)], ref(X', P', Gen')\Big).$$

Após o substantivo "man" ter sido analizado, a variável P' fica instanciada com "man(X')" e a variável Gen' fica instanciada com "masc"; portanto o token na pile torna-se:

$$\Big([(g(1)-1,f)], ref(X', \text{man}(X', \text{masc})\Big).$$

Quando o pronome relativo "who" é encontrado, de acordo com a regra gramatical (15), o seguinte token que é um vestígio é armazenado na stack:

$$(g(1)-1-3-1, g(1)-1, trace).$$

Agora, de acordo com a regra gramatical (14), uma sentença deve ser analisada. Desde que não há SN dentro da cláusula relativa para ser tomada como o sujeito da sentença, de acordo com a regra gramatical (9), um vestígio é consumido como um sintagma nominal. A análise então, automaticamente, faz duas coisas: primeiro, ela se assegura que a restrição (IV) seja satisfeita, isto é, que o sintagma nominal "the man" $C-$comande o "trace"; segundo, ela associa este novo nódulo etiqueta com a marca de referência para "the man" na pile. O token na pile se torna:

$$\Big([(g(1)-1-3-g(2)-1,t),(g(1)-1,f)], ref(X', \text{man}(X'), \text{masc})\Big).$$

Quando o pronome "himself" é encontrado, de acordo com a regra gramatical (8), um referente masculino é lido na pile e o pronome "himself" é tomado como se referindo à "the man". Além disso, o analisador checa para se assegurar que as restrições em correferência foram todas satisfeitas. O domínio mínimo de correferência do pronome, que é $g(1)-1-3-g(2)$, não

domina o antecedente, ou pelo menos o vestígio referente a ele, no nódulo $g(1) - 1 - 3 - g(2) - 1$.

Quando a regra gramatical (6) é chamada para satisfazer a segunda sub-meta na regra gramatical (17), que é a sub-meta "nounph", o seguinte token é escrito na pile:

$$\Big([(g(1) - 2 - 2, f)], ref(X'', X'' = P'', Gen'')\Big).$$

Após o substantivo próprio "mary" ter sido analisado a variável P'' fica instanciada com "mary" e a variável "Gen" fica instanciada com "fem", portanto o token na pile se torna:

$$\Big([(g(1) - 2 - 2, f)], ref(X'', X'' = \text{mary}, \text{fem})\Big).$$

Com isso terminamos, a análise com sucesso e a sentença é gramatical.

3.5 O formalismo *EXG2* (Endophora and Extraposition Grammar)

O formalismo *EXG2* é um formalismo gramatical declarativo que automaticamente implementa as restrições em catáfora, assim como, em anáfora e em extraposição. Existem duas classes de entrada numa gramática *EXG2*: regras gramaticais e declarações do domínio de correferência. As regras *EXG2* têm três formas.

A primeira forma é:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots \beta_n$$

onde α e δ variam sobre V_N e β_i sobre $(V_N \cup V_T \cup V_p)$.

A segunda forma é:

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

onde α varia sobre V_N e γ_i sobre $(V_T \cup V_N \cup V_p \cup \{\Theta \ /- \ \mu \ -/ \ \phi\}$, onde Θ varia sobre $(V_N \cup V_T)$, μ sobre {nom,acc,reflex} e ϕ sobre V_N, sendo nom o caso nominativo, acc o caso acusativo e reflex o caso reflexivo.

A terceira forma é:

$$\alpha \ldots \xi \longrightarrow \beta_1, \ldots, \beta_n$$

onde α e ξ variam sobre V_N e β_i sobre $(V_N \cup V_T \cup V_p)$.

A declaração de domínio de conferência tem a forma:

$$@\alpha$$

onde α varia sobre V_N.

Estas declarações metagramaticais são usadas quando restrições envolvendo a noção de domínio mínimo de correferência são introduzidos nos analisadores que o compilador gera das gramáticas no formalismo.

Uma regra $EXG2$ da forma

$$a \ // \ b \longrightarrow c.$$

pode ser lida como segue:

Um a que introduz uma marca de referência b pode ser reescrito como um c.

Uma regra $EXG2$ da forma

$$a \longrightarrow c \ /- \ d \ -/ \ b.$$

pode ser lida como:

Um a pode ser reescrito como uma endofora c, cujo caso é d, que correfere com alguma categoria (que poderia estar a sua direita ou a sua esquerda) que introduz uma marca de referência b.

Uma regra $EXG2$ da forma

$$a \ldots b \longrightarrow c, d.$$

pode ser lida como:

Uma sequência de categorias consistindo de um a e um vestígio b que está em algum lugar à direita do a, com ambas correferindo com a categoria dominante mínima que introduz uma marca de referência, se houve de fato uma marca de referência introduzida por uma categoria dominante, pode ser reescrito como um c seguido por um d desde que nem a nem b interrompam um conjunto de categorias que são reescrito por uma regra similar a esta, exceto quando a estiver acompanhado por b.

Neste formalismo a correferência está sujeita às seguintes restrições:

- Uma categoria C_1 que introduz uma marca de referência deve ser interpretada como não-correferencial a uma categoria C_2 que também introduz uma marca de referência se C_1 $C-comanda$ C_2.

- Uma categoria anafórica C_1 pode somente correferir com uma categoria C_2 que introduz uma marca de referência se C_1 não C-comanda C_2.

- Uma categoria anafórica C_1 cujo caso é "reflex" deve ser interpretada como correferencial com e somente com uma categoria $C-$comandante C_2 que está dentro do domínio mínimo de correferência de C_1.

- Uma categoria anafórica C_1 cujo caso não é "reflex" deve ser interpretada como não correferencial com uma categoria $C - comandante\ C_2$ em seu domínio mínimo de correferência.

- Uma categoria C_1 movida pode somente correferir com uma categoria vazia C_2 se $C_1\ C - comanda\ C_2$.

Como pode ser observado essas restrições são formulações abstratas das seguintes restrições:

(Ia) Um *SN* pleno deve ser interpretado como não-correferencial com outro *SN* pleno que ele $C-$comande.

(Ib) Um pronome deve ser interpretado como não-correferencial com outro *SN* pleno que ele $C-$comande.

(II) Um pronome reflexivo deve ser interpretado como correferencial com, e somente com, um *SN* $C-$comandante dentro de um domínio sintático especificado.

(III) Um pronome não-reflexivo deve ser interpretado como não-correferencial com outro *SN* $C-$comandante no domínio sintático que é especificado para (II).

(IV) Um constituinte movido deve $C-$comandar seu vestígio[2].

Uma regra *EXG2* da forma:

$$a\ //\ b \longrightarrow c.$$

[2]A restrição (IV) corresponde à restrição (VI) que se encontra na página 97 de CARVALHO [1989].

é compilada na seguinte regra *VPBSG*:

$$
\begin{aligned}
a(N) \quad \longrightarrow \quad & \langle \text{write\_pile}\,([[N, f)], b))\rangle, C\,(N\text{-}1), \langle \text{remove\_pile}\,((\text{List1}, b))\rangle, \\
& \{\text{is\_in}\,(\{N, f)\,, \text{List1})\}, \text{check\_pile2}\,(\text{List1}, \text{List2}, b), \\
& \text{check\_bag2}\,(\text{List2}, \text{List}, b), \langle \text{write\_pile}\,(\text{List}, b)\rangle.
\end{aligned}
$$

onde N− é usado para conectar nódulos que se ramificam e N⁻ para conectar nódulos que não se ramificam. Esta regra tem o seguinte significado:

Quando reescrevemos um símbolo não-terminal de classe a como um símbolo de classe c, escrevemos uma marca de referência, b, com uma lista contendo uma etiqueta de referência[3] associada a ela na Pile. O primeiro argumento dessa etiqueta de referência é o nódulo etiqueta de a e o segundo argumento é "f"[4] (seu tipo).

Neste ponto do processamento, se b tem argumentos, alguns deles (ou todos) serão variáveis não instanciadas. Quando terminar a análise de c, alguns (ou todos) os argumentos de b estarão instanciados. Removemos o b que foi introduzido na Pile, com sua lista de etiquetas de referência associada, e checamos a Pile para um outro b. Se não conseguirmos encontrar outro b, significa que b não foi referida previamente.

Se encontrarmos outro b, significa que esta entidade foi previamente referida. Portanto removemos este b prévio, com sua lista de etiquetas de referência, da Pile. Então devemos estar seguros de que toda referência prévia

[3] O termo etiqueta de referência é usado para designar uma estrutura com dois argumentos: o primeiro sendo o nódulo etiqueta da referência; e o segundo o seu tipo de referência, onde são usados para checar restrições em correferência.

[4] O tipo de referência pode assumir um dos três casos:

i. "f" (full): completo;

ii. "t" (trace): vestígio; ou

iii. "e" (endophora): endófora.

"f" a b não c-comande a última referência. Devemos assegurar também que o oposto seja verdadeiro, isto é, a última referência "f" a b não c-comande a referência prévia "f" a b. Se esta restrição é satisfeita, acrescentamos esta nova lista de etiquetas de referência para b à antiga lista, e checamos a Bag. Se não podemos remover um pedido para um referente b da Bag, devemos assegurar que qualquer nódulo etiqueta associado com o pedido satisfaça a restrição em correferência.

Após a restrição ser satisfeita, escrevemos b de volta na Pile, com a nova lista de etiquetas de referência, que consiste da lista de etiquetas de referência antes da Bag ser checada mais a nova lista para toda posição em que b foi encontrado como o referente para um pedido na Bag. Estas novas etiquetas de referência consistirão de um nódulo etiqueta e um tipo "e".

Uma regra $EXG2$ da forma:

$$a \longrightarrow c \,/\!-\, d \,-\!/\, b.$$

é compilada na seguinte regra $VPBSG$:

$$a(N) \longrightarrow C\,(N\hat{}1)\,,\mathrm{check\_pile\_or\_bag}\,(N,d,b)\,.$$

com o seguinte significado:

Quando reescrevemos um símbolo não-terminal de classe a como um símbolo de classe c, tentamos remover uma marca de referência, b, com a lista de etiquetas de referências correspondente da Pile. Esta lista pode conter um único elemento, no caso de b ter sido referido uma única vez, ou mais que um elemento, no caso de b ter sido referido mais de uma vez. Para cada vez que b foi referido, haverá uma etiqueta de referência correspondente. Então devemos assegurar que as restrições em correferência sejam satisfeitas. A

restrição (Ib) é satisfeita ao assegurar que toda referência prévia "f" a b não seja c-comandada por a.

A restrição (II) é satisfeita quando c é "reflex" e as seguintes condições sejam satisfeitas: exista uma referência a b que c-comande a, e esta referência a b esteja dentro do domínio mínimo de correferência de a.

A restrição (III) é satisfeita quando c não é "reflex", então as referências prévias "f" a b não são referências que c-comande a e, ao mesmo tempo, estejam dentro do domínio mínimo de correferência.

Após as restrições serem satisfeitas, escrevemos b de volta na Pile com uma nova lista de nódulo etiqueta, com a cabeça da lista sendo uma etiqueta de referência cujo primeiro argumento é o nódulo etiqueta de a (isto é, a mais recente referência a b), e cujo segundo argumento é "e", e o corpo da lista sendo a lista prévia de etiquetas de referência para b.

Se as restrições em correferência não podem ser satisfeitas, removemos um pedido para b da Bag (se já existir um pedido), com sua lista de nódulos etiqueta correspondente. Esta lista pode conter um único elemento, no caso de b ter sido referido uma única vez, ou mais de um elemento, no caso de b ter sido referido mais de uma vez. Antes de escrever o pedido para um referente b de volta na Pile, devemos assegurar que este novo pedido satisfaça as restrições em correferência (II) e (III). Caso isto aconteça, escrevemos o pedido para um referente b de volta na Bag com uma nova lista de nódulos etiqueta, com a cabeça da lista sendo o nódulo etiqueta de a (isto é, o mais recente pedido para um referente b) e corpo da lista sendo a lista prévia de nódulos etiqueta para b. Se não houver um pedido prévio para um referente b na Bag, escrevemos um pedido para um referente, com o nódulo etiqueta de a e o caso de c associado com ele, na Bag.

```
a(N)                          -->  c(N-1), d(N-2), min_dom_cat_ref(N, M),
                                   ⟨push((N, M, b)) ⟩.
a(N)                          -->  c(N-1), d(N-2), ⟨push((N, b)) ⟩.
min_dom_cat_ref(x-_, M)       -->  mdcr2(X, M).
min_dom_cat_ref(x~_, M)       -->  mdcr2(X, M).
mdcr2(X, X)                   -->  ⟨read_pile(L,R)⟩, {is_in((X, f), L)}.
mdcr2(Y-_, X)                 -->  mdcr2(Y, M).
mdcr2(Y~_, X)                 -->  mdcr2(Y, M).
is_in(X,[X|_]).
is_in(X,[_|T])                -->  is_in(X, T).
```

Quadro 3.5

Uma regra como:

$$a \ldots b \longrightarrow c, d.$$

é compilada usando as regras *VPBSG* do quadro 3.5.

Com o seguinte significado:

Quando reescrevemos um símbolo não terminal de classe a como um símbolo de classe c, seguido por um d, devemos armazenar na Stack um b associado com o nódulo etiqueta de a. Além disso, se existir qualquer categoria dominante que introduz uma marca de referência, então o nódulo etiqueta da categoria de domínio mínimo que introduz uma marca de referência deve também ser associado com b na Stack.

Uma definição metagramatical tal como:

$$@a$$

daria origem a uma regra da forma:

$$a \longrightarrow c, b.$$

que seria compilada na seguinte regra *VPBSG*:

$$a(g(N)) \longrightarrow c(g(N)-1), b(g(N)-2).$$

com o seguinte significado:

Quando reescrevemos um símbolo não-terminal de classe a, como um símbolo não-terminal de classe c, seguido por um símbolo não-terminal de classe b, o nódulo etiqueta de a será $g(N)$, estabelecendo que este nódulo é um nódulo de domínio de correferência.

Assumindo a implementação descendente da esquerda para a direita, apresentaremos a gramática $EXG2$.

3.5.1 Uma gramática no formalismo

Vejamos como a gramática do quadro 3.6 da p. 114 analisa a sentença.

(XXX) Because Fred encouraged him tom succeeded.

Quando a regra gramatical (6) é chamada para satisfazer a segunda sub-meta na regra gramatical (4), isto é, a sub-meta nounph, o seguinte token é escrito na pile

([g(1)-g(1)-2,f)],ref(X',X'=P',Gen')).

Após o substantivo próprio "fred" ter sido analisado, a variável P' fica instanciada com "fred" e a variável Gen' com "masc".

O token na pile fica:

([g(1)-g(1)-2,f)],ref(X',X'=fred,masc)).

Quando o pronome "him" é encontrado, de acordo com a regra gramatical (8), um referente masculino é lido da pile e o pronome "him" é tomado

```
@ sentence.
@ sub_clause.
sentence                                --> nounph(nom), verbph.                          (1)
sentence                                --> sub_clause, nounph(nom), verbph.              (2)
sentence                                --> nounph(nom), verbph, sub clause.              (3)
sub_clause                              --> s_conj, nounph(nom), verbph.                  (4)
s_conj                                  --> [because]                                     (5)
nounph(Case) // ref(X,X=P,Gen)          --> proper_noun(X,P,Gen),
                                            {Case=nom;Case=acc}.                          (6)
nounph(Case) // ref(X,P,Gen)            --> determiner,noun(X,P,Gen),rel_clause,          (7)
                                            {Case=nom;Case=acc}.
nounph(Case)                            --> pro(Case,Gen) /- Case -/ ref(_,_,Gen).        (8)
nounph(Case)                            --> trace(M),{Case\==reflex}.                     (9)
proper_noun(X,P,Gen)                    --> [P], {proper(P,Gen)}.                         (10)
determiner                              --> [D], {det(D)}.                                (11)
noun(X,P,Gen)                           --> [W], {n(W,X,P,Gen)}.                          (12)
rel_clause                              --> [ ].                                          (13)
rel_clause                              --> rel_pro(N-1), sentence(N-g(2)).               (14)
rel_pro...trace                         --> [who]                                         (15)
pro(Case,Gen)                           --> [Pr], {pro(Pr,Case,Gen)}.                     (16)
verbph                                  --> t_verb,nounph(acc).                           (17)
verbph                                  --> t_verb,nounph(reflex).                        (18)
verbph                                  --> i_verb.                                       (19)
t_verb                                  --> [V], {vt(V)}.                                 (20)
i_verb                                  --> [V], {vi(V)}.                                 (21)
/* Lexicon */
proper(fred,masc).
proper(tom,masc).
vt(encouraged).
vi(succeeded).
vi(persevered).
pro(he,nom,masc).
pro(him,acc,masc).
pro(himself,reflex,masc).
```

Quadro 3.6

como referente de "fred". Contudo, devido ao fato de que "fred" está dentro do domínio mínimo de correferência do pronome, ele é rejeitado como um referente pela reflexividade.

Com isso, um pedido para um referente masculino é escrito na *bag*. A *bag* agora contém o seguinte token:

([g(1)-g(1)-3-2,acc)],ref(_,_,masc)).

Quando a regra gramatical (6) é chamada para satisfazer a segunda sub-meta na regra gramatical (2), isto é, a sub-meta nounph, outro token é escrito na pile, com duas variáveis não instanciadas, P'' e Gen''; após o substantivo próprio "tom" ter sido analisado, a variável P'' fica instanciada com "tom" e a variável Gen'' com "masc". Os dois tokens na pile são os seguintes:

([g(1)-2,f)],ref(X'',X''=tom,masc))

([g(1)-1-2,f)],ref(X',X'=fred,masc)).

Então um pedido para um referente masculino é lido da *bag* e após as restrições em correferência serem checadas, o pronome "him" é tomado como referente para "tom". Após o verbo "succeeded" ter sido analisado, pois não há pedidos sem resolução na *bag*, a sentença é considerada gramatical.

Considere como a gramática rejeitaria a seguinte sentença:

(XXXI) He encouraged Fred.

Quando o pronome "He' é encontrado, de acordo com a regra gramatical (8), um token, que é um pedido para um referente masculino é escrito na bag. O token é da forma:

([g(1)-1,nom)],ref(_,_,masc)).

Quando o substantivo próprio é encontrado, de acordo com a regra gramatical (6), um pedido para um referente masculino é removido da *bag* e o pronome "He" é tomado como referente para "fred". Entretanto, devido ao fato de que o pronome "He" não pode $C-$Comandar seu antecedente (de acordo com a restrição (Ib)), a análise falha e a sentença "He encouraged Fred" é considerada agramatical.

Resumo

Neste capítulo Carvalho apresentou uma série de formalismos gramaticais onde algumas restrições em correferência foram implementadas pelo método automático-explícito. Iniciou com o formalismo gramatical $AG2$ que manipula anáfora e as restrições em correferência (Ib), (II) e (III); logo após o formalismo gramatical $AXG2$ que manipula extraposição, anáfora e as restrições em correferência (Ia) (Ib), (II), (III) e (IV). Finalmente desenvolveu o formalismo gramatical $EXG2$ que manipula endófora e extraposição, assim como as restrições em correferência do formalismo gramatical $AXG2$.

Capítulo 4

Ampliando o conceito de categoria mínima de regência

Porém, convém salientar que estes formalismos não conseguem analisar sentenças perfeitamente gramaticais como **(XXXII)**:

(XXXII) John expects that pictures of himself will be on sale.

em virtude do conceito empregado aqui da categoria mínima de regência de um nódulo A, que é a minimal S ou SN dominando A.

Isto ocorreria porque teríamos que dizer que "himself" é correferente com "pictures", pois a categoria de regência de "himself" seria o SN "pictures of himself"; logo "himself" não poderia ser correferente com "John" e teria de encontrar um correferente em sua própria SN, isto é, "pictures of himself", e como o único sintagma nominal diferente de "himself" é "pictures", "himself" seria considerado correferente com ele, violando com isso o filtro do i-sobre-i.

A proposta para contornar este problema é ampliar o conceito de categoria mínima de regência de um nódulo A, que passa a ser a primeira

categoria maior SN ou S que domina A, um regente de A e um SUJEITO accessível a A.

Com isso a análise da sentença **(XXXII)** ficaria da seguinte forma:

Como vimos "himself" não pode ser coindexado com "pictures", pois haveria a violação do filtro do i-sobre-i, e portanto "pictures of himself" não é um SUJEITO accessível a "himself". Como o regente de "himself" é a preposição "of" e seu SUJEITO accessível é o elemento de concordância da matriz, a categoria mínima de regência de "himself" é a sentença toda, passando com isso "himself" e "John" a serem correferentes, apesar de estarem em sentenças diferentes.

O formalismo gramatical que se encontra a seguir,é uma proposta de ampliação do formalismo gramatical *EXG2*, repetido aqui com as ampliações necessárias, para que o mesmo possa lidar com sentenças do tipo **(XXXII)**, onde aparecem sentenças encaixadas.

4.1 O formalismo *EXG3* (Endophora and Extraposition Grammar)

O formalismo *EXG3* é um formalismo gramatical declarativo que automaticamente implementa as restrições em catáfora, assim como, em anáfora e em extraposição. Existem duas classes de entrada numa gramática *EXG3*: regras gramaticais e declarações do domínio de correferência. As regras gramaticais do formalismo *EXG3* têm três formas.

A primeira forma é:

$$\alpha \,//\, \delta \longrightarrow \beta_1, \ldots \beta_n$$

onde α e δ variam sobre V_N e β_i sobre $(V_N \cup V_T \cup V_p)$.

A segunda forma é:

$$\alpha \longrightarrow \gamma_1, \ldots, \gamma_n$$

onde α varia sobre V_N e γ_i sobre $(V_T \cup V_N \cup V_p \cup \{\Theta \ /- \ \mu \ -/ \ \phi\}$, onde Θ varia sobre $(V_N \cup V_T)$, μ sobre {nom,acc,reflex} e ϕ sobre V_N, sendo nom o caso nominativo, acc o caso acusativo e reflex o caso reflexivo.

A terceira forma é:

$$\alpha \ldots \xi \longrightarrow \beta_1, \ldots, \beta_n$$

onde α e ξ variam sobre V_N e β_i sobre $(V_N \cup V_T \cup V_p)$.

As declarações de domínio de conferência têm as formas:

$$@\alpha \ \text{ e } \ \#\alpha$$

onde α varia sobre V_N.

Estas declarações metagramaticais são usadas quando restrições envolvendo a noção de domínio mínimo de correferência são introduzidos nos analisadores que o compilador gera das gramáticas no formalismo.

Uma regra $EXG3$ da forma

$$a \ // \ b \longrightarrow c.$$

pode ser lida como segue:

Um a que introduz uma marca de referência b pode ser reescrito como um c.

Uma regra $EXG3$ da forma

$$a \longrightarrow c \ /- \ d \ -/ \ b.$$

pode ser lida como:

Um a pode ser reescrito como uma endofora c, cujo caso é d, que correfere com alguma categoria (que poderia estar a sua direita ou a sua esquerda) que introduz uma marca de referência b.

Uma regra $EXG3$ da forma

$$a \ldots b \longrightarrow c, d.$$

pode ser lida como:

Uma sequência de categorias consistindo de um a e um vestígio b que está em algum lugar à direita do a, com ambas correferindo com a categoria dominante mínima que introduz uma marca de referência, se houve de fato uma marca de referência introduzida por uma categoria dominante, pode ser reescrito como um c seguido por um d desde que nem a nem b interrompam um conjunto de categorias que são reescrito por uma regra similar a esta, exceto quando a estiver acompanhado por b.

Neste formalismo gramatical a correferência está sujeita às mesmas restrições do formalismo gramatical $EXG2$, repetidos aqui:

- Uma categoria C_1 que introduz uma marca de referência deve ser interpretada como não-correferencial a uma categoria C_2 que também introduz uma marca de referência se C_1 $C-comanda$ C_2.

- Uma categoria anafórica C_1 pode somente correferir com uma categoria C_2 que introduz uma marca de referência se C_1 não C-comanda C_2.

- Uma categoria anafórica C_1 cujo caso é "reflex" deve ser interpretada como correferencial com e somente com uma categoria $C-$comandante C_2 que está dentro do domínio mínimo de correferência de C_1.

- Uma categoria anafórica C_1 cujo caso não é "reflex" deve ser interpretada como não correferencial com uma categoria C – *comandante* C_2 em seu domínio mínimo de correferência.

- Uma categoria C_1 movida pode somente correferir com uma categoria vazia C_2 se C_1 C – *comanda* C_2.

Como pode ser observado essas restrições são formulações abstratas das seguintes restrições:

(Ia) Um *SN* pleno deve ser interpretado como não-correferencial com outro *SN* pleno que ele C–comande.

(Ib) Um pronome deve ser interpretado como não-correferencial com outro *SN* pleno que ele C–comande.

(II) Um pronome reflexivo deve ser interpretado como correferencial com, e somente com, um *SN* C–comandante dentro de um domínio sintático especificado.

(III) Um pronome não-reflexivo deve ser interpretado como não-correferencial com outro *SN* C–comandante no domínio sintático que é especificado para (II).

(IV) Um constituinte movido deve C–comandar seu vestígio[1].

[1]A restrição (IV) corresponde à restrição (VI) que se encontra na página 97 de CARVALHO [1989].

4.1.1 Uma implementação para o formalismo

Uma regra *EXG*3 da forma[2]:

$$a \,//\, b \;\longrightarrow\; c.$$

é compilada na seguinte regra *VPBSG*:

$$\begin{aligned} a(N) \;\longrightarrow\; & \langle \text{write\_pile}\,([[(N,f)],b))\rangle,\ C\,(N\tilde{}1),\ \langle \text{remove\_pile}\,((\text{List1},b))\rangle, \\ & \{\text{is\_in}\,((N,f),\text{List1})\},\ \text{check\_pile2}\,(\text{List1},\text{List2},b), \\ & \text{check\_bag2}\,(\text{List2},\text{List},b),\ \langle \text{write\_pile}\,(\text{List},b)\rangle. \end{aligned}$$

com as regras extras fornecidas pelo compilador Prolog, como no quadro 4.1.

O significado desta regra, de acordo com o quadro 4.1, é o seguinte:

Quando reescrevemos um símbolo não-terminal de classe a como um símbolo de classe c, escrevemos uma marca de referência, b, com uma lista contendo uma etiqueta de referência[3] associada a ela na Pile. O primeiro argumento dessa etiqueta de referência é o nódulo etiqueta de a e o segundo argumento é "f"[4] (seu tipo).

Neste ponto do processamento, se b tem argumentos, alguns deles (ou todos) serão variáveis não instanciadas. Quando terminar a análise de c, alguns (ou todos) os argumentos de b estarão instanciados. Removemos o b

[2]onde N− é usado para conectar nódulos que se ramificam e N~ para conectar nódulos que não se ramificam.

[3]O termo etiqueta de referência é usado para designar uma estrutura com dois argumentos: o primeiro sendo o nódulo etiqueta da referência; e o segundo o seu tipo de referência, onde são usados para checar restrições em correferência.

[4]O tipo de referência pode assumir um dos três casos:

i. "f" (full): completo;

ii. "t" (trace): vestígio; ou

iii. "e" (endophora): endófora.

```
check_pile2(NewRefs, NewRefs, X)                --> ⟨absent_pile((_,X))⟩, {!}.
check_pile2(NewRefs, Refs, X)                   --> ⟨remove_pile((OldRefs, X))⟩,
                                                    {check_type1(NewRefs, OldRefs),
                                                    append(NewRefs, OldRef, Refs)}.

check_type1([],_).
check_type1([H|T], Old)                         :-  check_type2(H, Old), check_type1( T, Old).
check_type2((N, T), Old)                        :-  \+(T=f).
check_type2((N, f), [H|T])                      :-  check_type3((N, f), H), check_type2((N, f), T).
check_type3((N1, f), (N2, T))                   :-  \+(T=f).
check_type3((N1, f), (N2, f))                   :-  \+(c-commands(N1, N2)), \+(c-commands(N2, N1)
check_bag2(List2, List, X)                      --> ⟨remove_bag((List3, X))⟩,
                                                    {sat_cons(List2, List3, List2, List)}.

check_bag2(List, List, X)                       --> [].
sat_cons([], [], F, F).
sat_cons(Old, [(N,K)|R], Inter, Final)          :-  satisfy_constraints( K, N, Old),
                                                    sat_cons( Old, R, [(N,e)|Inter], Final).

satisfy_constraints(reflex,P,Alist)             :-  cons1b(P,Alist), cons2(P,Alist).
satisfy_constraints(Case,P,Alist)               :-  \+(Case=reflex),cons1b(P,Alist), cons3(P,Alist).
satisfy_constraints1(reflex,P,Alist)            :-  cons2(P,Alist).
satisfy_constraints1(Case,P,Alist)              :-  \+(case=reflex), cons3(P,Alist).
cons1b(P,Alist)                                 :-  \ +((is_in((A,f),Alist), c-commands(P,A))).
cons2(P,Alist)                                  :-  min_cor_dom(p,MCD), is_in(A,Alist),
                                                    c-commands(A,P), dominates(MCD,A).

cons3(P,Alist)                                  :-  min_cor_dom(P,MCD), \ +(is_in(A,Alist),
                                                    c-commands(A,P), dominates(MCD,A).

dominates(X,X-_).
dominates(X,X~_).
dominates(X,A-_).                               :-  \+(X=A), dominates(X,A).
dominates(X,A~_)                                :-  \+(X=A), dominates(X,A).
c-commands(A,B)                                 :-  first_branching_ancestor(A,C),
                                                    dominates(C,B).

first_branching_ancestor(X-N,X).
first_branching_ancestor(X~N,Y)                 :-  first_branching_ancestor(X,Y).
min_cor_dom(K-g(L)-g(L)-N-M, K-g(L)-g(L)-1)     :-  ((K-g(L)-1)=e(P)), ((K-g(L)-g(L)-N-M~)=reflex),
                                                    c-commands(K-g(L)-g(L)-1,K-g(L)-g(L)-N-M)
min_cor_dom(K-g(L)-g(L)-N, A)                   :-  ((K-g(L)-1)=e(P)), c-commands(A,K-g(1)).
min_cor_dom(g(1)-N,g(1))                        :-  \+(N=(_-_)), \+(N=(_~_)).
min_cor_dom(g(1)~N,g(1))                        :-  \+(N=(_~_)),\-(N=(_-_)).
min_cor_dom(K-g(L)-N,K-g(L))                    :-  \+(N=(_-_)),\+(N=(_-_)).
min_cor_dom(K-g(L)~N,K-g(L))                    :-  \+(N=(_-_)), \+(N=(_-_)).
min_cor_dom(K~g(L)N,K~g(L))                     :-  \+(N=(_-_)), \+(N=(_~_)).
min_cor_dom(K~g(L)~N,K~g(L))                    :-  \+(N=(_~_)), \+(N=(_-_)).
min_cor_dom(K-N,MCD)                            :-  \+(K=J-g(L), \+(K=J~g(L)), \+(K=g(1)),
                                                    min_cor_dom(K,MCD).
min_cor_dom(K~N,MCD)                            :-  \+(K=J~g(L)), \+(K=J-g(L)),\+(K=g(1)),
                                                    min_cor_dom(K,MCD).

is_in(X,[X|_]).
is_in(X,[_|T])                                  :-  is_in(X,T).
```

Quadro 4.1

que foi introduzido na Pile, com sua lista de etiquetas de referência associada, e checamos a Pile para um outro b. Se não conseguirmos encontrar outro b, significa que b não foi referida previamente.

Se encontrarmos outro b, significa que esta entidade foi previamente referida. Portanto removemos este b prévio, com sua lista de etiquetas de referência, da Pile. Então devemos estar seguros de que toda referência prévia "f" a b não c-comande a última referência. Devemos assegurar também que o oposto seja verdadeiro, isto é, a última referência "f" a b não c-comande a referência prévia "f" a b. Se esta restrição é satisfeita, acrescentamos esta nova lista de etiquetas de referência para b à antiga lista, e checamos a Bag. Se não podemos remover um pedido para um referente b da Bag, devemos assegurar que qualquer nódulo etiqueta associado com o pedido satisfaça a restrição em correferência.

Após a restrição ser satisfeita, escrevemos b de volta na Pile, com a nova lista de etiquetas de referência, que consiste da lista de etiquetas de referência antes da Bag ser checada mais a nova lista para toda posição em que b foi encontrado como o referente para um pedido na Bag. Estas novas etiquetas de referência consistirão de um nódulo etiqueta e um tipo "e".

Uma regra *EXG3* da forma:

$$a \longrightarrow c \,/\!-\, d \,-\!/\, b.$$

é compilada na seguinte regra *VPBSG*:

$$a(N) \longrightarrow C\,(N\tilde{}1)\,,\mathrm{check\_pile\_or\_bag}\,(N,d,b)\,.$$

com as regras extras fornecidas pelo compilador Prolog, como no quadro 4.2.

O significado desta regra, de acordo com o quadro 4.2, é o seguinte:

```
check_pile_or_bag(N,K,X)                          -->  check_pile3(N,K,X).
check_pile_or_bag(N,K,X)                          -->  check_bag3(N,K,X).
check_pile3(N,K,X)                                -->  ⟨remove_pile((List,X))⟩,
                                                       {satisfy_constraints(K,N,List)},
                                                       ⟨write_pile(([(N,k)—List],X))⟩.
check_bag3(N,K,X)                                 -->  ⟨remove_bag((List,X))⟩,
                                                       {satisfy_constraints1(K,N,List)},
                                                       ⟨write_bag(([(N,K)—List],X))⟩.
check_bag3(N,K,X)                                 -->  ⟨write_bag(([(N,K)—List],X))⟩.
satisfy_constraints(reflex,P,Alist)               :-   cons1b(P,Alist), cons2(P,Alist).
satisfy_constraints(Case,P,Alist)                 :-   \ +(case=reflex),cons1b(P,Alist), cons3(P,Alist).
satisfy_constraints1(reflex,P,Alist)              :-   cons2(P,Alist).
satisfy_constraints1(Case,P,Alist)                :-   \+(case=reflex), cons3(P,Alist).
cons1b(P,Alist)                                   :-   \ +((is_in((A,f),Alist), c-commands(P,A))).
cons2(P,Alist)                                    :-   min_cor_dom(p,MCD), is_in(A,Alist),
                                                       c-commands(A,P), dominates(MCD,A).
cons3(P,Alist)                                    :-   min_cor_dom(P,MCD), \ +(is_in(A,Alist),
                                                       c-commands(A,P), dominates(MCD,A).

dominates(X,X-_).
dominates(X,X~_).
dominates(X,A-_).                                 :-   \+(X=A), dominates(X,A).
dominates(X,A~_)                                  :-   \+(X=A), dominates(X,A).
c-commands(A,B)                                   :-   first_branching_ancestor(A,C),
                                                       dominates(C,B).

first_branching_ancestor(X-N,X).
first_branching_ancestor(X~N,Y)                   :-   first_branching_ancestor(X,Y).
min_cor_dom(K-g(L)-g(L)-N-M, K-g(L)-g(L)-1)       :-   ((K-g(L)-1)=e(P)), ((K-g(L)-g(L)-N-M~)=reflex),
                                                       c-commands(K-g(L)-g(L)-1,K-g(L)-g(L)-N-M)
min_cor_dom(K-g(L)-g(L)-N, A)                     :-   ((K-g(L)-1)=e(P)), c-commands(A,K-g(1)).
min_cor_dom(g(1)-N,g(1))                          :-   \+(N=(_-_)), \+(N=(_~_)).
min_cor_dom(g(1)~N,g(1))                          :-   \+(N=(_~_)),\+(N=(_-_)).
min_cor_dom(K-g(L)-N,K-g(L))                      :-   \+(N=(_-_)),\+(N=(_-_)).
min_cor_dom(K-g(L)~N,K-g(L))                      :-   \+(N=(_-_)), \+(N=(_-_)).
min_cor_dom(K~g(L)N,K~g(L))                       :-   \+(N=(_-_)), \+(N=(_~_)).
min_cor_dom(K~g(L)~N,K~g(L))                      :-   \+(N=(_~_)), \+(N=(_-_)).
min_cor_dom(K-N,MCD)                              :-   \+(K=J-g(L), \+(K=J~g(L)), \+(K=g(1)),
                                                       min_cor_dom(K,MCD).
min_cor_dom(K~N,MCD)                              :-   \+(K=J~g(L)), \+(K=J-g(L)),\+(K=g(1)),
                                                       min_cor_dom(K,MCD).

is_in(X,[X|_]).
is_in(X,[_|T])                                    :-   is_in(X,T).
```

Quadro 4.2

Quando reescrevemos um símbolo não-terminal de classe a como um símbolo de classe c, tentamos remover uma marca de referência, b, com a lista de etiquetas de referências correspondente da Pile. Esta lista pode conter um único elemento, no caso de b ter sido referido uma única vez, ou mais que um elemento, no caso de b ter sido referido mais de uma vez. Para cada vez que b foi referido, haverá uma etiqueta de referência correspondente. Então devemos assegurar que as restrições em correferência sejam satisfeitas. A restrição (Ib) é satisfeita ao assegurar que toda referência prévia "f" a b não seja c-comandada por a.

A restrição (II) é satisfeita quando c é "reflex" e as seguintes condições sejam satisfeitas: exista uma referência a b que c-comande a, e esta referência a b esteja dentro do domínio mínimo de correferência de a.

A restrição (III) é satisfeita quando c não é "reflex", então as referências prévias "f" a b não são referências que c-comande a e, ao mesmo tempo, estejam dentro do domínio mínimo de correferência.

Após as restrições serem satisfeitas, escrevemos b de volta na Pile com uma nova lista de nódulo etiqueta, com a cabeça da lista sendo uma etiqueta de referência cujo primeiro argumento é o nódulo etiqueta de a (isto é, a mais recente referência a b), e cujo segundo argumento é "e", e o corpo da lista sendo a lista prévia de etiquetas de referência para b.

Se as restrições em correferência não podem ser satisfeitas, removemos um pedido para b da Bag (se já existir um pedido), com sua lista de nódulos etiqueta correspondente. Esta lista pode conter um único elemento, no caso de b ter sido referido uma única vez, ou mais de um elemento, no caso de b ter sido referido mais de uma vez. Antes de escrever o pedido para um referente b de volta na Pile, devemos assegurar que este novo pedido satisfaça

```
a(N)                          -->  c(N-1), d(N-2), min_dom_cat_ref(N, M),
                                   ⟨push((N, M, b)) ⟩.
a(N)                          -->  c(N-1), d(N-2), ⟨push((N, b)) ⟩.
min_dom_cat_ref(x-_, M)       -->  mdcr2(X, M).
min_dom_cat_ref(x~_, M)       -->  mdcr2(X, M).
mdcr2(X, X)                   -->  ⟨read_pile(L,R)⟩, {is_in((X, f), L)}.
mdcr2(Y-_, X)                 -->  mdcr2(Y, M).
mdcr2(Y~_, X)                 -->  mdcr2(Y, M).
is_in(X,[X|_]).
is_in(X,[_| T])               :-   is_in(X, T).
```

Quadro 4.3

as restrições em correferência (II) e (III). Caso isto aconteça, escrevemos o pedido para um referente b de volta na Bag com uma nova lista de nódulos etiqueta, com a cabeça da lista sendo o nódulo etiqueta de a (isto é, o mais recente pedido para um referente b) e corpo da lista sendo a lista prévia de nódulos etiqueta para b. Se não houver um pedido prévio para um referente b na Bag, escrevemos um pedido para um referente, com o nódulo etiqueta de a e o caso de c associado com ele, na Bag.

Uma regra como:

$$a \ldots b \longrightarrow c, d.$$

é compilada usando as regras *VPBSG* do quadro 4.3.

Com o seguinte significado:

Quando reescrevemos um símbolo não terminal de classe a como um símbolo de classe c, seguido por um d, devemos armazenar na Stack um b associado com o nódulo etiqueta de a. Além disso, se existir qualquer categoria dominante que introduz uma marca de referência, então o nódulo etiqueta da categoria de domínio mínimo que introduz uma marca de referência deve também ser associado com b na Stack.

1=1 1=1

Uma definição metagramatical tal como:

$$@a$$

daria origem a uma regra da forma:

$$a \longrightarrow c, b.$$

que seria compilada na seguinte regra *VPBSG*:

$$a(g(N)) \longrightarrow c(g(N) - 1), b(g(N) - 2).$$

com o seguinte significado:

Quando reescrevemos um símbolo não-terminal de classe a, como um símbolo não-terminal de classe c, seguido por um símbolo não-terminal de classe b, o nódulo etiqueta de a será $g(N)$, estabelecendo que este nódulo é um nódulo de domínio de correferência.

Uma definição metagramatical tal como:

$$\#\mathrm{a}$$

daria origem a uma regra da forma:

$$\mathrm{a} \longrightarrow b.$$

que seria compilada na seguinte regra *VPBSG*:

$$\mathrm{a}(e(N)) \longrightarrow b(e(N)\text{-}1).$$

que teria o seguinte significado:

Quando reescrevemos um símbolo não-terminal de classe a, como um símbolo não-terminal de classe b o nódulo etiqueta de a será $e(N)$, estabelecendo que este nódulo é um nódulo de domínio de correferência. Este nódulo serve como identificação das sentenças encaixadas, possibilitando com isso, o controle da categoria mínima de regência das mesmas.

4.1.2 Uma gramática no formalismo

O quadro 4.5 que se encontra na p. 131, é uma proposta de ampliação da gramática *EXG*2, que figura no quadro 4.4 que se encontra na p. 130, para que a mesma possa analisar sentenças do tipo **(XXXII)**.

Note tamém que estas modificações analisam com sucesso sentenças gramaticais do tipo:

(XXXIII) Fred disse que Tom se matou.

onde o pronome reflexivo tem que encontrar seu correferente dentro da própria sentença encaixada.

```
@ sentence.
@ sub_clause.
sentence                              -->  nounph(nom), verbph.                          (1)
sentence                              -->  sub_clause, nounph(nom), verbph.              (2)
sentence                              -->  nounph(nom), verbph, sub clause.              (3)
sub_clause                            -->  s_conj, nounph(nom), verbph.                  (4)
s_conj                                -->  [because]                                     (5)
nounph(Case) // ref(X,X=P,Gen)        -->  proper_noun(X,P,Gen),
                                           {Case=nom;Case=acc}.                          (6)
nounph(Case) // ref(X,P,Gen)          -->  determiner,noun(X,P,Gen),rel_clause,          (7)
                                           {Case=nom;Case=acc}.
nounph(Case)                          -->  pro(Case,Gen) /- Case -/ ref(_,_,Gen).        (8)
nounph(Case)                          -->  trace(M),{Case\==reflex}.                     (9)
proper_noun(X,P,Gen)                  -->  [P], {proper(P,Gen)}.                         (10)
determiner                            -->  [D], {det(D)}.                                (11)
noun(X,P,Gen)                         -->  [W], {n(W,X,P,Gen)}.                          (12)
rel_clause                            -->  [ ].                                          (13)
rel_clause                            -->  rel_pro(N-1), sentence(N-g(2)).               (14)
rel_pro...trace                       -->  [who]                                         (15)
pro(Case,Gen)                         -->  [Pr], {pro(Pr,Case,Gen)}.                     (16)
verbph                                -->  t_verb,nounph(acc).                           (17)
verbph                                -->  t_verb,nounph(reflex).                        (18)
verbph                                -->  i_verb.                                       (19)
t_verb                                -->  [V], {vt(V)}.                                 (20)
i_verb                                -->  [V], {vi(V)}.                                 (21)
```

Quadro 4.4

```
#comp.

sentence                              -->  comp, sentence.
nounph(Case) // ref(X,X=P,Gen)        -->  proper_noun(X,P,Gen), sint_prep,
                                           {Case=nom;Case=acc}.
nounph(Case) // ref(X,P,Gen)          -->  determiner, noun(X,P,Gen), rel_clause,
                                           sint_prep, {Case=nom;Case=acc}.
nounph(Case)                          -->  pro(Case,Gen) /- Case -/ ref(_,_,Gen),
                                           sint_prep.
sint_prep                             -->  prepos, nounph(nom).
prepos                                -->  [P], {prep(P)}.
verbph                                -->  t_verb, sentence.
verbph                                -->  l_verb, nounph(nom).
comp                                  -->  [that].
comp                                  -->  [que].
verbph                                -->  aux, l_verb, sint_prep.
aux                                   -->  [will].
l_verb                                -->  [V], {vl(V)}.

/* Lexicon */
proper(fred,masc).
proper(tom,masc).
nom(pictures,X,pictures(X),neuter).
nom(sale,X,sale(X),neuter).
vt(encouraged).
vt(expects).
vt(disse).
vt(matou).
vi(succeeded).
vi(persevered).
vl(be).
vl(é).
pro(he,nom,masc).
pro(him,acc,masc).
pro(himself,reflex,masc).
pro(ele,nom,masc).
pro(se,reflex,neuter).
prep(of).
prep(...).
```

Quadro 4.5

Outra observação a ser feita, diz respeito à restrição (Ib), repetida aqui:

(Ib) Um pronome deve ser interpretado como não-correferencial com outro *SN* pleno que ele c-comande.

Note que esta restrição só é válida se as sentenças que forem análisadas pelo formalismo gramatical *EXG2*, não usarem verbos de ligação, por exemplo, a sentença:

(XXXIV) Ele é Fred.

que é perfeitamente gramatical, seria analisada pelo formalismo gramatical *EXG2* como agramatical, pelo simples fato de que este formalismo não foi elaborado para tratar das sentenças que usam verbos de ligação, da mesma forma que não tratou das sentenças que usam sentenças encaixadas, como a sentença **(XXXII)**.

Como trabalho futuro, qualquer formalismo gramatical que venha a manipular anáforas, deve de levar em consideração, além das restrições propostas neste trabalho, sentenças que usem verbos de ligação (que não foram tratadas aqui), e as restrições da lingüística que surgirem devido ao avanço da mesma.

Bibliografia

ABRAMSON, H.

[1984] *Defininte Clause Translation Grammars.* Proceedings IEEE Logic Programming Symposium, pp. 233–240.

CARVALHO, A. M. B. R.,

[1989] *Logic Grammars and Pronominal Anaphora. PhD Thesis.* University of Reading.

CHOMSKY, N. A.,

[1957] *Syntactic Structures.* Haia, Mouton.

[1965] *Aspects of the Theory of Syntax.* Cambridge, Mass., The MIT Press.

[1981] *Lectures on Government and Binding.* Dordrecht, Holanda, Foris.

CLOCKSIN, W. F., MELLISH, C. S.

[1987] *Programming in Prolog.* Heidelberg, Springer Verlag.

COLMERAUER, A.

[1978] *Metamorphosis Grammar.* Em L. BOLC, *Natural Language Communication with Computers.* Heidelberg, Springer Verlag. pp. 133–189.

DAHL, V.

[1984] *More on Gapping Grammars.* Proceedings International Conference on Fifth Generation Computers Systems, pp. 669–677.

DAHL, V.,ABRAMSON, H.

[1984] *On Gapping Grammars.* Proceedings of the 2nd. Logic Programming Conference, Uppsala Univ., Sweden.

DAHL, V., MCCORD, M.

[1983] "Treating Coordination in Logic Grammars", *American Journal of Computational Linguistics*, vol. IX, no. 2, pp. 69–91.

DAHL, V., SAINT-DIZIER, P.

[1986] *Constrained Discontinuous Grammars.* Rappors de Recherche 309, IRISA.

HIRSCHMAN, L., PUDER, K.

[1986] *Restriction Grammar: A Prolog Implementation.* Em: M. VAN CANEGAN and D. H. D. WARREN, Eds., *Logic Programming and Applications.* Ablex Publishing Corporation, pp. 245–261.

JOHNSON, M., KLEIN, E.

[1986] *Discourse, Anaphora and Parsing.* 11th. International Conference on Computational Linguistics (COLING'86), pp. 669–675.

KOWALSKI, R. A.

[1974] *Predicate Logic as a Programming Language. Proceedings IFIP 74*, pp. 569–574.

LASNIK, H., URIAGEREKA, J.

[1988] *A Course in GB Syntax. Lectures on Binding and Empty Categories.* Cambridge, Massachusetts, The MIT Press.

LOBATO, L. M. P.

[1986] *Sintaxe Gerativa do Português; da teoria padrão à teoria da regência e ligação.* Belo Horizonte, Vigília.

MAY, R.

[1985] *Logical Form. Its Structure and Derivation.* Cambridge, Massachusetts, The MIT Press.

PEREIRA, F. C. N.

[1981] "Extraposition Grammars", *American Journal of computational Linguistics*, vol. VII, no. 4, pp. 243–256.

PEREIRA, F. C. N., SHIEBER, S. M.

[1987] *Prolog and Natural-Language Analysis.* Stanford, CSLI.

PEREIRA, F. C. N., WARREN, D. H. D.

[1980] "Definite Clause Grammars for Language Analysis", *Artificial Intelligence*, vol. XIII, pp. 231–278.

RADFORD, A.

[1988] *Transformational Grammars. A first Course.* Cambridge, Cambridge University Press.

STABLER, E. JR.

[1986] *Restricting Logic Grammars.* Fifth National Conference on Artificial Intelligence (AAAI-86), pp. 1048–1052.

[1987] "Restricting Logic Grammars with Government-Binding Theory", *Computational Linguistics*, vol. XIII, no. 1–2, pp. 1–10.

Índice de figuras

Índice de Quadros