ALENCAR, Leonel Figueiredo. Sintaxe formal de línguas não configuracionais num ambiente computacional: o caso do latim. *Revista Virtual de Estudos da Linguagem – ReVEL*. Vol. 6, n. 10, março de 2008. ISSN 1678-8931 [www.revel.inf.br].

SINTAXE FORMAL DE LÍNGUAS NÃO CONFIGURACIONAIS NUM AMBIENTE COMPUTACIONAL: O CASO DO LATIM

Leonel Figueiredo de Alencar[1]

prof_leonel-sintaxe@yahoo.com.br

RESUMO: Neste trabalho, apresentamos uma análise formal de aspectos da sintaxe do latim, apontando algumas das dificuldades que línguas não configuracionais prototípicas como essa representam para um tratamento computacional no formalismo DCG na sua versão clássica. Propomos um formalismo alternativo que explora recursos avançados do SWI-Prolog para ampliar o poder expressivo da DCG. Nesse formato mais eficiente, construímos um fragmento de gramática gerativa do latim que dá conta da ordem livre de constituintes em frases com uma variada gama de verbos divalentes e trivalentes que seguem tanto padrões canônicos quanto não canônicos de caso e vinculação argumental. Para tanto, recorremos a duas regras gerais que formalizam a checagem de caso e a concordância, de modo que apenas duas regras de estrutura sintagmática são necessárias para gerar frases com verbos de dois e três lugares. Na DCG tradicional, ao contrário, precisa-se de centenas de regras para modelar o mesmo conjunto de frases. Por outro lado, graças ao algoritmo de parsing já embutido em Prolog, nosso fragmento de gramática não só reconhece e analisa frases com sujeitos nulos tanto expletivos quanto temáticos, mas também realiza o mapeamento entre estrutura morfossintática e estrutura semântica. Dessa maneira, é capaz de construir representações diferentes para frases ambíguas que resultam de mais de uma possibilidade de mapeamento sintático-semântico, explicitando as diferentes maneiras pelas quais a estrutura argumental do verbo é vinculada aos argumentos sintáticos na frase.
PALAVRAS-CHAVE: latim; não-configuracionalidade; interface léxico-sintaxe; Prolog.

INTRODUÇÃO

Uma característica geral a todos os modelos gerativos é a utilização, em maior ou menor grau, de uma linguagem formal, i.e. matematicamente explícita, para descrever línguas naturais individuais e formular hipóteses mais gerais sobre a linguagem humana. A principal tarefa dessa segunda linha de investigação é, por um

[1] Professor do Programa de Pós-Graduação em Lingüística e do Departamento de Letras Estrangeiras da Universidade Federal do Ceará.

lado, determinar, por meio da comparação entre línguas, o que não varia de uma língua para outra, i.e. os universais lingüísticos. Por outro, trata-se de fazer um levantamento dos tipos de variação, de modo a determinar os parâmetros da Gramática Universal (doravante UG, do inglês *Universal Grammar*) e seus possíveis valores.

Desse modo, a gramática gerativa constitui um dos fundamentos da lingüística computacional. Um sistema de processamento da linguagem natural baseado na UG oferece vantagens com relação a um sistema baseado apenas num subconjunto das línguas humanas. Com efeito, um modelo computacional da UG fornece uma base comum para a implementação das gramáticas de todas as línguas. A implementação da gramática de uma língua particular é, desse modo, facilitada enormemente, uma vez que basta especificar as instanciações de parâmetros dessa língua.

A relação entre gramática gerativa e lingüística computacional é, no entanto, uma via de mão dupla, pois a implementação computacional de um modelo formal de um fenômeno lingüístico extremamente complexo pode revelar inconsistências e lacunas que dificilmente seriam descobertas de outro modo, como ressaltam Beesley e Karttunen (2003:311).

Na lingüística computacional, a formalização deve ser completa, no sentido de que todos os resultados da investigação sobre um fenômeno devem ser expressos numa linguagem artificial. Por outro lado, um modelo meramente descritivo, por mais formal que seja, é insuficiente para habilitar um computador ao processamento da linguagem natural. É preciso dispor de um algoritmo que determine, passo a passo, como a máquina deve utilizar as informações constantes de uma gramática gerativa para realizar uma tarefa específica como reconhecer se uma determinada construção é gramatical ou não.

Um dos parâmetros de variação da UG mais importantes é o da configuracionalidade. Línguas como português, inglês etc. têm uma organização sintática essencialmente configuracional. Nesse tipo, as relações gramaticais (sujeito, objeto direto etc.) são identificadas principalmente por meio de posições na estrutura sintagmática:

(1) a. The queen will praise the poem. 'A rainha louvará o poema.'
b. The poem will praise the queen. 'O poema louvará a rainha.'

Um grupo de línguas numericamente importante, contudo, adota uma organização não configuracional. Nesse caso, a ordem das palavras é em grande parte

livre, ao passo que as relações gramaticais são identificadas por marcas nos núcleos e/ou nos elementos dependentes desses núcleos. No latim, ocorrem os dois tipos de marcas:

(2) a. Reginam carmina laudabunt. 'Os poemas louvarão a rainha.'
b. Regina carmen laudabit. 'A rainha louvará o poema.'
c. Scriptum carmina laudabunt 'Os poemas louvarão o escrito.'

No exemplo (2 a), as funções gramaticais de sujeito e objeto direto são identificadas tanto pelas flexões casuais, respectivamente, de nominativo e acusativo nos substantivos (os elementos dependentes), quanto pela flexão de pessoa e número no verbo (o núcleo, no caso), i.e. pela concordância verbal. Em (2 b), são as marcas de caso que permitem realizar um mapeamento não ambíguo entre funções sintáticas e estrutura argumental. Em (2 c), esse papel é desempenhado pela concordância.

As línguas não configuracionais representam um enorme desafio para a construção de programas de análise sintática automática, embora possam parecer, numa primeira análise superficial, mais fáceis de formalizar. Aparentemente, uma simples instrução do tipo "uma frase em latim constitui-se de um verbo acompanhado de zero, um ou mais de um sintagma nominal" seria capaz de produzir um importante fragmento dessa língua. No entanto, essa facilidade é apenas aparente, como veremos, pois é preciso dar conta de fenômenos como concordância e regência e do mapeamento entre sintaxe e semântica. Como a estrutura da frase latina é flexível, as relações entre núcleo e dependentes, como exemplificadas em (2), não são locais, como ocorre na maioria das vezes nas línguas configuracionais (cf. (1)).

Neste artigo, realizamos uma análise totalmente formalizada de aspectos da sintaxe latina, utilizando a linguagem de programação Prolog. Desenvolvemos incrementalmente um fragmento de gramática que, em sua versão final, é capaz de lidar com frases com verbos de diferentes valências, tanto com sujeito pleno quanto nulo, e com qualquer permutação entre o verbo e seus argumentos. Por outro lado, a gramática produz estruturas de constituintes distintas para frases estruturalmente ambíguas e realiza o mapeamento entre estruturas sintáticas e semânticas, explicitando as diferentes leituras de uma frase que decorrem de associações distintas entre funções sintáticas e estruturas argumentais.

O formalismo que elaboramos é uma extensão da Gramática de Cláusulas Definidas (doravante DCG, do inglês *Definite Clause Grammar*) em sua versão clássica, tal como amplamente difundida na lingüística computacional. Ao contrário desse modelo tradicional, que precisa de dezenas de regras para tratar da ordem livre de

constituintes, nossa proposta se vale de um número extremamente reduzido de regras, graças, sobretudo, à implementação de dois princípios sintáticos gerais: a concordância verbo-nominal e a checagem de caso entre um núcleo V e seus complementos.

O artigo está estruturado em quatro seções. Inicialmente, mostramos as dificuldades que a formalização da sintaxe de línguas não configuracionais como o latim representa para a gramática livre de contexto. Esse tipo de gramática constitui o esqueleto não só da tradicional DCG, mas também de sofisticados formalismos lingüísticos computacionalmente implementados, como a LFG e HPSG. Em seguida, mostramos as limitações da implementação de uma gramática do latim na DCG clássica. Na seção 3, apresentamos um fragmento de gramática latina em Prolog que supera essas deficiências. Esse fragmento, contudo, apenas reconhece as frases gramaticais, mas não é capaz de gerar representações que mostrem em que frases estruturalmente ambíguas diferem. Essa limitação é superada na seção 4, onde fazemos alterações na gramática de modo a capacitá-la para gerar representações sintáticas e semânticas. Isso se torna possível graças à introdução, no programa, de princípios de mapeamento entre estruturas morfossintáticas e estruturas argumentais.

1. MODELAÇÃO DE FRAGMENTO DO LATIM NO FORMALISMO CFG

Os principais sistemas de processamento computacional da sintaxe das línguas naturais derivam da gramática de estrutura sintagmática (doravante PSG, do inglês *phrase structure grammar*), que resultou da formalização, realizada por Chomsky na década de 50, do modelo estruturalista da gramática de estrutura de constituintes imediatos. A PSG comporta vários subtipos, cada um dos quais gera um tipo de língua formal. Esses diferentes tipos de línguas foram organizados por Chomsky numa hierarquia, conhecida hoje não só na lingüística, mas também, e principalmente, na informática teórica, como Hierarquia de Chomsky (Martín-Vide, 2004:159-160).

(3) $L_3 \subset L_2 \subset L_1 \subset L_0$

A cada um desses tipos de língua formal corresponde um tipo de gramática: Tipo 3, 2, 1 e 0. A relação entre os diferentes tipos de línguas é de inclusão. Desse modo, uma língua do Tipo 2 pode ser gerada por uma gramática do Tipo 2, 1 ou 0, mas não por uma gramática do Tipo 3.

Chomsky mostrou que a sintaxe das línguas naturais não pode ser descrita adequadamente por uma gramática do Tipo 3, também conhecido como classe das gramáticas regulares, que gera apenas línguas do Tipo 3. As línguas do Tipo 2 são geradas pelas gramáticas livre de contexto (doravante CFG, do inglês *context-free grammar*). Uma CFG pode dar conta de pelo menos algumas das construções sintáticas mais importantes do inglês. Esse formalismo, contudo, é bastante limitado quando se trata de modelar fenômenos como concordância e regência. Para superar essas limitações da CFG, Chomsky propôs regras de transformação, que são operações que incidem sobre o produto de uma CFG, que são árvores sintagmáticas, gerando novas árvores (Sag, Wasow e Bender, 2003:40-41).

Na lingüística computacional, essa solução de Chomsky para as limitações da CFG nunca teve muita aceitação, porque gramáticas transformacionais são do Tipo 0. Está matematicamente provado que, para qualquer gramática do Tipo 3, 2 ou 1, pode-se construir um algoritmo capaz de determinar se uma cadeia qualquer pertencente à língua gerada pela gramática é gramatical ou não. Essa prova não existe, contudo, para as gramáticas do Tipo 0 (Klenk, 2003:79). Em vez de gramáticas transformacionais, vários modelos gerativos alternativos foram propostos que tratam de fenômenos como concordância e regência por meio da operação matemática de unificação de estruturas de traços, entre os quais sobressaem, hoje em dia, a LFG e a HPSG. A CFG é, nesses modelos, apenas um esqueleto que é enriquecido com estruturas de traços. [2]

A primeira implementação computacional importante da CFG foi feita no formalismo DCG, desenvolvido no início da década de 80. Esse modelo formal tem sido utilizado ultimamente no Brasil por vários lingüistas para implementar computacionalmente gramáticas do português que tomam como ponto de partida uma descrição nos moldes da CFG.[3] Como sempre se ressalta na literatura, o formalismo DCG não consiste, porém, de uma simples tradução do modelo CFG, uma vez que na DCG podemos utilizar traços para dar conta, por meio da unificação, de fenômenos como concordância e regência. Na CFG isso só é possível de maneira extremamente pouco prática (Sag, Wasow e Bender, 2003:37-40). Computacionalmente, a DCG transcende a CFG, pois pode dar conta de línguas que não são do Tipo 2, como, por

[2] Sobre a dificuldade de implementação computacional de gramáticas transformacionais e as vantagens de modelos gerativos alternativos, ver, por ex., Falk (2001), Sag, Bender e Wasow (2003) e Kaplan (2004).

[3] Ver, entre outros, David (2007), Othero (2006) e Pagani (2004).

exemplo, a língua $L = \{a^i b^i a^i \mid i > 0\}$ (Klabunde, 2004:77). Conforme Blackburn, Bos e Striegnitz (2001:117), DCGs podem modelar qualquer tipo de língua computável.

Como a DCG é somente um formato mais prático para a descrição da sintaxe das línguas naturais em Prolog, e essa linguagem de programação é computacionalmente completa (Baader, 1999), parece não ser possível, a priori, determinar os limites do poder expressivo desse formalismo para a descrição dos fenômenos gramaticais das línguas naturais. A DCG que, por conveniência, chamarei de "clássica", no entanto, usada em autores como David (2007), Othero (2006) e Pagani (2004), tem limites bastante fáceis de serem estabelecidos. Isso fica claro, sobretudo, quando se aplica esse formalismo na descrição de línguas não configuracionais, de que o latim é exemplo prototípico.

Como vimos, a ordem dos constituintes imediatos da frase, em latim, é livre. Pelo critério da valência, entendida na acepção mais próxima da química, de onde se origina (o que os autores alemães chamam de *Wertigkeit*), podemos, conforme Weidmann (2007), distinguir as classes de verbos em latim (os parênteses indicam a realização facultativa do sujeito) do Quadro 1.

Ainda segundo Weidmann, a valência especifica a quantidade mínima de complementos de que se precisa para produzir uma frase gramatical com um determinado verbo. Como o latim é uma língua de sujeito nulo, os verbos de valência maior que zero podem ocorrer em frases com um complemento a menos do que especificado na valência, no caso, o sujeito, que pode ser expresso apenas pela morfologia verbal.

Tipo de verbo	Exemplo
avalentes	pluit 'chove'
monovalentes	(nos) ridemus '(nós) rimos'
divalentes	(ego) amo te '(eu) te amo'
trivalentes	(ego) tibi librum do '(eu) te dou um livro'

Quadro 1: Classes de verbos quanto à valência

Levando em conta a subcategorização verbal, podemos distinguir, com base na classificação acima, respectivamente quatro classes de verbos: impessoais, intransitivos, transitivos *stricto sensu* e bitransitivos. As frases abaixo exemplificam esses padrões:

(4) a. Pluit. 'Chove.'
b. Poeta ridet. 'O poeta ri.'
c. Poeta reginam amat. 'O poeta ama a rainha.'

d. Poeta reginae rosam dat. 'O poeta dá uma rosa à rainha.'

Essas classes verbais são as mais básicas, por instanciarem as associações canônicas entre argumentos verbais, molduras casuais e funções sintáticas. No Quadro 2, apresentamos essas associações. A quarta coluna classifica os verbos, com base na LFG, em termos de molduras funcionais, que consistem no conjunto de funções sintáticas subcategorizadas (Falk, 2001). OBJ corresponde geralmente ao objeto direto da gramática tradicional e OBJ2, o objeto secundário, ao objeto indireto.

Valência	**Classe verbal**	**Moldura casual**	**Moldura Funcional**
0	impessoal	[]	< >
1	intransitivo	[nom]	<SUBJ>
2	transitivo	[nom,acc]	<SUBJ,OBJ>
3	bitransitivo	[nom,dat,acc]	<SUBJ,OBJ2,OBJ>

Quadro 2: Associações canônicas entre valência, classe verbal, moldura casual e funcional

Uma CFG no formato clássico capaz de gerar o fragmento do latim constituído por todas e somente as frases gramaticais possíveis de se construir com verbos dessas quatro classes na terceira pessoa do singular, não levando em conta frases com omissão do sujeito nem com adjuntos, deve conter, por exemplo, as seguintes regras, que obedecem à notação do *Natural Language Toolkit* – NLTK (Bird, Klein e Loper, 2007). Nesse fragmento bem como nas gramáticas das seções seguintes, adotamos a teoria da LFG, segundo a qual línguas não configuracionais como malayalam são desprovidas das categorias IP e VP (Falk, 2001:50). Nesse modelo, S é uma categoria exocêntrica não projetiva .

(5) CFG de um fragmento do latim
S → V0 | SUBJ V1 | V1 SUBJ | SUBJ OBJ V2 | SUBJ V2 OBJ | OBJ V2 SUBJ | OBJ SUBJ V2 | V2 SUBJ OBJ | V2 OBJ SUBJ
V0 -> "pluit" V1-> "ridet" V2 -> "amat"
SUBJ -> "poeta" OBJ -> "reginam"

Além das frases (a – c) de (4), essa gramática reconhece as seguintes, onde se permuta a ordem dos constituintes:

(6) Ridet poeta. Reginam poeta amat. Reginam amat poeta.
Poeta amat reginam. Amat reginam poeta. Amat poeta reginam.

Por outro lado, as frases agramaticais de (7) não são reconhecidas:

(7) *Poeta pluit. *Poeta ridet reginam. *Poeta amat.
*Poeta poeta amat. *Reginam reginam amat.

O fragmento de (5) dá conta de todas as permutações possíveis entre os constituintes de frases com os verbos de valência 0, 1 e 2 do Quadro 1. Além das 9 regras de (5), contudo, são necessárias mais 24 para que todas as permutações de SUBJ, V, OBJ e OBJ2 em frases com verbos bitransitivos do tipo de (4 d) sejam geradas. Para dar conta da omissão do sujeito como (8), precisaríamos incluir ainda mais regras na gramática.

(8) a. Reginam amat.
b. Reginae rosam dat.

Em (9), mostramos que regras precisariam ser incluídas para que a gramática pudesse reconhecer frases com verbos divalentes e sujeito nulo:

(9) Regras frases com verbos divalentes e sujeito nulo
S -> OBJ V2 S -> V2 OBJ

A Tabela 1 mostra quantas regras são necessárias para cada classe valencial. O cálculo da quantidade de regras por classe valencial é feito com base na fórmula da permutação simples da análise combinatória, que é $P_n=n!$. Em latim, a quantidade de permutações entre o verbo e os argumentos verbais é determinada pela fórmula (v+1)!, onde v é a valência verbal (com a restrição v > 0).

Valência verbal	**Quantidade de regras**
0	1
1	2!=2 x 1=2
2	3!=3 x 2 x 1=6
3	4!= 4 x 3 x 2 x 1=24
Total	1+2+6+24=33

Tabela 1: Relação entre valência verbal e quantidade de regras em uma CFG.

2. UM FRAGMENTO DO LATIM NA DCG CLÁSSICA

Uma DCG, no formato que estamos chamando de clássico, capaz de reconhecer e gerar não só as frases do fragmento exemplificado em (4) e (6), mas também frases gramaticais com as demais formas verbais, poderia ter, em linhas gerais, a forma de (10). Nesse fragmento e em todos os demais deste trabalho, que estão disponíveis em nosso site,[4] não levamos em conta os traços de gênero dos substantivos, uma vez que esse aspecto não releva diretamente aos fenômenos que escolhemos para formalizar.

(10) Fragmento da sintaxe latina na DCG clássica

[4] URL **http://www.geocities.com/Athens/Crete/1546/latim/**.

```
s --> v([_,_,val(0)]). % frases com verbos impessoais
% frases com verbos monovalentes
s --> np([cas(nom),pers(X),num(Y)]),v([pers(X),num(Y),val(1)]).
s --> v([pers(X),num(Y),val(1)]), np([cas(nom),pers(X),num(Y)]).
s --> np([cas(nom),pers(X),num(Y)]), % frases com verbos divalentes
np([cas(acc),pers(_),num(_)]),v([pers(X),num(Y),val(2)]).
s --> np([cas(nom),pers(X),num(Y)]),
v([pers(X),num(Y),val(2)]),np([cas(acc),pers(_),num(_)]).
s --> np([cas(acc),pers(_),num(_)]),
v([pers(X),num(Y),val(2)]),np([cas(nom),pers(X),num(Y)]).
% mais 3 permutações de SUBJ, OBJ e V
s --> np([cas(nom),pers(X),num(Y)]), % frases com verbos trivalentes
np([cas(dat),pers(_),num(_)]),np([cas(acc),pers(_),num(_)]),
v([pers(X),num(Y),val(3)]).
s -->  v([pers(X),num(Y),val(3)]),np([cas(nom),pers(X),num(Y)]),
np([cas(dat),pers(_),num(_)]),np([cas(acc),pers(_),num(_)]),
% mais 22 permutações de SUBJ, OBJ2, OBJ e V
```

A regra do NP com substantivo como núcleo tem o formato de (11). Assumimos que, como o latim não tem artigo, não possui a categoria funcional D (Falk, 2001:50-51), responsável, em línguas com artigo como o português, inglês e o alemão, pelo traço de pessoa.

(11) Regra do sintagma nominal com núcleo substantivo

```
np([cas(X),pers(3),num(Y)]) --> n([cas(X),num(Y)]).
```

Note que, conforme a regra (11), o traço de pessoa é uma propriedade do NP e não do N, que não é marcado para pessoa. Em línguas que possuem a categoria D, o traço de pessoa não é inerente ao NP. Em vez disso, é projetado pela categoria funcional (cf. Radford, 1997:71,97).

No caso de NP com núcleo pronominal, o traço de pessoa é projetado em latim a partir do pronome:

(12) Regra do sintagma nominal pronominal

```
np([cas(X),pers(Z),num(Y)]) --> pron([cas(X),pers(Z),num(Y)]).
```

A seguir, damos exemplos de entradas lexicais:

(13) Fragmento de léxico do latim

```
n([cas(nom),num(sg)]) --> [poeta]; [regina].
n([cas(acc),num(sg)]) --> [poetam]; [reginam].
pron([cas(nom),pers(1),num(sg)]) --> [ego].
pron([cas(nom),pers(1),num(pl)]) --> [nos].
```

```
pron([cas(acc),pers(1),num(pl)]) --> [nos].
v([pers(3),num(sg),val(0)]) --> [pluit].
v([pers(3),num(sg),val(1)]) --> [ridet].
v([pers(1),num(sg),val(2)]) --> [amo].
v([pers(2),num(sg),val(2)]) --> [amas].
v([pers(3),num(sg),val(3)]) --> [dat].
```

Em comparação com uma gramática no formato CFG clássico, a gramática no formato DCG acima, graças à utilização dos traços de pessoa, número, caso e valência, representa um enorme ganho de economia. Mesmo assim, precisamos de um número ainda excessivo de regras.

Outra complicação advém da marcação não canônica de caso. Na gramática acima, os verbos de valência 2 e os de valência 3 subcategorizam, respectivamente, as molduras casuais [nom,acc] e [nom,dat,acc], como vimos no Quadro 2. Em línguas acusativas como latim, grego clássico e alemão, esse é o padrão para a maioria dos verbos dessas duas classes valenciais. Em todas essas línguas, contudo, há verbos divalentes e trivalentes com molduras casuais desviantes.[5] Por exemplo, o verbo latino *parere* 'obedecer' exige um objeto no dativo, assim como seu equivalente em alemão *gehorchen*:

(14) a. Poeta reginae paret. 'O poeta obedece à rainha.'
b. Der Dichter gehorcht der Königin. 'O poeta obedece à rainha.'

Outro exemplo de regência não canônica em latim é *docere* 'ensinar', que, tal qual seus equivalentes em grego clássico e alemão, rege dois acusativos.

(15) a. Poeta reginam fabulam docet. 'O poeta ensina a fábula à rainha.'
b. Der Dichter lehrt die Königin die Fabel. 'O poeta ensina a fábula à rainha.'

Na notação de (10) e (13), precisaríamos de mais duas classes valenciais para dar conta de frases como (14 a) e (15 a). Concomitantemente, seria necessário introduzir na sintaxe mais 30 regras para contemplar todas as permutações entre constituintes da frase (incluindo o verbo). A seguir, damos apenas alguns exemplos das permutações possíveis:

(16) a. Reginae poeta paret.
b. Paret reginae poeta.
(17) a. Docet poeta reginam fabulam.
b. Docet reginam poeta fabulam.
c. Docet reginam fabulam poeta.

[5] Sobre a distinção entre marcação de caso canônica e não canônica, ver, por exemplo, Wunderlich (2000).

No Quadro 3, mostramos algumas das molduras casuais não canônicas do latim (Glinz, 1994:232-242). Na notação clássica da DCG, cada moldura de dois casos implica o acréscimo na gramática de 6 regras. No caso de verbos de valência 3, cada moldura diferente implica mais 24 regras. Desse modo, uma DCG no formato clássico precisaria de mais (4 x 6)+(3 x 24)=96 regras para reconhecer todas as frases possíveis com os verbos de molduras casuais não canônicas do Quadro 3.

Valência	**Molduras casuais não canônicas**
2	[nom,dat] [nom,abl] [nom,gen] [acc,gen]
3	[nom,acc,acc] [nom,abl,acc] [nom,acc,gen]

Quadro 3: Verbos divalentes e trivalentes não canônicos.

Em (18) e (19), damos exemplos de frases com verbos divalentes e trivalentes, respectivamente, que apresentam uma realização não canônica da valência. A ordem dos exemplos obedece à listagem do Quadro 3.

(18) a. Regina poetae invidet. 'A rainha inveja o poeta.'
b. Terra abundat porco. 'A região abunda em porco.'
c. Regina vate reminiscitur. 'A rainha lembra-se do poeta.'
d. Reginam vatis miseret. 'A rainha comisera-se do poeta.'

(19) a. Regina poetam fabulam flagitat. 'A rainha exige do poeta uma fábula.'
b. Regina poetam somno privat. 'A rainha priva o poeta do sono.'
c. Regina poetam proditionis accusat. 'A rainha acusa o poeta de traição.'

O Quadro 3 está longe de ser exaustivo, uma vez que não considera a realização de argumentos verbais por meio de sintagmas preposicionados, orações infinitivas etc. Ele permite mostrar, contudo, que a notação clássica da DCG, por exigir um número excessivo de regras (mais de uma centena para um fragmento com os verbos do Quadro 2 e do Quadro 3), é totalmente inadequada para formalizar de maneira elegante a sintaxe latina. Na seção seguinte, propomos uma notação alternativa, que, explorando recursos avançados do SWI-Prolog, ultrapassa o formato tradicional da DCG e permite dar conta das frases como (18) e (19) por meio de apenas dois princípios gerais (checagem de casos e concordância) e somente duas regras de estrutura da frase.

3. COMO TRANSCENDER ALGUMAS LIMITAÇÕES DO FORMALISMO DCG CLÁSSICO

As línguas não configuracionais, como vimos, de que o latim é exemplo prototípico, licenciam uma enorme flexibilidade na ordem dos constituintes da frase. Isso representa uma dificuldade para a formalização da sintaxe dessas línguas em gramáticas livres de contexto ou na versão tradicional da DCG, devido ao excessivo número de regras exigido.

O SWI-Prolog disponibiliza vários predicados que permitem simplificar enormemente a descrição formal da sintaxe dessas línguas. Com esses predicados pré-definidos, podemos construir outros que expressam de forma elegante generalizações sobre fenômenos gramaticais dessas línguas, como a Checagem de Caso e a Concordância. Comecemos pela primeira operação:

(20) Checagem de Caso
Uma frase é gramatical se o conjunto de casos dos NPs que ocorrem na frase for idêntico ao conjunto de casos da moldura de subcategorização do verbo.

Note que em (20) estabelecemos uma identidade entre conjuntos. Desse modo, uma frase como (21), em que temos a seqüência de casos [dat,acc,nom], é licenciada pelo verbo *dat*, subcategorizado no léxico pela moldura [nom,dat,acc], pois há identidade de conjunto entre essas listas. De fato, pela teoria dos conjuntos, {nom,dat,acc} e {dat,acc,nom} são representações alternativas de um mesmo conjunto, pois a ordem não é relevante nesse caso.

(21) Reginae.DAT rosam.ACC poeta.NOM dat.

Para expressar formalmente em Prolog o princípio (20), precisamos alterar o formato das entradas lexicais. Em vez de utilizar índices valenciais atômicos do tipo de 0, 1, 2 e 3 como em (10), recorremos a molduras casuais sob a forma de listas:

(22)

```
v(subcat([nom,dat,acc]),agr([pers(3),num(pl)]))--> [dant].
v(subcat([nom,acc,acc]),agr([pers(3),num(pl)]))--> [docent].
v(subcat([nom,acc]),agr([pers(3),num(pl)])) --> [amant].
```

O fragmento de código em (23), baseado parcialmente na proposta de Lehner (1989:145) para a checagem do caso em alemão, implementa em Prolog o princípio de (20). Para facilitar a exposição, utilizamos o formato interno de Prolog para representar as regras do formalismo DCG, que é a notação de listas diferenciais (*difference lists*).

(23) Regras da frase com checagem de caso (1ª versão)

```
s(A,E):- np(casus(X),_,A,B),np(casus(Y),_,B,C),
np(casus(Z),_,C,D),v(subcat(L),_,D,E),
msort([X,Y,Z],R1),msort(L,R2),R1=R2.
s(A,D):- np(casus(X),_,A,B),np(casus(Y),_,B,C),
v(subcat(L),_,C,D),msort([X,Y],R1),msort(L,R2),R1=R2.
```

O predicado msort/2 transforma uma lista L num lista R cujos elementos estão ordenados alfabética ou numericamente. Isso pode ser testado na linha de comando do SWI-Prolog:

```
?- msort([nom,acc,dat],R).
R = [acc, dat, nom]
?- msort([3,5,1],R).
R = [1, 3, 5]
```

Em (23), os casos dos NPs são unificados às variáveis X, Y e Z e agrupados alfabeticamente na lista R1. Os casos da moldura de subcategorização dos verbos são também ordenados alfabeticamente numa lista R2. A última diretiva de cada regra checa se há identidade entre R1 e R2.

As duas regras de (23) licenciam, desse modo, qualquer permutação de SUBJ, OBJ2 e OBJ em frases com V final. Nessas construções, podemos ter tanto um verbo de moldura casual canônica como *amare* e *dare*, quanto um de moldura não canônica como *parere* e *docere*. Na DCG clássica, como vimos, em vez de apenas duas regras, são necessárias, ao todo, 16 regras para modelar todas as permutações dos argumentos desses verbos em estruturas de V final. Para contemplar outras subclasses não canônicas de verbos divalentes e trivalentes mais regras seriam necessárias. Na notação que propomos em (23), contudo, as únicas modificações que precisam ser feitas restringem-se ao léxico. Para analisar frases com verbos que regem genitivo e acusativo, como *accusare* 'acusar', basta introduzir entradas lexicais como a seguinte:

```
v(subcat([nom,gen,acc]),agr([pers(3),num(sg)]))--> [accusat].
```

As duas regras de (23), contudo, não contemplam a concordância verbo-nominal. Em latim, como noutras línguas acusativas, o verbo finito concorda com o NP que está no caso nominativo.[6]

[6] Veremos mais adiante como dar conta da concordância em frases com verbos impessoais, em que não há um NP (pelo menos de forma explícita) e o verbo fica na 3ª pessoa do singular.

Essa generalização é expressa formalmente na versão seguinte da regra de (23) por meio da operação de concordância (*agree*).

(24) Introdução da operação de concordância na regra da frase

```
s2(A,E):- F1 =..[np,casus(X),_,A,B],F2 =.. [np,casus(Y),_,B,C],
F3 =..[np,casus(Z),_,C,D],F4 =.. [v,subcat(L),_,D,E],
call(F1),call(F2),call(F3),call(F4),agree([F1,F2,F3,F4]),
msort([X,Y,Z],R1),msort(L,R2),R1=R2.
```

Nessa regra, utilizamos o predicado =.. (chamado univ) para repassar para o predicado agree/1 informações sobre propriedades morfossintáticas dos constituintes.

A operação agree, por sua vez, tem, inicialmente, a seguinte definição:

(25) Operação de concordância agree (1ª versão)

```
num(0). num(1). num(2). num(3). num(4).
agree(List):-  length(List,L), num(N), N<L, nth0(N,List,E),
arg(1,E,casus(nom)),member(M,List),functor(M,v,_),
arg(2,E,AGR), arg(2,M,AGR).
```

O predicado num/1 tem por função criar um índice N que induz Prolog a percorrer todos os elementos da lista de constituintes da frase, a qual constitui argumento do predicado agree/1, a fim de localizar tanto o NP que tem o traço de caso nominativo quanto o verbo, com o qual deve concordar. O NP no nominativo e o verbo podem estar em qualquer posição relativa um ao outro. As duas diretivas finais com o predicado arg/3 exigem a unificação dos traços de concordância (representados pela variável AGR) do NP no nominativo e do verbo. Caso essa unificação fracasse, a frase será agramatical.

A regra (24) é ainda limitada porque só contempla estruturas de V final. Essa limitação é eliminada pela regra (26), que, sozinha, contempla todas as 24 permutações possíveis dos constituintes de frases com verbos trivalentes, tanto canônicos quanto não canônicos:

(26) Regra da frase para verbos trivalentes (1ª versão)

```
s3(A,E):- permutation([ [np,casus(X)],[np,casus(Y)],
                        [np,casus(Z)],[v,subcat(L)]],P),
nth0(0,P,C1),nth0(1,P,C2),nth0(2,P,C3),nth0(3,P,C4),
append(C1,[_,A,B],P1),append(C2,[_,B,C],P2),
append(C3,[_,C,D],P3),append(C4,[_,D,E],P4),
F1 =..P1,F2 =.. P2,F3 =..P3,F4 =.. P4,
```

```
call(F1),call(F2),call(F3),call(F4),agree([F1,F2,F3,F4]),
msort([X,Y,Z],R1),msort(L,R2),R1=R2.
```

Nessa regra, utilizamos o predicado permutation/2 do SWI-Prolog, o qual gera todas as permutações de uma lista de elementos, no nosso caso, a lista de constituintes da frase (incluindo o verbo). Para cada uma dessas permutações, são construídas automaticamente diretivas de Prolog por meio dos predicados nth0/3, append/3 e =../2, correspondendo às diretivas que constituem o lado direito das regras da DCG clássica. Desse modo, essa única regra que propomos faz o mesmo trabalho de pelo menos 24 regras da abordagem tradicional.

Com entradas lexicais apropriadas, frases com qualquer das permutações entre constituintes (incluindo o verbo) são reconhecidas pela gramática. Alguns exemplos: com verbos de molduras casuais não canônicas (cf. Quadro 3):

(27) a. Regina fabulam poetam flagitat. 'A rainha exige do poeta uma fábula.'
Regina flagitat poetam fabulam.
Flagitat poetam fabulam regina.
b. Regina poetam somno privat. 'A rainha priva o poeta do sono.'
Somno regina poetam privat.
Privat poetam regina somno.
c. Regina proditionis poetam accusat. 'A rainha acusa o poeta de traição.'
Regina accusat proditionis poetam.
Accusat proditionis poetam regina.

A regra (26) dá conta de frases com um verbo e três argumentos nominais, em qualquer ordem. Para reconhecer frases de um verbo e dois argumentos nominais, precisamos de uma versão reduzida dessa regra, o que não é difícil fazer, como podemos constatar em (28).

(28) Regra da frase para verbos divalentes (1ª versão)

```
s3(A,D):-   permutation([[np,cas(X)],[np,cas(Y)],
                         [v,subcat(L)]],P),
nth0(0,P,C1),nth0(1,P,C2),nth0(2,P,C3),
append(C1,[_,A,B],P1),append(C2,[_,B,C],P2),
append(C3,[_,C,D],P3),
F1 =..P1,F2 =.. P2,F3 =..P3,
call(F1),call(F2),call(F3),agree([F1,F2,F3]),
msort([X,Y],R1),msort(L,R2),R1=R2.
```

Em (28), repetimos quase integralmente a última linha da regra (26). Como a operação de checagem de caso é necessária em qualquer tipo de frase, obtemos um

modelo formal mais elegante da sintaxe latina criando uma regra geral de checagem de caso, que independe do número de argumentos verbais, o que fazemos em (29).

(29) Regra da checagem de caso

```
check_case(Arguments,Verb):-
msort(Arguments,R1),msort(Verb,R2),R1=R2.
```

Com essa operação, podemos simplificar as regras de estrutura da frase. Para verbos trivalentes, temos a regra (30).

(30) Regra da frase para verbos trivalentes (2ª versão)

```
s4(A,E):- permutation([[np,cas(X)],[np,cas(Y)],
                       [np,cas(Z)],[v,subcat(L)]],P),
nth0(0,P,C1), nth0(1,P,C2), nth0(2,P,C3),nth0(3,P,C4),
append(C1,[_,A,B],P1),append(C2,[_,B,C],P2),
append(C3,[_,C,D],P3),append(C4,[_,D,E],P4),
F1 =..P1,F2 =.. P2,F3 =..P3,F4 =.. P4,
call(F1),call(F2),call(F3),call(F4),
agree3([F1,F2,F3,F4]),check_case([X,Y,Z],L).
```

Façamos um resumo do que vimos até agora. Mostramos, inicialmente, que, na notação clássica do formalismo DCG, dezenas e dezenas de regras são necessárias para dar conta de todas permutações possíveis de verbo e argumentos em línguas não configuracionais típicas, como exemplificamos por meio do latim. Em seguida, formulamos em Prolog duas regras de estrutura sintagmática e dois princípios gerais que, estendendo o poder expressivo da DCG, permitem construir gramáticas capazes de reconhecer qualquer frase latina, tanto com verbos divalentes e trivalentes canônicos quanto com vários tipos de verbos não canônicos.

Um problema, contudo, permanece não resolvido, se nosso objetivo for contemplar todos os padrões casuais do Quadro 3. Frases com verbos impessoais como (31) não são reconhecidas, mesmo quando introduzimos no léxico a entrada (32) para a forma em questão do verbo *miserere*.

(31) Reginam vatis miseret. 'A rainha comisera-se do poeta.'

(32) Exemplo de entrada para verbo impessoal

```
v(subcat([acc,gen]),agr([pers(3),num(sg)])) --> [miseret].
```

Por que isso? A explicação é simples: a operação agree verifica se há concordância entre os traços de pessoa e número do verbo e do sintagma nominal com

caso nominativo. É preciso, portanto, que haja um NP no nominativo, para que a frase seja considerada gramatical. Como *miseret* não apresenta esse caso na sua moldura de subcategorização, essa verificação fracassa e a frase não é reconhecida.

Cremos que uma solução elegante para esse tipo de verbo consiste em, inicialmente, introduzir na moldura de subcategorização de verbos impessoais um caso nominativo.

(33) Reformulação da entrada do verbo impessoal *miseret*

```
v(subcat([nom,acc,gen]),agr([pers(3),num(sg)])) --> [miseret].
```

Esse caso é satisfeito por um pronome de 3ª pessoa do singular representado por uma categoria vazia, pronome esse que devemos incluir no léxico, sob a forma de uma entrada lexical apropriada. Em (34), apresentamos uma primeira possibilidade de formulação de entrada lexical de pronome nulo.

(34) Entrada lexical do pronome foneticamente nulo (1ª versão)

```
pron(cas(nom),agr([pers(3),num(sg)])) --> [].
```

Com essas alterações, frases do tipo de (31) são reconhecidas. No entanto, a gramática passa a hipergerar, reconhecendo como gramaticais construções agramaticais do tipo de (35), em que o nominativo é realizado pelo pronome *illa* 'ela'.

(35) *Illa reginam vatis miseret.

Observemos agora estes exemplos:

(36) a. Poetam proditionis accusant. 'Acusam o poeta de traição.'
b. Illae poetam proditionis accusant. 'Elas acusam o poeta de traição.'

O latim, como língua de sujeito nulo, admite construções como (36 a), com pronome sem matriz fonética deflagrando a flexão número-pessoal do verbo. Esse pronome pode, contudo, ter uma expressão fonética, sob a forma de um demonstrativo (cf. (36 b)). Verbos impessoais, contudo, não admitem a realização fonética do pronome sujeito, o que explica a agramaticalidade de (35). É importante, desse modo, distinguir dois tipos de pronomes foneticamente vazios: um pronome nulo expletivo, que não porta papel temático, restrito à 3 ª pessoa do singular, e um pronome nulo temático, que pode assumir qualquer valor para os traços de pessoa e número (cf. Mioto, Silva e Lopes 2005:245). Por outro lado, é preciso modificar o quadro de subcategorização do verbo impessoal *miseret*. Essas duas modificações estão em (37) e (38).

(37) Entradas dos dois tipos de pronomes foneticamente nulos

```
% pronome nulo expletivo
pron(cas([nom,expl]),agr([pers(3),num(sg)])) --> [].
% pronomes nulos temáticos de 3ª pessoa
pron(cas([nom]),agr([pers(3),num(sg)])) --> [].
```

```
pron(cas([nom]),agr([pers(3),num(pl)])) --> [].
```

(38) Reformulação da entrada do verbo impessoal *miseret*

```
v(subcat([[nom,expl],acc,gen]),agr([pers(3),num(sg)])) --> [miseret].
```

Conforme (38), o verbo *miseret* exige um NP não apenas no nominativo, mas também de natureza expletiva. Essa exigência somente é satisfeita pelo pronome nulo expletivo. Desse modo, frases agramaticais como (35) não são mais reconhecidas.

Essas modificações que introduzimos no léxico implicam uma ligeira alteração na regra de concordância, de modo a aceitar tanto NPs plenos e pronominais foneticamente nulos quanto sob a forma de um expletivo:

(39) Regra de concordância (versão final)

```
agree3(List):- length(List,L), num(N), N<L, nth0(N,List,E),
arg(1,E,Functor), Functor=..[cas,Cas], nth0(0,Cas,nom),
member(M,List), functor(M,v,_), arg(2,E,AGR), arg(2,M,AGR).
```

O leitor atento deve já ter se perguntado sobre os verbos latinos que têm sintagmas preposicionados no seu quadro de subcategorização, a exemplo de verbos do português como *morar*, *depender*, *pôr* etc. Alguns exemplos:

(40) Regina aliquid ex nuntio quaeret. 'A rainha pergunta algo ao mensageiro.'
(41) Poeta reginam a latrone defendet. 'O poeta defende a rainha do ladrão.'
(42) Regina venit sub/in arborem. 'A rainha vem para baixo/dentro da árvore.'
(43) Poeta habitat sub/in/pro arbore. 'O poeta mora sob/dentro/diante de uma árvore.'
(44) Poeta equum pro urbe ponit. 'O poeta põe o cavalo em frente à cidade.'

O fragmento de gramática latina exposto nesta seção não contempla frases desse tipo, uma vez que as regras de (30) manipulam apenas sintagmas nominais. Esses exemplos mostram, por outro lado, que, no latim, a subcategorização envolvendo PPs é mais complexa do que em português, uma vez o verbo não seleciona apenas uma preposição específica (cf. (40) – (41)) ou uma classe de preposições (direcionais em (42) versus não direcionais em (43) e (44), mas, pelo menos aparentemente, também o caso do NP objeto da preposição. De fato, as preposições *in* e *sub* regem tanto acusativo quanto ablativo. O contraste entre (42) e (43), em que temos essas duas preposições regendo acusativo na primeira frase e ablativo na segunda, sugere que é o verbo que determina o caso da preposição numa determinada frase. Por exemplo, *venire* exigiria acusativo no objeto da preposição, ao passo que *habitare* e *ponere* exigiriam ablativo, como sugerem as frases de (42) a (44). Nessa hipótese, com o verbo atribuindo caso ao

NP que se encontra dentro do PP complemento do verbo, teríamos, no quadro da TRL, a violação de uma barreira à regência, que é o PP (cf. Mioto, Silva e Lopes, 2005:207). Veremos, contudo, que uma solução elegante é possível para o tratamento da subcategorização desse tipo de verbo em latim, sem precisar pressupor que o verbo subcategoriza tanto o PP quanto o NP que é complemento do PP.

O primeiro passo para modelar formalmente os diferentes tipos de subcategorização envolvendo PPs é definir as classes de preposições exigidas por cada verbo. Em (45), introduzimos nas molduras casuais dos verbos os casos preposicionais dir e sit, para representar, respectivamente, preposições direcionais e situativas, i.e., não direcionais.[7] Os casos preposicionais correspondem ao que Glinz (1994:229) chama de *Präpokasus* 'prepocaso' e na LFG é denominado de PCASE (Falk, 2001).

(45)

```
v(subcat([[nom],dir]),agr([pers(3),num(sg)]))--> [venit].
v(subcat([[nom],sit]),agr([pers(3),num(sg)]))--> [habitat].
v(subcat([[nom],acc,sit]),agr([pers(3),num(sg)]))--> [ponit].
```

Conforme (45), não há diferença alguma entre um caso tradicional como acc, realizado pela flexão nominal, e casos como dir ou sit, realizados sintaticamente por uma preposição de determinado tipo. Em línguas como finlandês, dir e sit são realizados morfologicamente por casos como alativo, ilativo, locativo (que existia no latim arcaico) e inessivo.

No léxico, as preposições direcionais e situativas dos exemplos (40) e (41) são representadas da seguinte maneira:

(46) Entradas de preposições direcionais e situativas

```
p(cas(dir),subcat([acc])) --> [in]; [sub].
p(cas(sit),subcat([abl])) --> [in]; [sub].
p(cas(sit),subcat([abl])) --> [pro].
```

Nessas entradas, o caso preposicional (i.e. o prepocaso na terminologia de Glinz) é o valor do atributo de caso, representado formalmente em (46) pelo funtor cas/1. Trata-se de um traço de núcleo sob a perspectiva da sintaxe minimalista (Radford, 1997:67), uma vez que é uma propriedade gramatical intrínseca do núcleo preposicional, a qual determina o comportamento sintático do PP (como, por exemplo, a

[7] Essa classificação das preposições se baseia em Wunderlich (2000:263), que propõe os traços +DIR e –DIR para tratar de fenômeno semelhante em alemão.

capacidade de ser argumento de um determinado tipo de verbo). Nesse arcabouço, o caso que se encontra como argumento do funtor subcat/1, na forma de único elemento de uma lista (preposições são geralmente monovalentes), é um traço de complemento, que deve ser satisfeito pelo núcleo do NP regido pela preposição. Por exemplo, a preposição *ante* exige sempre um NP no acusativo. Um PP como *ante urbem* "diante da cidade" porta caso preposicional situativo, podendo satisfazer as exigências de subcategorização de verbos como *habitare* ou *ponere*, conforme (45). Preposições como *in* e *sub* são, por outro lado, ambíguas: a variante que tem traço de núcleo dir tem traço de complemento acusativo, ao passo que a variante com traço de núcleo sit rege ablativo.

Enquanto verbos como *habitare*, *manere* "permanecer" etc. subcategorizam PPs com preposições pertencentes a uma classe ampla como as preposições situativas, outros verbos exigem preposições específicas como *ex* ou *a/ab* (cf. (40) e (41)). A mesma notação de (45) pode ser empregada para esses verbos. Ao contrário de *dir* e *sit*, o caso preposicional de uma preposição como *ex* denota uma classe que tem exatamente essa preposição como membro.

(47)

```
v(subcat([[nom],acc,ex]),agr([pers(3),num(sg)]))--> [quaeret].
v(subcat([[nom],acc,ab]),agr([pers(3),num(sg)]))--> [defendet].
```

(48)

```
p(cas(ex),subcat([abl])) --> [ex].
p(cas(ab),subcat([abl])) --> [a]; [ab].
```

A regra do PP é, por um lado, semelhante à do NP (como a regra da preposição é semelhante à do substantivo), pois tanto o predicado pp/4 quanto np/4 contém o atributo de caso como primeiro argumento:

(49)

```
np(cas(X),agr([pers(3),NUM])) --> n(cas(X),NUM).
pp(cas(P),_)--> p(cas(P),subcat([C])),np(cas(C),_).
```

Por outro lado, preposições e verbos compartilham o atributo subcategorização. Em nossa gramática, o caso do complemento da preposição não é verificado pela regra check_case, mas localmente, na própria regra do PP, por meio da variável C, que deve ser ligada a um mesmo valor tanto no funtor subcat/1 da preposição quanto cas/1 do NP.

Não seria difícil, contudo, formular a regra do PP nos moldes da regra da frase (cf. (30)), de modo a se verificar a compatibilidade de caso entre preposição e NP por meio de check_case, o que não fazemos aqui apenas para não sobrecarregar a exposição.

Para que a gramática reconheça permutações de V, NPs e PPs, definimos os predicados categories/2 e contituents/1. O primeiro contém uma lista com as categorias que realizam argumentos verbais. Essa lista poderia facilmente ser expandida para dar conta de argumentos realizados sob a forma de sintagmas adjetivais, frases infinitivas etc. O predicado constituents/1 contrói listas de dois ou três argumentos, com todas as possibilidades de combinação entre esses elementos, como [np, np], [np, pp] e [pp, pp].

(50) Predicados auxiliares

```
categories([np,pp]).
constituents([M1,M2]):- categories(Cat),member(M1,Cat),
                member(M2,Cat).
constituents([M1,M2,M3]):- categories(Cat),member(M1,Cat),
                member(M2,Cat),member(M3,Cat).
```

Agora podemos redefinir a regra da frase nos seguintes moldes, de modo a contemplar frases com verbos trivalentes com qualquer moldura de subacategorização envolvendo qualquer combinação de NPs e PPs, em qualquer ordem.

(51) Regra da frase para verbos trivalentes (versão pré-final)

```
s5(A,E):- constituents([M1,M2,M3]),
append([M1],[cas(X)],Arg1),
append([M2],[cas(Y)],Arg2),
append([M3],[cas(Z)],Arg3),
append([Arg1,Arg2,Arg3],[[v,subcat(L)]],Const),
permutation(Const,P),
nth0(0,P,C1),nth0(1,P,C2),nth0(2,P,C3),nth0(3,P,C4),
append(C1,[_,A,B],P1),append(C2,[_,B,C],P2),
append(C3,[_,C,D],P3),append(C4,[_,D,E],P4),
F1 =..P1,F2 =.. P2,F3 =..P3,F4 =.. P4,
call(F1),call(F2),call(F3),call(F4),
agree3([F1,F2,F3,F4]),check_case([X,Y,Z],L).
```

Com apenas essa única regra, damos conta de um fragmento do latim que necessitaria de mais de uma centena de regras na notação da DCG clássica. Esse fragmento inclui qualquer uma das 24 combinações entre o verbo e seus três argumentos possíveis para cada uma das frases de (40) a (44). Variantes dessas frases

com pronome nulo temático são também reconhecidas, do mesmo modo que frases com verbos que subcategorizam um pronome nulo expletivo.

Um exame atento do funcionamento dessa regra, contudo, revela que ainda precisa ser aperfeiçoada. A resposta de Prolog à seguinte diretiva revela uma deficiência grave (o predicado corpus/2 está definido no arquivo corpus.pl disponível no nosso site):

```
?- corpus(N,Frase),phrase(s5,Frase).
N = 2
Frase = [poeta, reginae, rosam, dat] ;
N = 2
Frase = [poeta, reginae, rosam, dat]
Yes
```

Como podemos observar, Prolog gera duas frases idênticas. Na verdade, a definição de (51) é de tal modo redundante que ao todo 6 frases idênticas são geradas para cada combinação gramatical entre um verbo trivalente e seus argumentos. Isso se deve ao fato de que as variáveis X, Y e Z nessa regra, que representam os diferentes valores do atributo caso, não estão vinculadas quando Prolog realiza as permutações entre verbos e argumentos.

Esse problema pode ser facilmente corrigido, portanto, ligando essas variáveis às combinações de casos existentes no léxico. Para tanto, criamos um predicado listall/0 com a seguinte definição:

(52) Levantamento de todas as molduras casuais do léxico verbal

```
listall:- findall(L,lex(_,v,[subcat(L)|_]),List),
          list_to_set(List,Set),
          member(M,Set),
          assert(case_list(M)),fail.
```

Antes de testar a gramática, precisamos armazenar na memória de Prolog as diferentes molduras, num processo conhecido como *caching* (Blackburn, Bos e Striegnitz, 2001:162). Isso é feito pela seguinte diretiva, que deve ser dada a Prolog no início de uma sessão de teste da gramática:

```
?- listall.
No
```

Com isso, os diferentes tipos de molduras presentes no léxico passam a ficar disponíveis na memória:

```
?- case_list(Moldura).
Moldura = [[nom], dir] ;
Moldura = [[nom], sit] ;
Moldura = [[nom], acc, ex]
Yes
```

Uma chamada ao predicado case_list/1 deve ser feita no corpo de cada regra da frase (por ex. case_list([X,Y,Z]) para verbos trivalentes).

Com essa pequena modificação, a gramática deixa de gerar cópias redundantes das frases, o que pode ser facilmente verificado da seguinte forma:

```
?- corpus(2,Frase),phrase(s5,Frase).
Frase = [poeta, reginae, rosam, dat] ;
No
```

A gramática continua gerando frases em duplicata, mas apenas no caso de construções sintaticamente ambíguas:

```
?- corpus(31,Frase), phrase(s5,Frase).
Frase = [poetae, reginae, parent] ;
Frase = [poetae, reginae, parent] ;
No
```

A ambigüidade dessa frase decorre de sincretismo na marcação de caso do latim. A terminação *-ae* da primeira declinação corresponde tanto ao dativo singular quanto ao nominativo plural. Dessa forma, obtemos as duas interpretações abaixo, conforme analisemos *poetae* ou *reginae* como sujeito (nominativo plural) ou objeto (dativo singular):

(53) a. Poetae.NOM reginae.DAT parent.
'Os poetas obedecem à rainha.'
b. Poetae.DAT reginae.NOM parent.
'As rainhas obedecem ao poeta.'

A gramática, porém, não explicita as diferentes análises sintáticas da frase, às quais correspondem diferentes representações semânticas:

(54) a. parent(agent(poetae),recipient(reginae))
b. parent(agent(reginae),recipient(poetae))

Conforme (54 a), no evento designado pelo verbo *parent*, o referente da expressão *poetae* desempenha o papel de agente, ao passo que o referente da expressão *reginae* desempenha o papel de recipiente. Em (54 b), invertem-se os papéis.

Na próxima seção, mostraremos como ampliar a nossa gramática de modo a dar conta do fenômeno da ambigüidade estrutural e da construção de representações semânticas a partir das árvores sintáticas. Outros problemas do modelo até agora utilizado também serão resolvidos.

4. MAPEAMENTO ENTRE SINTAXE E SEMÂNTICA

Nesta seção, vamos corrigir vários problemas da versão anterior da gramática, ao mesmo tempo em que passamos a dar conta formalmente da análise sintática e semântica das frases.

Como teoria modular, a gramática gerativa, em seus diferentes modelos, postula a separação entre o léxico e a sintaxe (o "sistema computacional" no Programa Minimalista), considerados subcomponentes distintos da língua-I (Raposo, 1999:16-18). Nas duas seções anteriores, não levamos isso em conta. As regras do léxico são modeladas da mesma forma que as da sintaxe na DCG tradicional.

Desenvolvendo sugestão de Blackburn, Bos e Striegnitz (2001:119), reformulamos as entradas lexicais utilizando o predicado lex/3. O primeiro argumento contém a forma da palavra (já flexionada se for o caso), tal como é usada pela sintaxe. O segundo argumento é a categoria sintática. O terceiro argumento consiste numa lista com as propriedades morfossintáticas, que incluem tanto os traços de núcleo (por ex. os atributos de caso e concordância nos substantivos e pronomes) quanto de complemento (por ex. o atributo de subcategorização em verbos e preposições).

(55) Entradas lexicais no novo formato

```
lex(poeta,n,[cas([nom]),num(sg)]).
lex(ego,pron,[cas([nom]),agr([pers(1),num(sg)])]).
lex(urbe,n,[cas(abl),num(sg)]).
lex(habitat,v,[subcat([[nom],sit]),agr([pers(3),num(sg)])]).
lex(in,p,[cas(sit),subcat([abl])]).
lex(parent,v,[subcat([[nom],dat]),agr([pers(3),num(pl)])]).
```

Na parte sintática da gramática, essas representações são manipuladas por Prolog, que, com base em predicados que elaboramos, constrói regras no formalismo DCG capazes de processar as frases não só sintática, mas também semanticamente. Para as categorias lexicais plenas (i.e. excluindo os pronomes nulos) construímos a seguinte regra:

(56)

```
w(F,Cat,GR,[Form|A],A):-      lex(Form,Cat,GR),
                              functor(F,Cat,1),arg(1,F,Form).
```

Por essa regra, o predicado w/5 gera, por exemplo para preposição *in* da entrada em (55), a seguinte representação:

```
w(p(in),p, [cas(sit), subcat([abl])],[in],[]).
```

Conseqüentemente, o PP é definido com base diretamente no predicado w/5 e não no predicado lex/3:

```
pp(pp(F1,F2),[Pcase])-->
w(F1,p,[Pcase,subcat([Cas])]),np(F2,[cas(Cas),_]).
```

Uma das questões mais discutidas em lingüística gerativa, nas duas últimas décadas, diz respeito à projeção de estruturas morfossintáticas a partir das estruturas semânticas de itens lexicais. Diferentes teorias da projeção ou vinculação argumental (*argument linking*) postulam papéis temáticos, ou construtos semânticos alternativos como estrutura aspectual ou eventiva, como primitivos, dos quais são derivadas as realizações dos argumentos.[8]

Neste trabalho, optamos por considerar as molduras casuais como primitivos. Essas estruturas, que integram as entradas lexicais dos verbos, constituem o input de um mecanismo de "mapeamento" (do inglês *mapping*, i.e. projeção) que constrói as respectivas grades de papéis temáticos.

[8] Uma excelente discussão das principais abordagens nessa área é oferecida por Levin e Hovav (2005).

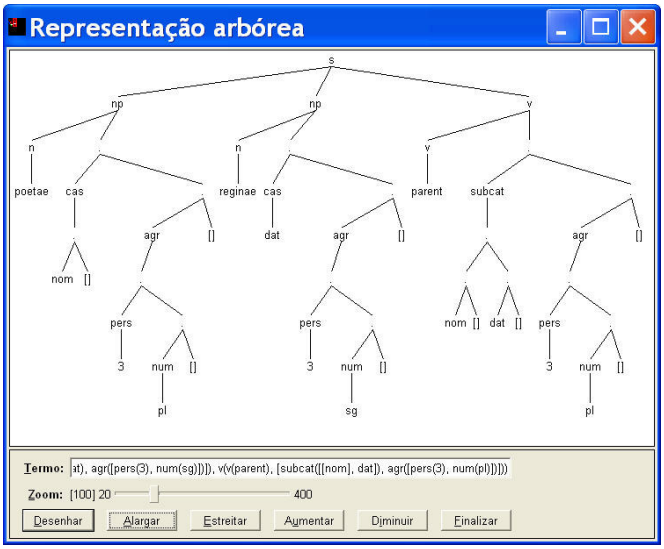

Figura 1: Representação sintática de (53 a)

Para tanto, propomos uma teoria da vinculação argumental bastante simples. Inicialmente, definimos o predicado mapping/2, que estabelece todas as correspondências entre molduras casuais e grades de papéis temáticos. Alguns exemplos:

(57)

```
mapping([[nom], sit],[th(_),loc(_)]).
mapping([[nom], dat],[ag(_),rec(_)]).
mapping([[nom], acc, ex],[ag(_),th(_),loc(_)]).
mapping([[nom], acc, sit],[ag(_),th(_),loc(_)]).
mapping([[nom, expl], acc, gen],[nul(_),exper(_),stim(_)]).
```

O primeiro argumento desse predicado é uma moldura casual, que pode incluir, como vimos, não só casos morfológicos como nominativo e dativo, mas também prepocasos como sit ou ex. O segundo argumento de mapping/2 é uma grade de papéis temáticos.

Papel temático	Representação em Prolog	Exemplos de frases e representações semânticas
AGENTE	ag(_)	Poeta reginae rosam dat. 'O poeta dá ma rosa à rainha.' dat(ag(np(n(poeta))),rec(np(n(reginae))),th(np(n(rosam))))
RECIPIENTE	rec(_)	
TEMA	th(_)	
LOCATIVO	loc(_)	Poeta in urbe habitat. 'O poeta mora na cidade." habitat(th(np(n(poeta))),loc(pp(p(in),np(n(urbe)))))
NULO	nul(_)	Reginas vatis miseret. 'As rainhas comiseram-se do vate.' miseret(nul(np(pron(expl))), exper(np(n(reginas))), stim(np(n(vatis))))
EXPERIENCIADOR	exper(_)	
ESTÍMULO	stim(_)	

Quadro 4: Inventário de papéis temáticos

Nossa gramática utiliza o inventário de papéis temáticos do Quadro 4, que se baseia, quase que totalmente, em Kroeger (2004). A diferença mais importante com relação a essa proposta é que não distinguimos entre TEMA e PACIENTE, uma vez que esses papéis, geralmente, são atribuídos a um mesmo participante do evento (Falk, 2001:103).

O papel temático NULO que propomos não tem um estatuto lingüístico, visando apenas a facilitar a formulação dos princípios de mapeamento dos verbos que subcategorizam um pronome nulo expletivo. Com esse papel sem interpretação semântica, não há discrepância entre o número de casos regidos pelo verbo e o número de elementos na grade temática. As representações dos exemplos sob a forma de termos de Prolog constituem, como veremos, uma simplificação das representações que a gramática produz.

O papel LOC, que na abordagem de Kroeger se desdobra em ORIGEM, ALVO e CAMINHO, é utilizado na nossa gramática para caracterizar semanticamente argumentos não só de verbos locativos *stricto sensu*, que envolvem literalmente movimento ou localização, mas também verbos que expressam noções de "pseudo-espaço" (Falk, 2001:103), como *quaerere* 'perguntar' (cf. ex. (40)) (que envolve transmissão de informações) e *defendere* 'defender'.

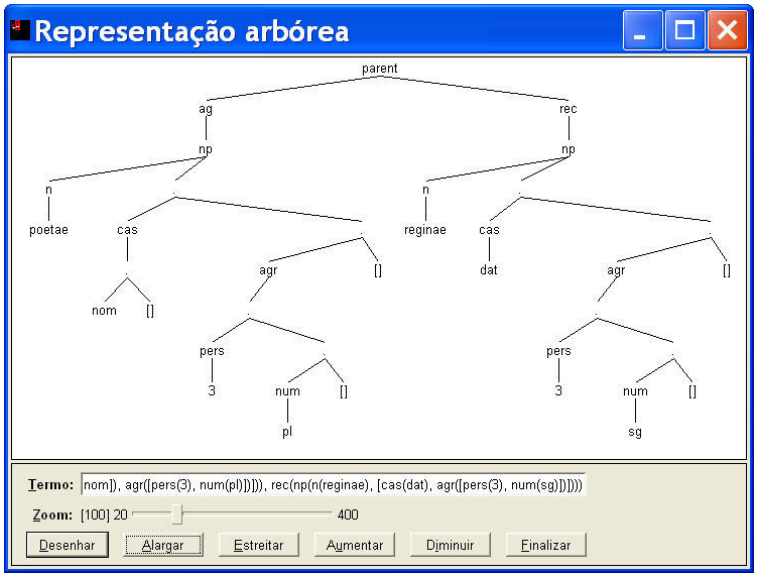

Figura 2: Análise da frase (53 a) em termos de papéis semânticos (leitura (54 a))

Com base nas estruturas morfossintáticas geradas pelas regras da frase, no princípio da concordância (que localiza o NP com caso nominativo com o qual o verbo concorda, i.e. o sujeito da frase) e nas associações entre molduras casuais e grades temáticas formalizadas pelo predicado mapping/2, as representações dos NPs e PPs regidos pelos verbos são inseridas nas respectivas estruturas lógico-semânticas. Para uma forma verbal como *parent* 'obecedem', é gerada automaticamente a representação parent(ag(_),rec(_)). Tudo isso é feito por um outro predicado, o mapping/3, que, por ser bastante complexo, deixamos de comentar em detalhe. O leitor pode baixar a gramática de nosso site e testar esse predicado, por meio da seguinte diretiva com o predicado s/3:

```
?- s(Synt,Sem,[poetae,reginae,parent]).
```

Com as modificações apresentadas, a gramática agora gera as diferentes estruturas de constituintes de frases ambíguas como (53). A Figura 1 traz, em formato de árvore, a primeira das representações sintáticas geradas. As estruturas de traços morfossintáticos dos constituintes estão no formato gráfico de listas de Prolog (Blackburn, Bos e Striegnitz, 2001:129). A árvore pode ser obtida por meio da seguinte diretiva:[9]

[9] O predicado desenha_termo/1 (*zeichne_term/1* no original alemão) faz parte de programa em Prolog/XPCE, elaborado por Gerhard Röhner (autor do SWI-Prolog-Editor), que interpreta graficamente termos de Prolog: **http://lernen.bildung.hessen.de/informatik/swiprolog/term.htm**.

```
?- s(Synt,Sem,[poetae,reginae,parent]), desenha_termo(Synt).
```

Analogamente, os dois mapeamentos diferentes entre molduras casuais e estruturas argumentais de (54) também são produzidos, conforme podemos constatar na Figura 2 e na Figura 3.

As árvores da Figura 2 e Figura 3 ainda contêm muita informação sintática, como caso e propriedades de concordância dos constituintes que realizam os papéis temáticos verbais. Conforme o Programa Minimalista, os traços de caso, ao contrário dos traços de concordância, não são interpretáveis e devem ser apagados no curso da derivação da Forma Lógica, depois de checados (Radford, 1997: 70-74).

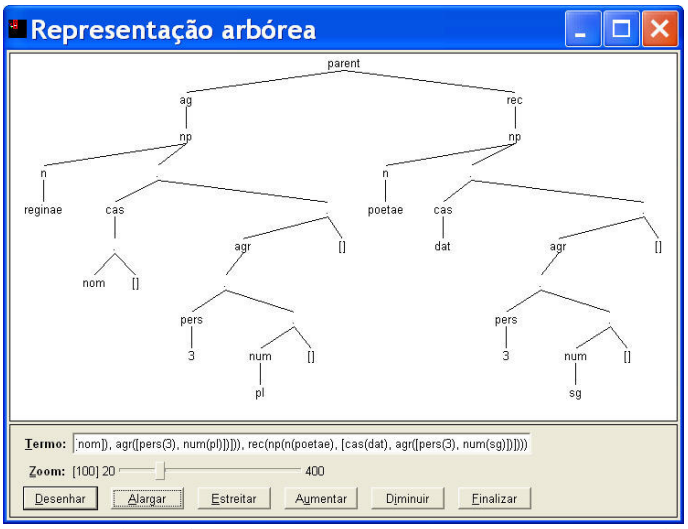

Figura 3: Análise da frase (53 b) em termos de papéis semânticos (leitura (54 b))

Tendo em vista uma possível utilização da gramática no ensino e aprendizagem do latim, porém, decidimos manter os traços de caso em nossas árvores semânticas. Cremos que, nesse campo de aplicação, sobretudo no caso de alunos iniciantes, é conveniente visualizar esse tipo de informação morfossintática simultaneamente à informação semântica. O aluno pode, então, se familiarizar melhor com as relações sistemáticas entre esses dois planos.

CONCLUSÃO

Neste trabalho, elaboramos um fragmento de gramática do latim na linguagem de programação Prolog. Trata-se, portanto, de uma descrição completamente formal de fenômenos sintáticos característicos dessa língua, como a ordem livre de constituintes, a possibilidade de sujeito nulo e a marcação dos argumentos verbais por meio da flexão casual e preposições e pela concordância verbo-nominal.

O fragmento de gramática que desenvolvemos, por outro lado, graças à natureza declarativa de Prolog, pode ser aplicado automaticamente não só no reconhecimento, análise sintática e semântica de frases, mas também na geração de frases. Tudo isso é possível porque Prolog já disponibiliza o algoritmo necessário para tanto. A gramática pode ser, então, testada exaustivamente, de forma automática, o que facilita a descoberta de erros que, de outro modo, poderiam passar despercebidos.

Como língua não configuracional, em que a ordem dos constituintes da frase é completamente livre, o latim representa uma enorme dificuldade de modelação no formalismo DCG em sua versão tradicional. Nessa notação, necessitamos de um número excessivo de regras para dar conta de todas as permutações admissíveis entre verbos de diferentes valências e seus argumentos, que, no caso do sujeito, ainda podem ser expressos por pronomes nulos.

Ao longo do trabalho, mostramos como deficiências da DCG clássica podem ser contornadas, quando utilizamos todo o potencial de Prolog para modelar formalmente as regularidades de uma língua. Em vez de centenas de regras de estrutura da frase, formulamos apenas duas para dar conta de verbos divalentes e trivalentes. Na nossa proposta, a gramaticalidade das frases é assegurada por duas regras gerais, que verificam a concordância entre sujeito e verbo e checam se os constituintes da frase satisfazem as exigências expressas nas molduras de subcategorização dos verbos.

Nossa gramática é capaz, também, de identificar frases sintaticamente ambíguas, apresentando as diferentes árvores de estrutura sintagmática e suas contrapartes semânticas.

REFERÊNCIAS BIBLIOGRÁFICAS

1. BAADER, Franz. Logic-based knowledge representation. In: WOOLDRIDGE, Michael J.;VELOSO, Manuela (Orgs.). *Artificial intelligence today: recent trends and developments*. Berlin: Springer, 1999. p.13-41.
2. BEESLEY, Kenneth R.; KARTTUNEN, Lauri. *Finite state morphology*. Stanford: CSLI Publications, 2003.
3. BIRD, Steven; KLEIN, Ewan; LOPER, Edward (2007). *Natural Language Processing in Python*. Disponível em: <http://nltk.sourceforge.net/doc/en/book.pdf> Acesso em: 15.05.2007.
4. BLACKBURN, Patrick; BOS, Johan; STRIEGNITZ, Kristina. *Learn Prolog now!* Saarbrücken: Universität des Saarlandes, 2001. Disponível em:< http://www.coli.uni-saarland.de/~kris/learn-prolog-now/html/prolog-notes.pdf> Acesso em: 29.04.2005.
5. DAVID, Karine Alves. *Sintaxe das expressões nominais no português do Brasil: uma abordagem computacional*. 2007. 117p. Dissertação (Mestrado) – Programa de Pós-Graduação em Lingüística, Universidade Federal do Ceará, Fortaleza.
6. FALK, Yehuda N. *Lexical-Functional Grammar: an introduction to parallel constraint-based syntax*. Stanford: CSLI Publications, 2001.
7. GLINZ, Hans. *Grammatiken im Vergleich: Deutsch – Französisch – Englisch – Latein. Formen – Bedeutungen – Verstehen*. Tübingen: Niemeyer, 1994.
8. KAPLAN, Ronald M. Syntax. In: MITKOV, Ruslan. *The Oxford handbook of computational linguistics*. Oxford: Oxford University Press, 2004. p.70-90.
9. KLABUNDE, Ralf. Automatentheorie und formale Sprachen. In: _____ et al. (Orgs.). *Computerlinguistik und Sprachtechnologie: eine Einführung*. Heidelberg: Spektrum Akademischer Verlag, 2004. p. 63-90.
10. KLENK, Ursula. *Generative Syntax*. Tübingen: Narr, 2003.
11. KROEGER, Paul R. *Analyzing syntax: a lexical-functional approach*. Cambridge: Cambridge University Press, 2004.
12. LEHNER, Christoph. *Prolog und Linguistik*. Hildesheim: Universität Hildesheim, 1989. Disponível em:< http://www.uni-hildesheim.de/~chlehn/p_ueb/probuch.pdf> Acesso em: 02.08.2005.
13. LEVIN, Beth; HOVAV, Malka Rappaport. *Argument realization*. Cambridge: Cambridge University Press, 2005.

14. MARTÍN-VIDE, Carlos. Formal grammars and languages. In: MITKOV, Ruslan. *The Oxford handbook of computational linguistics*. Oxford: Oxford University Press, 2004. p. 157-177.
15. MIOTO, Carlos; FIGUEIREDO, Maria Cristina; LOPES, Ruth. *Novo manual de sintaxe*. Florianópolis: Insular, 2005.
16. OTHERO, Gabriel. *Teoria X-barra: descrição do português e aplicação computacional*. São Paulo: Contexto, 2006.
17. PAGANI, Luiz Arthur. Analisador gramatical em Prolog para gramáticas de estrutura sintagmática. *Revista Virtual de Estudos em Linguagem – ReVEL.* Ano 2, n.3, ag. 2004. [www.revel.inf.br].
18. RADFORD, Andrew. *Minimalist syntax: exploring the structure of English*. 2. ed. Cambridge: Cambridge University Press, 2004.
19. RAPOSO, Eduardo Paiva. Da Teoria dos Princípios e Parâmetros ao Programa Minimalista: algumas idéias-chave. In: CHOMSKY, Noam. *O Programa Minimalista*. Tradução de Eduardo Paiva Raposo. Lisboa: Caminho, 1999. p.15-37.
20. SAG, Ivan A.; WASOW, Thomas; BENDER, Emily. *Syntactic theory: a formal introduction*. 2. ed. Stanford: CSLI Publications, 2003.
21. WEIDMANN, Clemens. *Lateinische Grammatik – Grammatica Latina*. Wien: Universität Wien, 2007. Disponível em:<http://www.univie.ac.at/latein/gr/grinh.htm> Acesso em: 10.11.2007.
22. WUNDERLICH, Dieter. Predicate composition and argument extension as general options: A study in the interface of semantic and conceptual structure. In: STIEBELS, Barbara; _____ (Orgs.). *Lexicon in Focus*. Berlin: Akademie Verlag, 2000. p. 247-269.

RESUMO: Neste trabalho, apresentamos uma análise formal de aspectos da sintaxe do latim, apontando algumas das dificuldades que línguas não configuracionais prototípicas como essa representam para um tratamento computacional no formalismo DCG na sua versão clássica. Propomos um formalismo alternativo que explora recursos avançados do SWI-Prolog para ampliar o poder expressivo da DCG. Nesse formato mais eficiente, construímos um fragmento de gramática gerativa do latim que dá conta da ordem livre de constituintes em frases com uma variada gama de verbos divalentes e trivalentes que seguem tanto padrões canônicos quanto não canônicos de caso e vinculação argumental. Para tanto, recorremos a duas regras gerais que formalizam a checagem de caso e a concordância, de modo que apenas duas regras de estrutura sintagmática são necessárias para gerar frases com verbos de dois e três lugares. Na DCG tradicional, ao contrário, precisa-se de centenas de regras para modelar o mesmo conjunto de frases. Por outro lado, graças ao algoritmo de parsing já embutido em Prolog, nosso fragmento

de gramática não só reconhece e analisa frases com sujeitos nulos tanto expletivos quanto temáticos, mas também realiza o mapeamento entre estrutura morfossintática e estrutura semântica. Dessa maneira, é capaz de construir representações diferentes para frases ambíguas que resultam de mais de uma possibilidade de mapeamento sintático-semântico, explicitando as diferentes maneiras pelas quais a estrutura argumental do verbo é vinculada aos argumentos sintáticos na frase.
PALAVRAS-CHAVE: latim; não-configuracionalidade; interface léxico-sintaxe; Prolog.

ABSTRACT: In this paper, we present a formal analysis of some aspects of Latin syntax, pointing out difficulties prototypical non-configurational languages like this represent to a computational treatment within the DCG formalism in its classical version. We propose an alternative notation that exploits SWI-Prolog's advanced features to extend the expressive power of DCG. In this more efficient format, a generative Latin grammar fragment is built that copes with free order of constituents in sentences with a variety of divalent and trivalent verbs exhibiting both canonical and non-canonical case and argument linking patterns. In our proposal, we resort to two general rules formalizing case checking and agreement, so that only two phrase structure rules are needed to generate sentences with two-place and three-place verbs. By contrast, hundreds of rules are necessary in the traditional DCG format to model the same corpus. On the other hand, thanks to Prolog built-in parsing algorithm, our grammar fragment not only recognizes and analyses sentences with both expletive and thematic null subjects, but also does the mapping from morphosyntactic to semantic structure. By this means, it is capable of building different representations of ambiguous sentences resulting from more than one possible syntactic-semantic mapping, expliciting the different ways how the verb's argument structure is linked to syntactic arguments in the sentence.
KEYWORDS: Latin; non-configurationality; lexicon-syntax interface; Prolog.

Recebido no dia 04 de dezembro de 2007.

Artigo aceito para publicação no dia 26 de fevereiro de 2008.