Demonstrando uma intuição equivocada:[*] *o uso da lógica e da computação na lingüística*

On a mistaken intuition: the use of logic and computation in linguistics

Luiz Arthur Pagani

1 Introdução

Em um artigo anterior nesta Revista Letras (GONÇALVES; PAGANI, 2004), na tentativa de desqualificar, no processamento lingüístico, uma mudança categorial da palavra *past* em duas derivações diferentes, que supostamente deveriam explicar a eliminação do efeito labirinto da sentença *The horse raced past the barn fell*,[1] os autores afirmam que

> a mudança da preposição, de ((S\NP)\(S\NP))/NP para (NP\NP)/NP, não se mostra problemática após a escolha da categoria de *raced*, já que, sendo a primeira a representação de uma categoria de adjun-

* O presente texto foi preparado no sistema Latex, a partir de sua implementação para Linux – o TeTEX – e de um programa de integração (IDE) – o Kile –, tudo isso instalado em computadores funcionando com o sistema operacional Kurumin, dentro das diretrizes do chamado *software* livre. Expresso assim o meu reconhecimento à enorme equipe anônima que desenvolve todos esses recursos.

1 O termo *labirinto* está traduzindo aqui o termo em inglês *garden-path*, empregado em expressões como *garden-path sentence* e *garden-path effect*, que, na psicolingüística, designa dificuldades de processamento, como a do famoso exemplo mencionado na citação acima; em português, um exemplo de labirinto é *O navio angolano entrava no porto o navio brasileiro*, na qual a forma *entrava* tende a ser interpretada como o pretérito imperfeito do verbo *entrar*, e não como o presente do verbo *entravar*, causando uma dificuldade de integração do objeto direto *o navio brasileiro*, já que *entrar* não é transitivo direto.

> *to adverbial e a segunda a representação de uma categoria de adjunto adnominal, a escolha feita em raced determina automaticamente qual das duas versões de análise para a preposição deve ser escolhida. As regras da divisão R5 e R6 (BORGES NETO, 1999, p. 56) parecem dar conta da mudança, já que é plausível que ela se aplique apenas ao funtor e não ao argumento, da mesma forma como se pode dividir, num divisor composto, o seu divisor e o seu dividendo por um mesmo número, sem que se faça a mesma divisão no dividendo principal (ou seja, a operação (2/3)/5 é equivalente à operação ((2/7)/(3/7))/5. (GONÇALVES; PAGANI, 2004, p. 191).*

No entanto, diferentemente da intuição inicial dos autores, a derivação da categoria ((S\NP)\(S\NP))/NP a partir da categoria (NP\NP)/NP (ao contrário também do que sugeria a redação da nota mencionada) não pode ser obtida numa Gramática Categorial que respeite os princípios tradicionais de demonstração lógica.

Assim, o objetivo do presente artigo é demonstrar essa inderivabilidade. Curiosamente, apesar do objetivo modesto, a demonstração exigirá a apresentação de alguns recursos formais, desenvolvidos na Lógica, que são empregados nas Gramáticas Categoriais, com os quais os lingüistas estão pouco familiarizados. Isso será feito na seção 2; depois dela, apresentaremos as derivações possíveis para as divisões compostas, na seção 3. A demonstração da inderivabilidade será feita na seção 4, e depois dela apresentaremos, na seção 5, um demonstrador implementado em Prolog, que também confirma computacionalmente a inderivabilidade. O artigo termina com as conclusões a que se pôde chegar depois de tudo o que será apresentado aqui (seção 6).

2 CÁLCULO DE SEQÜENTES DO LAMBEK

Para a demonstração que se pretende fazer aqui, será empregada uma parte do cálculo de seqüentes de Gentzen, usado pelo Lambek (1958)

[2] *A regra de corte (cut) não será apresentada aqui porque, além de ser desnecessária para os propósitos do presente artigo, ela é desnecessária de uma forma geral (como o próprio Carpenter (1997, p. 150-153) demonstrou); como estamos interessados também apenas na parte sintática, desconsideraremos sua parte semântica. Uma apresentação mais simples das Gramáticas Categoriais foi feita por Wood (1993, p. 34), e uma apresentação deste mesmo cálculo, inclusive com uma implementação em Prolog na qual me inspirei para elaborar o demonstrador da seção 5, já havia sido proposta por Moortgat (1988); em português, temos a introdução de Borges Neto (1999).*

para definir sua axiomatização da Gramática Categorial, mas recorremos à sua apresentação feita por Carpenter (1997, p. 140-150).[2]

Na Lógica, os seqüentes são fórmulas do tipo $\Gamma \Rightarrow \phi$, que são normalmente interpretados como "a proposição ϕ pode ser provada (ou demonstrada) a partir da seqüência de proposições em Γ"; o termo Γ, anterior ao símbolo de demonstração, é chamado de **antecedente***, e o termo ϕ, posterior a este símbolo, é chamado de* **conseqüente***.*[3]

No uso que Lambek faz da Lógica, o que normalmente seriam proposições e conjuntos de proposições passam a ser reinterpretados como categorias e seqüências de categorias.[4] *Assim, por exemplo, pode-se representar o fato de que uma expressão da categoria N seguida de uma expressão da categoria N\S resulta numa expressão da categoria S por meio do seqüente N, N\S ⇒ S.*

A parte do cálculo que nos interessa aqui pode ser representada pelos seguintes esquemas de derivação de seqüentes:

1. $$\frac{}{A \Rightarrow A}\ Id$$

2. $$\frac{\Delta \Rightarrow B \qquad \Gamma_1, A, \Gamma_2 \Rightarrow C}{\Gamma_1, A/B, \Delta, \Gamma_2 \Rightarrow C}\ /A$$

3. $$\frac{\Delta \Rightarrow B \qquad \Gamma_1, A, \Gamma_2 \Rightarrow C}{\Gamma_1, \Delta, B\backslash A, \Gamma_2 \Rightarrow C}\ \backslash A$$

4. $$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow B/A}\ /C \qquad [\Gamma \text{ não vazio}]$$

5. $$\frac{A, \Gamma \Rightarrow B}{\Gamma \Rightarrow A\backslash B}\ \backslash C \qquad [\Gamma \text{ não vazio}]$$

[3] *A notação mais comum na Lógica para os seqüentes talvez seja $\Gamma \vdash \phi$; como observa o próprio Carpenter (1997, p. 140), o símbolo $\Rightarrow$ é mais empregado na Computação. Vou preferir este último exatamente por este motivo: na implementação do demonstrador em Prolog, apresentado na seção 5, é bem mais simples definir um operador mais parecido com $\Rightarrow$ do que com $\vdash$.*

[4] *Essa reinterpretação foi motivada pela semelhança entre o esquema de dedução do modus ponens (p, p → q ⇒ q) e os esquemas de simplificação categorial (X/Y, Y → X e Y, Y \X → X). Usaremos aqui a notação categorial do Lambek, ao contrário da notação do Steedman usada por Gonçalves e Pagani (2004); assim, em vez de NP usaremos N e qualquer categoria X\Y será escrita como Y \X (sobre diferenças notacionais nas Gramáticas Categoriais, ver as observações de Wood (1993, p. 11)).*

$$\dfrac{\dfrac{}{N \Rightarrow N}\,Id \qquad \dfrac{}{S \Rightarrow S}\,Id}{N,\ N\backslash S \Rightarrow S}\,\backslash A$$

Figura 1 - Demonstração de que N, $N\backslash S \Rightarrow S$

Nestes esquemas, os seqüentes que aparecem acima da barra horizontal são chamados de **premissas** *e o seqüente que aparece abaixo é chamado de* **conclusão***; com eles, é possível demonstrar que a partir de uma seqüência de premissas se chega uma conclusão.*[5] *Assim, de volta ao nosso exemplo, a demonstração de que efetivamente uma expressão da categoria N seguida de uma expressão da categoria $N\backslash S$ resulta numa expressão da categoria S pode ser representada pela derivação apresentada na figura 1.*

Para fazer esta derivação, empregou-se o esquema 3, com $\Gamma_1 = \phi$, $\Delta = N$, $B = N$, $A = S$, $\Gamma_2 = \phi$ e $C = S$. Como todos os ramos desta derivação terminam no axioma da identidade (1), esta derivação é uma demonstração.

Mais importante do que demonstrar que N, $N\backslash S \Rightarrow S$ (o que, do ponto de vista de um lingüista, poderia ser mais simplesmente demonstrado a partir da dedução natural de Prawitz, em vez do cálculo de seqüentes, de acordo com a discussão feita por Pagani (2004)), esse tipo de demonstração serve mais claramente para testar o tipo de afirmação feita sobre a relação entre as categorias $(N\backslash N)/N$ e $((N\backslash S)\backslash(N\backslash S))/N$ (que são as versões, na notação de Lambek, para as categorias $(NP\backslash NP)/NP$ e $((S\backslash NP)\backslash(S\backslash NP))/NP$, empregadas por Gonçalves e Pagani (2004)).

A partir desse tipo de derivação é possível, por exemplo, demonstrar que a regra R4 (a chamada regra de promoção de tipo – type raising) é válida, como se pode ver na figura 2. Em 2(a), $X \Rightarrow Y/(X\backslash Y)$ é transformado em X, $X\backslash Y \Rightarrow Y$, pelo esquema 4, com $\Gamma = X$, $B = Y$ e $A = X\backslash Y$; depois, a aplicação do esquema 3, para $\Gamma_1 = \phi$, $\Delta = X$, $B = X$, $A = Y$, $\Gamma_2 = \phi$ e $C = Y$, termina a demonstração em duas instâncias do axioma da identidade. Em 2(b), $X \Rightarrow (Y/X)\backslash Y$ é primeiro transformado em Y/X, $X \Rightarrow Y$, pelo esquema 5, para $\Gamma = X$, $A = Y/X$ e $B = Y$, que, agora pela aplicação do esquema 2), com $\Gamma_1 = \phi$, $A = Y$, $B = X$, $\Delta = X$, $\Gamma_2 = \phi$ e $C = Y$, nos leva a duas instâncias do axioma da identidade, encerrando com sucesso a demonstração.

[5] *Ao contrário das demonstrações nos cálculos clássicos (o proposicional e o quanticacional (MORTARI, 2001, p. 63)), que partem de um conjunto de premissas, o cálculo de seqüentes se caracteriza como uma lógica linear, na qual a ordem das premissas é relevante.*

(a)

$$\dfrac{\dfrac{\dfrac{}{X \Rightarrow X}\,Id \qquad \dfrac{}{Y \Rightarrow Y}\,Id}{X,\ X\backslash Y \Rightarrow Y}\,\backslash A}{X \Rightarrow Y/(X\backslash Y)}\,/C$$

(b)

$$\dfrac{\dfrac{\dfrac{}{X \Rightarrow X}\,Id \qquad \dfrac{}{Y \Rightarrow Y}\,Id}{Y/X,\ X \Rightarrow Y}\,/A}{X \Rightarrow (Y/X)\backslash Y}\,\backslash C$$

Figura 2 - Demonstração para a regra R4

3 DEMONSTRAÇÃO DA DIVISÃO GENERALIZADA

A fim de demonstrar quais são as possíveis aplicações das regras de divisão a uma categoria complexa que é parte de uma categoria também complexa, mas que só toma um argumento posterior simples,[6] do tipo (X/Y)/Z ou (X\Y)/Z, recapitularemos aqui as demonstrações das regras de divisão R5 e R6.[7]

3.1 DEMONSTRAÇÃO DE R5

A regra R5 é uma regra unária que se aplica a uma categoria complexa (do tipo X/Y ou Y \X), tornando o resultado ainda mais complexo pelo acréscimo de uma mesma categoria em ambos os lados do conectivo. A idéia, como já foi mencionado, é a mesma que a das frações: se dividirmos

[6] Como, no caso que nos interessa especificamente aqui, a função principal só toma seu argumento depois do funtor ((N\N)/N e ((N\S)\(N\S))/N) e este argumento não é complexo, não apresentaremos as demonstrações dos outros casos possíveis da generalização (quando o argumento é complexo); no entanto, o leitor que compreender as demonstrações apresentadas poderá facilmente executar estas outras demonstrações.

[7] Na Gramática Categorial Clássica, R5 e R6 são os nomes atribuídos por Moortgat (1988) a duas regras que podem ser deduzidas no cálculo do Lambek, que é o que será feito a seguir; estas regras são representadas, respectivamente, pelos seguintes pares de fórmulas: X/Y ⇒ (X/Z)/(Y/Z) e Y \X ⇒ (Z\Y)\(Z\X), e X/Y ⇒ (Z/X)\(Z/Y) e Y \X ⇒ (Y \Z)/(X\Z). Como já foi dito, estou me restringindo aqui apenas à parte sintática das regras.

o numerador e o denominador de uma fração por um mesmo número, a fração resultante será equivalente.

Aplicando os esquemas de derivação e os procedimentos exemplificados na demonstração da promoção de tipo, ambos apresentados na seção 2, podemos demonstrar a validade de $X/Y \Rightarrow (X/Z)/(Y/Z)$, como se pode ver na figura 3(a). Nesta derivação, a fórmula $X/Y \Rightarrow (X/Z)/(Y/Z)$ é transformada primeiro em $X/Y, Y/Z \Rightarrow X/Z$ e depois em $X/Y, Y/Z, Z \Rightarrow X$, a partir de dois usos consecutivos do esquema 4, e depois se chega aos axiomas de identidade por dois usos também consecutivos do esquema 2.

A demonstração de $Y\backslash X \Rightarrow (Z\backslash Y)\backslash(Z\backslash X)$ é apresentada na figura 3(b). Semelhantemente ao caso anterior, dois usos consecutivos do esquema 5 transformam a fórmula $Y\backslash X \Rightarrow (Z\backslash Y)\backslash(Z\backslash X)$ primeiro em $Z\backslash Y, Y\backslash X \Rightarrow Z\backslash X$ e depois em $Z, Z\backslash Y, Y\backslash X \Rightarrow X$. Ainda como antes, o esquema 3 usado duas vezes seguidas revela os axiomas de identidade, encerrando a demonstração.

3.2 DEMONSTRAÇÃO DE R6

Como a regra R6 também é uma regra unária, com as mesmas condições de aplicação que R5, a demonstração de suas duas versões será bastante semelhante às duas demonstrações anteriores. Mas, ainda que aqui também comecemos com dois usos consecutivos dos esquemas que operam no conseqüente, a diferença agora é que teremos em cada um deles uma função tomando argumentos para lados distintos.

Na figura 4(a), a demonstração começa transformando a fórmula $X/Y \Rightarrow (Z/X)\backslash(Z/Y)$ em $Z/X, X/Y \Rightarrow Z/Y$, a partir do esquema 5, e depois em $Z/X, X/Y, Y \Rightarrow Z$, pelo esquema 4. A seguir, esta fórmula é reduzida duas vezes pelo esquema 2, encerrando a demonstração com sucesso.

Para a demonstração de $Y\backslash X \Rightarrow (Y\backslash Z)/(X\backslash Z)$, na figura 4(b), esta fórmula é primeiro tranformada em $Y\backslash X, X\backslash Z \Rightarrow Y\backslash Z$, pelo esquema 4, e depois em $Y, Y\backslash X, X\backslash Z \Rightarrow Z$, pelo esquema 5. Recorrendo-se duas vezes ao esquema 3, para quando a função está no antecedente e toma um argumento anterior, a demonstração é encerrada com sucesso.

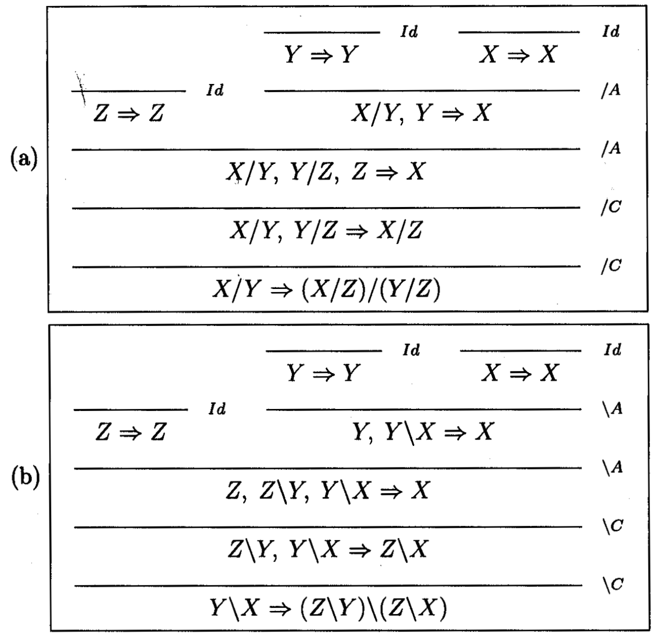

Figura 3 - Demonstração para a regra R5

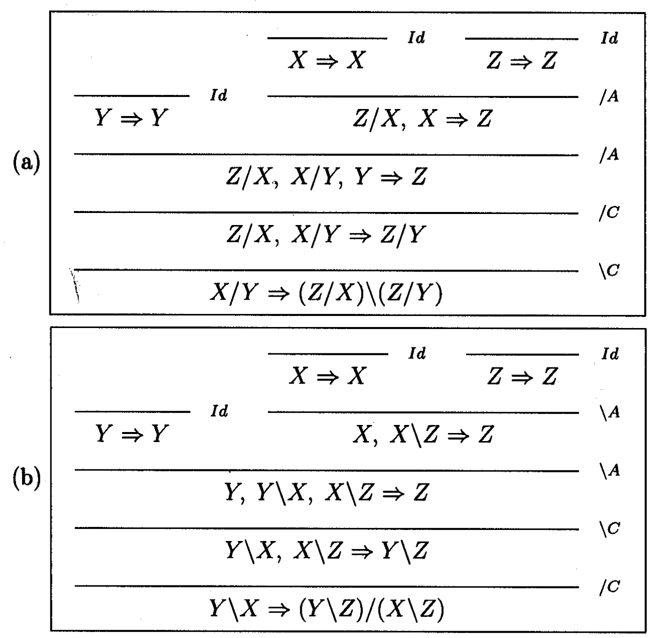

Figura 4 - Demonstrações para a regra R6

3.3 DEMONSTRAÇÃO DAS GENERALIZAÇÕES PERTINENTES

Se nos ativermos apenas ao padrão (X\Y)/W da categoria (N\N)/N (que é a categoria mais básica que deveria resultar na suposta ((N\S)\(N\S))/N),[8] e se considerarmos ainda que, neste caso, só poderíamos aplicar R5 e R6 à parte inicial deste padrão (Y\X), chegaríamos somente a duas possibilidades: (Y\X)/W ⇒ ((Z\Y)\(Z\X))/W (aplicação de R5) e (Y\X)/W ⇒ ((Y\Z)/(X\Z))/W (aplicação de R6).

A demonstração do primeiro caso aparece na figura 5. Nela, a fórmula (Y\X)/W ⇒ ((Z\Y)\(Z\X))/W é transformada em (Y\X)/W, W ⇒ (Z\Y)\(Z\X) pelo esquema 4, e depois desmembrada numa instância do axioma da identidade e na fórmula Y\X ⇒ (Z\Y)\(Z\X), pelo esquema 2. A seguir, esta última fórmula é transformada primeiro em Z\Y , Y\X ⇒ Z\X e depois em Z, Z\Y, Y\X ⇒ X, por dois usos consecutivos do esquema 3, que terminam a demonstração em três outras instâncias do mesmo axioma da identidade.

Na figura 6, vemos a demonstração de (Y\X)/W ⇒ ((Z\Y)\(Z\X))/W. Ela começa com a aplicação do esquema 4, levando à fórmula (Y\X)/W, W⇒ (Z\Y)\(Z\X), que depois é separada pelo esquema 2 numa instância do axioma da identidade e na fórmula Y\X ⇒ (Z\Y)\(Z\X). Uma série de duas aplicações do esquema 5 transforma esta última fórmula primeiro em Z\Y, Y\X ⇒ Z\X e depois em Z, Z\Y, Y\X ⇒ X. A demonstração termina com sucesso em mais três instâncias do axioma da identidade depois de mais duas aplicações do esquema 3.

[8] A impossibilidade de se provar ((N\S)\(N\S))/N ⇒ (N\N)/N pode ser comprovada pela seguinte derivação:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{N \Rightarrow N}\,Id \qquad \dfrac{}{N,\ (N\backslash S)\backslash(N\backslash S)}\,\times}{N,\ ((N\backslash S)\backslash(N\backslash S))/N,\ N \Rightarrow N}\,/A}{((N\backslash S)\backslash(N\backslash S))/N,\ N \Rightarrow N\backslash N}\,\backslash C}{((N\backslash S)\backslash(N\backslash S))/N \Rightarrow (N\backslash N)/N}\,/C$$

Observe que aqui, ao contrário das outras derivações no texto, preferi reduzir completamente a categoria no conseqüente, para só depois reduzir as categorias no antecedente; essa diferença, no entanto, só afeta a ordem de apresentação da derivação, não alterando em nada os resultados obtidos.

$$\dfrac{\dfrac{\dfrac{}{W \Rightarrow W}\,\mathit{Id} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{Z \Rightarrow Z}\,\mathit{Id} \qquad \dfrac{\dfrac{}{Y \Rightarrow Y}\,\mathit{Id} \qquad \dfrac{}{X \Rightarrow X}\,\mathit{Id}}{Y,\ Y\backslash X \Rightarrow X}\,\backslash A}{Z,\ Z\backslash Y,\ Y\backslash X \Rightarrow X}\,\backslash A}{Z\backslash Y,\ Y\backslash X \Rightarrow Z\backslash X}\,\backslash C}{Y\backslash X \Rightarrow (Z\backslash Y)\backslash(Z\backslash X)}\,\backslash C}{(Y\backslash X)/W,\ W \Rightarrow (Z\backslash Y)\backslash(Z\backslash X)}\,/A}{(Y\backslash X)/W \Rightarrow ((Z\backslash Y)\backslash(Z\backslash X))/W}\,/C$$

Figura 5 - Demonstração da aplicação de R5 à função subordinada

$$\dfrac{\dfrac{\dfrac{}{W \Rightarrow W}\,\mathit{Id} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{Y \Rightarrow Y}\,\mathit{Id} \qquad \dfrac{\dfrac{}{X \Rightarrow X}\,\mathit{Id} \qquad \dfrac{}{Z \Rightarrow Z}\,\mathit{Id}}{X,\ X\backslash Z \Rightarrow Z}\,\backslash A}{Y,\ Y\backslash X,\ X\backslash Z \Rightarrow Z}\,\backslash A}{Y\backslash X,\ X\backslash Z \Rightarrow Y\backslash Z}\,\backslash C}{Y\backslash X \Rightarrow (Y\backslash Z)/(X\backslash Z)}\,/C}{(Y\backslash X)/W,\ W \Rightarrow (Y\backslash Z)/(X\backslash Z)}\,/A}{(Y\backslash X)/W \Rightarrow ((Y\backslash Z)/(X\backslash Z))/W}\,/C$$

Figura 6 - Demonstração da aplicação de R6 à função subordinada

Como se pode perceber, nenhuma das duas fórmulas ((Y\X)/W ⇒ ((Z\Y)\(Z\X))/W e (Y\X)/W ⇒ ((Y\Z)/(X\Z))/W), que regulamentam a aplicação da idéia de dividir, no dividendo composto de uma fração,[9] o seu dividendo e o seu divisor ambos por um mesmo número é semelhante à fórmula (N\N)/N ⇒ ((N\S)\(N\S))/N, que representaria a suposta relação entre (N\N)/N e ((N\S)\(N\S))/N.

[9] *Vale aqui também uma pequena observação sobre a confusão dos nomes das partes de uma divisão: na nota referida no início do texto, se dizia "dividir, num divisor composto, o seu divisor e o seu dividendo por um mesmo número, sem que se faça a mesma divisão no dividendo principal"; no exemplo apresentado, porém, a divisão pelo mesmo número é feita no dividendo composto: "a operação (2/3)/5 é equivalente à operação ((2/7)(3/7))/5". Numa divisão, o dividendo é o número que será dividido (e que fica acima do traço da divisão; ou à esquerda, na notação linearizada), enquanto o divisor é o número pelo qual o dividendo será dividido (e que fica abaixo do traço da divisão; ou à direita, ainda na notação linearizada):* $\dfrac{\textit{dividendo}}{\textit{divisor}}$ *(ou dividendo/divisor).*

Na comparação com a primeira, a ordem dos conectivos é a mesma em ambas as fórmulas, mas as categorias que são introduzidas na parte direita da fórmula (Z e S), depois do $\Rightarrow$, não se equiparam: o Z aparece antes do \, na primeira, enquanto o S vem depois, na segunda, como podemos observar logo abaixo.[10]

$$(Y\backslash X)/W \Rightarrow ((Z\backslash Y)\backslash(Z\backslash X))/W$$
$$(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N$$

Na comparação com a segunda, ainda que não haja problemas em relação à identidade das categorias que são introduzidas na parte direita da regra, o que não bate agora é a ordem dos conectivos; mais especificamente, apenas o segundo conectivo da parte direita não é o mesmo de uma fórmula para a outra, como se observa logo a seguir.

$$(Y\backslash X)/W \Rightarrow ((Y\backslash Z)/(X\backslash Z))/W$$
$$(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N$$

Assim, percebe-se que as duas possibilidades pertinentes da aplicação da idéia da divisão do dividendo não chegam ao resultado intuído inicialmente.

4 CONSTATAÇÃO DA INDEMONSTRABILIDADE

Uma alternativa mais direta para se constatar a impossibilidade da relação entre $(N\backslash N)/N$ e $((N\backslash S)\backslash(N\backslash S))/N$ consistiria em tentar demonstrar, sem sucesso, a fórmula $(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N$. Isso é feito na figura 7.

$$\dfrac{\dfrac{}{N \Rightarrow N}\,Id \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{N \Rightarrow N}\,Id \qquad \dfrac{}{S, N\backslash N \Rightarrow S}\,\times}{N, N\backslash S, N\backslash N \Rightarrow S}\,\backslash A}{N\backslash S, N\backslash N \Rightarrow N\backslash S}\,\backslash C}{N\backslash N \Rightarrow (N\backslash S)\backslash(N\backslash S)}\,\backslash C}{\dfrac{(N\backslash N)/N, N \Rightarrow (N\backslash S)\backslash(N\backslash S)}{(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N}\,/C}\,/A$$

Figura 7 - Indemonstrabilidade de $(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N$

[10] *Numa terminologia típica do Prolog, se considerássemos X, Y, W e Z variáveis, e N e S constantes, poderíamos dizer que as duas fórmulas não seriam unificáveis, porque tanto o Y quanto o X teriam de valer N do lado esquerdo do $\Rightarrow$, mas S do seu lado direito.*

Nesta última derivação, começamos transformando a fórmula (N\N)/N) ((N\S)\(N\S))/N em (N\N)/N,N ⇒ (N\S)\(N\S), pelo esquema 4, para em seguida desmembrá-la no axioma da identidade N ⇒ N e na fórmula N\N) (N\S)\(N\S), pelo esquema 2. Depois de mais duas aplicações do esquema 5, esta última fórmula é transformada primeiro em N\S,N\N ⇒ N\S e depois em N,N\S,N\N ⇒ S. Finalmente, na última operação possível para esta derivação, o uso do esquema 3 separa uma outra ocorrência do axioma da identidade N ⇒ N e a fórmula S,N\N ⇒ S. Como, nesse ponto da derivação, não é possível empregar mais nenhum dos esquemas do cálculo,[11] mas como um dos ramos do diagrama termina com uma fórmula que não é o axioma da identidade, essa derivação não se constitui numa demonstração; assim, (N\N)/N ⇒ ((N\S)\(N\S))/N não é um teorema válido – ou seja, de (N\N)/N não se chega a ((N\S)\(N\S))/N.

Observando a fórmula que restou no topo do diagrama (S,N\N ⇒ S), observa-se que, se não houvesse aquele N\N, teríamos chegado a mais uma instância do axioma da identidade, o que teria resultado na demonstração de (N\N)/N ⇒ ((N\S)\(N\S))/N. Essa observação reforça a conclusão das comparações feitas no final da seção anterior, segundo a qual o que há de errado é a inserção de uma categoria no lado inadequado; por isso, em vez de podermos cancelar um N com o outro, sobram os dois como N\N.

5 DEMONSTRADOR EM PROLOG

Para encerrarmos esse nosso exercício de confirmação de uma intuição em relação à nossa teoria, vamos apresentar um pequeno programa em Prolog que funciona como um demonstrador de teoremas para o tipo de fórmulas que estávamos usando nas demonstrações acima.[12] A listagem do programa é dada a seguir.

[11] *Esta condição é representada no diagrama pelo símbolo ×.*

[12] *O Prolog usado aqui foi o SWI-Prolog, que, além de ser gratuito, oferece a vantagem de uma interface gráfica que facilita a visualização das unificações – ou seja, de como as variáveis dos esquemas assumem os valores que resultam nos teoremas a serem demonstrados. Esse Prolog pode ser obtido na internet a partir do endereço http://www.swi-prolog.org. No entanto, o programa apresentado aqui deve funcionar em qualquer outra versão de Prolog; em alguns casos, talvez, as definições de append/3 e lenght/2 podem não estar pré-construídas (built-in) – neste caso, basta acrescentar ao programa as definições abaixo. (Para uma introdução ao Prolog, ver o manual de Clocksin e Mellish (1987).)*

```
append([ ], [ ], Lista).
append([Primeiro | Resto1], Lista, [Primeiro | Resto2]) :-
append(Resto1, Lista, Resto2).
lenght([ ], 0).
lenght([_ | Resto], N) :-
lenght(Resto, M).
N is M + 1
```

```
:- op(600, xfx, ==>).
:- op(550, xfx, /).
:- op(550, xfx, \).
% Axioma da identidade
[A] ==> A :-
!.
% Função de argumento posterior no antecedente
Gama ==> C :-
append(Gama1, [A/B | Resto], Gama),
append(Delta, Gama2, Resto),
Delta ==> B,
append(Gama1, [A | Gama2], Lista),
Lista ==> C.
% Função de argumento anterior no antecedente
Gama ==> C :-
append(Gama1, Resto, Gama),
append(Delta, [B\A | Gama2], Resto),
Delta ==> B,
append(Gama1, [A | Gama2], Lista),
Lista ==> C.
% Função de argumento anterior no conseqüente
Gama ==> A\B :-
( length(Gama, 0)
-> fail
; [A | Gama] ==> B
).
% Função de argumento posterior no conseqüente
Gama ==> B/A :-
( length(Gama, 0)
-> fail
; append(Gama, [A], Lista),
Lista ==> B).
```

As primeiras três cláusulas do programa definem os operadores que serão usados: os conectivos de construção das categorias complexas (/ e \) e o símbolo $\Rightarrow$ (que, no programa, é representado por ==>). Na seqüência, são definidos os cinco esquemas de derivação; a principal diferença notacional, já que, nos esquemas apresentados na seção 2, o lado esquerdo do símbolo $\Rightarrow$ apresenta uma lista de categorias, é o uso dos colchetes, que, em Prolog, é o recurso para representar as listas.

Assim, o axioma da identidade é representado no programa de forma quase completamente idêntica ao esquema 1 (o ! – cut – indica que, depois de satisfeita esta cláusula, nenhuma outra precisa ser mais testada).

Na definição dos esquemas para o conseqüente há uma estrutura comum, responsável por implementar a exigência do Γ *não vazio: se o tamanho da lista* Gama *for 0* (length(Gama, 0))*, então esta cláusula não pode ser satisfeita (*fail*); caso contrário, o seu processamento pode continuar. Para o esquema da função de argumento anterior, basta acrescentar a categoria A ao começo da lista de categorias* Γ*, o que é feito por* [A | Gama] ==> B. *Como, no esquema para a função com argumento posterior, o acréscimo precisa ser feito no final da lista, é preciso usar o predicado append/3:* append(Gama, [A], Lista); *assim, no final, para satisfazer* Gama ==> B/A, *é preciso então satisfazer* Lista ==> B.

Para os esquemas relativos ao antecedente, suas definições também apresentam uma estrutura comum: em ambas é preciso separar da lista inicial Gama *as sublistas* Gama1, Gama2 *e* Delta *(relativas a* Γ_1*,* Γ_2 *e* Δ*), além das categorias* A\B *e* B/A. *Quando a função é de argumento posterior, a lista de categorias* Gama *precisa ser decomposta na lista* Gama1 *e em outra lista* [A/B | Resto] (append(Gama1, [A/B | Resto], Gama))*, e, por sua vez, a lista* Resto *também precisa ser decomposta nas listas* Delta *e* Gama2 (append(Delta, Gama2, Resto))*; desta forma, a satisfação de* Gama ==> C *depende da satisfação de* Delta ==> B *e de* Lista ==> C*, de modo que* Lista *contenha o resultado da concatenação de* Γ_1*, A e* Γ_2 (append(Gama1, [A | Gama2], Lista))*. Na função de argumento anterior, a lista de categorias* Gama *é separada nas listas* Gama1 *e* Resto (append(Gama1, Resto, Gama))*, e esta última ainda é decomposta em* Delta *e em* [B\A | Gama2] (append(Delta, [B\A | Gama2], Resto))*; como no caso da função de argumento posterior, a satisfação de* Gama ==> C *também depende da satisfação de* Delta ==> B *e de* Lista ==> C.

5.1 Exemplos de uso do demonstrador

Podemos usar o programa acima para, por exemplo, demonstrar que N, N\S ⇒ S. Depois de carregá-lo no interpretador de Prolog, essa demonstração pode ser solicitada da seguinte maneira:

```
?- [n, n\s] ==> s.<enter>
Yes
```

A resposta Yes que obtemos significa que o interpretador conseguiu provar esta fórmula. A título de ilustração, vejamos como esta demonstração é executada.

A cláusula cuja prova foi requerida unifica com a cabeça da definição da função com argumento anterior no antecedente, a partir das seguintes atribuições de valores às variáveis:

- Gama = Resto = [n, n\s],
- C = A = s,
- Gama1 = Gama2 = [],
- Delta = [n],
- B = n,
- Lista = [s].

A partir desta unificação, duas novas cláusulas precisam ser provadas: [n] ==> n (Delta ==> B) *e* [s] ==> s (Lista ==> C)*, que são claramente instâncias do axioma da identidade (primeiro para a unificação* A = n*, e depois para* A = s)*. Como todos os objetivos são completamente satisfeitos, a solicitação termina, indicando o seu sucecsso.*

Vejamos agora como é executada a demonstração de $X \Rightarrow Y/(X\backslash Y)$*, que é solicitada da seguinte forma.*

?- [x] ==> y/(x\y).<enter>

Yes

Esta cláusula unifica com a cabeça da definição da função com argumento posterior no conseqüente, para as seguintes atribuiçõs de valores às variáveis:

- Gama = [x],
- B = y,
- A = x\y,
- Lista = [x, x\y].

Depois de constatado que a lista Gama *não é vazia (*length(Gama, 0))*, o resultado de sua concatenação a* [A] *aparece em Lista; assim, a nova cláusula a ser satisfeita é* [x, x\y] ==> y (Lista ==> B)*. Esta nova cláusula unifica com a cabeça da definição para a função com argumento anterior no antecedente, atribuindo os seguintes valores às variáveis:*

- Gama = Resto = [x, x\y],
- C = A = y,
- Gama1 = Gama2 = [ ],
- Delta = [x],
- B = x,
- Lista = [y].

Assim, chegamos novamente a duas instâncias do axioma da identidade [x] ==> x (Delta ==> B) *e* [y] ==> y (Lista ==> C)*, terminando com* *sucesso a demonstração, com todas as cláusulas exaustivamente satisfeitas.*

Como o procedimento é bastante mecânico, e mais fácil de ser acompanhado usando os recursos do próprio interpretador de Prolog,[13] *parece excessivo comentar também passo-a-passo o processamento das quatro cláusulas das regras R5 e R6. O leitor curioso que tiver interesse pode constatar que o programa também é capaz de demonstrá-las a partir de:*

```
?- [x/y] ==> (x/z)/(y/z).<enter>,
?- [x\y] ==> (z\y)\(z\x).<enter>,
?- [x/y] ==> (z/x)\(z/y).<enter>,
?- [y\x] ==> (y\z)/(x\z).<enter>.
```

Fica também a cargo do leitor a execução das provas de $(Y\backslash X)/W \Rightarrow ((Z\backslash Y)\backslash(Z\backslash X))/W$ *e* $(Y\backslash X)/W \Rightarrow ((Y\backslash Z)/(X\backslash Z))/W$*, invocadas respectivamente por:*

```
?- (y\x)/w ==> ((z\y)\(z\x))/w.<enter>,
?- (y\x)/w ==> ((y\z)/(x\z))/w.<enter>.
```

5.2 Processamento da indemonstrabilidade

Antes de passarmos às conclusões, seria interessante observar mais detidamente a impossibilidade de se provar $(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N$*:*[14]

```
?- [(n\n)/n] ==> ((n\s)\(n\s))/n.<enter>
No
```

Esta cláusula unifica com a cabeça da definição para uma função de argumento posterior no conseqüente, atribuindo os seguintes valores às suas variáveis:

- Gama = [(n\n)/n],
- B = (n\s)\(n\s),
- A = n,
- Lista = [(n\n)/n, n].

[13] *Caso se esteja usando o SWI-Prolog, é possível usar o seu acompanhador gráfico (graphical tracer), solicitado por meio dos comandos a seguir.*

```
?- guitracer.<enter>
?- trace.<enter>
```

Logo após estes comandos, a execução de qualquer prova abre uma janela na qual se podem acompanhar todas as unificações que vão sendo feitas à medida que as cláusulas vão sendo processadas.

[14] *Também é possível testar a indemonstrabilidade de* $((N\backslash S)\backslash(N\backslash S))/N \Rightarrow (N\backslash N)/N$*, usando* ?- [((n\s)\(n\s))/n] ==> (n\n)/n.<enter>.

Com isto, gera-se um novo objetivo: satisfazer a cláusula [(n\n)/n, n] ==> (n\s)\(n\s) (Lista ==> B). *Como esta cláusula unifica com a cabeça da definição para as funções com argumento posterior no antecedente, suas variáveis recebem os seguintes valores:*

- Gama = [(n\n)/n, n],
- C = (n\s)\ (n\s),
- Gama1 = Gama2 = [],
- A = n\n,
- B = n,
- Resto = Delta = [n],
- Lista = [n\n].

Chega-se assim a dois novos objetivos: [n] ==> n (Delta ==> B) *e* [n\n] ==> (n\s)\(n\s) (Lista ==> C). *Como a primeira cláusula é uma instanciação do axioma da identidade, ela é facilmente satisfeita. Com a unificação da segunda cláusula com a cabeça da definição para funções com argumento anterior no conseqüente, chegamos a uma nova atribuição de valores às variáveis:*

- Gama = [n\n],
- A = B = n\s.

Passamos então a ter um novo objetivo: satisfazer [n\s, n\n] ==> n\s ([A | Gama] ==> B). *Pela sua unificação novamente com a cabeça da definição para funções com argumento anterior no conseqüente, obtemos uma nova série de atribuições:*

- Gama = Resto = [n, n\s, n\n],
- C = A = s,
- Gama1 = [],
- Delta = [n],
- B = n,
- Gama2 = [n\n],
- Lista = [s, n\n].

Depois disso, passamos a ter mais dois objetivos: [n] ==> n (Delta ==> B) *e* [s, n\n] ==> s (Lista ==> C). *Novamente o primeiro deles é o axioma da identidade, e o segundo agora não é capaz de unificar com a cabeça de nenhuma das definições. Como não existem outras alternativas disponíveis,*[15] *chegamos à resposta No, que indica que o interpretador de*

[15] Na verdade, observamos apenas as principais unificações, dentre todas as suas possibilidades. Mas isso não interfere no fato de que, na presente prova, nenhuma unificação possível leva à demonstração requerida.

Prolog não foi capaz de satisfazer a cláusula usando o programa apresentado aqui. Ou seja, se o programa está correto, $(N\backslash N)/N \Rightarrow ((N\backslash S)\backslash(N\backslash S))/N$ não é demonstrável.

6 CONCLUSÕES

De forma alguma se pretendeu aqui invalidar a solução proposta para o efeito labirinto apresentada por Gonçalves e Pagani (2004). Mesmo sem a derivação de uma categoria a partir da outra, ainda haveria uma alternativa de processamento, supondo que houvesse dois itens lexicais, ambos com a mesma forma past, mas cada um de uma dessas categorias, e sua escolha poderia ser resolvida pelos critérios do analisador gramatical, tornando desnecessária sua atribuição a uma relação entre categorias que se demonstrou impossível para uma Gramática Categorial.[16] O objetivo principal foi mesmo o de relatar o percurso que levou à constatação do equívoco em relação à intuição de que de $(N\backslash N)/N$ se chegaria a $((N\backslash S)\backslash(N\backslash S))/N$.

Neste relato, esperava-se descrever os trabalhosos (tediosos, diriam alguns) exercícios de demonstração dos teoremas (ou da impossibilidade de uma prova, como era o caso aqui) e de acompanhamento da execução do demonstrador em Prolog. O objetivo secundário, segundo sugere o subtítulo do presente texto, era o de exemplificar como essas tarefas podem auxiliar o lingüista a não cometer esse tipo de equívoco (ou, ainda no nosso caso, admitir o erro cometido).

Se, por um lado, o emprego da lógica e da computação exige dos lingüistas um conhecimento para o qual eles ainda não estão sendo institucionalmente instruídos,[17] por outro, estas ferramentas retribuem o esforço despendido em sua utilização por meio da precisão que a primeira oferece e da automatização das tarefas oferecida pela segunda.

Esse tipo de ferramenta nos possibilita ainda supor que uma das fontes do equívoco talvez tenha sido a diferença entre a mencionada divisão do dividendo e do divisor, de um dividendo composto, por um mesmo número (para a qual o lado em que a operação ocorre é indistinto) e a natureza das categorias (que é sensível ao lado pelo qual o argumento é tomado).

16 Contudo, devo ressaltar que está ainda é uma opinião intuitiva, que também deve ser submetida à comprovação formal – mais provavelmente, pela implementação de um analisador gramatical que executasse a operação descrita.

17 Por institucionalmente intruídos, pretendia me referir tanto ao fato de que nos cursos de Lingüística (tanto de graduação, quanto de pós-graduação) não se costuma ensinar esse tipo de habilidade, quanto à constatação de que a própria Lingüística (enquanto instituição que se manifesta em congressos e publicações que adotam esse rótulo) nem sempre vê com bons olhos o emprego desses recursos.

Podemos ainda atribuir uma parte desse erro às diferenças entre a notação usada no artigo mencionado para as categorias (a do Steedman) e a empregada nas apresentações das regras R5 e R6 (tanto Borges Neto (1999) quanto Moortgat (1988) escolheram a do Lambek; Wood (1993), porém, apresenta as regras nas duas notações).

Finalmente, o objetivo mais remoto (mas não por isso o menos importante) sempre é fazer com que o leitor se interesse pelo assunto e busque, a partir das referências bibliográficas mencionadas aqui, aprofundar sua compreensão sobre estas questões, para que tenhamos cada vez mais interlocutores para discutir este tipo abordagem para os fenômenos lingüísticos.

RESUMO

O presente texto procura corrigir uma pequena falha cometida em uma nota do artigo "O efeito labirinto além da sintaxe"(Letras, n. 63, p. 177-195). Nesta nota se afirmava que deveria ser possível derivar uma certa operação na Gramática Categorial; o que se demonstra aqui é que, infelizmente, essa derivação é impossível.

Palavras-chave: sentenças-labirinto; processamento de língua natural; gramática categorial.

ABSTRACT

The present essay aims to rectify a mistake made at a footnote of the article "Garden-path effect beyond syntax" (Letras, n. 63, p. 177-195). At that footnote, it was alleged that it could be possible to derive a certain operation in Categorial Grammar– unfortunately, this derivation is proved to be impossible.

Key-words: garden-path sentences; natural language processing; categorial grammar.

REFERÊNCIAS

BORGES NETO, J. *Introdução às gramáticas categoriais*. Curitiba, 1999.

CARPENTER, B. *Type-logical semantics*. Cambridge, Mass.: The MIT Press, 1997.

CLOCKSIN, W.; MELLISH, C. *Programming in Prolog*. Berlin: Springer, 1987.