informações linguísticas (tais como tempo, flexão, definitude, gênero). Tal seleção possui uma natureza puramente gramatical, isto é, não há restrições ao tipo semântico do argumento selecionado – logo, as categorias em questão (T e D) são funcionais.

O complemento de um ST é sempre um SV, enquanto o complemento de um SD é sempre um SN. É em função dessa relação entre sintagmas funcionais e sintagmas lexicais que, normalmente, os SVs são realizados nas línguas naturais em alguma forma verbal específica (num dado tempo, com algum modo e aspecto, em alguma pessoa e número gramaticais) e os SNs realizam-se por meio de uma referência concreta (são definidos ou indefinidos, apresentam uma quantidade específica, pertencem ao masculino ou feminino, estão no singular ou plural).

Ao descrevemos os sintagmas lexicais e funcionais, utilizamos aqui a sigla em língua portuguesa da respectiva abreviatura do sintagma: SN, SV, SA, SP, SC, ST (ou SF) e SD. Ao consultar bibliografia relevante sobre sintaxe gerativa, pode ser que você encontre essas abreviaturas conforme o original em inglês. É o que acontece com o importante livro *Novo manual de sintaxe* (Mioto; Silva e Lopes, 2013). Nesse caso, você deve ter em mente que, em inglês, o termo "sintagma" é traduzido com a palavra *phrase*. Sendo assim, as traduções equivalentes serão as seguintes.

SN = NP (*noun phrase*)
SV = VP (*verb phrase*)
SA = AP (*adjectival phrase*)
SP = PP (*prepositional phrase*)
SC = CP (*complementizer phrase*)
ST = TP (*temporal phrase*)
SF = IP (*inflectional phrase*)
SD = DP (*determiner phrase*)

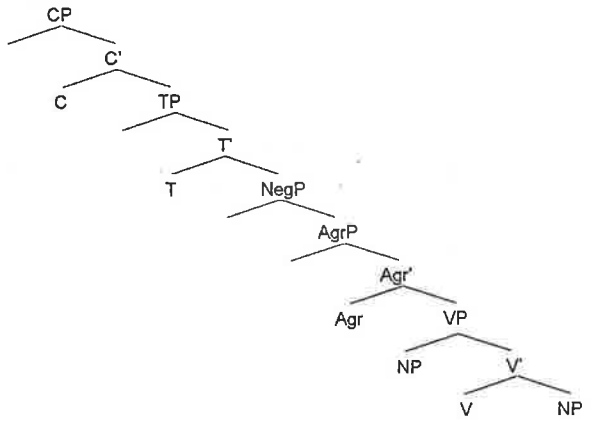
Note que, em inglês, a abreviatura dos sintagmas sempre termina na letra "P", de *phrase* (tradução de sintagma, e não frase – não confunda!). É por essa razão que, jocosamente, as tecnicalidades dos sintaticistas de orientação gerativista são chamadas de "língua do pê".

Na verdade, existem outros sintagmas funcionais importantes nas línguas naturais. Contudo, o estudo mais aprofundado das categorias gramaticais deve acontecer noutro momento, num estágio mais avançado da sua formação acadêmica. Por ora, você deve ter em mente que SC, ST e SD são os principais sintagmas funcionais computados pela linguagem humana. Quando, na próxima unidade, analisarmos as computações sintáticas que realizam a derivação de representações linguísticas, veremos que existem muitas relações estruturais entre categorias lexicais e funcionais. Portanto, fixe esses sintagmas em sua memória e em seu conhecimento.

Para finalizarmos esta unidade, resta-nos descrever rapidamente os argumentos e os adjuntos que se apresentam numa estrutura sintagmática de tal maneira complexa que se caracterizam como orações: os argumentos e adjuntos oracionais. Trata-se, na verdade, de mais uma instância de nosso conhecido fenômeno da recursividade, a qual recebe bastante destaque nas aulas de língua portuguesa sob o nome de *orações subordinadas*. Este é o tópico da próxima seção.

A cisão dos sintagmas

Na literatura gerativista, desde o trabalho seminal de Pollock (1989), os sintagmas SC, ST e SV vem sendo reanalisados e descritos por meio de outros sintagmas mais específicos que os constituem. Assim, por exemplo, o SC pode ser dividido num "sintagma de tópico' (STop) e um "sintagma de foco" (SFoc). O ST pode ser reinterpretado como um sintagma de concordância para o sujeito (AgrS) e um sintagma de concordância para o objeto (AgrO) – note que *Agree* é o termo inglês que significa *concordância*. Também o SV vem sendo redescrito como a articulação entre um verbo leve ("v" – lido como *vezinho*), que dá origem ao Sv, que domina o SV. Na figura abaixo, você pode conferir a estrutura de uma sentença com o ST (TP) cindidos num sintagma de negação (NegP) e num sintagma de concordância (AgrP).



Exemplo de um sintagma cindido de acordo com Pollock (1989).

A motivação para a cisão de sintagmas depende do objetivo específico do sintaticista, em sua tarefa de explicar um determinado fenômeno. Pollock, por exemplo, teve a intenção de indicar que o sintagma de tempo comportava, na verdade, mais do que somente a expressão do tempo verbal. Ele dava conta também da negação, da concordância, do aspecto, dentre outras noções funcionais.

computação binária que constrói um sintagma com a propriedade fundamental da recursividade, chegaremos à construção de estruturas tão intrincadas como a oração e o período composto. Dominar a noção de sintagma, diferenciar sintagmas lexicais e funcionais, identificar a fronteira entre orações etc. são apenas instrumentos metalinguísticos úteis e imprescindíveis para compreendermos como é que a nossa cognição produz e compreende frases no nosso uso cotidiano da linguagem. Esperemos que você tenha já o domínio dessas ferramentas. Você fará uso corrente delas na próxima unidade e durante toda a sua vida acadêmica como estudioso da sintaxe, caso deseje prosseguir com seus estudos.

Exercícios

1) O que se deve entender por "sintaxe" nos estudos de linguística gerativa?
2) Quais são as unidades mínimas e máximas da análise sintática?
3) O que se deve entender pela noção de sintagma?
4) Use os testes de identificação de sintagma para descrever a ambiguidade estrutural presente na seguinte frase.
"Recebi uma fotografia de Petrópolis."
5) O que os sintaticistas querem dizer quando afirmam que a constituição de sintagmas é sempre binária e pode ser recursiva? Dê um exemplo.
6) Identifique os sintagmas presentes nas estruturas que se seguem.
a) Lemos muitos textos.
b) Lemos muitos textos de linguística.
c) Lemos muitos textos de linguística gerativa.
7) Represente os seguintes sintagmas em árvores sintáticas.
a) Vi televisão.
b) João gosta de doces.
c) Paulo dormiu por horas.
8) Descreva os três sintagmas funcionais apresentados nesta unidade. Dê um exemplo de cada.
9) Como você compreendeu a diferença entre períodos simples e períodos compostos a partir desta unidade? O que essa diferença tem a ver com o fenômeno da recursividade?

UNIDADE 9
O Sistema Computacional

Na unidade anterior, aprendemos os fundamentos da análise sintática gerativista. Estudamos os diferentes tipos de sintagma existentes nas línguas naturais e aprendemos a desenhar suas relações estruturais por meio dos diagramas arbóreos – as árvores sintáticas. Todo esse instrumental técnico que acabamos de estudar possui uma função bem clara no empreendimento da linguística gerativa: trata-se de ferramentas que nos ajudam a compreender e representar o conhecimento sintático que inconscientemente dominamos quando falamos uma dada língua. Isso quer dizer que precisamos saber identificar sintagmas, desenhar árvores e delimitar orações num dado período porque essa é uma habilidade muito útil, indispensável ao sintaticista que assume a tarefa de desvendar os mecanismos cognitivos por meio dos quais os humanos são capazes de produzir e compreender frases.

É correto dizer que tudo o que aprendemos na unidade 8 foi descrever formalmente as estruturas sintáticas da linguagem humana. Pois bem, na presente unidade, passaremos da descrição linguística **formal** para a descrição computacional das operações sintáticas subjacentes ao nosso uso da linguagem. Chegamos, portanto, ao momento do curso em que faremos uso do que já aprendemos sobre léxico e sintaxe para começar a analisar o Sistema Computacional da linguagem humana.

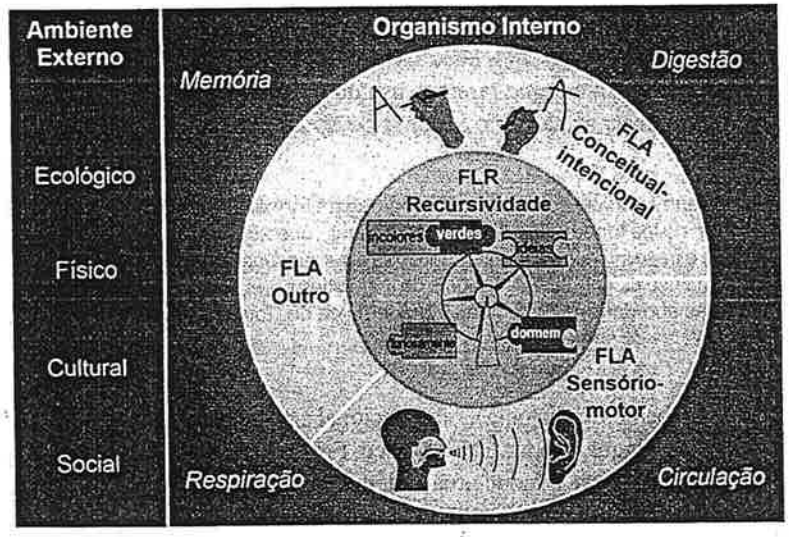
> Dizemos que a sintaxe gerativa é uma abordagem *formal* acerca da linguagem humana porque se ocupa fundamentalmente das "formas" que estruturam fonemas, morfemas, palavras, e frases – independentemente das diversas *funções* que essas formas podem assumir no uso da língua. Os manuais de linguística normalmente opõem à abordagem "formal" as abordagens "funcionais" que, em vez da "forma", privilegiam as "funções" comunicativas da linguagem.

Como já sabemos, o Sistema Computacional é somente um dentre os diversos componentes da linguagem humana. Léxico, Forma Fonética (FF), Forma Lógica (FL) e sistemas de interface (conceitual-intencional e articulatório-perceptual) são outros elementos no complexo edifício de nosso conhecimento linguístico. Cada

um deles é responsável por uma determinada fração no conjunto de habilidades que compõem a nossa cognição linguística. Ocorre que, para o gerativismo, o Sistema Computacional ocupa uma posição privilegiada na arquitetura da linguagem. Conforme já estudamos, o modelo de língua-I formulado por Chomsky assume que a sintaxe seja o componente central na estrutura e no funcionamento de qualquer língua natural. Essa hipótese é de tal maneira importante e forte para os gerativistas que, no ano de 2002, Chomsky elaborou uma nova dicotomia para destacar a proeminência do Sistema Computacional (e em especial a recursividade) em relação aos demais sistemas linguísticos: a distinção entre faculdade da linguagem *lato sensu* e faculdade da linguagem *stricto sensu*.

De acordo com Chomsky, todos os sistemas cognitivos envolvidos direta ou indiretamente no conhecimento linguístico humano compõem um vasto campo de nossa cognição denominado "faculdade da linguagem em sentido amplo" (*lato sensu*). Assim, Léxico, Sistema Computacional, FF, FL e Interfaces comporiam o grande conjunto da linguagem humana em seu *sentido amplo*: a FLA. Porém, um desses componentes é de tal forma importante para o funcionamento das línguas naturais que se torna necessário denominá-lo de maneira especial. Esse componente é o Sistema Computacional (e a recursividade de suas operações). Chomsky (2002) propôs que tal Sistema seja referido como "faculdade da linguagem em sentido restrito" (*stricto sensu*), a FLR.

Figura 9.1: Faculdade da linguagem em sentido amplo (FLA) e Faculdade da linguagem em sentido restrito (FLR).



Fonte: Chomsky, Hauser e Fitch, 2002, p. 1570. (adaptado)

FLR, isto é, o Sistema Computacional com as suas operações recursivas, é o componente linguístico dedicado à criação de estruturas sintáticas como sintagmas e sentenças. Na dinâmica de funcionamento da linguagem, esse Sistema recebe do Léxico as unidades operacionais mínimas (palavras) com as quais é capaz de criar representações complexas. Tais representações, por sua vez, são enviadas pelo Sistema para as interfaces cognitivas responsáveis pela articulação fonológica das frases e pela integração discursiva de nossas elocuções. Isso quer dizer que o Sistema Computacional é uma espécie de máquina (ou software, usando uma metáfora da informática) capaz de gerar informação sintática a partir de informações lexicais, criando representações linguísticas que podem ser usadas pelos sistemas fonético-fonológicos e semântico-discursivos. Esse Sistema é, por conseguinte, o elemento que integra todos os diferentes componentes da arquitetura da linguagem.

"Ideias verdes incolores dormem furiosamente"

Em seu texto inaugural da linguística gerativa (1957), Chomsky apresentou essa frase estranha como argumentação em favor da relativa independência entre sintaxe e semântica. A frase, como vemos, é perfeita do ponto de vista sintático, mas anômala semanticamente – afinal, "ideias" não têm cores, e "verdes" não podem ser incolores, além disso "ideias" não dormem e não se pode "dormir furiosamente". A única maneira de conferir algum significado a essa estrutura sintática é imaginar conotações metafóricas ou contextos imaginativos livres, que, portanto, ultrapassarão o valor semântico básico de cada palavra e serão licenciadas em FL.

Diante dessa breve descrição acerca da FLR, você talvez esteja se perguntado: mas de que maneira, afinal, o Sistema Computacional gera sintagmas e frases? Quais são as operações desse Sistema e como ele se relaciona com as suas interfaces? Pois bem, é isso o que veremos em detalhes ao longo desta unidade. Com efeito, muitos dos conceitos e termos técnicos que serão aqui estudados com vagar já foram introduzidos brevemente ao longo do curso. Teremos agora a oportunidade de retomar os temas e as figuras já apresentados para integrá-los e desenvolvê-los com mais detalhes e aprofundamento.

Estrutura profunda *versus* estrutura superficial

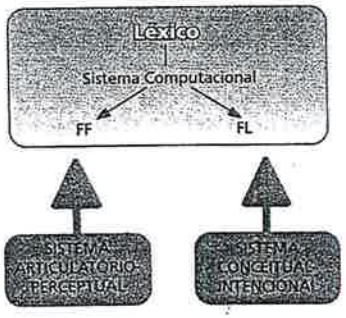
Nos modelos gerativistas anteriores ao minimalismo, Chomsky defendia a existência de dois outros níveis de representação linguística, além de π e λ. Eram eles: estrutura profunda e estrutura superficial. A estrutura profunda era descrita como aquela em que as informações semânticas básicas do Léxico seriam representadas, tais como a saturação de argumentos e a atribuição de papéis temáticos. Sobre essa estrutura, o Sistema Computacional, então chamado simplesmente de "sintaxe", aplicaria diversas operações, tais como apagamento, substituição e deslocamento, as quais produziriam uma nova representação, a estrutura superficial. Essa estrutura superficial seria então enviada para FF e FL, a fim de sofrer novas modificações antes da pronúncia e da interpretação final. Para saber mais sobre esse momento na história do gerativismo, veja Mioto, Silva e Lopes (2013).

Assumindo que o conhecimento sintático existente em nossas mentes seja eminentemente derivacional, precisamos então caracterizar quais são as operações utilizadas pelo Sistema Computacional durante a derivação de sintagmas e sentenças. Da mesma forma, precisamos descrever em que momento, no curso de uma derivação, essas operações computacionais podem ou devem ser ativadas. É precisamente isso o que começaremos a estudar na seção seguinte.

Operações computacionais

Pelo que você já aprendeu ao longo deste curso, é possível compreender que o Sistema Computacional constrói representações sintáticas derivacionalmente, a partir de unidades retiradas do Léxico. O Léxico é, por assim dizer, a fonte de alimentação do Sistema Computacional. É de lá que esse Sistema retira todas as informações necessárias para disparar suas operações computacionais.

Figura 9.2: A arquitetura da linguagem.



Entretanto, o acesso do Sistema Computacional aos itens do Léxico que devem compor uma dada representação linguística é bastante restrito e ordenado. Ora, o Sistema não poderia acessar todo o Léxico de uma só vez, em suas centenas de milhares de unidades, para selecionar apenas as seis ou sete, dez ou mesmo vinte palavras que devem compor uma sentença específica. Imagine, por exemplo, como seria dispendioso se você quisesse criar uma frase em sua língua materna e, para isso, tivesse de vasculhar um volumoso dicionário da língua portuguesa, inspecionando uma a uma todas as palavras ali presentes até que encontrasse as que desejasse usar. Seria uma atividade muito demorada, tediosa ou mesmo inútil, não? Com essa alegoria, o que queremos dizer é que, na cognição linguística humana, deve existir uma instância intermediária entre o Léxico e o Sistema Computacional. Essa instância deve conter apenas o pequeno conjunto das informações lexicais necessárias para alimentar uma derivação sintática específica. Tal instância é denominada Numeração.

Conforme aprendemos na unidade 6, a Numeração é, fundamentalmente, o conjunto de referência que contém as informações linguísticas que orientarão o Sistema Computacional durante a derivação de uma estrutura sintática particular. Para entender melhor isso, lembre-se de um assunto muito importante que estudamos na unidade 7: a noção de "traço" linguístico. São os traços formais de um dado item lexical (como estrutura argumental e grade temática, dentre outros) que dispararão as operações do Sistema. Ora, os traços de um item lexical se tornam visíveis para o Sistema Computacional justamente quando estão inseridos numa Numeração e são de lá introduzidos no espaço derivacional da sintaxe. Podemos compreender a Numeração como a compilação do conjunto de instruções que o Sistema Computacional deve seguir no curso de uma derivação. Mas note bem: a Numeração não é em si mesma uma operação computacional, tampouco uma representação sintática. Ela é somente uma instância intermediária entre o Léxico e o Sistema Computacional. Do ponto de vista psicológico, a Numeração é uma entidade abstrata que corresponde aproximadamente ao nosso planejamento de fala – aqueles milésimos de segundo inconscientes em que os itens que vamos usar numa frase são selecionados.

Figura 9.3: Numeração, uma instância entre o Léxico e o Sistema Computacional.

Os sintaticistas representam uma Numeração pela letra "N", à qual se seguem, entre chaves, os itens a serem usados na derivação. Por exemplo, sabemos que, na representação "João ama Maria", os itens lexicais manipulados pelo Sistema Computacional são "João", "ama" e "Maria". Sendo assim, a Numeração que dá origem a essa representação é inicialmente descrita como se segue.

(1) N = {João, ama, Maria}

O que falta para que essa Numeração possa ser considerada completa é indicarmos o número de vezes que cada um desses itens lexicais deverá ser retirado da Numeração e inserido no espaço derivacional. No caso, digamos que cada item presente em (1) deva ser retirado de N uma e somente uma vez durante a derivação. Para indicarmos isso, devemos anexar a cada um dos itens da Numeração 1 um *índice* subscrito – tal como vemos abaixo.

(2) N = {João$_1$, ama$_1$, Maria$_1$}

Essa Numeração indica claramente que cada item nela presente deverá ser introduzido no espaço computacional apenas uma vez. Você deve notar que a indicação do número de vezes em que um item deve ser retirado de N é importante porque sabemos, de antemão, que uma suposta frase como "João, João, João, ama, Maria" ou "João Maria ama Maria João" é muito diferente de "João ama Maria". Tal diferença, no caso, decorre da quantidade de vezes em que um item é inserido na frase – e essa limitação é determinada pelo índice do item conforme inscrito na Numeração.

Quando um item é retirado de N e é inserido numa derivação, seu índice é reduzido em 1. No momento em que o índice de um item atinge 0 (zero), isso

significa que esse item não será mais inserido na derivação. Por exemplo, quando tiramos o item "Maria" da Numeração {João$_1$, ama$_1$, Maria$_1$} e o inserimos na derivação, nosso N será igual ao que se segue.

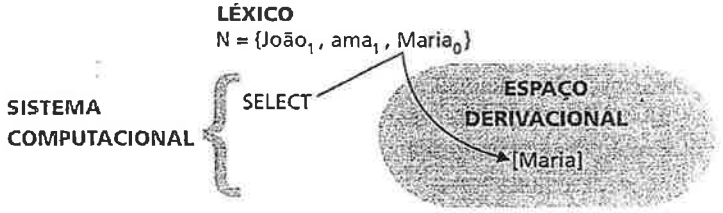
(3) N = {João$_1$, ama$_1$, Maria$_0$}

É esse novo N em (3) – que poderíamos chamar de N1, para diferenciá-lo do N em (2) –, já com o índice de "Maria" reduzido a zero, que estará disponível para as demais operações computacionais que serão disparadas no curso da derivação de "João ama Maria".

Se você já compreendeu o que é uma Numeração, então deverá estar se perguntando de que maneira ela é finalmente transformada numa derivação. Muito bem, a primeira coisa que precisa acontecer para que uma derivação tenha início é a retirada de um dado item lexical, diretamente da Numeração, e sua respectiva introdução no espaço derivacional. A operação computacional que realiza esse procedimento é denominada Select (termo inglês que, como já estudamos, significa *selecionar*).

Selecionar (ou Select, se quisermos usar o termo estrangeiro) é uma operação computacional muito simples. Tudo o que ela faz é retirar um determinado item de N e introduzi-lo no espaço derivacional, reduzindo o seu índice em 1 (por exemplo, se o índice for 3 passa a ser 2, se for 2 passa a ser 1, se for 1 passa a ser zero). Já sabemos que, se temos a Numeração {João$_1$, ama$_1$, Maria$_1$}, a operação Select poderá introduzir o item "Maria" no espaço derivacional uma única vez. É isso que vemos representado na figura a seguir. Note que, em N, o item "Maria" apresenta índice zero, justamente porque já foi introduzido na derivação o número máximo de vezes permitido.

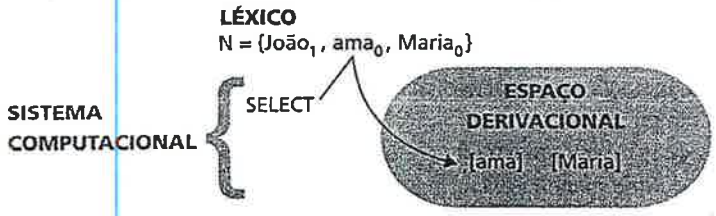
Figura 9.4: A operação Select (selecionar) retira itens de N e os introduz no espaço da derivação.



Você já compreendeu a função da operação Select na arquitetura da linguagem humana? Se sim, devemos agora lhe apresentar como é que uma derivação segue em frente, logo depois da inserção de um item lexical em nosso espaço derivacional.

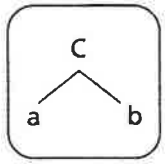
Continuando com a Numeração {João$_1$, ama$_1$, Maria$_0$}, suponhamos agora que a operação Select introduza no espaço da derivação, logo ao lado do item "Maria", a palavra "ama", justamente como ilustrado na figura a seguir. E agora? O que deverá acontecer?

Figura 9.5: Dois itens de N encontram-se inseridos no espaço da derivação.



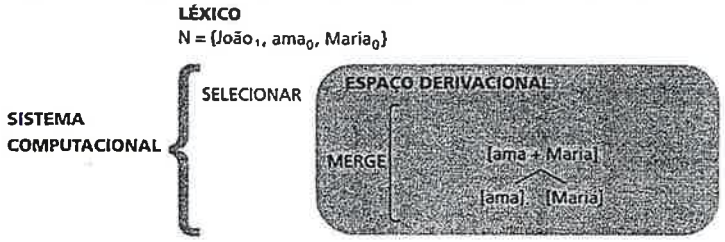
Quando pelo menos dois itens lexicais estão inseridos no espaço derivacional, a operação Merge (combinar), que também já conhecemos da unidade 6, pode ser ativada. Merge é a operação mais básica e fundamental do Sistema Computacional. Ela consiste em combinar dois elementos, digamos "a" e "b" – que podem ser itens lexicais ou sintagmas complexos –, e a partir desses dois formar um terceiro item, por exemplo "C". A computação "a" + "b" = C é representada na figura 9.6.

Figura 9.6: Merge é a operação computacional básica por meio da qual dois elementos quaisquer são combinados de modo a formar um terceiro elemento.



No caso específico do exemplo que estamos analisando, a existência de "ama" e "Maria" no espaço derivacional já permite a ativação da operação Merge. Essa operação será então acionada e, assim, o verbo [ama] será concatenado ao SN [Maria]. Por meio dessa combinação, Merge fará surgir o SV [ama [Maria]]. Tal combinação é representada como se segue.

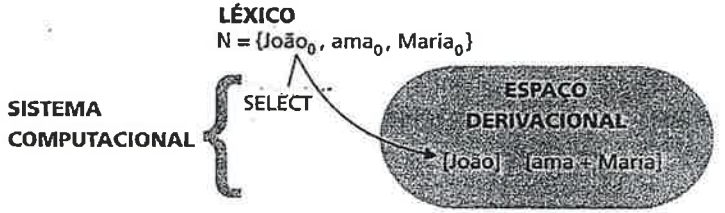
Figura 9.7: Merge combina [ama] e [Maria] fazendo surgir o SV [ama [Maria]].



Com esse simples exemplo, você já pode deduzir que essa operação computacional não foi desencadeada por acaso. Com efeito, ela foi disparada pelo traço de seleção do predicador "ama". Sabemos que esse item seleciona um argumento interno, e satisfazer essa seleção é uma das obrigações do Sistema Computacional. Sendo assim, o Sistema ativa a operação Merge, que toma "Maria" como complemento de "ama" e, dessa forma, satisfaz sua seleção de argumento interno.

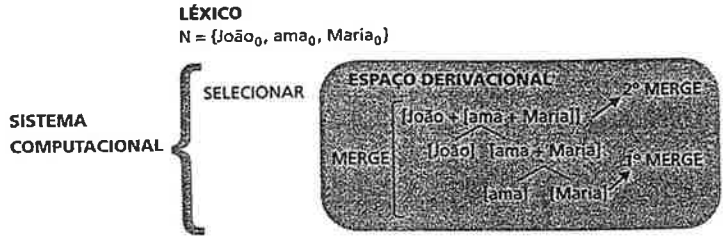
Pois bem, uma vez que, no espaço derivacional, o sintagma [ama [Maria]] tenha sido formado, qual será a próxima operação que o Sistema Computacional deverá utilizar na sequência de nossa derivação? Você está pensando corretamente se disse o seguinte: o que vai acontecer agora é a seleção de "João" e sua introdução, através de Select, no espaço da derivação. Vemos isso acontecendo na figura seguinte.

Figura 9.8: A operação Select insere [João] no espaço da derivação, ao lado do objeto completo já formado [ama [Maria]].



Até aqui, tudo bem. Mas e agora? Temos o item lexical [João] e o sintagma [ama [Maria]] disponíveis em nosso espaço derivacional. O que deverá acontecer? Nesse momento, a operação Merge é disparada mais uma vez (trata-se do segundo Merge na derivação). Dessa vez, Merge combinará o item lexical [João] e sintagma [ama [Maria]], tal como é ilustrado na figura 9.9.

Figura 9.9: A operação Merge combina [João] com o sintagma [ama [Maria]].



Nesse momento, nossa representação [João [ama [Maria]]] está basicamente formada. É nesse momento que o Sistema Computacional deve ativar a operação Spell-Out (termo inglês que significa "dividir", "separar"). Já aprendemos que Spell-Out é uma espécie de momento computacional que separa a derivação em duas partes: a representação (π que será encaminhada para a FF, e a representação (λ), que será enviada para a FL. Isso significa que, mais do que uma "operação", Spell-Out é uma bifurcação no curso de uma derivação sintática, uma espécie de fronteira divisória com duas direções: FF e FL.

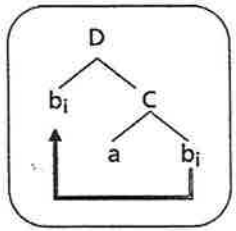
Se você for um estudante curioso, talvez tenha agora em mente a seguinte pergunta: mas, afinal, como o Sistema Computacional pode saber que é chegado o momento de Spell-Out? Como o Sistema é capaz de identificar que deve cindir a derivação em duas partes, uma fonética e outra semântica? Essa é uma pergunta importante. Deixaremos para respondê-la em detalhes na próxima seção desta unidade. Por enquanto, começaremos a descrever uma operação computacional que também já é nossa conhecida: Move (mover) – o deslocamento de constituintes de uma posição para outra dentro de uma derivação.

Move é uma operação computacional bastante complexa. Ela é fundamental para compreendermos um grande número de fenômenos sintáticos das línguas naturais. É por essa razão que retomaremos Move mais à frente ao longo desta unidade, para analisarmos com mais profundidade de que maneira essa operação satisfaz inúmeras condições derivacionais. Para já, o que precisamos entender é que Move é uma forma especial de aplicação da operação Merge.

O Merge simples, também chamado de Merge externo, acontece quando combinamos dois itens que se encontram disponíveis no espaço derivacional. O Merge simples (que devemos chamar apenas de Merge) já foi representado na figura 9.6. Por sua vez, Move é o nosso Merge complexo, ou Merge interno. Ele acontece, numa descrição inicial, da seguinte maneira: 1°) o Sistema Computacional faz uma *cópia* de um item já existente no espaço da derivação, seja um item lexical ou um sintagma; 2°) o Sistema Computacional combina, via Merge, essa *cópia* com outro elemento existente no espaço derivacional. Vejamos isso em detalhes.

Imagine que, conforme está ilustrado na figura a seguir, o Sistema Computacional precisasse combinar um item linguístico qualquer – digamos "b" – com um sintagma qualquer – por exemplo, C. O que ele deveria fazer se "b" não estivesse mais disponível na Numeração e se C fosse um sintagma já construído no curso de uma derivação?

Figura 9.10: Move = cópia + Merge.

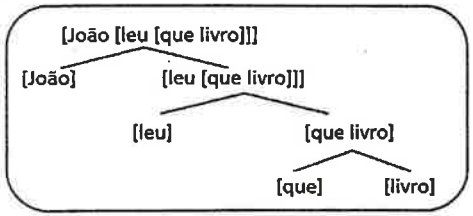


A alternativa do Sistema seria fazer uma cópia de "b" e, então, combiná-la com C. O resultado da combinação da cópia de "b" com o sintagma C seria um novo sintagma, representado na figura 9.10 por D. Note que a letra "i" subscrita nas duas instâncias de "b" indica exatamente que ambos os elementos recebem o mesmo índice, isto é, são o mesmo elemento, sendo um a cópia do outro. A seta utilizada na ilustração indica-nos a posição na derivação de onde a cópia do constituinte foi deslocada e qual é posição final do movimento. Muito bem, é precisamente esse o recurso usado pelo Sistema Computacional quando se torna necessário combinar um elemento já inserido numa derivação com outro também disponível no espaço derivacional, mas com índice zerado na Numeração: o Sistema faz uma cópia do constituinte que precisa ser movido e então procede ao Merge de tal cópia com outro constituinte sintático. A operação complexa {cópia + Merge} é, portanto, o que chamamos de Move.

Se essa explicação formal está parecendo muito abstrata, vejamos um exemplo linguístico. Suponha que uma derivação tenha criado a seguinte representação: "João leu que livro?". Essa representação é derivada em algumas etapas. Primeiramente, temos a Numeração = {João$_1$, leu$_1$, que$_1$, livro$_1$}. A operação Select retira "livro" de N e o introduz no espaço derivacional. Logo depois, a mesma operação insere "que" no espaço da derivação. Agora, a operação Merge é ativada e os itens "que" e "livro" são combinados entre si num só sintagma: o SN [que livro]. Em seguida, Select introduz "leu" na derivação. Então, Merge é acionado mais uma vez, procedendo à concatenação do verbo "leu" e do SN [que livro], gerando o SV [leu [que livro]]. Por fim, Select insere "João" na derivação e, logo a seguir, Merge combina esse SN com o SV [leu [que livro]], que culmina na representação [João [leu [que livro]]]. A figura a seguir apresenta visualmente os passos dessa

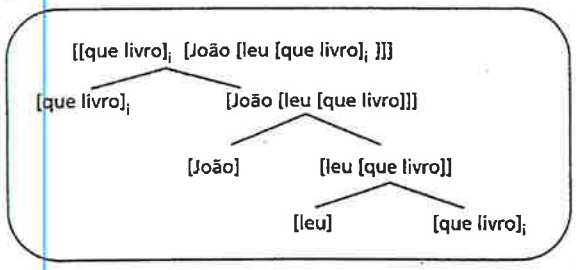
derivação (se você está curioso para saber como o Sistema identifica que se trata de uma interrogação, saiba que a Numeração que apresentamos acima é simplificada e está, na verdade, incompleta. Nela faltou indicar a presença do traço "interrogação", que marcaria a força ilocucionária da representação).

Figura 9.11: A derivação da representação [João [leu [que livro]]].



Se você entendeu o processo dessa derivação até a formação da representação "João leu que livro?", então pense no seguinte. Como o Sistema Computacional teria de proceder caso o SN [que livro] tivesse de ocupar o início da sentença, e não o final, como está nessa representação? Nesse caso, o Sistema não teria alternativa senão acionar a operação Move. Dessa forma, ele faria uma cópia do SN [que livro] e combinaria essa cópia com a representação [João [leu [que livro]]]. Após essa aplicação de Move, o SN [que livro] terá sido deslocado para o início da construção, e teremos a representação [[que livro]$_i$ [João [leu [que livro]$_i$]]]. A figura 9.12 ilustra essa nova representação. Note que o constituinte [que livro] aparece duas vezes na representação, em função do deslocamento de sua cópia para o início da frase.

Figura 9.12: Ilustração da aplicação de Move sobre o SN [que livro].



É certo que uma dessas duas ocorrências terá de ser apagada antes de a frase tomar corpo fonético numa pronúncia específica em FF – mas trataremos disso mais à frente nesta unidade.

Bare Phrase Structure

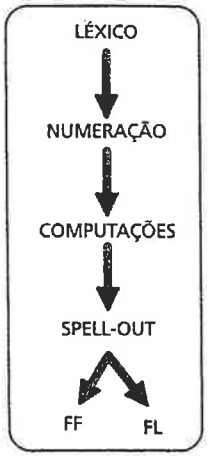
Você deve ter notado que, nas figuras 9.11 e 9.12, não utilizamos as representações arbóreas no modelo X-barra. A razão para isso é que X-barra é um modelo mais representacional e menos derivacional. Saiba que o modelo X-barra é ainda muito útil e bastante usado pelos gerativistas (tanto que, noutras figuras desta unidades, voltaremos a usá-lo), no entanto ele não é muito claro para indicar a derivação de uma sentença e é, dessa forma, mais adequado para ilustrar a sua representação, inclusive com a explicitação de rótulos gramaticais como N, N', SN etc. O modelo *Bare Phrase* (ou "estrutura simplificada", numa tradução livre), por sua vez, apresenta uma representação minimalista, preocupada mais em ilustrar a derivação de uma estrutura, e menos a sua representação.

Antes de concluir esta seção, vamos recapitular o que acabamos de estudar. Vimos que o Sistema Computacional utiliza suas operações para, a partir dos itens lexicais presentes numa dada Numeração, derivar representações sintáticas que serão acessadas e usadas pelos sistemas de interface. Durante a derivação, Select, Merge e Move são as principais operações computacionais de que o Sistema pode lançar mão. Como dissemos, num determinado momento derivacional – denominado Spell-Out –, a derivação se divide em duas partes: a que será direcionada à FF e a que será direcionada à FL. Vejamos, na seção a seguir, como o Sistema consegue identificar o momento correto dessa bifurcação derivacional.

Derivação por fases

Em função do que estudamos até aqui, acreditamos que a interpretação da seguinte figura seja uma tarefa fácil para você.

Figura 9.13: O curso de uma derivação antes e depois de Spell-Out.



Vemos aqui que uma derivação, compreendida como um conjunto de computações, é alimentada pelos traços lexicais compilados numa dada Numeração. Essa derivação aplicará operações como Select, Merge e Move até um ponto em que será dividida em duas: uma direcionada à FF e outra à FL. Nesse percurso até essas interfaces, após Spell-Out, as computações continuam – com a ressalva de que as operações a caminho de FL não serão mais visíveis na expressão fonética da representação sintática em construção, dado que informações fonético-fonológicas são encaminhadas apenas para FF. Pelo que se vê, Spell-Out é uma espécie de *comando* do Sistema Computacional, o qual é responsável por enviar a representação até então construída para o processamento computacional nos sistemas de interface. Mas quando é que esse comando é ativado no curso de uma derivação?

Desde seu importante texto de 1999, Chomsky vem defendendo a ideia de que as derivações acontecem por "fases", isto é, por pequenos ciclos (ou fatias de informação) ao final dos quais a derivação é bifurcada entre FF e FL. Uma fase, para Chomsky, é o suporte sintático das estruturas de significado inspecionadas em FL. São o que o autor chama de "estruturas proposicionais", em que um *sujeito* diz alguma coisa sobre algum *predicado*. Além disso, Chomsky também interpreta que as fases sejam uma espécie de *memória de trabalho*, já que, segundo ele, FL só é capaz suportar pequenas fatias de estrutura em sua *memória ativa*. Para além dessas interpretações chomskianas, existem muitas outras acerca da natureza das fases, formuladas por diversos e respeitados linguistas. Você terá oportunidades de conhecê-las, caso se aprofunde nos estudos gerativistas.

Em termos derivacionais, uma fase é um SC ou um SV, isto é, é uma proposição completa, com a sua força ilocucionária devidamente marcada (no SC), ou é uma estrutura de predicação completa, com um verbo e sua respectiva estrutura argumental devidamente saturada (no SV). Isso significa que uma derivação é enviada para FF e FL tão logo um SV tenha sido saturado com seus argumentos ou tão logo um SC tenha sido representado, com seus constituintes e sua força ilocucionária assinalados. SC e/ou SV são, portanto, os momentos de uma derivação em que a operação Spell-Out é ativada. O interessante é que, uma vez que tenha sofrido Spell-Out, um constituinte sintático não estará mais disponível para futuras operações do Sistema Computacional. Vejamos um exemplo disso, passo a passo.

Uma representação como "João ama Maria" é enviada para as interfaces assim que "João" sofre Merge com o SV "ama Maria", pois é nesse momento que uma estrutura argumental plenamente saturada é construída – note que, com o Merge de "João", o verbo "amar" terá saturados os dois argumentos exigidos em sua estrutura argumental. Da mesma forma, a representação "Que livro o João leu que livro" é enviada, por meio de Spell-Out, para as interfaces da linguagem na hora em que o constituinte "que livro" é movido para o início da sentença, numa posição dentro do SC, já que, após a aplicação de Move, uma proposição completa, é construída. Depois de enviados para FF e FL, esses elementos não poderão sofrer Merge ou Move, já que não estarão mais acessíveis ao Sistema Computacional.

Se você está gostando do formalismo chomskiano, vai achar interessante o fato de a noção de "fase" provocar uma reformulação no conceito de Numeração. Ora, para que a operação Select possa inserir itens lexicais numa fase correta (e não, digamos, em qualquer uma das fases de uma dada derivação), esses itens devem estar "arranjados", isto é, devem estar organizados dentre os elementos que irão compor uma dada fase (e não outra) numa derivação específica. Por exemplo, imagine que quiséssemos construir uma representação sintática como "Paulo acha que João está feliz". Como é que o Sistema Computacional poderia saber que "Paulo" deve ser combinado com "acha" (e não com "está feliz"), ao mesmo tempo em que "João" deve ser combinado com "está feliz" (e não com "acha")? Pois bem, o Sistema saberá disso porque "Paulo" estará arranjado, na Numeração, no mesmo grupo de "achar", bem como "João" estará arranjando no mesmo conjunto de "feliz". Cada um desses grupos corresponde ao total de itens lexicais que devem ser inseridos em uma e somente em uma das fases da derivação de nosso exemplo. A noção de fase conduz, portanto, à noção de "arranjos", também chamados de "subarranjos", que são os grupos de itens que compõem uma Numeração, separados por seus respectivos núcleos de fase, isto é, C e V. Para visualizar isso, compare as duas numerações a seguir.

(4) a. N = {Paulo$_1$, acha$_1$, que$_1$, João$_1$, está$_1$, feliz$_1$}
b. N = {{acha$_1$, Paulo$_1$}, {que$_1$, João$_1$, está$_1$, feliz$_1$}}

Você deve ter percebido que a Numeração em (4b) encontra-se organizada em dois grupos, correspondentes às duas fases em que a derivação deverá ser computada. Cada grupo desses é iniciado por um núcleo de fase (V, núcleo de SV, ou C, núcleo de SC). Logo, podemos dizer que (4b) é uma Numeração organizada em subarranjos, enquanto (4a) não é. É justamente esse subarranjo que permitirá ao Sistema identificar corretamente que item deve ser inserido em qual fase da derivação.

Se você está achando que a Numeração organizada em subarranjos parece ser em si mesma uma *representação*, saiba que muitos gerativistas pensam como você. Veremos, ao final desta unidade, que existem críticas importantes ao modelo computacional chomskiano que aqui descrevemos. Uma delas diz respeito ao caráter mais representacional e menos derivacional de certas entidades teóricas, que Chomsky (1995) nomeia como **necessidade conceitual virtual**.

> Uma necessidade conceitual virtual é uma espécie de princípio básico do qual os gerativistas não podem abrir mão. É um axioma descritivo. Sem ele, não haveria como explicar o funcionamento do Sistema Computacional.

Passemos agora para a próxima seção desta unidade. Veremos como os princípios da Interpretação Plena e da Economia Computacional norteiam uma derivação sintática. São eles que dizem o Sistema Computacional o *que* fazer e *como* fazer durante a derivação de representações linguísticas.

Princípios derivacionais

Na unidade 6 deste curso, você estudou o Princípio da Interpretação Plena – que os gerativistas às vezes chamam simplesmente de "FI", que é a sigla em inglês desse Princípio (*Full Interpretation*). Você deve lembrar-se de que o Princípio da Interpretação Plena é, por assim dizer, o agente regulador do Sistema Computacional. Segundo esse Princípio, o Sistema deve gerar representações que possam "ser lidas" nas interfaces da linguagem, isto é, as derivações do Sistema devem cumprir a condição de gerar representações que possam ser acessadas e usadas pelo sistema articulatório-perceptual e pelo sistema conceitual-intencional. É como se os sistemas de interface tivessem de dar uma espécie de "veredito" para cada representação gerada pelo Sistema Computacional. Se uma representação for

licenciada nas interfaces, então poderemos pronunciá-la e interpretá-la corretamente – logo, seu veredito será positivo. Caso a representação não seja licenciada nas interfaces, então não poderemos ou pronunciá-la ou interpretá-la corretamente – portanto, seu veredito será negativo.

Em termos mais técnicos, dizemos que uma derivação é *convergente* quando ela produz uma representação que é licenciada pelas duas interfaces da linguagem. Dizendo de outra forma, uma derivação é convergente (veredito positivo) quando tanto o sistema articulatório-perceptual quanto o sistema conceitual-intencional conseguem acessar e usar sua respectiva representação. Por outro lado, dizemos que uma derivação não é convergente quando sua respectiva representação não é licenciada por uma das interfaces ou por ambas. Isto é, uma derivação não é licenciada (veredito negativo), e assim sofre um *crash* nas interfaces, quando o sistema articulatório-perceptual e/ou o sistema conceitual-intencional são incapazes de utilizar a representação construída. Ilustremos isso com um rápido exemplo.

Vimos que a aplicação da operação Move envolve a cópia de um dado constituinte presente na derivação e o respectivo Merge dessa cópia com outro elemento no espaço derivacional. Após a aplicação de Move, haverá portanto duas ocorrências do constituinte copiado, tal como já analisamos num exemplo anterior, que aqui reproduzimos.

(5) [[que livro]$_i$ [João [leu [que livro]$_i$]]]

Da maneira em que se encontra, essa representação não poderia ser acessada e usada pelo sistema articulatório-perceptual. Note que o SN [que livro] está duplicado e assim teria de ser pronunciado duas vezes. Isso faria com que nenhuma ordenação linear coerente fosse estabelecida entre as duas cópias: o SN estaria ao mesmo tempo antes e depois de si mesmo. Essa derivação, com a pronúncia repetida do item [que livro], seria considerada não convergente (*crash*!) na interface fonológica.

Talvez você esteja se perguntando: o Sistema Computacional seria capaz de modificar essa representação de alguma maneira a fim de licenciá-la nas interfaces? A resposta é sim. Para licenciar tal derivação, tudo o que o Sistema terá de fazer é apagar uma das cópias do constituinte movido. Esse apagamento é representado pelo tachado duplo, conforme representado a seguir.

(6) [[que livro]$_i$ [João [leu ~~[que livro]~~$_i$]]]

Essa representação indica que [que livro] só será pronunciado uma única vez, pois somente a cópia no início da frase poderá ser realizada foneticamente, sendo a última silenciada. Com essa operação (apagar cópia), nossa derivação poderá ser

acessada e utilizada por ambos os sistemas de interface e, assim, será considerada convergente, em acordo com o Princípio da Interpretação Plena.

Essa análise trata de um problema de convergência na interface fonológica da linguagem, mas você pode imaginar que as mesmas imposições de FI valem para a interface semântica. Lembre-se de que, para ser convergente, uma derivação deverá ser licenciada ao mesmo tempo tanto em FF quanto em FL. A clássica frase "Ideias verdes incolores dormem furiosamente" seria um bom exemplo de falta de convergência na interface conceitual da linguagem, pelas razões que já comentamos ao início desta unidade.

Com efeito, o Princípio da Interpretação Plena não é a única condição imposta ao funcionamento do Sistema Computacional. Também as condições de Economia Computacional devem ser levadas em consideração para o licenciamento de uma derivação sintática. Analisemos isso melhor.

O linguista Juan Uriagereka, da Universidade norte-americana de Maryland, propôs, em importante estudo publicado em 1999, que o "minimalismo" na linguística gerativa deve assumir dois sentidos. O primeiro deles é o chamado minimalismo **ontológico**. Esse tipo de minimalismo assume que a faculdade de linguagem é, em si mesma, simples, econômica, rápida e não redundante. O minimalismo ontológico afirma essencialmente que a natureza da linguagem humana é "mínima": uma língua natural é uma espécie de organismo bem estruturado, capaz de gerar as estruturas de maneira eficiente e rápida, quase perfeita. Nesse sentido, podemos deduzir que as operações do Sistema Computacional serão tão simples e econômicas quanto possível. Se a linguagem humana é minimalista, então as operações computacionais mais simples serão sempre preferidas em detrimento de operações mais custosas e complexas, não é mesmo?

> Na filosofia, entendemos por *ontologia* o estudo da natureza das coisas existentes no mundo. Ontologia = como as coisas são? Já os termos *epistemologia* e *metodologia* reportam-se à maneira pela qual podemos estudar as coisas existentes no mundo. Epistemologia / Metodologia = como podemos conhecer as coisas.

A outra interpretação de "minimalismo" na teoria gerativa denomina-se, segundo Uriagereka, minimalismo **metodológico**. Esse minimalismo é uma espécie de instrução para o sintaticista e quer dizer, basicamente, que ao descrevermos a linguagem humana, analisando por exemplo uma derivação sintática, devemos ser minimalistas. Ora, ser minimalista metodologicamente significa usar o mínimo de artifícios descritivos possível. Por exemplo, se um sintaticista puder explicar uma derivação postulando, digamos, duas operações computacionais, então essa explicação deve ser preferível a outra que postule três ou quatro operações. O minimalismo metodológico

visa à formação de linguistas objetivos, concisos e diretos, que não criem explicações desnecessariamente complexas para os fenômenos de uma língua natural.

Para os objetivos deste curso, é o minimalismo ontológico o que mais nos interessa. A partir dele, poderemos prever que certas operações computacionais serão mais ou menos ativadas pelos Sistema conforme provoquem ou não mais complexidade durante uma derivação. Por exemplo, sabemos que a operação Move é computacionalmente muito mais complexa do que Merge. Sendo assim, e tendo em conta a natureza minimalista da linguagem, podemos prever que a operação Merge será preferível à operação Move sempre que possível. Move só será ativado pelo Sistema como *último recurso*, isto é, como a única maneira de satisfazer uma imposição dos traços presentes numa Numeração e, assim, preservar o Princípio da Interpretação Plena. Vejamos isso na prática.

Imagine que tenhamos, num certo momento numa dada derivação, a Numeração {Maria$_0$, parece$_0$, estar$_0$, triste$_0$}. Com esse N, teríamos a seguinte derivação em processo.

(7) [parece [Maria [estar triste]]]

Essa derivação, no momento em que se encontra, ainda não seria convergente, logo não poderia ser enviada para as interfaces com a ativação de Spell-Out. No intuito de licenciar essa construção, o Sistema deverá realizar duas operações. A primeira é retirar o item "Maria" da posição em que se encontra, na qual não pode ser pronunciada – note que, como o verbo "estar" não se encontra flexionado, nenhum Caso pode ser atribuído ao SN "Maria" (veremos os detalhes sobre a identificação de Caso mais à frente nesta unidade). A segunda é atribuir um sujeito ao verbo "parecer", já que esse ainda não foi assinalado na derivação. Mas como o Sistema faria isso? Lembre-se de que os itens da Numeração já se encontram com os seus índices zerados, de forma que nenhuma operação Merge parece ser possível. Qual seria, então, a solução?

Nesse caso, a única possibilidade derivacional é aplicação da operação Move. Move fará uma cópia do constituinte "Maria" e, então, procederá ao Merge dessa cópia com o verbo "parece", dando origem à seguinte representação.

(8) [Maria$_i$ [parece [Maria$_i$ [estar triste]]]]

Em seguida, deve acontecer o apagamento da cópia de "Maria" em sua posição original, ao lado de "estar triste". Após esse apagamento, a representação estará completa, pronta para ser acessada pelas interfaces, conforme representamos a seguir.

(9) [Maria$_i$ [parece [~~Maria~~$_i$ [estar triste]]]]

O importante nesse exemplo é você notar que a operação complexa Move só foi acionada pelo Sistema porque nenhuma outra mais simples, como Merge, estava disponível. Pois bem, esse é um bom exemplo do segundo tipo de Princípio imposto às derivações: a Economia Derivacional – operações mais complexas só são ativadas como último recurso.

Alçamento de constituintes

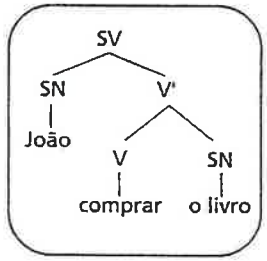
Se você analisou com atenção a representação [Maria$_i$ [parece [~~Maria~~$_i$ [estar triste]]]], certamente terá notado que o sujeito da oração [estar triste] foi deslocado para a posição de sujeito da outra oração [parece]. Quando um constituinte é, como ocorreu no exemplo, deslocado de uma oração para outra, dizemos tecnicamente ter ocorrido um alçamento (*raising*, no termo em inglês). No caso, tivemos um alçamento de sujeito, mas também objetos e outros tipos de complemento podem ser alçados de uma oração para outra na derivação de uma sentença.

Você deve ter percebido que, ao ilustrar o Princípio da Interpretação Plena e o Princípio de Economia Derivacional, que são os dois grandes princípios computacionais da linguagem, usamos exemplos com a operação Move. Com efeito, Move é uma das operações mais intrigantes do Sistema Computacional. Apesar de sua complexidade, ela é acionada em diversas circunstâncias derivacionais. Vejamos isso em detalhes nas próximas seções.

Regras de Movimento

Acreditamos que você já tenha aprendido que a sintaxe das línguas naturais caracteriza-se como um conjunto de operações combinatórias – que aqui estamos chamando de *operações computacionais* ou simplesmente *computações*. Vimos, por ocasião da unidade 8, quando estudamos a teoria X-barra, que um núcleo sintático X qualquer seleciona um dado complemento e com ele se combina para projetar o nível X' (pronunciado x-barra). Por exemplo, um núcleo V como o verbo "comprar" seleciona um SN como "o livro" como seu argumento interno e, dessa combinação, resulta a projeção V' "comprar o livro". Por sua vez, a projeção V' seleciona um argumento externo como "João", criando uma combinação sintática da qual resulta uma projeção máxima, o SV "João comprar o livro". Essas computações estão representadas na árvore a seguir.

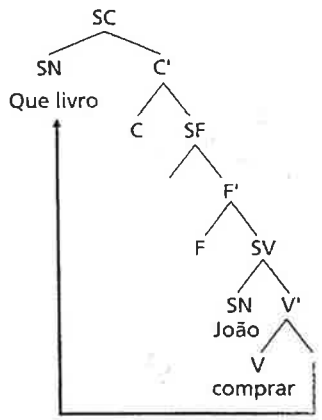
Figura 9.14: Um SV com dois argumentos saturados.
(Os triângulos indicam omissão na representação de estruturas.)



O interessante das computações da figura 9.14 e das demais até aqui apresentadas é que todas elas são *locais*, isto é, ocorrem com constituintes que estão contíguos: um se posiciona visivelmente ao lado do outro. Existem, não obstante, combinações sintáticas de *longa distância*. Essas combinações ocorrem entre constituintes que se afastam na linearidade da sentença. Trata-se de sintagmas que estabelecem relações sintáticas não locais, com distância variável (curta, média ou longa). Por exemplo, na sentença interrogativa "Que livro João comprou?", sabemos que o SN "que livro" é o objeto direto do V "comprar", e, assim, deve ser selecionado como seu argumento interno, na posição local em que recebe, do verbo predicador, o papel temático de "tema". Entretanto, ao lermos a frase percebemos que o constituinte "que livro" não aparece na sua posição original, logo após o verbo. Ele não ocupa a posição de base na qual estabelece relação local com V e onde recebe papel temático. O SN "que livro" está, na verdade, localizado no início da frase, na posição que já aprendemos ser denominada como especificador de SC. Essa posição destina-se especialmente aos sintagmas que exprimem, dentre outras coisas, valores de força ilocucional da sentença – note que o constituinte "que livro" possui valor interrogativo e, por isso, deve normalmente aparecer no início de uma frase que tenha valor ilocucional de "pergunta". Ora, como isso é possível? Como o complemento de um verbo pode realizar-se, na sentença, numa posição distante de V?

Já aprendemos que certos constituintes numa dada sentença podem ser pronunciados em posições sintáticas diferentes daquela em que são originalmente selecionados. Quando isso acontece, dizemos que tal constituinte foi "movido", em decorrência da aplicação de Move, que é também chamado de *regra de Movimento*. Uma regra de Movimento é, portanto, um tipo de combinação sintática que desloca sintagmas ou núcleos sintáticos de uma posição para outra dentro da sentença no curso de uma derivação. Esse deslocamento terá como consequência o estabelecimento de relações sintáticas não locais entre constituintes, conforme podemos ver na ilustração que se segue.

Figura 9.15: Uma aplicação de Move – a regra de Movimento.



Vemos no exemplo que o SN "que livro" é gerado na posição de complemento de "comprar". Tal posição, que denominamos *posição de base*, é aquela em que esse SN é selecionado como argumento interno de V. É nessa mesma posição que "que livro" recebe papel temático. No entanto, na estrutura final da frase, essa relação do SN com o seu núcleo predicador correspondente (V) é apenas indireta, não local, afinal o SN está linearizado no início da sentença e V está no final. Dizemos que ocorreu, nesse caso, uma regra de Movimento – uma aplicação da operação Move: o SN foi deslocado de sua posição de base para a posição de especificador do SC, de modo a caracterizar a força ilocucional interrogativa da sentença.

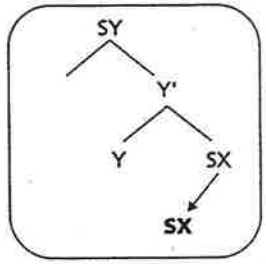
Regra de Movimento é o artifício técnico da descrição linguística por meio do qual conseguimos entender como os constituintes mantêm relações sintáticas não locais. Trata-se de uma ferramenta descritiva muito importante, pois muitas vezes encontramos nas línguas naturais constituintes com funções descontínuas na sentença, isto é, muitas vezes encontramos sintagmas ou núcleos que desempenham função linguística em diferentes posições da frase. Esse fenômeno, que denominamos como **descontinuidade**, é de extrema produtividade em todas as línguas humanas – e a noção de regra de Movimento é uma maneira de explicá-lo. Todavia, tenha sempre em vista que falamos em "movimento" de uma maneira conotativa, apenas para entendermos melhor o que está acontecendo com os constituintes descontínuos. Isso não significa que, na nossa cabeça ou no meio das frases que pronunciamos, constituintes estejam de fato se movendo de uma posição para outra, num dada velocidade mensurável. O Movimento é apenas aparente. É mais ou menos como quando falamos do movimento do Sol, que parece se mover pelo céu, mas na verdade não está em movimento. Trata-se de uma metáfora descritiva.

Dizemos que um constituinte é *descontínuo* quando ele desempenha funções distintas em diferentes posições na sentença. Na figura 9.15, o constituinte "que livro" desempenha a função de tópico interrogativo no início da frase e, ao mesmo tempo, é o "tema" do predicador "comprar". Ele é pronunciado numa posição (em SC) e interpretado tematicamente noutra (em SV).

Vejamos como uma regra de Movimento se aplica, seus mecanismos e suas etapas. Primeiramente, temos o constituinte que será movido de sua posição de base para uma posição de destino. Esse constituinte pode ser um núcleo X ou um sintagma SX qualquer (X quer dizer justamente que isso pode acontecer com qualquer categoria lexical ou funcional). Tal constituinte já se encontra presente no curso de uma dada derivação. O que o Sistema Computacional fará com ele para gerar o Movimento é, como já estudamos, (1°) produzir uma cópia desse constituinte, (2°) deslocar uma cópia do constituinte para a posição de destino e (3°) apagar a cópia deixada na posição de base.

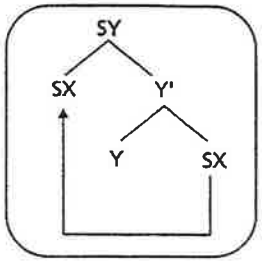
Suponha que o SX a seguir seja escolhido para ser movido. A primeira coisa que deve acontecer é a cópia desse sintagma.

Figura 9.16: A cópia do constituinte a ser movido é o primeiro passo da operação Move.



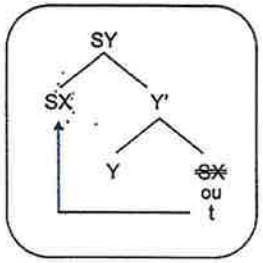
Uma vez que a cópia de SX tenha sido feita, o sistema linguístico irá deslocá-la para a posição alvo do Movimento. Suponhamos que esse alvo seja o especificador de SY, que é a posição sintática mais próxima vazia e disponível. E para lá então que nossa cópia de SX será movida. Tecnicamente, dizemos que a cópia de SX é combinada, via Merge, com a projeção Y'. Do Merge entre a cópia de SX e Y' projeta-se o sintagma hipotético SY.

Figura 9.17: A cópia do constituinte é deslocada para outra posição, na qual entra em Merge com outro elemento sintático.



A última etapa para a conclusão da regra de Movimento é o apagamento da cópia mais baixa do constituinte movido (em português, quase sempre é essa a cópia a ser apagada, mas noutras línguas o apagamento pode ser diferente – por exemplo, pode ser que, numa dada língua natural, seja a cópia em posição mais alta a que deve ser apagada). Conforme explicamos na seção anterior desta unidade, tal cópia precisa ser apagada por uma questão de linearização em FF. Geralmente, indicamos que a cópia de um constituinte foi apagada usando o tachado duplo sobre a sigla desse constituinte (~~X~~ ou ~~SX~~) ou, numa notação mais antiga, usamos um "t" (do inglês, *trace* – que quer dizer "vestígio" do Movimento).

Figura 9.18: A cópia mais baixa do constituinte movido deve ser apagada, para linearização em FF; uma cadeia forma-se entre as duas ocorrências de SX.



Entendemos, desta forma, que uma regra de Movimento é um composto de três operações computacionais: copiar, combinar (Merge) e apagar. Nesse composto, há uma relação entre as duas cópias (ou mais, se for o caso – já que o movimento pode ser **cíclico**) geradas pelo Movimento. Tal relação é capturada pela noção de *cadeia*.

Movimento cíclico

O movimento sintático pode acontecer diversas vezes com um mesmo constituinte. Isto é, a operação Move pode mover um sintagma para uma posição e de lá movê-lo para outra, tantas vezes quanto seja possível e necessário para que a derivação se torne convergente – respeitando, é claro, as condições de legibilidade nas interfaces (o Princípio FI). Vemos isso na ilustração que se segue.

$[_{sc}$ que livro$_i$ [eles disseram $[_{sc}$ ~~que livro~~$_i$ que o João estava lendo ~~que livro~~$_i$]

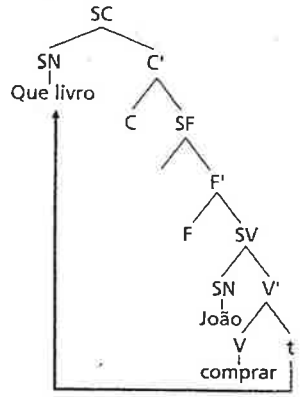
A cadeia permite-nos interpretar que as cópias nela presentes constituem, na verdade, um único objeto sintático, que se apresenta distribuído descontinuamente na sentença. Nas ilustrações que veremos na seção seguinte da unidade, as três operações que ocorrem numa regra de Movimento já terão sido aplicadas e tudo o que veremos representado na árvore sintática é a cadeia.

Existem três tipos de regra de Movimento: o movimento A', o movimento de núcleo e o movimento A. Vejamos os detalhes de cada uma dessas instâncias de Move.

Movimento A', movimento de núcleo e movimento A

A cadeia que podemos ver na figura 9.19 a seguir ilustra bem um movimento A' (lê-se A-barra). Nesse tipo de movimento, o local de chegada do constituinte movido é uma posição não argumental, isto é, é uma posição A'. Uma posição A' é, por conseguinte, aquela em que nenhum argumento é selecionado (lembre-se de que "o livro" é argumento interno selecionado por V e é nessa posição de complemento que recebe papel temático).

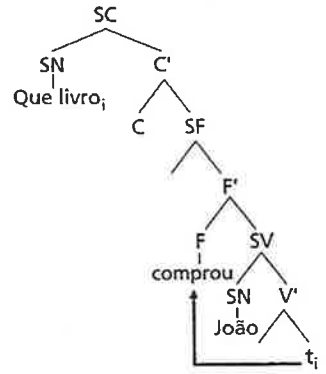
Figura 9.19: Movimento A'.



O movimento A' é característico das sentenças interrogativas, orações relativas, topicalizações e demais deslocamentos à periferia esquerda das frases. Os constituintes que se movem para posições A' geralmente já são argumentos de algum núcleo sintático presente na frase e movem-se para uma posição não argumental de modo a desempenhar outra função linguística, tais como interrogação, relativização, topicalização etc. Muitas vezes, o movimento A' é também denominado movimento de *qu-* (já que acontece com expressões interrogativas como *quem, quando, de que maneira, em que lugar*).

Por sua vez, o movimento de núcleo, tal como o seu nome já anuncia, acontece quando o elemento a ser deslocado é um núcleo sintático X. Ele move-se de uma posição de núcleo para outra posição de núcleo. Na figura 9.20 a seguir, vemos que o verbo "comprar" ocupa a posição de V, núcleo do SV. Notamos que o núcleo "comprar", em sua forma nominal infinitiva, não possui especificações de tempo, modo, aspecto, número ou pessoa, portanto não manifesta nenhuma flexão finita. Para que esse verbo receba flexão finita e, dessa forma, licencie a estrutura da frase com o *status* de sentença, o núcleo V deve sofrer movimento para T, o núcleo do sintagma temporal (ST – que às vezes, como já vimos, pode ser referido como SF, o sintagma flexional). É somente em T/F que estão codificadas as informações flexionais – como, por exemplo, tempo pretérito, aspecto perfeito, terceira pessoa, número singular etc. Movendo-se para T/F, o verbo assumirá uma morfofonologia finita como a de "comprou", na qual o núcleo verbal "comprar" amalgama-se aos morfemas flexionais do exemplo. Isso é o que ilustramos a seguir.

Figura 9.20: Movimento de núcleo.



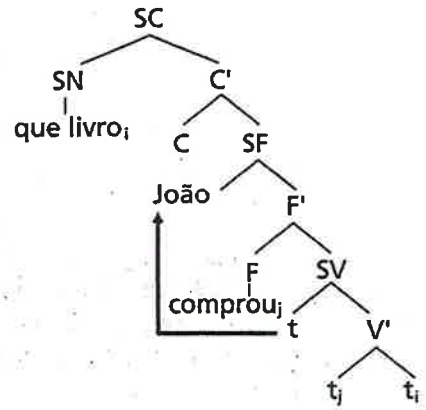
Uma nota importante. Nesta altura do curso, você deve compreender que a flexão verbal ocorre, na verdade, no interior do ST/SF. É somente no núcleo T/F que marcas funcionais como tempo, modo, aspecto, número e pessoa são computadas na arquitetura da linguagem. No domínio do SV, apenas informações lexicais encontram-se acessíveis para o Sistema. É por essa razão que, em sentenças finitas, os verbos devem sofrer movimento de núcleo, de V para T/F, de modo a assumir uma flexão. Tenha atenção a esta nota, pois até aqui, nas ilustrações de nosso curso, vínhamos simplificando as árvores sintáticas e apresentando verbos flexionados já no interior do SV. Ora, isso era apenas um recurso didático. A flexão verbal ocorre, de fato, no interior do ST/SF. Tome nota disso!

Se compararmos o movimento de núcleo com o movimento A', veremos que se trata de operações diferentes. Como você pode perceber, no movimento de núcleo, a cópia mais alta do constituinte movido ocupa uma posição de núcleo (T, no exemplo) e a cópia mais baixa também ocupa posição de núcleo (V, no exemplo). Já no movimento A', a cópia mais alta do constituinte movido ocupa uma posição não argumental (A', que no exemplo é o especificador de SC), enquanto a cópia mais baixa ocupa uma posição argumental (A, que no exemplo é o argumento interno de V). Se você entendeu essa lógica, deve estar imaginando como deve ser, então, o movimento A.

No movimento A, o constituinte que sofre regra de Movimento desloca-se para uma posição argumental, isto é, uma posição A. No caso, essa posição é a de especificador do sintagma temporal. Essa é a posição do sujeito de uma sentença. Tal posição é argumental, mas não é temática. Trata-se de uma posição para um argumento puramente sintático, sem o valor lexical das posições argumentais que são também temáticas.

No exemplo que se segue, o SN "João" é o argumento externo do verbo comprar. Esse argumento é deslocado para a posição argumental de sujeito da sentença, no especificado do ST/SF. Isso quer dizer que "João" possui duas funções na representação criada pelo Sistema Computacional: ele é o argumento externo de "comprar" e, após a aplicação de Move, é também o sujeito da sentença.

Figura 9.21: Movimento A.



Uma questão importante acerca do movimento A é entender, afinal, porque ele deve acontecer. Já sabemos que a posição de destino desse tipo de movimento é uma posição A, mas não é uma posição temática. Tal posição deve ser obrigatoriamente ocupada nas línguas naturais – ou seja, todas as sentenças nas diferentes línguas humanas devem manifestar a posição sintática do sujeito, ainda que na forma de um constituinte sem matriz fonética. Mas por que isso acontece? Veremos a seguir que a chave para a resposta dessa pergunta encontra-se na noção linguística de Caso.

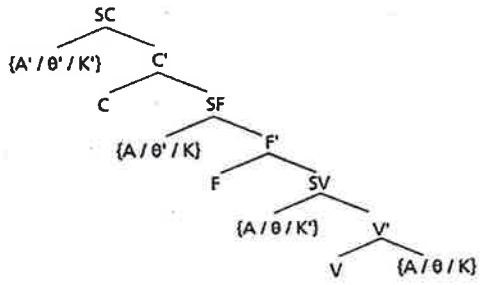
Caso e EPP

O Caso é, conforme estudamos rapidamente durante a unidade 7, uma propriedade importante das línguas naturais. A função do Caso num sistema linguístico qualquer é permitir a discriminação dos argumentos de um dado predicador. Por exemplo, se um predicador possui um argumento externo e um argumento interno, como poderíamos saber, numa determinada frase, quem é quem? Ou seja, como poderíamos identificar corretamente um argumento externo de um predicador,

separando-o de um argumento interno? Lembre-se de que a correta identificação do *status* de um argumento (se externo ou interno) é essencial para sabermos qual é o seu papel temático – portanto, identificar a categoria de um argumento é também conhecer a sua interpretação semântica. Pois bem, o Caso é justamente o fenômeno linguístico que permite a discriminação dos argumentos de um predicador. Cada tipo de argumento recebe, numa frase, um Caso específico, como, por exemplo, o *nominativo* associado ao argumento externo, o *acusativo* associado ao argumento interno SN, o *oblíquo* associado ao argumento interno SP, e assim por diante. É essa marca de Caso que permitirá a identificação do papel temático dos argumentos.

Posição argumental não temática

O especificador de ST é uma posição argumental, já que é ocupada pelo argumento externo de T – o sujeito da sentença. Contudo, T não é um núcleo lexical e, dessa forma, não pode atribuir papel temático. O especificador de T é, portanto, uma posição argumental, mas não temática. Na figura a seguir, vemos um inventário das posições de uma sentença. Atenção à legenda das posições: A = posição argumental, A' = posição não argumental; θ = posição temática, θ' = posição não temática; K = posição com Caso, K' = posição sem Caso. Lembre-se de P, como núcleo lexical, tem como complemento uma posição {A, θ, K}. Estudaremos mais sobre a noção de Caso ainda nesta unidade.



Na figura acima, representamos ST como SF (e, consequentemente, T como F). Tenha atenção, pois às vezes podemos representar um ST usando a notação mais antiga SF.

Para ilustrar melhor o que estamos dizendo, imagine que você ouça ou leia uma frase com o predicador "morder" e os argumentos "menino" e "cachorro". A primeira coisa que desejará saber é "quem mordeu quem", isto é, você terá de

identificar quem é o agente desse verbo e quem é o seu paciente. Tipicamente, o agente de "morder" será o "cachorro", mas e se for o "menino"? Seria o caso de um evento extraordinário, não é mesmo? Ora, saberemos quem é o agente e quem é o paciente dessa frase se pudermos identificar o Caso dos SNs "menino" e "cachorro". Vejamos isso com mais detalhes.

Em português, identificamos o Caso de um SN (na verdade, de um SD, que é o sintagma funcional que domina SN, como aprendemos na unidade 8) por meio de sua posição linear na frase. Assim, em "O cachorro mordeu o menino", sabemos que "o cachorro" é o argumento externo de "morder" porque esse item encontra-se à frente do verbo, na posição em que tipicamente o Caso nominativo (que licencia a função de sujeito) é identificado. O Caso nominativo sinaliza, portanto, que se trata de um argumento externo (sujeito). Com base nessa informação, saberemos que esse sujeito – no caso, o SN "o cachorro" –, deve ser interpretado como "agente" da ação de "morder", pois tal informação encontra-se inscrita na grade temática desse verbo. Da mesma forma, sabemos que "o menino" é o argumento interno de "morder" em razão de sua posição imediatamente posterior ao verbo, na qual identificamos o Caso acusativo (que licencia a função de objeto direito). Ora, se sabemos que "o menino" está posposto ao verbo, e essa é a posição típica do Caso acusativo, deduziremos então que esse SN é o argumento interno de "morder", o qual deve ser interpretado como "paciente" da ação, de acordo com os traços de seleção desse predicador verbal.

Mas o que aconteceria se a posição desses SNs fosse invertida? Nesse caso, identificaríamos o "menino" com o Caso nominativo e o "cachorro" com o Caso acusativo – e como resultado teríamos a estranha declaração de que um menino (sujeito / nominativo / agente) mordeu um cachorro (objeto / acusativo / paciente). Em suma, o que você deve sempre ter em mente é que o Caso de um SN é o mecanismo linguístico que permite a identificação de seu *status* como argumento (externo ou interno) e é essa identificação que nos indicará o papel temático a ser atribuído a tal sintagma.

A ordenação linear não é, contudo, a única maneira pela qual as línguas naturais identificam os Casos associados às expressões nominais. Muitas línguas possuem sistemas morfológicos complexos, por meio dos quais morfemas específicos são adjungidos a um nome de acordo com o Caso que ele assume. Dezenas de línguas indígenas brasileiras possuem intrincados sistemas morfológicos para marcar Caso em suas expressões nominais. Não obstante, o exemplo mais famoso de uma língua que possui morfologia de Caso é o latim.

Línguas com sistemas morfológico de Caso são chamadas simplesmente de "línguas de Caso", enquanto línguas sem sistema morfológico de Caso denominam-se "línguas de Caso abstrato".

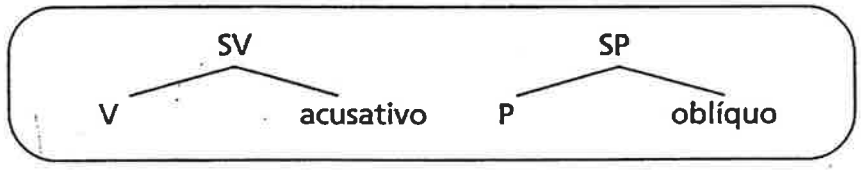
Em latim ou em qualquer outra língua com morfologia de Caso, um nome mudará a sua expressão morfológica de acordo com o Caso que receber numa sentença específica. Por exemplo, a frase "Os cachorros mordem o menino" é traduzida para o latim como "*Canes mordent puer*". Vemos que "*canes*" e "*puer*" assumem, respectivamente, a forma do nominativo e do acusativo, já que são o sujeito e o objeto na frase. Entretanto, se a frase fosse "O menino morde os cachorros", então a respectiva tradução seria "*Puerum mordet canibus*" – note que as palavras mudaram de forma, pois a expressão morfológica de "menino" é, em latim, diferente nas formas nominativa e acusativa: *puerum* (nominativo/sujeito) e *puer* (acusativo/objeto). O mesmo ocorre com a expressão de "cachorros": *canes* (nominativo/sujeito) e *canibus* (acusativo/objeto).

Existem diversos Casos dentre as línguas naturais, os quais servem para identificar os distintos tipos de relação entre predicadores e argumentos. Eis os Casos mais conhecidos pelos linguistas: ablativo, absolutivo, acusativo, dativo, ergativo, genitivo, locativo, nominativo, oblíquo, partitivo e vocativo. Para os propósitos de nossa discussão nesta unidade, os Casos mais relevantes a destacar são nominativo, acusativo e oblíquo. O nominativo, como já dissemos, é o Caso que identifica o sujeito da sentença, que tipicamente será o argumento externo do predicador de uma dada frase. Por sua vez, o acusativo indica o objeto direto, que é o argumento interno de um predicador verbal. Por fim, o oblíquo identifica o Caso do complemento preposicionado de um predicador, seja objeto indireto, complemento nominal ou outro (a distinção para cada uma dessas funções pode ser feita em detalhes, mas aqui usaremos o termo "oblíquo" para simplificar).

A pergunta que deve passar pela sua cabeça neste momento é a seguinte: como o Sistema Computacional sinaliza o Caso (morfológico ou abstrato) a ser identificado nas expressões nominais?

Para os argumentos internos, a posição de identificação de Caso coincide com a posição de atribuição de papel temático. Ou seja, o acusativo é identificado na posição de complemento de V, enquanto o oblíquo torna-se visível na posição de complemento de P. Note que esse sistema de identificação mantém-se o mesmo ainda que ocorra aplicação de Move, já que a existência de uma cadeia permite o licenciamento do papel temático e do caso de um constituinte que tenha sofrido regra de Movimento.

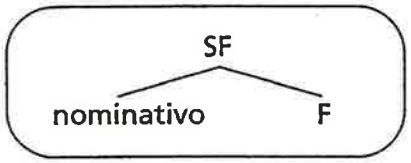
Figura 9.22: Identificação dos Casos acusativo e oblíquo.



Dizendo de outra forma, a posição de licenciamento de Caso é, para os argumentos internos, coincidente com a de atribuição de papel temático: a posição de complemento de um núcleo lexical X. A exceção é o argumento interno de verbos inacusativos (veja a unidade 7, para relembrar o conceito de inacusatividade, se precisar), que serão licenciados com o nominativo após aplicação da operação Move.

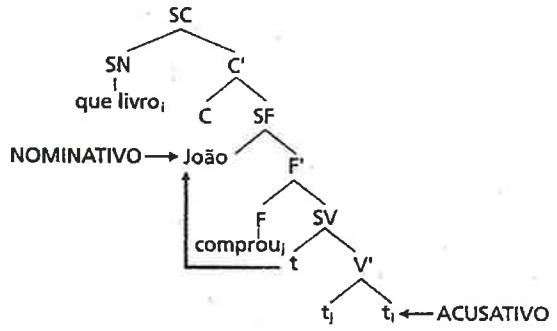
Se acusativo e oblíquo são normalmente licenciados em sua posição de base, o que acontece com o nominativo? Você deve ter reparado que os núcleos que estão associados ao acusativo (V) e ao oblíquo (P) são lexicais. Pois bem, o nominativo é, por seu turno, associado a uma categoria funcional: o núcleo T do sintagma temporal. Isso significa que o nominativo, sendo o Caso de identificação do sujeito da sentença, está relacionado à flexão verbal finita, mas não ao núcleo lexical do verbo. Para entender isso, lembre-se de que uma forma de palavra como "cantaremos" possui, na verdade, duas categorias sintáticas distintas. Uma é o núcleo lexical V: "cant-". Outra é o conjunto de morfemas que expressam tempo, modo, aspecto, número e pessoa, que chamamos de flexão T: "-aremos". Enquanto o núcleo lexical V licencia o acusativo em seu complemento, o núcleo funcional T (com uma flexão finita) licencia o nominativo no seu especificador.

Figura 9.23: Identificação do Caso nominativo.



A identificação do nominativo é, portanto, uma propriedade da sentença, e não da camada lexical de uma frase. Para ser identificado com o Caso nominativo, um SN deve sofrer regra de Movimento para o especificador de ST, pois é somente nessa posição funcional que o Caso do sujeito da sentença pode ser identificado. É isso o que se ilustra na figura a seguir.

Figura 9.24: Um SN deve ser movido para especificador de ST de modo a ser licenciado com o nominativo.



Tipicamente, são os argumentos externos de um predicador que são movidos para a posição de especificador de T, de modo a serem licenciados com o Caso nominativo. Veja que foi exatamente isso o que aconteceu com [João] na frase "Que livro João comprou?", representada na figura 9.24. Não obstante, o mesmo acontece com o argumento interno de verbos inacusativos. Numa frase como "O livro chegou", o SN livro é licenciado com nominativo, mas ele é o argumento interno do verbo inacusativo "chegar". Podemos, então, deduzir que "o livro" deve ser movido da posição de complemento de V para a de especificador de T, de modo a licenciar-se com o nominativo.

É importante ressaltarmos que mesmo no caso dos verbos que não selecionam argumento, como os que indicam fenômenos meteorológicos, tais como "chover", "nevar" ou "ventar", o Sistema Computacional também licencia um sujeito com o Caso nominativo. Trata-se dos sujeitos expletivos, que possuem uma expressão puramente funcional (e não lexical), como o *it* do inglês, o *il* do francês e os expletivos nulos do português, que estudamos nas unidades anteriores. Isso quer dizer que, independente das propriedades lexicais manifestadas numa sentença, o Caso nominativo associado ao sujeito, no interior do sintagma temporal, sempre deve ser identificado.

Marcação excepcional de caso

Em todos os exemplos que estamos analisando, a identificação de Caso acontece dentro da oração em que o SN/SD a ser licenciado encontra-se, ao lado de seu respectivo predicador. Mas pode ser que um argumento de um predicador seja licenciado com um Caso numa oração diferente daquela em que é gerado, junto a outro predicador. Quando isso acontece, dizemos haver uma marcação excepcional de caso. Um exemplo desse tipo de marcação é bem conhecido dos brasileiros. Trata-se das frases em que um pronome como "mim" aparece ao lado de um verbo no infinitivo, tal como em "Ele pediu para mim sair da sala". Nesse caso, o argumento externo sujeito de "sair" (o pronome na primeira pessoa do singular) não consegue ser licenciado com o Caso nominativo (pronome "eu"), pois na oração em que se encontra não há flexão finita – note que o verbo "sair" encontra-se no infinitivo não flexionado. É por isso que o pronome é alçado para a outra oração, na qual consegue receber o Caso oblíquo (pronome "mim"), na posição de complemento da preposição P. Como um argumento de um predicador ("sair") recebe identificação de Caso em associação com outro predicador ("para"), estamos diante de uma marcação excepcional de Caso. Na escola, aprendemos a evitar essa marcação excepcional e somos treinados a fazer outro tipo de licenciamento de Caso, no qual usamos a forma "eu" associada a uma suposta forma de infinitivo flexionado na primeira pessoa do singular.

Essa obrigatoriedade de identificação do sujeito de uma sentença, com o nominativo, é representada na linguística gerativa pelo chamado **traço EPP**. Tal traço é de fato uma imposição formal do sintagma temporal/flexional: todo ST deve licenciar um sujeito, em seu especificador, com o Caso nominativo.

O termo EPP é a sigla da expressão inglesa *Extended Projection Principle* (Princípio da Projeção Estendido). O "Princípio da Projeção (PP)" estabelece que os traços formais inscritos num item lexical devem ser projetados para a sintaxe. Já o "Princípio da Projeção Estendido (EPP)" afirma que, independente das informações do léxico, um item deve ser licenciado como sujeito da sentença, com Caso nominativo identificado em ST.

Se você vier a se aprofundar nos estudos de sintaxe gerativa, descobrirá que há muitos tópicos de pesquisa importantes a serem discutidos sobre a identificação e a expressão do Caso nas diferentes línguas humanas. Por exemplo, é preciso entender melhor se os Casos são apenas checados pelo Sistema Computacional, e já vêm estabelecidos desde o Léxico, ou se são na verdade valorados ou marcados pelo Sistema. É igualmente necessário estabelecer como acontece a inserção dos morfemas que explicitam Caso, na especificidade das línguas que possuem Caso morfológico. Além disso, a postulação de que o nominativo seja identificado na relação "especificador → núcleo" (no ST), enquanto o acusativo e oblíquo são identificados na relação "núcleo → complemento" (no SV e no SP) incomoda muitos sintaticistas. Muitos deles vêm propondo diferentes soluções teóricas para uniformizar a relação local em que todos os Casos devem ser identificados. Enfim, depois de ser introduzido no mundo do formalismo sintático, você descobrirá que há muitos problemas sobre o fenômeno "Caso" esperando por uma solução. Uma boa leitura para você começar a familiarizar-se com os estudos gerativistas mais avançados é o *Entendendo o minimalismo* (*Understanding minimalism*, de Hornstein, Nunes e Grohmann – 2005), livro que infelizmente só está disponível no original em língua inglesa.

Na próxima seção, analisaremos um tipo especial de constituinte linguístico. Trata-se das categorias vazias. Esses são elementos idênticos a um SN (ou um SD), com a particularidade de serem "nulos", isto é, são categorias que não manifestam nenhum conteúdo fonético.

Categorias vazias

Uma categoria vazia é um constituinte sintático manipulado pelo Sistema Computacional que possui a particularidade de não manifestar conteúdo fonético. É o que podemos chamar, metaforicamente, de *constituinte invisível* ou *conjunto vazio*. Por exemplo, o objeto direito de uma pergunta como "Você viu o João?" é o SN "o João", que possuiu manifestação fonética nítida, ao passo que o objeto de uma resposta como "Eu vi" é uma categoria vazia, uma espécie de sintagma sem pronúncia. Se você está se lembrando do "sujeito nulo" e do "objeto nulo", que aqui já estudamos, saiba que esses são bons exemplos de categorias vazias.

Existem quatro tipos de categoria vazia: "pro" (prozinho), "PRO" (prozão), *cópia* e *variável*. Esses tipos se distinguem em função de três propriedades linguísticas: Caso, papel temático e referentes potenciais. Vejamo-los um de cada vez.

A categoria vazia "pro" é gerada na base, isto é, é retirada de uma Numeração e inserida no curso de uma derivação, pela operação Select como qualquer outro tipo de item lexical. Devemos entender "pro" como um pronome nulo, uma forma de zero fonético de valor pronominal. Gramaticalmente, "pro" deve ser licenciado com papel temático e Caso da mesma forma que qualquer outro SN. Eis dois exemplos de ocorrência de "pro" em português.

(10) [pro Dormi cedo hoje]
(11) [Você viu o jogo? Eu não vi pro]

Em (10), "pro" é identificado com o argumento externo do verbo "dormir", que é licenciado com o Caso nominativo e interpretado com o papel temático de *experienciador*. Já no exemplo (11), "pro" é o argumento interno da segunda ocorrência do verbo "ver". Seu Caso é identificado como acusativo, o que conduz à interpretação de *tema* como seu papel temático. Como se vê, o comportamento de "pro" é, em português, idêntico ao de uma categoria nominal foneticamente realizada.

O comportamento sintático de "PRO" é um pouco diferente. Vejamos a seguir uma ocorrência de "PRO".

(12) [João prometeu [PRO estudar a matéria]]

"PRO" também deve ser selecionado da Numeração que alimenta uma derivação e ser combinado com os demais constituintes de uma representação, bem como deve receber papel temático de algum predicador. No entanto, "PRO" não é licenciado com Caso. Note que, no exemplo dado, "PRO" é o argumento externo de "estudar" e, assim, é licenciado com o papel temático de experienciador, mas não é identificado com nenhum Caso. Isso ocorre porque na oração [estudar a matéria] não acontece nenhuma flexão finita e, como já sabemos, é justamente a flexão finita que licencia, em ST, o Caso nominativo do sujeito da sentença. Vemos, portanto, que, diferentemente de "pro", "PRO" não precisa ser licenciado por uma identificação de Caso.

Você deve entender que "PRO" é o único tipo de argumento nominal que pode figurar numa representação sintática sem ser identificado com algum Caso. E isso acontece em virtude de "PRO" ser uma categoria vazia. Se no lugar de "PRO" usarmos uma categoria preenchida, isto é, um SN foneticamente realizado, então ele terá de ser licenciado com algum Caso, do contrário a derivação não será convergente. É isso o que vemos acontecer no exemplo que se segue.

(13) *[João prometeu [ele estudar a matéria]]

A agramaticalidade dessa frase acontece porque o SN "ele" não foi identificado com nenhum Caso. Para licenciar essa frase, teríamos de atribuir alguma flexão ao verbo "estudar", tal como a seguinte.

(14) [João prometeu que [ele vai estudar a matéria]]

Nessa frase, o verbo auxiliar flexionado "vai" oferece a flexão que licencia o Caso nominativo ao pronome "ele", o que torna a derivação convergente. "PRO"

é, por tudo o que dissemos, uma categoria vazia gerada na base que recebe papel temático, mas não é identificado com Caso – já que a obrigatoriedade do Caso se restringe aos constituintes com realização fonética. Tipicamente, "PRO" será o sujeito de oração infinitivas não flexionadas.

A terceira categoria vazia existente nas línguas naturais é a *cópia*. Já sabemos que uma cópia é um sintagma que foi apagado na FF em decorrência da aplicação de regra de Movimento, às vezes representada como "t". A especificidade dessa categoria vazia reside no fato de ela ser gerada no curso de uma derivação, isto é, ela não está presente na Numeração. Mais precisamente, uma cópia é gerada pelo movimento de um constituinte para uma posição argumental. Como estudamos na seção anterior, o ST de uma sentença possui uma posição A, em seu especificador, na qual se satisfaz o traço EPP com a identificação do Caso nominativo. Uma expressão nominal que ainda não tenha sido identificada com um Caso na camada lexical de uma derivação deve ser deslocada para o especificador de ST, de modo a licenciar-se com o nominativo. Ora, é justamente a cópia apagada (t) do sintagma movido para essa posição argumental em ST que denominamos como a categoria vazia *cópia*.

(15) $[_{ST}$ João$_i$ $[_T$ vai $[_{SV}$ t_i estudar a matéria]]]

Essa definição nos levará a identificar na cópia propriedades semelhantes às de "PRO", já que ela possui papel temático, estabelecido na posição de argumento de um predicador, mas não é identificada com Caso. No exemplo, podemos ver que a cópia (t) é o argumento externo de "estudar", licenciado com o papel temático de experienciador. "Estudar" não apresenta flexão finita, sendo assim não possui elemento T que possa licenciar o nominativo, é por isso que "João" foi movido para ST, numa posição em que é identificado com Caso e satisfaz o traço EPP.

Por fim, a categoria vazia *variável* é também um tipo de cópia, mas possui a especificidade de ser uma cópia de um constituinte movido para uma posição não argumental. Vejamos um exemplo:

(16) $[_{SC}$ Que matéria$_j$ $[_{ST}$ João$_i$ $[_T$ vai $[_{SV}$ t_i estudar $t_j]]]]$

Aqui já sabemos identificar t_i como uma cópia, já que se refere ao constituinte "João", que foi movido para uma posição argumental. O fato novo é a ocorrência do segundo t, identificado como t_j. Esse t_j é a cópia do sintagma "que matéria", que foi movido para uma posição não argumental, em SC. Trata-se, portanto, de uma variável.

Como categoria vazia, a variável distingue-se da cópia em função da posição de destino da regra de Movimento. Enquanto a cópia possui um constituinte movido para uma posição A, a variável possui um constituinte movido para uma posição A'. Isso fará com que a variável tenha propriedades semelhantes às de "pro", uma vez que possui tanto papel temático quanto Caso. No exemplo, você pode conferir que t_j é o argumento interno do verbo "estudar". É nessa posição de complemento que a variável recebe papel temático (tema) e Caso (acusativo).

Numa última palavra sobre as categorias vazias, precisamos descrever seus antecedentes potenciais. "pro" é o equivalente a um pronome pessoal nulo e, assim, não precisa ter obrigatoriamente um referente anafórico na frase ou no discurso – pense, por exemplo, nos pronomes "eu", "nós" ou "você", que são dêiticos e não anafóricos e podem todos ser substituídos por "pro" numa língua com o parâmetro [+ sujeito nulo]. Trata-se de uma categoria vazia classificada, portanto, com os traços [+ pronominal] e [- anafórico].

Por sua vez, "PRO" tipicamente possui referente anafórico, mas também pode ter uma interpretação pronominal arbitrária (sem referente ou antecedente), como acontece em [PRO viver é preciso]. Dessa forma, "PRO" é uma categoria vazia que conjuga os traços [+ pronominal] e [+ anafórico].

Já a cópia sempre possui seu referente em posição A, mas não veicula valor pronominal, sendo, por conseguinte, caracterizada como uma categoria [- pronominal] e [+ anafórico]. Por fim, o referente de uma variável ocupa uma posição A', caracterizando-se como uma categoria [- pronominal] e [- anafórico], uma vez que não pode ser interpretado como um pronome, tampouco como anáfora, tal como ocorre com expressões referenciais.

Aprimoramento do modelo

Com esta unidade, procuramos apresentar a você os conceitos e as tecnicalidades básicas acerca da natureza e do funcionamento do Sistema Computacional da linguagem humana. Esse conhecimento lhe será útil quando iniciar-se em estudos gerativistas mais avançados. Não obstante, você não deve, de maneira alguma, ficar com a impressão de que a descrição das computações sintáticas seja um capítulo concluído no empreendimento da linguística gerativa. Na verdade, ainda existem muitas questões teóricas e metodológicas em plena discussão entre os estudiosos do gerativismo interessados no Sistema Computacional.

É com esse espírito de "pesquisa em desenvolvimento" que gostaríamos de convidar você a fazer, ao final desta unidade, uma reflexão sobre certos aspectos do gerativismo que poderão ser modificados futuramente a fim de aprimorar o poder descritivo e explanatório do modelo. Com efeito, o momento atual da linguística gerativa – o Programa Minimalista – não é uma teoria fechada. Antes, como o

próprio nome o indica, trata-se de um *programa* de pesquisa aberto, que poderá reformular-se continuamente na busca da melhor descrição possível de nossa cognição linguística. Nesse sentido, apresentaremos a seguir questões e questionamentos que poderão aguçar a sua curiosidade científica. Você verá que, mais do que "*aprender sintaxe*", a linguística gerativa convidará você a "*fazer sintaxe*".

A primeira questão a apresentar diz respeito ao lugar do componente morfológico das línguas naturais na arquitetura da linguagem e, particularmente, na dinâmica de funcionamento do Sistema Computacional. As propostas de Chomsky (desde 1995 até o presente) não atribuem à estrutura interna das palavras papel relevante nas computações do Sistema durante a derivação do par (π, λ). Na interpretação chomskiana clássica, as computações morfológicas dão-se ou no interior do Léxico, antes portanto das operações que geram as representações sintáticas, ou em FF, quando os itens abstratamente computados pelo Sistema assumem a representação morfofonológica que será enviada para pronúncia na interface articulatório-perceptual. Muitos linguistas discordam dessa interpretação e argumentam que, na verdade, a morfologia desempenha papel crucial na arquitetura da linguagem. Por exemplo, os linguistas norte-americanos Morris Hale (do Instituto de Tecnologia de Massachusetts) e Alec Marantz (da Universidade de Nova York) propuseram, em 1993, um modelo linguístico denominado Morfologia distribuída. Nesse modelo, que apresentamos rapidamente ao final da unidade 7, a morfologia é interpretada como um componente de interface com o Sistema Computacional e também com FF e FL, aos quais fornece listas de palavras reais e valores enciclopédicos de referência para seus significados. Trata-se de um modelo muito importante, com forte possibilidade de integração com a arquitetura minimalista de linguagem que aprendemos neste curso. É possível, portanto, que os modelos gerativistas do futuro representem o componente morfológico no interior do Sistema Computacional ou, pelo menos, em alguma instância de interface com ele.

Outro ponto de discussão relevante refere-se ao quanto de representação e de derivação há na faculdade da linguagem. Como vimos, a proposta minimalista assume que existem duas e somente duas representações linguísticas: π e λ. Nesse sentido, os sintaticistas devem caracterizar todo o funcionamento da linguagem, desde a Numeração, passando pelas operações do Sistema Computacional até a formação de representações em FF e FL, de maneira eminentemente derivacional. No entanto, conforme apontamos durante esta unidade, o conceito de Numeração organizada em subarranjos, com índices e núcleos de fase devidamente organizados, parece conferir ao modelo uma indesejável natureza representacional. Diversos gerativistas têm indicado que a noção de subarranjo ressuscita o nível de representação que era conhecido, nos anos 60, 70 e 80, como *estrutura profunda*. Por exemplo, Kenedy (2007) abordou o assunto em sua tese de doutoramento e, para ele, uma boa solução para o problema é reconhecer a existência de uma interface cognitiva entre o Léxico e o Sistema Computacional. Tal interface cor-

responderia ao planejamento conceitual de fala humana e seria nela que os itens lexicais que alimentam uma derivação sintática seriam organizados em estruturas proposicionais, tais como as fases, de modo a criar uma Numeração organizada.

Na verdade, existem muitas propostas sobre o tema "representação *versus* derivação". Gerativistas importantes, como Juan Uriagereka e Michael Brody, defendem abertamente a necessidade de o modelo gerativista voltar a ser mais representacional e menos derivacional, assumindo a existência de pelo menos três representações linguísticas: estrutura profunda, π e λ. Outras propostas, muito pelo contrário, defendem um *derivacionalismo radical*. Segundo linguistas como Samuel Epstein, as interfaces articulatório-perceptual e conceitual-intencional devem acessar diretamente as derivações do Sistema Computacional, sem a necessidade das representações em FF e FL. O acesso direto das interfaces às computações do Sistema pode ser uma hipótese científica interessante para o desenvolvimento do gerativismo. O que você pensa a respeito? Eis aqui um bom tema de pesquisa para a sua pós-graduação.

Por fim, um dos temas mais interessantes para a agenda futura do gerativismo é articulação entre o minimalismo ontológico e o minimalismo metodológico. Conforme aprendemos, o minimalismo ontológico diz respeito a como compreendemos a natureza da faculdade da linguagem e suas relações com as interfaces, ao passo que o minimalismo metodológico refere-se à maneira pela qual o sintaticista conduz o seu trabalho descritivo.

Em termos ontológicos, a interação dinâmica entre a faculdade da linguagem e as interfaces está claramente estabelecida como questão relevante para o debate na linguística gerativa. Há uma grande quantidade de livros e artigos que vem sendo publicados recentemente sobre esse tema. Já em termos metodológicos, a relação entre as interfaces e as operações do Sistema Computacional ainda não parece estar muito bem fixada na ordem do dia. Muitos linguistas assumem, ainda que tacitamente, que as interfaces sejam um tópico de pesquisa relevante para os estudiosos do desempenho linguístico, mas não para os teóricos da competência linguística. Outros estudiosos, por sua vez, assumem uma postura contrária e defendem a hipótese de que os estudos das interfaces devem ser trazidos para o interior da teoria linguística, de modo que modelos de competência e de desempenho possam ser integrados ou, pelo menos, articulados. É isso o que propõem, por exemplo, as linguistas brasileiras Letícia Sicuro Corrêa (da PUC-Rio) e Marina Augusto (da UERJ). Elas formularam, no ano de 2007, o Modelo Integrado de Computação On-line (MINC), que apresenta uma conjugação muito interessante entre derivação minimalista e modelos psicolinguísticos de produção e compreensão da linguagem em tempo real. Por detrás de modelos integrados como esse, está a necessidade de, na caracterização das operações do sistema computacional (o minimalismo metodológico), levarmos em consideração a natureza cognitiva da linguagem em seu nicho natural de interação com suas interfaces (minimalismo ontológico).

Não sabemos como essas e outras questões relevantes serão desenvolvidas pelos gerativistas num futuro próximo. Talvez você venha a ser um deles! Seja então bem-vindo à nossa busca por adequação observacional, descritiva e explanatória.

Conclusão

Nesta unidade, aprendemos a maneira pela qual os gerativistas concebem o Sistema Computacional da linguagem. Esse Sistema retira unidades lexicais, compiladas numa Numeração, para gerar derivacionalmente representações sintáticas que serão enviadas aos sistemas de interface por FF e FL. Nesse processo, o Sistema lança mão de algumas operações computacionais, como Select, Merge e Move, sob a regulação do Princípio da Interpretação Plena e do Princípio de Economia Derivacional. Há diversos tipos de exigência derivacional que o Sistema deve atender ao gerar representações, tais como a identificação de Caso, a satisfação do traço EPP e o licenciamento das categorias vazias. A visão de Sistema Computacional que aqui aprendemos é uma construção teórica, fruto do trabalho epistemológico do gerativista. Num futuro próximo, diversas questões poderão ser reformuladas e novas questões poderão ser apresentadas de modo a aprimorar o modelo teórico do gerativismo.

Exercícios

1) O que se deve entender por derivação e representação em linguística gerativa?
2) Descreva os passos derivacionais que geram a representação: [Paulo comprou uma blusa].
3) Descreva os passos derivacionais que geram a representação: [Que roupa Paulo comprou].
4) Qual é a diferença entre Numeração e subarranjo?
5) Caracterize e exemplifique os três tipos de regra de Movimento existentes.
6) Qual é a relação existente entre Caso e traço EPP?
7) O que você entendeu por "Sistema Computacional" e "operações derivacionais"? Por que essas noções são relevantes para o estudo da dimensão cognitiva da linguagem humana?
8) Explique como os Princípio da Interpretação e o Princípio da Economia Computacional orientam a derivação de estruturas sintáticas.