PROJETO PUL∅

Maria das Graças Volpe Nunes (coordenadora)
Jorge Marques Pelizzoni
Juliana Galvani Greghi
Ricardo Hasegawa
Ronaldo Teixeira Martins

Relatório do projeto de desenvolvimento da versão experimental do módulo de tradução português-transcrição de libras para um tradutor semi-automático global português-libras.

Núcleo Interinstitucional de Lingüística Computacional - NILC
São Carlos
Maio de 2003

SUMÁRIO

1. Introdução

O presente relatório tem por objetivo reportar o desenvolvimento do PUL∅ – Portuguese-UNL-LIST deOralizer, versão experimental de um sistema de tradução automática unidirecional de uma língua oral-auditiva, o português, para a representação linear (*LIbras Script for Translation – LIST*) de uma língua gestual-visual, a língua brasileira de sinais, ou libras. O principal objetivo do PUL∅ é converter uma sentença originalmente produzida em língua portuguesa para uma transcrição especializada de libras que seja um compromisso entre (i) simplificação do processo de tradução e (ii) suficiência para a *síntese de fala* em libras.

O PUL∅ acompanha a tecnologia de sistemas de tradução automática auxiliada por humanos (*human-aided machine translation)*, na medida em que exige, no processo de tradução, alguma interação com o usuário humano proficiente em língua portuguesa. Não é objetivo da ferramenta realizar toda a conversão português-LIST de forma completamente automática, mas contar com o conhecimento humano – particularmente com o conhecimento de difícil ou ainda impossível formalização computacional – para resolver as ambigüidades e os desvios lingüísticos que possam ser observados nas sentenças de entrada do sistema. No entanto, a expectativa da ferramenta é a de que seu usuário final possa ser monoglota e não-especialista, ou seja, de que possa constituir mão-de-obra mais barata e menos qualificada do que a requerida em uma atividade de tradução comum. Principalmente, a ferramenta parte do compromisso de não exigir desse usuário final nenhum conhecimento de libras ou de qualquer outra língua gestual-visual.

O PUL∅ toma, como entrada, uma subvariedade simplificada da língua portuguesa (aqui chamada "português normalizado"), desprovida de elipses, topicalizações, anacolutos, anáforas, ambigüidades léxicas e sintáticas e outros acidentes lógico-gramaticais que pudessem vir a afetar o desempenho da ferramenta. Este processo de normalização corresponde exatamente à parte interativa da ferramenta e segue, em linhas gerais, as estratégias disponíveis para a produção de "línguas controladas" (*controlled languages*), descritas na Seção 6.1.

Para a análise semântica desse subconjunto do português, utilizou-se uma linguagem de representação do conhecimento (a *Universal Networking Language - UNL*[1]), que operou, no protótipo, como interlíngua, para a qual era convertida a sentença em língua portuguesa, e da qual era gerada a representação linearizada da língua brasileira de sinais (LIST). A representação UNL ofereceu a perspectiva de desenvolvimento de um sistema de tradução baseado, principalmente, em informação de natureza semântica, em detrimento das estruturas sintáticas, que são consideravelmente diferentes entre português e libras. Em última instância, esta escolha representou a adoção de técnicas de transdução de informação originalmente empacotada em estruturas lineares (a sentença do português), que foi

[1] UCHIDA, H.; ZHU, M.; DELLA SENTA; T. A *gift for a millenium.* Tóquio: IAS/UNU, 1999.

inicialmente convertida para estruturas reticuladas (os grafos UNL) e, em seguida, reapresentada como nova estrutura linear (as listas de LIST).

A ausência de recursos e de tecnologia previamente disponíveis e o curto prazo de tempo para a implementação do protótipo fizeram que a prototipagem do PUL∅ se visse limitada a uma única história em quadrinhos da Turma da Mônica, apresentada no Anexo I, selecionada pela coordenação geral do projeto, e composta de 12 frases, de complexidade média. Embora este corpus possa parecer excessivamente limitado, deve-se salientar que nosso objetivo primeiro envolvia apenas a verificação da viabilidade da abordagem escolhida, e não o desenvolvimento de uma ferramenta genérica, robusta, de aplicabilidade ilimitada. As conclusões parciais extraídas desta experiência, embora remetam diretamente aos fatos lingüísticos e problemas computacionais associados a esse pequeno conjunto de 12 frases, permitiram a antecipação de muitos outros problemas e a revisão de algumas estratégias adotadas, que se espera possam ser testadas e eventualmente referendadas para um conjunto maior de dados, em uma desejável segunda etapa do projeto.

Durante o desenvolvimento do protótipo, inúmeros recursos lingüísticos foram desenvolvidos. São eles: a notação LIST, o formalismo gramatical NL-UNL, a gramática português-UNL, o dicionário português-UNL, a gramática UNL-LIST e o dicionário UNL-LIST. Paralelamente foram também produzidos subsistemas computacionais que atuam de forma modular na plataforma PUL∅: o normalizador do português (Kaapor), a ferramenta de análise genérica NL-UNL (HERMETO) e o conversor UNL-LIST (ManateCo), construído a partir dos aplicativos de geração (DeConverter, versão 2.5) e de construção de dicionários (DicBld, versão 2.1) fornecidos pela *Universal Networking Digital Language Foundation*, de Genebra. O principal objetivo deste relatório, a par da apresentação geral da abordagem adotada, consiste no detalhamento de cada um desses módulos e recursos, bem como das escolhas e decisões técnicas a eles associadas.

A estrutura deste relatório é a que segue. Na Seção 2, recupera-se, em linhas gerais, o suporte teórico das decisões metodológicas que balizaram o desenvolvimento do protótipo. É apresentada também a estrutura geral do PUL∅, com a descrição sumária de seus módulos. A Seção 3 introduz a notação proposta para saída do PUL∅ e que deve servir de entrada ao sintetizador de fala. Trata-se de LIST, uma representação linearizada de libras a partir do vocabulário do português. A Seção 4 apresenta, de forma bastante abreviada e sintética, as características da *Universal Networking Language*, linguagem de representação de conhecimento utilizada como representação intermediária entre português e LIST. O corpus de treinamento é apresentado, analisado e discutido na Seção 5. Na Seção 6, descrevem-se os mecanismos e os procedimentos utilizados para a tradução português-UNL. São apresentados, principalmente, as rotinas de normalização do português, o formalismo da gramática de análise e a estrutura do dicionário português-UNL. São também discutidos os resultados obtidos para o *corpus* analisado. A Seção 7 apresenta o processo de geração de LIST a partir da representação UNL. Faz-se aqui breve referência às ferramentas genéricas disponibilizadas pela UNDL Foundation e ao processo de customização para que se pudesse adaptá-las para a geração de LIST. A Seção 8 descreve as estratégias de integração de todos esses módulos e recursos em uma mesma ferramenta. A Seção 9, por fim, apresenta as conclusões gerais e avança algumas possíveis estratégias

de desenvolvimento futuro. Nos anexos são apresentadas, entre outros, as bases de dados lingüísticos compiladas ao longo do desenvolvimento do projeto.

Este projeto se ressente de uma série de restrições, principalmente relativas ao tempo de desenvolvimento (de setembro de 2002 a maio de 2003) e à interlocução com a equipe responsável pela síntese de fala, que impediram que as conclusões parciais dele extraídas pudessem ser analisadas à luz de um conjunto maior de informações. Em que pese a larga experiência do NILC no campo do processamento automático da língua portuguesa, o grupo não tinha nenhuma experiência anterior relativa ao processamento de línguas de sinais, e as informações disponíveis a respeito de libras e do processamento automático de libras revelaram-se bastante escassas. O trabalho de desenvolvimento da ferramenta revestiu-se, portanto, em todos os momentos, de um caráter de inovação acentuada e de desbravamento de um território de pesquisa que, pela relevância social, inspira maiores investimentos e deveria evitar soluções de continuidade.

A equipe do projeto é particularmente grata à professora Tanya Felipe, da Universidade do Estado e Pernambuco e do Instituto Nacional de Educação dos Surdos, pelo fornecimento do material didático sobre libras e pelas duas visitas feitas a São Carlos para o esclarecimento de dúvidas relativas às escolhas do projeto.

2. Tradução Intersemiótica Automática

A atividade prevista no projeto global – a recodificação de uma mensagem originalmente produzida no registro da escrita de uma linguagem verbal natural (o português brasileiro) para uma linguagem gestual-visual (libras) - enquadra-se no que vem sendo chamado, nos estudos da linguagem, de tradução intersemiótica ou, mais especificamente, *modality translation*. Trata-se, até onde pudemos observar, de um domínio já razoavelmente explorado dentro dos estudos da tradução automática, para o qual não foram encontradas, no entanto, outras iniciativas para a língua portuguesa ou para a língua brasileira de sinais[2]. A bibliografia recenseada, principalmente relativa a sistemas de tradução do inglês para a *American Sign Language*, parece convergir para três pontos principais: a) a idéia de que os sistemas de tradução automática intermodal acompanham, em linhas gerais, os princípios, abordagens e técnicas já desenvolvidos para os sistemas intramodais (de uma língua oral-auditiva para outra língua oral-auditiva), a despeito das diferenças de suporte; b) a idéia de que os sistemas de tradução intermodal se subdividem, na verdade, em dois subsistemas: o de tradução de uma língua oral-auditiva para um sistema de escrita da língua gestual-visual; e o de síntese de sinais (gestual-visuais) a partir desse sistema de escrita; e c) a idéia de que a complexidade da tarefa está evidentemente relacionada ao sistema de escrita da língua gestual-visual adotado. Nesta seção, cada um desses tópicos será referido separadamente, para que se possam estabelecer os fundamentos teóricos para as escolhas que foram feitas.

Em primeiro lugar, cabe referir a idéia de que não há diferenças expressivas entre as tecnologias de tradução intermodal e as tecnologias de tradução intramodal. A área conhecida como Tradução Automática (*Machine Translation*), precursora dos estudos sobre o processamento automático das línguas naturais, e cujos primeiros ensaios já contam mais de 50 anos, tem proposto, ao longo de sua tumultuada história, vários paradigmas e técnicas de tradução. Verifica-se a existência de pelo menos três abordagens predominantes: a tradução baseada exclusivamente em conhecimento lingüístico, ou seja, em dicionários e gramáticas (*Language-Based Machine Translation* – LBMT); a tradução baseada em conhecimento, ou seja, em dicionários, gramáticas e, adicionalmente, enciclopédias e bases de conhecimento (*Knowledge-Based Machine Translation - KBMT*); e a tradução baseada em exemplos, ou seja, em dicionários, gramáticas e *corpora (Example-Based Machine Translation – EBMT)*. Os dois primeiros casos constituiriam, principalmente, modelos de tradução baseada em regras, ou na explicitação do conhecimento lingüístico inato do falante; o último seria particularmente amparado em análises e dados estatísticos. O primeiro modelo, em função do custo relativamente mais baixo se comparado aos demais,

[2] Em uma pesquisa feita, em outubro de 2002, no motor de busca Google em várias tentativas de combinação das palavras-chave "libras", "LSB", "língua brasileira de sinais", "língua de sinais brasileira", "língua de sinais dos centros urbanos brasileiros", "LSCB", "Brazilian Sign Language", de um lado; e "tradução automática" ou "machine translation" de outro, não surgiram documentos ou referências que satisfizessem o critério. Em contrapartida, para a combinação "sign language" e "machine translation" foram disponibilizados mais de 1.700 documentos.

seria mais adequado para sistemas mais genéricos e mais robustos, mas produziria resultados menos satisfatórios e mais sujeitos a erro. Os dois últimos, por envolverem o desenvolvimento de recursos mais dispendiosos (enciclopédias e *corpora* convenientemente anotados, separados por domínio do conhecimento), produziriam resultados mais exatos, mas seriam indicados apenas para sistemas mais especializados, de domínio restrito.

Do ponto de vista da técnica, são citadas duas linhas principais: a tradução direta e a tradução indireta. A tradução direta prevê, em linhas gerais, que a língua-alvo seja considerada o próprio instrumento de análise da língua-fonte. Ou seja, não haveria, em princípio, nenhum estágio intermediário entre língua-fonte e língua-alvo. O vocabulário da sentença de entrada seria automaticamente vertido para a língua-alvo por meio de um dicionário bilíngüe, com a ajuda, talvez, de algum processamento morfológico. Uma vez geradas as equivalências lexicais na língua-alvo, haveria algum reordenamento (bastante superficial e localizado) dos itens lexicais, para produzir resultados mais aceitáveis (como a posposição do adjetivo, por exemplo, no caso das traduções do inglês para o português). Não haveria propriamente processamento sintático das sentenças originais da língua-fonte, ou qualquer outro tipo de processo semântico. Por sua simplicidade, produz resultados bastante ruins, e não seria adequada senão para o desenvolvimento de sistemas de tradução de menus, expressões formulaicas e palavras isoladas.

A tradução indireta prevê o desenvolvimento de uma forma de representação intermediária entre a língua-fonte e a língua-alvo. Esta forma de representação pode ser dependente das línguas envolvidas, no sentido de constituir uma interface específica (unidirecional ou bidirecional), ou pode ser independente tanto da língua-fonte quanto da língua-alvo, procurando organizar-se como uma outra língua, artificial, autônoma, neutra, porém mais adequada ao processamento automático (porque livre de ambigüidade, por exemplo). No primeiro caso, fala-se em tradução indireta baseada em transferência (Fig.1); no segundo, em tradução indireta baseada em interlíngua (Fig.2).

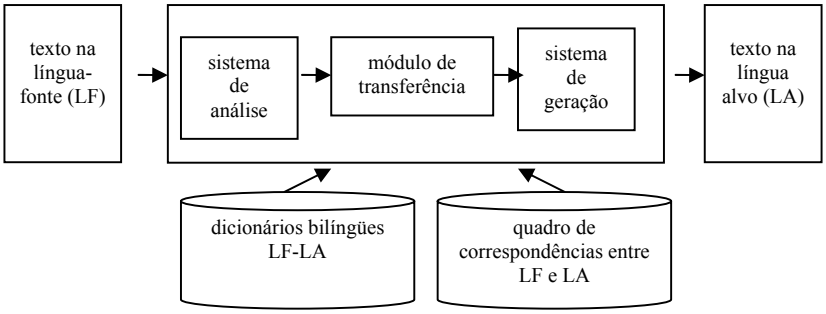


Figura 1 - Sistema de Tradução baseado em Transferência

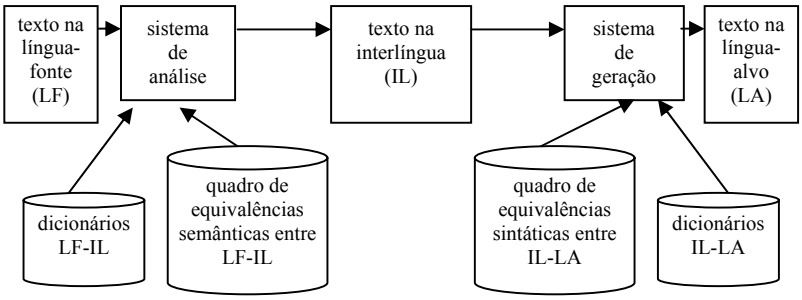


Figura 2 - Sistema de Tradução baseado em Interlíngua

Os sistemas de tradução intermodal poderiam ser classificar, como qualquer sistema de tradução automática, em qualquer dessas abordagens e técnicas, a depender sempre dos recursos disponíveis e das estratégias de tradução adotadas. Os dicionários língua natural-língua de sinais, na medida em que traduzem, isoladamente, palavras por sinais, acompanham, por exemplo, a técnica de tradução direta; sistemas que prevêem, entre a linearidade da língua oral-auditiva e a tridimensionalidade da língua gestual-visual, uma representação intermediária (como um sistema de escrita linear para a língua de sinais), se enquadrariam entre as técnicas de tradução indireta.

O PUL∅, protótipo projetado pelo NILC para compor um módulo do sistema de tradução português-libras, foi deliberadamente proposto como um sistema de tradução automática a) baseado exclusivamente em conhecimento lingüístico e b) que utiliza a estratégia de tradução indireta por interlíngua. A primeira opção, caracterizada pelo fato de o sistema prever apenas o desenvolvimento de dicionários e gramáticas e dispensar a construção de outros repositórios de informação, se explica, em larga medida, por uma série de restrições operacionais que diziam respeito, particularmente, à definição do *corpus* a ser trabalhado. Na medida em que se definiu que se trabalharia sobre histórias em quadrinho, de temática bastante variada, com a predominância de gêneros primários do discurso[3], não se revelou viável a construção de uma base de conhecimento que permitisse o equacionamento de todas as ambigüidades relativas aos textos de entrada. Por este motivo, decidiu-se, desde o início, que o sistema proveria essas informações por meio da interação com o usuário humano, que faria as vezes, portanto, de repositório adicional de informações, substituindo enciclopédias e outras bases de conhecimento normalmente adotadas em outros sistemas de tradução.

A adoção de uma estratégia de tradução indireta baseada em interlíngua foi principalmente derivada da experiência prévia do grupo com um modelo desta natureza (o sistema UNL); da aposta nas virtudes desta proposta, particularmente no que tange à facilidade de expansão do modelo, com a incorporação de outras línguas, o que permitiria

[3] Cf. BAKHTIN, M. Gêneros do discurso. In A estética da criação verbal. São Paulo: Martins Fontes, 2000[1953].

que pudéssemos desenvolver, no futuro próximo, uma plataforma multilíngüe de tradução para libras; e da verificação das diferenças – semânticas, principalmente – entre o português e libras, particularmente quando se percebeu que a organização fonológica e morfossintática da língua de sinais seria afetada por um conjunto de marcadores semânticos (de natureza visual) que normalmente não está disponível nas descrições gramaticais das línguas naturais. Desta forma, previu-se que a tradução português-libras mereceria contar com um nível de representação intermediário, que fosse suficientemente autônomo, seja em relação à língua-fonte, seja em relação à língua-destino.

A modularização dos sistemas de tradução intermodal parece ser uma opção mais pacífica do que a determinação de suas abordagens e estratégias. Quando se confrontam os dois diferentes suportes – bidimensional, no caso da língua oral-auditiva, e tridimensional, no caso da língua gestual-visual – parece ser uma escolha razoavelmente óbvia a subdivisão do processo em duas etapas muito distintas: (i) tradução da língua oral-auditiva para algum tipo de transcrição simplificada da língua gestual-visual e (ii) síntese de fala (no sentido lato), em que a transcrição é "executada", ou seja, convertida num enunciado gestual-visual sintetizado. No caso do projeto em tela, estas duas etapas foram, desde o início, distribuídas entre atores diferentes: o NILC, em função de sua já considerável tradição no desenvolvimento de ferramentas de análise de língua natural, ficou responsável pela parte da tradução; o ???? ficou responsável pela parte gráfica relativa à síntese dos sinais. Entre os dois módulos, no entanto, observou-se, desde sempre, a necessidade da compatibilidade das informações: a saída provida pelo tradutor deveria obrigatoriamente respeitar as necessidades estabelecidas pelo sintetizador. No entanto, dada a falta de sincronia entre os desenvolvedores das duas partes do sistema, particularmente porque o trabalho de análise teria precedido bastante a contratação de uma equipe de síntese, o NILC acabou por definir unilateralmente um formato de saída, inspirado na notação linear proposta por Felipe (1998)[4], modificada porém para eliminar as ambigüidades estruturais e os problemas de digitação derivados desta proposta. Esta nova notação – batizada de LIST – será apresentada na Seção 3.

A arquitetura geral do PUL∅ é apresentada na Figura 3. O dado de entrada – a fala de uma personagem em cada um dos quadrinhos de uma história da Turma da Mônica - é digitado pelo usuário e é desmembrado automaticamente, pelo *Recognizer*, em unidades de processamento (sentenças), a partir de algumas fronteiras sentenciais pré-estabelecidas (ponto, ponto-de-interrogação, ponto-de-exclamação, etc). Em seguida, a sentença isolada é normalizada semi-automaticamente, com a interação do usuário, para que se possam resgatar as informações não disponíveis em sua estrutura superficial (elipses, referências anafóricas, etc.) e para que se possam resolver todas as ambigüidades observadas. O *Normalizer* também verifica a consistência ortográfica do vocabulário utilizado, reconhecendo interjeições e grafias desviantes do padrão da língua ("*obrigadooooo...*", por exemplo), muito comuns na reprodução da fala das personagens, principalmente do Cebolinha, que troca sistematicamente os "r" pelos "l". A sentença normalizada é subdividida em itens lexicais, pelo *Tokenizer*, que recolhe as informações relativas a cada

[4] FELIPE, T. A. F. S. *A Relação Sintático-Semântica dos Verbos e seus Argumentos na Língua Brasileira de Sinais*. Tese de doutorado, UFRJ, 1998.

entrada no dicionário e as repassa para o *parser*, que opera a análise sintática automática a partir das possibilidades combinatórias contidas na gramática português-UNL. As regras sintáticas estão associadas a regras de projeção semântica, ativadas pelo *Interpreter*, que gera finalmente o grafo UNL para a sentença analisada. A representação UNL, um conjunto de relações binárias unidirecionais entre itens do vocabulário da linguagem UNL, serve de entrada para o DeCo, ferramenta genérica de decodificação de UNL, que converte a representação reticulada em uma nova lista, agora em LIST, com o auxílio da gramática e do dicionário UNL-LIST. Este resultado final – a sentença em LIST – serviria de entrada ao segundo módulo do tradutor geral português-libras, responsável pela síntese de fala.

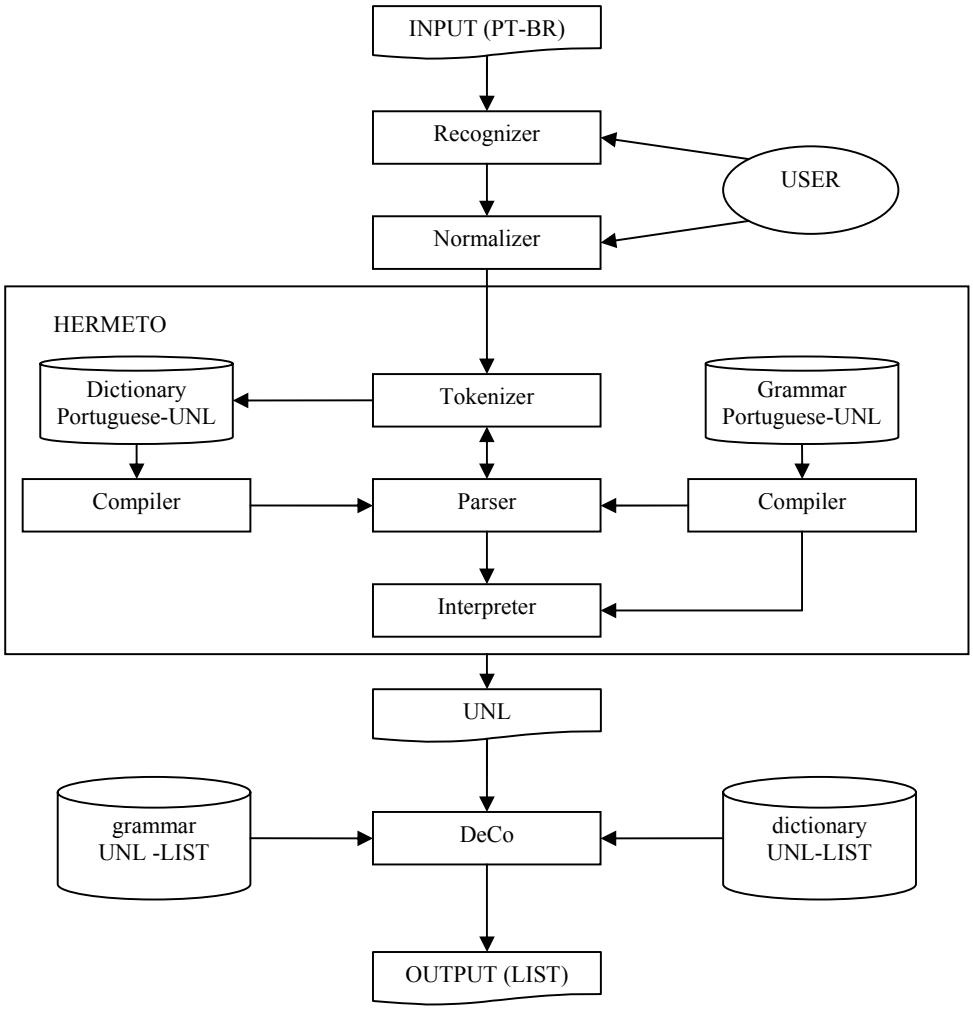


Figura 3 - Arquitetura Geral do PULØ.

3. LIST – LIbras Script for Translation

O objetivo desta seção é apresentar uma proposta de transcrição de libras (i) afeita ao tratamento computacional, (ii) concebida especialmente para fins de tradução (semi)automática intermodal e (iii) que seja um compromisso entre (iii.i) simplificação do processo de tradução e do desenvolvimento do tradutor e (iii.ii) suficiência para a *síntese de fala* na língua brasileira de sinais. Mais especificamente, a formalização ora proposta constitui o formato de saída de PUL∅ e o formato de entrada do sintetizador de fala. Seu objetivo seria representar, de forma não ambígua, todas e apenas as informações necessárias para a geração de sinais de libras a partir de uma sentença em língua portuguesa, considerando, particularmente:

a) que português e libras são línguas completamente distintas, não apenas em relação à estrutura (gramática e léxico), mas ao próprio suporte;

b) que a sentença em língua portuguesa possui redundâncias (como a concordância de número, de gênero e de pessoa gramatical) que seriam suprimidas no processo de representação em libras;

c) que a sentença em língua portuguesa possui informações (como a ordem dos constituintes) que nem sempre são relevantes para a representação em libras;

d) que a sentença em língua portuguesa possui lacunas (informação sobre classificadores, elipses, anáforas e outras pró-formas, nominais e verbais) que deverão ser preenchidas, ainda que de forma arbitrária, para que se possa assegurar a gramaticalidade e a semanticidade dos enunciados de libras.

Para efeito de clareza, a especificação de LIST será apresentada em três diferentes níveis de descrição: a) o vocabulário LIST; b) a estrutura frasal de LIST; c) a estrutura dos documentos em LIST.

É essencial entender que esta seção propõe simplesmente o ferramental formal básico que, em princípio, deverá dar conta de uma transcrição de libras que atenda aos requisitos citados. Muito de LIST não está nem poderia estar definido aqui ou em qualquer outro lugar neste momento, já que depende diretamente de volume de transcrição executada, mediante o qual (i) um dicionário robusto LIST será montado; (ii) um conjunto de atributos LIST, definido (tarefa parcialmente inclusa na anterior); e (iii) a adequação e suficiência das entidades aqui propostas, verificada. Por esse motivo, no final desta seção, incluímos um resumo do estado corrente de LIST resultante da experiência obtida nas transcrições realizadas.

3.1. Vocabulário LIST

O vocabulário LIST é constituído de *LIST Words* (*LWs*), que corresponderiam à representação linear (bidimensional) dos sinais de libras. Por razões mnemônicas, este vocabulário será definido a partir do conjunto de itens lexicais de língua portuguesa, ao invés de procurar representar índices para a síntese de fala, como propõem muitos outros sistemas de escrita para línguas de sinais (HamNoSys, por exemplo), de aplicação diversa à pretendida para LIST, no entanto. Cada LW é composta de uma *headword*, necessariamente, seguida de uma matriz atributo-valor, opcionalmente.

Para as headwords, será adotada a seguinte convenção: todos os sinais de natureza fonográfica (ou seja, que procuram imitar, em sua formação, a estrutura fonológica das palavras da língua portuguesa, por meio da datilologia), serão representados em letras maiúsculas, sem hífen; os demais sinais, de natureza não-fonográfica (ou seja, logográfica, ideográfica, pictográfica, etc.), serão representados por letras minúsculas[5]. A par desta convenção, serão adotados ainda os seguintes princípios:

a) lexias simples de libras que não selecionam classificadores serão representadas pelos itens lexicais da língua portuguesa correspondentes: ‘casa’, ‘estudar’, ‘criança’;

b) lexias compostas e lexias complexas do português que correspondem a lexias simples de libras serão representadas pelos itens lexicais da língua portuguesa correspondentes conjugados com hífen: ‘cortar-com-faca’, ‘querer-não’, ‘meio-dia’;

c) lexias simples do português que correspondem a lexias compostas de libras serão representadas pela combinação dos itens lexicais da língua portuguesa correspondentes conjugados pelo símbolo "^": ‘cavalo^listra’ (zebra).

Dada a estrutura morfológica dos sinais de libras, as headwords de LIST não compreenderão nenhum sufixo flexional específico, sendo representados pelas formas do masculino singular, no caso dos substantivos e palavras de natureza substantiva; do masculino singular, no caso dos adjetivos e palavras de natureza adjetiva; e do infinitivo impessoal, no caso dos verbos. Espaços em branco seriam proibidos dentro de headwords. De resto, o espaçamento entre quaisquer elementos é livre, incluindo a possibilidade de inserção de quebras de linha na codificação de uma mesma lista.

As lexias simples de libras que selecionam classificadores serão representadas pelas headwords seguidas de uma matriz atributo-valor. Trata-se de uma lista, entre chaves, de atribuições do tipo “Atributo:Valor”, separadas por espaço em branco, que farão a indicação explícita da função sintática ou do caso semântico subcategorizado. Por exemplo, a LW correspondente ao item lexical do português “colocar”, que, em libras, tem raiz icônica que concorda com o objeto a ser colocado e faz referência ao sinal do local onde se coloca o objeto, seria assim representada: ‘colocar{genobj:coisa-redonda loc:embaixo{ref:i}}’, indicando, neste caso, que a ação descrita pelo verbo e que será recuperada na geração do sinal corresponde ao ato de colocar um objeto de forma

[5] A opção pelas letras maiúsculas e minúsculas diverge da estabelecida em Felipe 2001 para evitar ambigüidade. Na proposta da autora, o item lexical (hipotético) "X-Y-Z" poderia corresponder, do ponto de vista computacional, (a) à lexia composta "X-Y-Z", que poderia ser representada por um único sinal (como "CORTAR-COM-FACA"; "MEIO-DIA"); e (b) à lexia simples "X-Y-Z", a ser representada por meio da datilologia (como "J-O-Ã-O").

arredondada embaixo de outro [i] referido anteriormente, abaixo do qual o novo sinal deverá ser gerado. A aplicação de matrizes atributo-valor será detalhada na subseção seguinte.

O quadro abaixo ilustra a correspondência entre português, LIST e o sistema de notação em palavras proposta em Felipe (1998):

português	**Felipe (1998)**	**LIST**
casa	CASA	casa
estudei	ESTUDAR	estudar
cortar	CORTAR-COM-FACA	cortar-com-faca
zebra	CAVALO^LISTRA	cavalo^listra
João	J-O-Ã-O	JOÃO
amiga	AMIG@	amigo
árvores	ÁRVORE+	árvore.@plural
colocar	$_{\text{coisa-arredondada}}$COLOCAR	colocar{genobj:coisa-arredondada}
perguntar (você para eles)	$_{2s}$PERGUNTAR$_{3p}$	perguntar{psuj:2s pobj:3p}

Tabela 1 - Correspondência entre os vocabulário do português, notação de libras e LIST

3.2. Estrutura frasal de LIST

Como o próprio nome sugere, LIST tenta, na medida do possível, aproximar-se de uma estrutura linear, uma simples lista de sinais a serem executados em seqüência, separados por espaços em branco. Entretanto, introduzimos alguns "operadores/estruturadores frasais" para melhor dar conta de alguns fenômenos de libras. Após a apresentação dessas estruturas de maneira mais informal, esta seção termina com uma especificação sintática formal da estrutura frasal de LIST (Seção 3.2.1).

- **grupos rítmicos:** em princípio, não pretendemos mais que um rudimento de representação rítmica, que, inclusive, é de uso e consideração opcional, por parte tanto dos codificadores LIST quanto dos decodificadores (síntese). Quando o codificador

achar que um grupo de sinais forma um grupo rítmico que mereça destaque, basta colocá-los entre colchetes (é possível aninhamento). Por exemplo, isso pode ser interessante para denotar topicalização:

[[... tópico ...] ... enunciado sobre o tópico ...]

Como se pode observar acima, toda sentença deve ficar entre colchetes. Tecnicamente, os colchetes não criam propriamente sublistas, mas estabelecem assertivas/restrições rítmicas ou sintagmáticas. Por exemplo, dados A = [a b c d], B = [a [b c d]] e C = [[a b c] d], podemos dizer que:

i) A é a forma menos assertiva/restritiva, representando todas as possíveis partições rítmicas da lista (a b c d);
ii) B/C é mais restritivo, representando todas as partições em que (b c d)/(a b c) acabam pertencendo a um mesmo grupo; e, portanto,
iii) $A \supset B$, $A \not\subset B$, $A \supset C$, $A \not\subset C$ e $B \cap C = \varnothing$.

A idéia de criar grupos prosódicos dentro de LIST parece coincidir com a observação de que o processo de articulação dos sinais em libras acompanha esse tipo de estruturação. Ou seja, o grupo que trabalhar no processo de síntese de fala deve entender que o processo de inserção de silêncios em libras respeitaria essa organização. Silêncios deveriam vir entre essas fronteiras (sintagmáticas). A única dúvida é se, como na prosódia das línguas naturais, esses silêncios seriam todos de mesma duração, ou se seria necessária uma especificação mais refinada.

- **comentários:** comentários quaisquer podem ser inseridos no meio da sentença entre aspas. Para todos os efeitos, é como se esses comentários não existissem. Ex.:

["1ª sentença de LIBRAS – Você é legal." você legal]

- **grupos de precedência ou comentário:** como introduziremos algumas operações binárias, o uso de parênteses é permitido (i) para ajuste de precedência (como nas expressões matemáticas), bem como (ii) *para "comentar" a anotação*, podendo ser usados pelo codificador para explicitar partes da estrutura sintagmática de uma sentença complicada e assim facilitar sua interpretação por outros humanos. Tais parênteses de comentário devem ser necessariamente ignorados pela ferramenta de síntese, diferentemente dos colchetes, cuja consideração é opcional. Ex.:

[(gata menina) ": a gata da menina" comer sapo]

Importante: tecnicamente, o escopo dos operadores é alterado *tanto por colchetes como por parênteses*, mas apenas estes *não são assertivos*. Portanto:

[a b c d] = [a (b c (d))] = [(a b c) d]

- **paralelismo:** dois sinais S_1 e S_2 quaisquer podem ser paralelizados por meio do operador ||, resultando num sinal obtido pela execução simultânea de ambos, o qual é denotado, portanto, S1||S2. Naturalmente, o codificador apenas paralelizará elementos cuja paralelização fizer sentido e for possível. Como apoio notacional, estendemos a definição para operar entre um sinal e toda uma lista de sinais, assim:

$$S_1\ S_2\ S_3 \ldots S_n \parallel S = (S_1 \parallel S)\ (S_2 \parallel S)\ (S_3 \parallel S) \ldots (S_n \parallel S)$$

  Ou seja, o resultado de *Lista* || *Sinal* é uma nova lista de sinais cujos elementos são obtidos pela paralelização de *Sinal* com cada elemento de *Lista*. Alguns exemplos de uso (e não-uso):

[eu copo quebrar || não-facial]
[eu (copo quebrar || não-facial)]
[eu copo (quebrar || não-facial)]
[eu copo quebrar não]

  Como se pode ver acima, o paralelismo pode ser estabelecido em vários níveis, incluindo a possibilidade (talvez inarticulável) de mais de dois sinais em paralelo. Exemplos de estruturas possíveis:

lista || sinal1 || sinal2
sinal2 sinal3 (sinal5 sinal6 || sinal4) sinal7 || sinal1

  No último exemplo, o fluxo principal de sinais é [sinal2 sinal3 sinal5 sinal6 sinal7], [sinal4] é executado paralelamente à subseqüência [sinal5 sinal6], e [sinal1] ocorre todo o tempo, inclusive em paralelo com [sinal4]. Se quiséssemos alternância entre esses sinais:

(sinal2 sinal3 || sinal 1) (sinal5 sinal6 || sinal4) (sinal7 || sinal 1)

  Tecnicamente, o operador || é binário, de associatividade à esquerda e precedência mínima (inclusive menor que a precedência do espaço em branco separador de elementos de lista!).

- **matriz atributo-valor:** um *conjunto*, entre chaves, de atribuições do tipo *Atributo:Valor* separadas por espaço em branco pode ser associado a qualquer headword ou grupo rítmico ao se lhe justapor à direita. Os atributos seguem as regras para headwords e valores praticamente podem ser expressões LIST quaisquer (embora devam ser geralmente mais simples. Para a definição formal geral de valor, consulte a especificação sintática no final da seção). Veja um exemplo na seguinte codificação:

[eu bola cama{id:1} colocar{genobj:coisa-redonda loc:embaixo{ref:1}}]
"Eu coloquei a bola embaixo da cama."

Exemplificamos acima e definimos já o único atributo "inato" de LIST e que se aplica a qualquer LW, se necessário: id. Por meio da atribuição {id:I}, a LW em questão é identificado como I (qualquer, *exclusivo* e não necessariamente numérico) de forma a poder ser referenciado por valores (no exemplo, "embaixo{ref:1}") de atributos ("loc") de outros sinais ("colocar"). É exata e tão-somente esse recurso que permite a LIST extrapolar sua linearidade temporal e pretender tratar a espacialidade de libras.

Como apoio notacional, estendemos a definição do operador de atribuição de matriz, {}, para abarcar listas inteiras de sinais, assim:

$$(S_1\ S_2\ S_3\ ...\ S_n)\ \{M\} = (S_1\{M\}\ \ S_2\{M\}\ \ S_3\{M\}\ \ ...\ S_n\{M\})$$

Caso um sinal S_i já seja da forma S{M'}, não há problema, porque:

$$S\{M_1\}\{M_2\} = S\{M_1 \cup M_2\}, \text{ se atributos}(M_1) \cap \text{atributos}(M_2) = \varnothing$$

ou seja, a operação é possível se as matrizes em questão não puderem se contradizer, especificando conjuntos disjuntos de atributos. Perceba, entretanto, a seguinte desigualdade drástica:

$$[(S_1\ S_2\ S_3\ ...\ S_n)\{M\}] \neq [S_1\ S_2\ S_3\ ...\ S_n]\ \{M\}$$

No lado direito da desigualdade, não há distribuição da operação de atribuição de matriz; antes, trata-se da atribuição da matriz M ao grupo rítmico $[S_1\ S_2\ S_3\ ...\ S_n]$ propriamente dito, como um todo, e não aos seus elementos S_i, individualmente.

Por fim, adicionamos um "açúcar sintático" para o tratamento de atributos booleanos, ditos *traços*:

{Atributo} = {Atributo:true}

Adicionalmente, diz-se que uma entidade apresenta o traço T se e somente se estiver especificada para {T} ou, equivalentemente, {T:true}. Agora podemos expressar algo do tipo (note que aí está a sentença inteira, sem necessidade de mais um par de colchetes em volta de tudo):

[viajar férias você]{interrogativa}

3.2.1. Especificação sintática formal da estrutura frasal LIST

Legenda: **S:** símbolo inicial
X∗: zero ou mais repetições de X
X+: uma ou mais repetições de X
X | Y: X ou Y
[X]: aplicação opcional de X
‘[’,‘]’ ,‘(’, etc.: literais que poderiam ser confundidos com metacaracteres

$S^{(obs.2)} \rightarrow$ List (‘||’ List) $*^{(obs.3)}$
List → AttribExpr+
AttribExpr → Nested (‘{’ M ‘}’)∗
Nested → Id | ‘(’ S ‘)’ | ‘[’ S ‘]’
M → (Id [‘:’ AttribExpr])+

Observações:

1. da primeira aplicação de S (nível da sentença) deve-se requerer que seja uma lista unitária, composta de um único grupo rítmico (provavelmente não unitário);
2. se, na primeira regra de produção, o caminho opcional é tomado (ou seja, há aplicação do operador de paralelismo), todas as Lists “opcionais” devem ser unitárias e não conter grupos rítmicos, já que a paralelização está definida entre uma lista e um sinal.

3.3. Estrutura dos documentos em LIST

A estrutura do documento LIST acompanharia a estrutura de um documento XML, em que seriam utilizadas as seguintes etiquetas (*tags*):

<d>	início do documento
</d>	fim do documento
	início de conjunto de enunciados inter-relacionados (parágrafo)
	fim de conjunto de enunciados inter-relacionados (parágrafo)
<s>	início do enunciado (sentença)
</s>	fim do enunciado (sentença)
{list}	início da representação em LIST
{/list}	fim da representação em LIST
{pt-br}	início da sentença em língua portuguesa
{/pt-br}	fim da sentença em língua portuguesa
{unl}	início da representação em UNL
{/unl}	fim da representação em UNL

Tabela 2 - Lista de Etiquetas dos Documentos em LIST

A etiqueta </d> deverá ser interpretada, pelo sintetizador, como uma interrupção prolongada no fluxo da gesticulação (correspondente ao encerramento da produção de um texto). A etiqueta </p> deverá ser interpretada como uma interrupção de duração intermediária (correspondente à mudança de tópico). A etiqueta </s> constitui uma interrupção mais breve (correspondente à mudança de enunciado). Para assegurar a legibilidade da representação LIST, é recomendável que seja também representada, no mesmo documento, a expressão-fonte (em língua portuguesa), que deverá ser, porém, ignorada pelo sintetizador.

Assim, um documento em LIST que procurasse representar a seqüência de sinais correspondente à expressão portuguesa "Declaração de Salamanca" teria a seguinte estrutura (abreviada):

```
<xml>
<head>
...
</head>
<body>
<p>
<s>
{pt-br}
Declaração de Salamanca
{/pt-br}
{unl}
mod(declaration(icl>document).@entry, Salamanca(icl>place))
{/unl}
{list}
[declaração SALAMANCA]
{/list}
</s>
</p>
</body>
</xml>
```

Figura 4 - Representação esquemática de um documento LIST

3.4. LIST hoje – resultados parciais

A partir da experiência em transcrição LIST adquirida até o momento, além de um conjunto pequeno de headwords já estabelecido (apresentado no dicionário UNL-LIST, à Seção 7.2), estão em vigência as seguintes determinações, relativas a atributos:

- **quanto à força ilocucionária:** os traços (atributos booleanos) “excl”, “imp”, “int” e “neg” são apresentados por sentenças/sinais com padrão entoacional exclamativo, imperativo, interrogativo e negativo, respectivamente. Uma sentença é afirmativa se e somente se não apresenta nenhum desses traços;

- **quanto à concordância:** de acordo com as relações de concordância verificadas para um verbo de libras, este deve especificar zero ou mais dos seguintes atributos: "psuj/pobj" (pessoa do sujeito/objeto), "gensuj/genobj/gentampaobj" (gênero do sujeito/objeto/tampa do objeto[6]). Os atributos de pessoa assumem valores combinando pessoa ("1", "2" ou "3") e número ("s(ingular)" ou "p(lural)"). Observaram-se no corpus os seguintes gêneros: "coisa-redonda" e "coisa-achatada". Ex.:

  [DENGUE vamos vencer{pobj:3s psuj:1p}]{excl imp}
  "Vamos vencer a dengue!"

  [por-isso caixa água precisar
  fechar-com-tampa{gentampaobj:coisa-achatada}]{excl}
  "Por este motivo, nós temos que manter a caixa d´água fechada!"

- **quanto à "autoconcordância":** observou-se um sinal que "concordava" com a própria idéia que exprimia, o que denominamos *autoconcordância*. Trata-se, evidentemente, de um sinal que a especialista em libras houve por bem considerar como parametrizado, por identificar um morfema intercambiável na sua constituição. Os sinais autoconcordantes em gênero devem especificar o atributo "gen". Ex.:

  [garrafa ferro{gen:coisa-redonda}]
  "... garrafas, latas ..."

- **quanto a vocativos:** embora vocativos não sejam freqüentes em libras, optamos por mantê-los em nossas versões LIST de sentenças em português. Para que, no entanto, possam ser facilmente identificados e devidamente executados ou até apagados pela equipe de síntese, o vocativo deve constituir um grupo rítmico com o traço "voc". Ex.:

  "[água largado [cascão]{voc}]{excl}"
  "Agua parada, Cascão!"

- **quanto a enumerações:** enumerações longas costumam apresentar um padrão rítmico próprio também em libras; portanto, definimos que a enumeração longa (de três elementos ou mais) deve constituir um grupo rítmico distinto, com o traço "enum" e composto exclusivamente de tantos (sub)grupos rítmicos quantos forem seus elementos. Ex.:

  [SI você [[febre alto] [músculo dor] [dor-de-cabeça] [joelho dor]
  [cotovelo dor] [MAL corpo todo]]{enum}]
  "Se você tiver febre alta, dores musculares, dores de cabeça, dores nos joelhos, dores nos cotovelos, e mal estar geral..."

[6] Essa concordância verbal surpreendente, em *gênero com a tampa do objeto* (!), foi verificada no corpus para o verbo "fechar-com-tampa".

4. Universal Networking Language (UNL)

Entre os objetivos assinalados para o desenvolvimento do protótipo, foram duas principalmente as questões que nos conduziram à adoção do modelo de tradução baseada em interlíngua:

a) a idéia de que a representação LIST, proposta como saída do sistema, deveria ser autoconsistente, autônoma e independente em relação ao processo de tradução e síntese de fala, de forma a permitir que as estratégias empregadas em ambos os processos possam ser substituídas sem que haja necessidade de alteração da representação e sem que sejam inutilizados os documentos já codificados; e

b) que seria desejável que a representação LIST fosse semelhante a outros formalismos computacionais de representação do conhecimento, para que se pudesse assegurar a intercambialidade entre os códigos e o desenvolvimento de filtros que permitissem, à LIST, servir de representação-alvo de outros sistemas de análise sintático-semântica (do português e também de outras línguas).

Estes dois compromissos, aliados à perspectiva de ampliar o sistema, transformando-o em uma plataforma efetivamente multilíngüe, permitiram perceber que seria interessante incorporar, como estratégia de tradução, uma representação intermediária, supostamente não-marcada e independente em relação à língua-fonte e à língua-alvo, para que se pudesse permitir uma maior intercambialidade de informações, e para assegurar portabilidade e robustez a todo o sistema. Entre os vários formalismos disponíveis (PATR-II, LFG, CG, FUG, TAG, etc.), optou-se pela *Universal Networking Language*, tendo em vista: a) a experiência anterior do grupo com este modelo de formalização; b) o seu caráter efetivamente plurilingüístico, dado que a iniciativa envolve grupos lingüísticos muito mais variados do que as outras abordagens disponíveis; c) o seu caráter público, na medida em que as patentes associadas pertenceriam à ONU; e d) sua abrangência, já que a flexibilidade da representação permitiria à UNL contemplar as informações de natureza visual freqüentemente expurgadas dos outros modelos, de base estritamente oral-auditiva.

O Projeto *Universal Networking Language*, ou UNL, teve início em 1996, e gravita em torno da *Universal Networking Digital Language Foundation*, sediada em Genebra, na Suíça. O projeto esteve originalmente vinculado ao Instituto de Estudos Avançados da Universidade das Nações Unidas, em Tóquio, e atualmente conta com 15 equipes de pesquisadores, que desenvolvem ferramentas para 13 diferentes línguas. O português do Brasil integra o Projeto desde 1997, através do Núcleo Interinstitucional de Lingüística Computacional (NILC), que vincula pesquisadores ligados à USP-S.Carlos, à UNESP-Araraquara e à UFSCar.

O Projeto tem por objetivo, principalmente, o desenvolvimento de um formalismo lógico-computacional capaz de representar, para a máquina, o conteúdo ideacional veiculado pelas sentenças das línguas naturais. Essa representação, que recebe o nome *Universal Networking Language* em virtude de seu caráter desejavelmente universal e reticulado, foi concebida para representar de forma única o conteúdo semântico de uma

sentença escrita em qualquer língua natural. Mais especificamente, a UNL é uma metalinguagem que serve para descrever aspectos especiais do significado de sentenças, tais como as relações semânticas que podem ser representadas por relações formais (morfológicas ou sintáticas) entre palavras de uma sentença. Apesar de simples, a UNL fornece uma visão muito prática de um sistema de processamento automático das línguas naturais: a de permitir o tratamento de aspectos semânticos de forma sistemática e independente do conhecimento profundo sobre questões não textuais. Ela se restringe, por exemplo, ao significado literal (e, logo, denotativo) de sentenças e trata de descrições de significados frasais seguindo a abordagem tradicional de relacionamento de papéis semânticos a entidades ou objetos componentes da frase. Ela não abrange alguns pressupostos teóricos importantes para a comunicação, tais como as questões comunicativas, estilísticas, intencionais ou retóricas, que levam ao significado conotativo, de natureza pragmática. Desse modo, suas limitações indicam também a exclusão do tratamento da recepção da linguagem, tida no Projeto UNL como um problema exclusivo do leitor.

Apesar dessas limitações, a UNL permite reconhecer algumas expressões idiomáticas, que são lexicalizadas com base em interpretações literais. Este é o elo mais explícito, na UNL, da tentativa de se processar informações de caráter conotativo. Entretanto, a UNL visa a promover somente a comunicação básica entre indivíduos de línguas nativas distintas e não a comunicação natural e abrangente que existiria entre indivíduos que partilhassem a mesma língua. A característica principal da UNL é a de privilegiar a predicação entre elementos lingüísticos, por sua caracterização individual ou relacional (com outros elementos lingüísticos) durante a ocorrência de eventos ou outras relações complexas entre eventos. Neste caso, os próprios argumentos verbais ou adjuntos frasais se manifestam como predicados, sendo responsáveis por preservar a relevância da informação textual. As predicações podem, portanto, ser relativamente simples, como um simples adjetivo, ou complexas, como em uma subordinação.

Privilegiando os aspectos estruturais do significado, ao invés dos aspectos da semântica lexical, a UNL se baseia em conceitos relacionados à idéia de caso semântico. A semântica lexical é incorporada ao léxico, como um ponto de partida para o processamento subseqüente. Desse modo, parte da representação semântica já é recuperada pelas próprias entradas lexicais. Em especial, incorporam-se ao léxico as interpretações particulares da língua em cada um de seus verbetes e são esses que passam a corresponder ao vocabulário UNL. A UNL serve, portanto, para projetar entidades de sistemas de representação da estrutura gramatical em sistemas de representação de interpretações semânticas e vice-versa. Por exemplo, no processo de codificação, ela é usada para mapear uma sentença escrita em linguagem natural (sentença de origem) em um conjunto de relações entre significados (sentença UNL), sendo que cada uma das relações consiste em uma proposição no sistema de representação universal. No processo de decodificação, regras gramaticais de uma língua destino qualquer são aplicadas à sentença UNL, para gerar a estrutura gramatical correspondente, a partir da qual se obtém a forma superficial na língua destino.

São 3 os principais constituintes da UNL: i) Palavras Universais (*Universal Words* - UWs), i.e., palavras que retêm o significado dos componentes frasais; ii) Rótulos de

relações (*Relation Labels* - RLs), que expressam as relações entre as UWs; iii) Rótulos de atributos (*Attribute Labels* - ALs), que expressam informações adicionais e, em geral, restritivas, sobre as UWs.

A função de uma UW é denotar um significado específico. Sua representação genérica é um rótulo simples (que indica o significado genérico de uma palavra em inglês) ou um rótulo limitado por um intervalo específico, que denota significados distintos (quando há, p.ex., ambigüidade em relação à palavra original). Por exemplo, os significados ativados pela palavra do inglês "book" permitem a representação das seguintes UWS: *book*, *book(icl>publication)*, *book(icl>account)* e *book(obj>room)*. A primeira UW, *book*, é a representação mais genérica do significado, válido para todas as acepções da palavra inglesa, ainda que não estejam, muitas vezes, inter-relacionadas. As demais limitam este significado a conceitos particulares. No caso, a *publicações*, *livro de contabilidade* ou *reserva de um quarto* (em um hotel), respectivamente.

RLs servem para expressar relações binárias entre significados, i.e., entre duas UWs distintas. Sua representação geral é dada por um par ordenado do tipo *relation_label(*UW_1*,* UW_2*)*, onde UW_1 e UW_2 são duas UWs diferentes relacionadas pela relação semântica indicada por *relation_label*. Há diversas classes de RLs (agente, objeto, instrumento, beneficiário, local, duração, direção, etc.), perfazendo um total de 38 RLs. A relação **agt** (de agente) expressa, por exemplo, a relação entre o evento e seu iniciador, ainda que não-intencional. Neste sentido, a sentença do português '*A lebre corre' seria* representada em UNL por '*agt(run,hare)'*. A relação **obj** (de objeto) representa, em contrapartida, a relação entre o evento e aquele que sofre diretamente suas conseqüências (paciente). A relação entre 'come' e 'maçãs' em *'Pedro come maçãs'* seria representada por '*obj(eat,apple)'*. Algumas relações, como **icl** (de hiponímia), **pof** (partonímia), **equ** (de sinonímia) e **iof** (de instanciação), por exemplo, não definiriam casos sentenciais, mas seriam utilizadas para inter-relacionar UWs no dicionário, restringindo sua acepção e compondo uma espécie de ontologia de UWs, o *UW System*, que pode ser mobilizado no processo de análise e síntese para efeito de inferenciação.

ALs servem para limitar o significado de uma UW genérica, i.e., para particularizar seu significado. Informações adicionais tais como tempo verbal, aspecto, intenção ou estrutura sentencial são exemplos de atributos específicos de uma UW. A representação genérica de um AL é dada pela UW, seguida por tantos atributos quantos forem os sugeridos na sentença de origem. Cada um deles é identificado pelo símbolo inicial "@". Por exemplo, a representação genérica de uma UW com *n* atributos tem a forma:

UW.@atrib$_1$.@atrib$_2$.....@atrib$_n$

Se não houver ALs vinculados a uma UW, esta representa o significado mais genérico de sua classe. Da mesma forma que para RLs, pode haver diferentes classes de ALs. O atributo .@def, por exemplo, indica uma UW definida; .@past indica um evento no passado; .@topic indica topicalização; etc. Alguns dos ALs descritos acima se aplicam a elementos intra-sentenciais, outros à sentença como um todo. No primeiro caso, o AL é associado ao componente específico que ele modifica. No segundo, o AL é associado à UW nuclear, i.e., àquela que expressa o predicado principal da sentença. Formas mais

complexas de combinações de UWs podem ainda ser expressas por meio de ALs. Além disso, línguas naturais distintas podem ter representações mais detalhadas para expressar informações restritivas sobre conceitos semânticos. Neste caso, a UNL permite a definição de subcategorizações pela inclusão de novos atributos e, assim, é possível tratar as particularidades de cada língua, inclusive das línguas de sinais.

Uma sentença UNL consiste em uma estrutura reticulada de dados (um hipergrafo), cujos nós corresponderiam às UWs, e os arcos [entre nós] às relações binárias (orientadas) de dependência semântica (os RLs) entre esses itens lexicais. Nós e arcos poderiam ser modificados por atributos (os ALs) de natureza variada (intensificadores e outros operadores lógicos: conjunção, disjunção, negação, interrogação, etc.), por meio do acréscimo de etiquetas específicas com indicação clara de seu escopo de aplicação. Uma sentença UNL é ilustrada na Figura 5 abaixo.

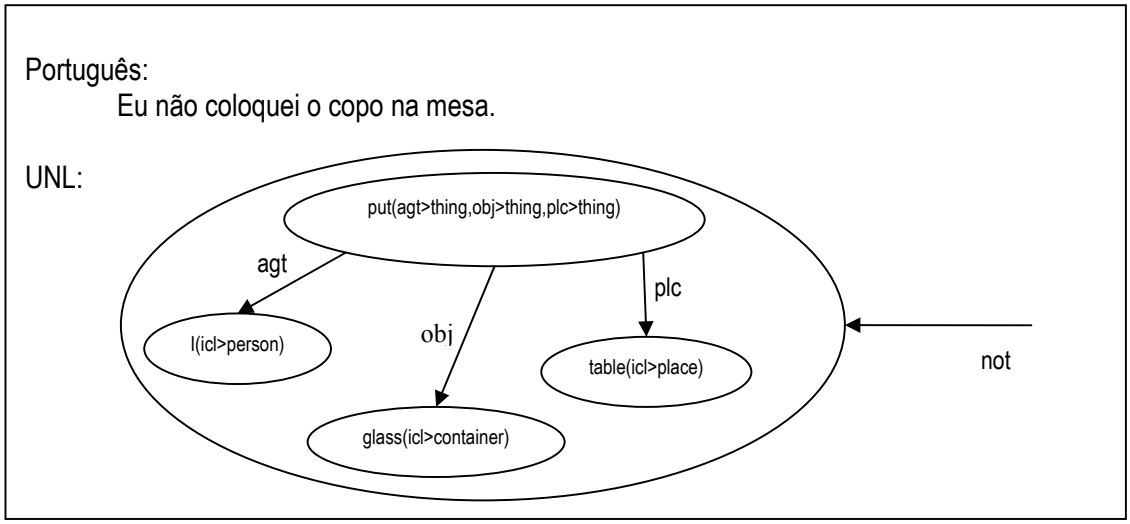


Figura 5 - Representação UNL de uma sentença do português

Este mesmo grafo poderia ser representado como uma lista (não-ordenada) de relações binárias entre itens lexicais de libras, da seguinte forma:

```
{unl}
01.@not
suj:01(put(agt>thing,obj>thing,plc>thing, I(icl>person))
obj:01(put(agt>thing,obj>thing,plc>thing, glass(icl>container))
plc:01(put(agt>thing,obj>thing,plc>thing, table(icl>place))
{/unl}
```

Figura 6 - Representação linear de um grafo UNL

A ordem de apresentação das relações não seria relevante, na medida em que se pretende apenas a representação da estrutura semântica, e não da estrutura sintática dos

enunciados. Para que se possa referir nós co-referentes, todas as pró-formas seriam indexadas e compartilhariam os mesmos índices. O escopo seria representado como atributo das relações e funcionaria como um hipernó, podendo integrar, como argumento, outras relações. Os atributos seriam anexados, por meio de ".@" diretamente aos nós ou hipernós a que se referissem.

Além da construção de uma especificação semântica comum, o Projeto envolve, em cada grupo, a produção de ferramentas de codificação para a UNL e decodificação a partir da UNL. A partir de um algoritmo comum, implementado pelo UNL-Center, todos os grupos devem desenvolver: a) um dicionário língua natural-UNL, b) uma gramática bidirecional língua natural-UNL, e c) uma matriz de probabilidades de aplicação das regras gramaticais. Como todos os grupos operam sobre o mesmo algoritmo, a sintaxe das regras e a forma do dicionário são idênticas entre os diferentes grupos, pouco importando a filiação lingüística. O desenvolvimento das ferramentas de codificação e decodificação, e os problemas dele derivados, orientam o processo de revisão da especificação UNL, e conformam a razão mesma das alterações.

Para maiores informações sobre a UNL, a webpage da UNDL Foundation, em http://www.undl.org.

5. Corpus

Para o desenvolvimento do protótipo, decidiu-se pelo trabalho em um *corpus* constituído, inicialmente, por histórias infantis. Duas histórias chegaram a ser codificadas, mas se percebeu, muito cedo, que as histórias infantis escritas em língua portuguesa, quando recontadas em libras, envolveriam adaptações e alterações que muito dificilmente poderiam ser replicadas em um ambiente computacional com a tecnologia disponível. A presença freqüente de um vocabulário diferenciado (e infantilizado), o revezamento entre discurso direto e discurso indireto, e o apoio indispensável nas ilustrações, com as quais a representação em sinais deveria competir, indicaram que, nessa primeira etapa, seria mais prudente trabalharmos com um corpus de outra natureza, em que algumas dessas variáveis pudessem ser mais bem controladas. Por este motivo, preferiu-se trabalhar com histórias em quadrinho, porque se atingiria o mesmo público-alvo (as crianças surdas); porque se restringiria o gênero de trabalho apenas ao diálogo, ou discurso direto; porque se poderia criar uma interface mais atraente, em que os sinais correspondentes a cada balão seriam gerados no momento em que este fosse clicado pelo leitor, garantindo um procedimento de leitura simples, intuitivo e que respeitasse o ritmo de cada usuário. Entre as histórias em quadrinho, optou-se pelas tirinhas semanais da Turma da Mônica, editadas pela Editora Globo, e disponíveis no Portal da Mônica, em http://www.portaldamonica.com.br.

Selecionou-se, inicialmente, uma só história – a de número 74 – que consistia em um panfleto instrutivo a respeito dos cuidados para se evitar a disseminação da dengue. A história é apresentada no Anexo I deste relatório. As 12 sentenças da história foram originalmente codificadas pela especialista Tanya Felipe, do INES, sendo em seguida revistas pelos integrantes da equipe, que geraram as representações LIST e UNL desejáveis (ou possíveis) para cada um dos enunciados apresentados. A partir dessas representações, foram produzidos os recursos lingüísticos (dicionários e gramáticas) necessários para que pudesse ser emulado computacionalmente o comportamento humano.

Abaixo são apresentadas as principais questões atinentes ao *corpus* de trabalho. A codificação completa do *corpus* também é apresentada no Anexo I deste relatório.

O primeiro problema observado consistiu no fato de, na história em questão, as fronteiras entre as sentenças não coincidirem exatamente com os balões que representam as falas das personagens. Muitas sentenças começavam em um quadrinho e terminavam em outro, quando não se espalhavam entre vários quadrinhos distintos. Essa continuidade era indicada, via de regra, pelo uso de reticências no início e no fim das sentenças, o que prejudicava, evidentemente, a idéia de unidade semântica da frase, necessária tanto para a representação UNL quanto para a geração do enunciado libras. A única solução viável encontrada foi conservar as fronteiras indicadas pelo próprio quadrinhista, porque qualquer alternativa implicaria em reorganizar a diagramação do texto original. A fragmentariedade resultante da nossa solução deve ser equacionada, portanto, pelo próprio leitor, ficando fora do alcance da ferramenta de tradução.

Outro problema evidente – já aguardado – foram as elipses e as substituições por zero observadas nos enunciados analisados. Como muitas dessas elipses não podem existir em libras, cumpria preencher toda e qualquer lacuna, para que não se gerassem enunciados LIST que pudessem produzir seqüências de libra agramaticais. Algumas dessas elipses

revelaram-se mais simples (caso de “Vamos combater a dengue!”, com a elipse do sujeito ‘nós’ da forma do plural de convite expresso por ‘vamos’, gramaticalizado finalmente como imperativo, com a ajuda da presença do ponto-de-exclamação); em outros casos, no entanto, a recuperação dessas referências envolvia conhecimento de mundo, que não poderia ser provido automaticamente. É o caso, por exemplo, de “Se você tiver febre alta, dores musculares, de cabeça, nos joelhos, cotovelos, e mal estar geral...”, em que se faz, inicialmente, a elipse de ‘dores’ (“dores musculares, ___ de cabeça, __ nos joelhos”) e, mais tarde, a elipse também da preposição (“___ ___ cotovelos”, em lugar de “dores nos cotovelos”). Embora o leitor humano consiga resgatar facilmente essas relações, particularmente dada a contigüidade dos antecedentes, é forçoso admitir que tal se observa apenas porque já existe o conhecimento prévio de que a) “cotovelo” não é uma doença, mas uma parte do corpo (assim como “cabeça” e “joelho”), e b) pode haver “dor nos cotovelos”. A complexidade desse conhecimento fica particularmente mais nítida se substituímos “cotovelo” por uma doença qualquer, como “vômito”, na seqüência referida: “Se você tiver febre alta, dores musculares, de cabeça, nos joelhos, vômitos, e mal estar geral....” oferece uma possibilidade de análise sintática diferente da sentença original, embora “vômitos” e “cotovelos” compartilhem os mesmos traços gramaticais. Para evitar que problemas como esses degradassem exageradamente o desempenho da ferramenta, optou-se por delegar ao usuário, no módulo da normalização, o equacionamento do preenchimento das elipses e de outros fenômenos de natureza semelhante.

6. O pulo português-UNL

No processo de análise do português, para sua tradução em UNL, foram desenvolvidos vários recursos, lingüísticos e computacionais, descritos nesta seção. Muitos desses recursos, embora tenham sido elaborados especificamente para o projeto em tela e para o *corpus* especificado na Seção 5, são customizáveis para outros projetos e para outros *corpora*.

6.1 Normalização do português – Kaapor

O *corpus* selecionado para esse projeto, como visto na Seção 5, é composto por histórias em quadrinhos da Turma da Mônica, de Mauricio de Sousa. Alguns personagens, como Cebolinha e Chico Bento, apresentam desvios na linguagem em relação ao registro padrão neutro. Exemplos desses desvios podem ser vistos no conjunto de sentenças abaixo:

“Mentila! Você quer que eu me levante só pla aceltar dileto na minha cala!” (Cebolinha)

“Tô atlasado plo encontlo com a Litinha!” (Cebolinha)

“Eu vô dexá o mio bem aqui, i quem quisé vai si servindo! Tá bão ansim?” (Chico Bento)

“Ai, ai... Nada mior qui uma rede depois do armoço!” (Chico Bento)

Além dessas, existem outras ocorrências que precisam ser tratadas, por exemplo, as onomatopéias presentes nas histórias em quadrinhos (“Blam”, “Kabum”, “Psiiii”, “Uóóóm”,”Uau!!!” etc.). Essas ocorrências devem ser previamente tratadas para que possa ser realizada a tradução LIST. Entretanto, neste primeiro momento, serão tratados os desvios de linguagem do personagem Cebolinha.

O módulo Kaapor é o responsável pela normalização, ou seja, por realizar as correções necessárias no texto para que o mesmo possa ser traduzido. Tal módulo tenta, na medida do possível, realizar as correções automaticamente, sem a intervenção do usuário; mas existem alguns casos em que o auxílio de um usuário falante da língua portuguesa é indispensável. Isso acontece, por exemplo, no tratamento de uma sentença que contenha a palavra “calo” dita pelo Cebolinha, que poderá representar tanto “carro” quanto “caro” (mas não “calo”, já que a palavra estará em negrito no balão, indicando desvio de pronúncia).

As histórias são originalmente apresentadas como uma figura (extensão *.gif*) e devem ter suas falas transpostas para um arquivo-texto (extensão *.txt*), na medida de um balão por linha. Neste arquivo, as falas do Cebolinha devem ser formatadas de maneira especial: na figura original, as palavras que têm a letra R trocada pela letra L são grafadas em negrito; correspondentemente, elas devem vir no arquivo-texto delimitadas pelas

etiquetas <n> e </n> no início e fim da palavra respectivamente. O módulo oferece ao usuário a opção de visualização da história em quadrinhos original (Figuras 7 e 8).

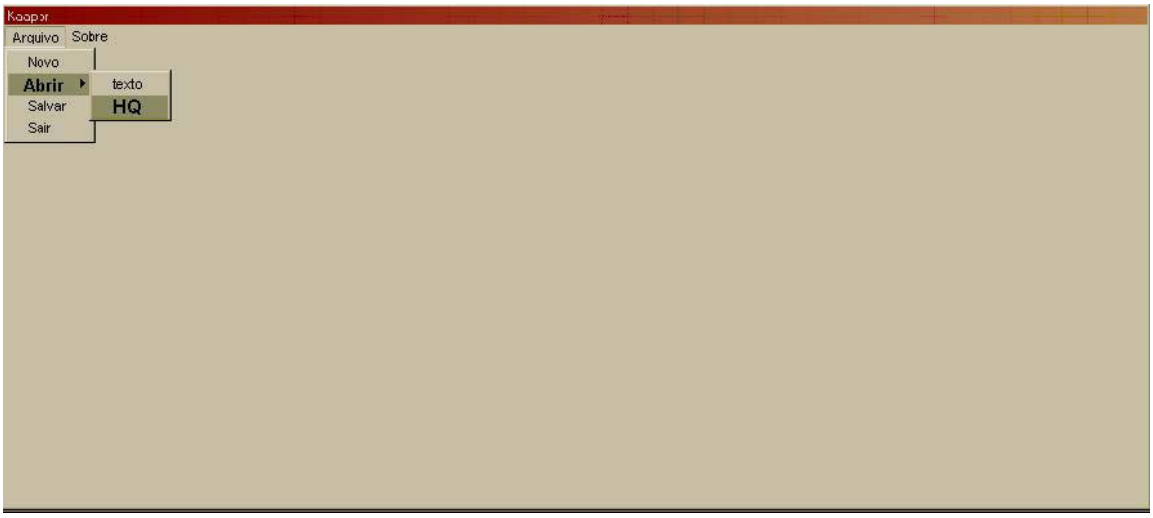


Figura 7 – Tela de interação inicial do módulo Kaapor

Figura 8 – História em quadrinhos original

O processamento do texto se inicia a partir do arquivo-texto: a ferramenta de normalização começa pela análise/conversão das sentenças. O primeiro passo realizado pelo módulo é a identificação de palavras com a marcação <n> e </n> e a correção das mesmas. Para gerar alternativas de correção, o módulo realiza a substituição das letras L por R ou RR (ou mesmo L, no caso de haver a ocorrência de mais de uma letra L na palavra, como em “Clalo!”), em todas as combinações possíveis, retendo apenas as cadeias

resultantes que pertencerem a um banco de palavras da língua portuguesa. No caso de restar uma única alternativa, esta substituirá automaticamente a palavra original na sentença normalizada; no caso de mais de uma entrada identificada, como no exemplo da palavra "calo", o usuário será consultado e a entrada selecionada por ele irá substituir a palavra original na sentença normalizada (Figura 9).

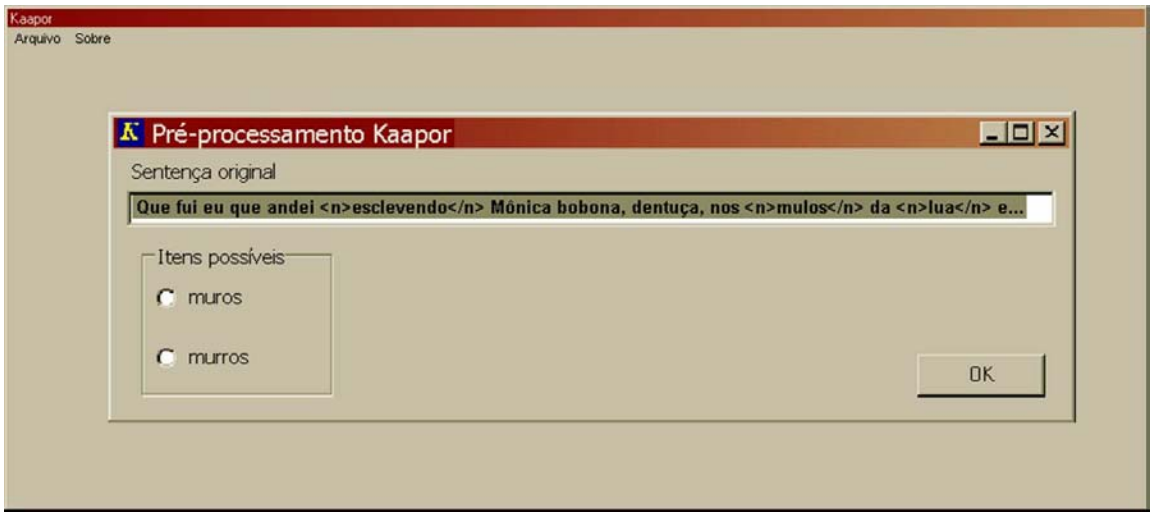


Figura 9 – Interação do módulo com o usuário

Depois dessa primeira fase, as sentenças são apresentadas ao usuário para que ele faça as alterações que achar pertinentes (Figura 10). Caso o usuário decida não realizar nenhuma alteração, ele pode optar pelo encerramento da aplicação (opção "Sair" no menu "Arquivo"), ou seja, nenhuma tarefa seja realizada, ou pode optar pela conclusão do processamento, ou seja, a geração em LIST (opção "Concluir..." no menu "Arquivo").

O resultado da primeira fase, ou seja, a correção/edição das sentenças, é armazenado em um arquivo com extensão .kpr, que será utilizado pelo próximo módulo executado, o Hermeto. Vale ressaltar que a ferramenta mantém o nome do arquivo original no demais arquivos que manipula, alterando apenas a extensão dos arquivos criados.

Antes que as sentenças sejam enviadas ao Hermeto, elas são analisadas a fim de identificar possíveis referências anafóricas que devam ser corrigidas pelo usuário. As sentenças analisadas que apresentam possíveis anáforas são apresentadas ao usuário para que opte por editá-las ou não.

A identificação de possíveis anáforas é realizada, neste momento, com o auxílio de um etiquetador morfossintático, que identifica a classe gramatical que cada palavra assume na sentença analisada, e de um conjunto finito de palavras que, quando classificadas como pronome, podem indicar uma referência anafórica na sentença, por exemplo, "aquela", "naquele" e "esse" (o conjunto completo de palavras com tal característica pode ser encontrado no Anexo VIII). A tarefa de identificação é realizada da seguinte maneira: uma sentença é submetida ao etiquetador morfossintático e o resultado é analisado. Caso a sentença contenha alguma das palavras pertencentes ao conjunto previamente definido e essa palavra seja classificada como pronome, essa sentença é apresentada ao usuário para

que ele faça alterações, caso seja necessário, ou seja, caso aquela palavra esteja indicando realmente uma anáfora. Vale lembrar que sempre que existirem anáforas nas sentenças, essas devem ser corrigidas para que o processamento possa continuar com sucesso. Todas as alterações são armazenadas no arquivo de resultados mencionado anteriormente.

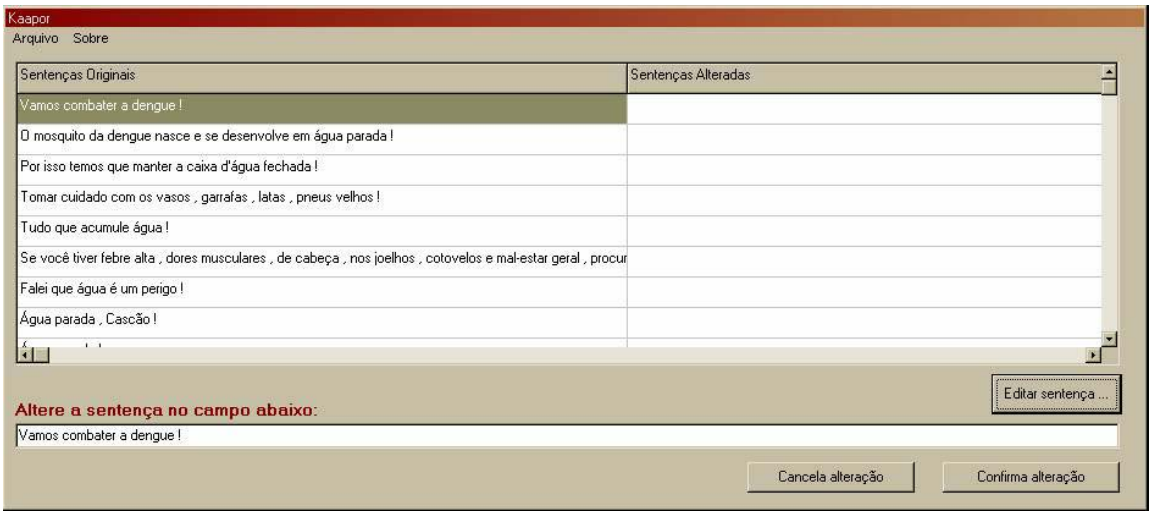


Figura 10 – Sentenças pré-analisadas

A função principal do módulo Kaapor é gerar o arquivo .kpr. Entretanto, por coveniência, esse módulo já integra e executa os módulos Hermeto e ManateCo. Essa integração será detalhada na Seção 8.

6.2 Gramática português-UNL

A gramática português-UNL, construída no âmbito deste projeto, pode ser definida pela sêxtupla <N, T, P, I, W, S> , em que:

N = conjunto de símbolos não-terminais
T = conjunto de símbolos terminais
P = conjunto de regras de produção sintática
I = conjunto de regras de interpretação semântica
W = prioridade de aplicação das regras de produção sintática (inteiro de 0 a 255)
S = símbolo inicial

Para efeito de simplificação, o conjunto de regras de produção sintática (P) e o de interpretação semântica (I) foram representados conjuntamente, como campos distintos de uma mesma regra. Assim, as regras da gramática possuem o seguinte formato:

p -> i
em que $p \in P$, e $i \in I$

As regras (p) de produção sintática acompanham uma gramática livre-de-contexto, com indicação explícita de prioridade de aplicação de cada regra, definida pela fórmula:

a[w] := b

em que $a \in N$, $b \in N \cup T$, e $w \in W$

Na fórmula acima, quanto maior o valor de 'w', tanto menor a prioridade de aplicação da regra (0 = prioridade máxima de aplicação). Se duas ou mais regras compartilharem a mesma prioridade, sua ordem de aplicação corresponderá à ordem das regras no arquivo da gramática.

Para simplificação do número de regras e para restrições, foram utilizados os seguintes operadores:

{x,y}	indica mútua exclusão (ou x ou y)
'x'	indica entrada do dicionário
[x]	indica opcionalidade (ou x ou nada)
a(x:y)	indica que y é o valor do atributo x de a
a(x=y)	indica que **a** é ativada se e apenas se a classe **x** possuir o valor **y** (em que y deve corresponder, necessariamente, a um símbolo terminal, isto é, proveniente do dicionário)
<x>	indica todas as flexões que tomam por canônica a forma x
+	indica espaço em branco entre os símbolos (x + y = x y)
-	indica contigüidade espacial entre os símbolos (x - y = xy)
#	indica fronteira entre palavras
##	indica fronteira entre sintagmas
###	indica fronteira entre sentenças
.@entry	indica o principal nó (núcleo) de uma regra (sua atribuição é obrigatória para que a regra possa ser referenciada por outra regra)
i	indica valor co-indexado de um atributo específico (utilizado para restringir a aplicação de uma regra, exigindo a concordância entre os valores dos atributos dos termos conjugados)

São exemplos de regras de produção sintática:

S[1]:= adv - ',' + S.@entry
S[2]:= 'se' + S.@entry
S[3]:= NOP(per:i,nbr:i) + ver(per:i,nbr:i,rol:cop) + NOP(nbr:i).@entry
S[4]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:be).@entry
S[5]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:do).@entry
S[6]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:occur).@entry
S[7]:= NOP(per:i,nbr:i,gen:i) + ver(per:i,nbr:i,rol:cop) + adj(nbr:i,gen:i).@entry
S[8]:= VP.@entry
S[9]:= NOP.@entry - ',' + ppn

S[10]:= NOP.@entry

As regras (i) de interpretação semântica são definidas pela fórmula:

atr_1, atr_2, ..., atr_n, rel_1, rel_2, ..., rel_n

- em que **atr** corresponde a regras de atribuição
- e **rel** corresponde a regras de relacionamento

As regras de atribuição (**atr**) consistem na adjudicação de um ou mais atributos UNL a um token que ocupa determinada posição no lado direito da regra de produção sintática. Essa adjudicação é feita pela fórmula:

posição.@atributo

- em que **posição** (representada por ":%%", em que % = [0..9]) indica a posição do token no lado direito da regra de produção sintática
- e **atributo** é o atributo previsto pela especificação UNL a ser adjudicado ao token que ocupa esta posição

São exemplos de regras de atribuição:

:01.@entry
:04.@entry.@past,
:01.@entry, :04.@past.@progress

As regras de relacionamento (**rel**) consistem no estabelecimento de uma relação UNL a dois tokens que ocupam posições determinadas no lado direito da regra de produção sintática. Esse relacionamento é feito pela fórmula:

relação(posição, posição)
relação(posição, 'constante')
relação('constante', posição)

- em que **relação** é uma das relações previstas pela especificação UNL;
- **posição** (representada por ":%%", em que % = [0..9]) indica a posição do token no lado direito da regra de produção sintática; e
- **constante** corresponde a uma entrada do dicionário UNL (UW).

São exemplos de regras de relacionamento:

agt(:01, :02)
agt(:01, :02), obj(:01, :03)

São exemplos de regras de interpretação:

:01.@entry
agt(:01, :02)
:01.@entry, agt(:01, :02)
:01.@entry, :04.@def, agt(:01, :02), obj(:01, :03)

Exemplos completos de regras da gramática correspondem a:

S[1]:= adv - ',' + S.@entry -> man(:03,:01)
S[2]:= 'se' + S.@entry -> man(:02,'if(icl>how)')
S[3]:= NOP(per:i,nbr:i) + ver(per:i,nbr:i,rol:cop) + NOP(nbr:i).@entry -> aoj(:03,:01)
S[4]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:be).@entry -> aoj(:02,:01)
S[5]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:do).@entry -> agt(:02,:01)
S[6]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:occur).@entry -> obj(:02,:01)
S[7]:= NOP(per:i,nbr:i,gen:i) + ver(per:i,nbr:i,rol:cop) + adj(nbr:i,gen:i).@entry -> aoj(:03,:01)
S[8]:= VP.@entry
S[9]:= NOP.@entry - ',' + ppn -> :03.@vocative, and(:03,:01)
S[10]:= NOP.@entry

A atual versão da gramática – apresentada no Anexo II deste relatório – foi construída especialmente para o conjunto de 12 sentenças constantes no *corpus* e soma 88 linhas de código.

6.3 Dicionário português-UNL

O dicionário português-UNL utilizado pelo projeto correspondeu a um arquivo texto em que as entradas foram representadas como indicado abaixo.

[ENTRY] {ID} LEMMA "CONCEPT" (FEATURES) <LANGUAGE, FREQUENCE, PRIORITY>;

> **ENTRY** = pode ser caracterizada por formas livres (lexias simples, lexias compostas, lexias complexas) ou por formas presas (como raízes, afixos e partes de *collocations*). Não deve ser sensível ao caso.
> **ID** = identificador (número inteiro utilizado para efeito de indexação do dicionário - um número diferente por entrada)
> **LEMMA** = forma canônica (lematizada), a partir da qual possa ser traçada uma rede de flexões e/ou derivações associadas na língua representada;
> **CONCEPT** = representação semântica da entrada na rede UNL de conceitos (UW);
> **FEATURES** = conjunto (não-ordenado) de pares de atributo-valor, de natureza fonético-fonológica, morfossintática, semântica, pragmático-discursiva e outras, na medida das necessidades da aplicação. Sugere-se que os traços sejam sempre compostos por seqüências de três caracteres, modificadas, se necessário.
> **LANGUAGE** = identificação da língua do dicionário (utilizou-se 'pt-br' para português brasileiro);
> **FREQUENCE** = número inteiro entre 0 e 255, que indica a freqüência de ocorrência da entrada (0 para mais freqüente, 255 para menos freqüente) para efeito de análise. Default = 0.
> **PRIORITY** = número inteiro entre 0 e 255, que indica a prioridade de geração da entrada (0 para mais prioritária, 255 para menos prioritária) para efeito de geração. Default = 0.

Exemplos de entradas são apresentados a seguir:

```
[a] {} o "the" (pos:art;typ:def;gen:fem;nbr:sgn) <PT-BR,0,0>;
[acumule] {} acumule "gather(icl>do)"
(pos:ver;typ:do;tst:dts;per:3per;nbr:sgn;ten:pres;moo:sbj) <PT-BR,0,0>;
[alta] {} alta "high(mod<thing)" (pos:adj;gen:fem;nbr:sgn) <PT-BR,0,0>;
```

```
[água] {} água "water(icl>matter)" (pos:nou;gen:fem;nbr:sgn) <PT-BR,0,0>;
[cabeça] {} cabeça "head(pof>body)" (pos:nou;gen:fem;nbr:sgn) <PT-BR,0,0>;
[caixa d'água] {} caixa_d'água "water tank(icl>container)"
(pos:nou;gen:fem;nbr:sgn) <PT-BR,0,0>;
[Cascão] {} Cascão "Cascão(icl>person)" (pos:ppn;gen:mcl;nbr:sgn) <PT-
BR,0,0>;
```

A atual versão do dicionário – apresentada no Anexo III deste relatório – é composta de 58 entradas.

6.4 Hermeto

O Ambiente Hermeto é uma ferramenta computacional para ser usada por usuários que saibam apenas criar as regras da gramática e o dicionário e manipular o ambiente. Para tanto, ela é composta por uma interface baseada em janelas e um compilador que utiliza a gramática português-UNL descrita acima para gerar uma biblioteca de ligação dinâmica (DLL).

Esta ferramenta foi desenvolvida para facilitar o trabalho e a depuração no desenvolvimento de qualquer *parser* para qualquer língua origem. Portanto, ela permite tanto a criação e edição de uma gramática como a manipulação de um *corpus* e o uso deste *corpus* na depuração da gramática e dicionário. O uso da DLL que incorpora o tokenizer, o parser, o interpretador e a gramática já compilados mais o dicionário no formato descrito acima resulta em uma árvore sintática e no código UNL da frase de entrada normalizada (Para melhor compreensão, veja o módulo Hermeto na arquitetura do PUL∅ na figura 3). Assim, a DLL admite, como dado de entrada, apenas sentenças isoladas. Conseqüentemente, o Hermeto, embora não faça qualquer restrição quanto à dimensão, quanto à forma, ou quanto ao domínio da sentença de entrada, não é capaz de resgatar relações de natureza extra-sentencial, seja por referência ao co-texto imediato, seja por referência ao contexto situacional, incluídas as substituições, as elipses, as pronominalizações e todas as relações anafóricas.

As janelas do Hermeto foram desenvolvidas para permitir de forma clara a edição, configuração e obtenção de toda a informação da análise realizada pelas regras do *parser*. Nas figuras de 11 a 15, são apresentadas as principais janelas e caixa de diálogo utilizadas para criar, editar e depurar o *corpus* e regras da gramática, e configurar o ambiente de trabalho.

A janela principal do Hermeto é divida em três áreas principais: 1) lista dos parágrafos – mostra uma listagem dos parágrafos no *corpus* atual. Nesta área com um simples clique do *mouse*, é possível criar, alterar e apagar um parágrafo; 2) lista das sentenças – mostra a listagem das sentenças contidas no parágrafo selecionado na lista dos parágrafos. Nesta área, de maneira semelhante à lista dos parágrafos, é possível criar, alterar e apagar uma sentença; e 3) área de edição de sentenças e visualização dos resultados do código UNL e da árvore sintática da sentença selecionada. Abaixo da caixa de edição há dois botões usados para executar a DLL (Gerar Código) e mostrar o resultado da análise feita para a sentença contida na caixa de edição (Mostrar Análise).

Na barra de menus, é possível acessar todas as funcionalidades disponíveis no ambiente, tais como, carregar, importar, editar e salvar um *corpus*, configurar o ambiente de trabalho, entre outros. Já a barra de ferramentas contém os botões e uma caixa de listagem que permitem acesso as principais funções da interface. Trata-se de um tipo de atalho aos recursos da ferramenta por meio de linguagem gráfica e não textual como se faz na barra de menus.

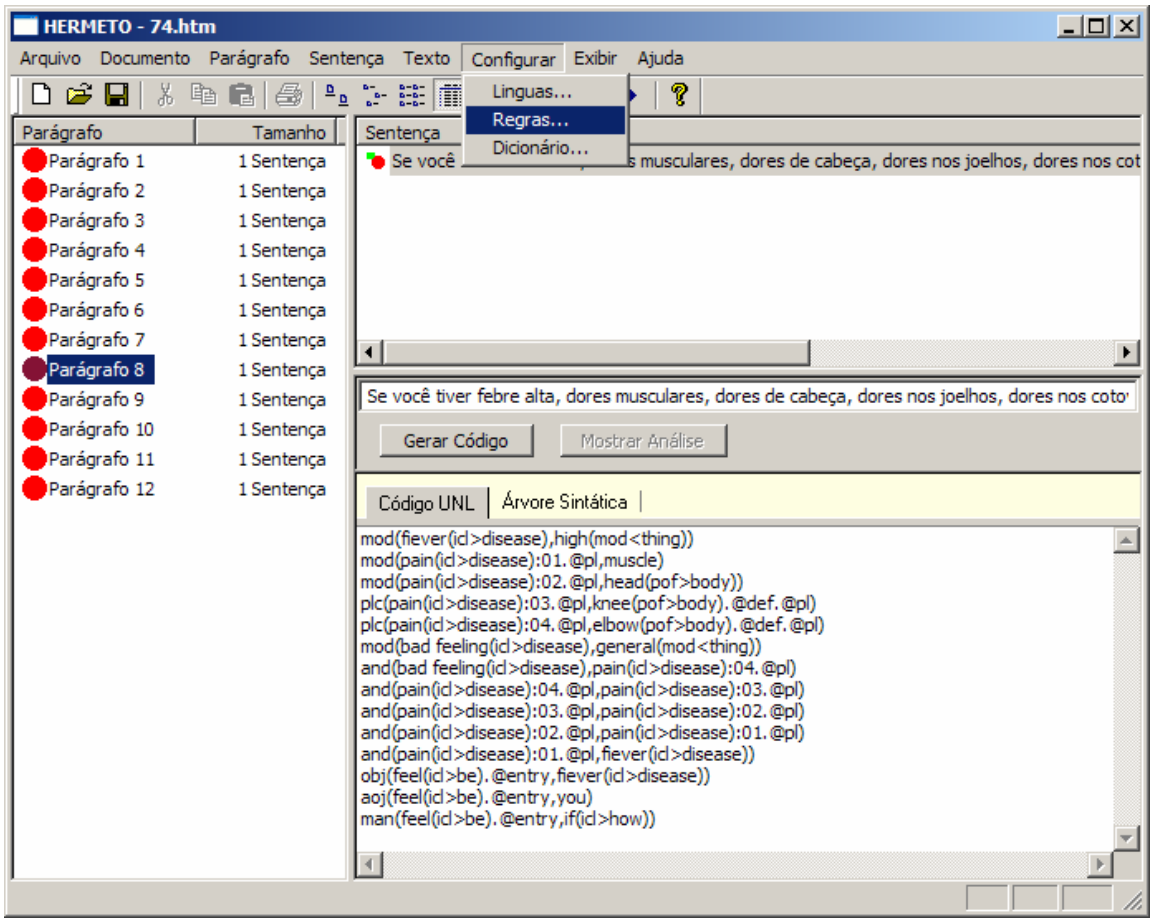


Figura 11 – Ambiente de janelas do Hermeto com o corpus e o resultado em UNL da única sentença do parágrafo 8

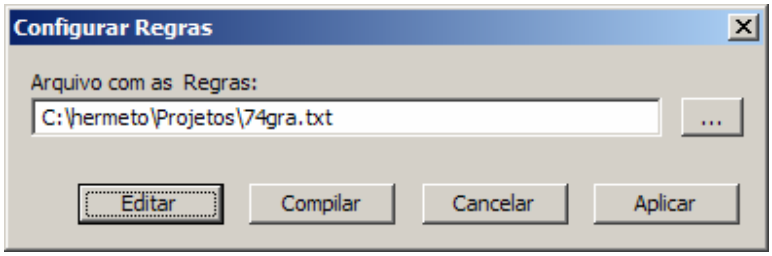


Figura 12 – Caixa de diálogo usada para configurar o ambiente de compilação

Para os recursos de manipulação de *corpus*, não serão dados mais detalhes por serem simples de usar e serem acessados com um simples clique do *mouse*. Entretanto, a configuração do ambiente de trabalho e a criação e edição das regras da gramática por serem mais trabalhosas serão dadas mais atenção.

A configuração e edição das regras da gramática são realizadas ao abrir a caixa de diálogo acessível através da barra de menus. Nesta caixa de diálogo, através da caixa de texto e botões é possível acionar o nome e caminho do arquivo com as regras da gramática, e chamar o editor de *parser* e o compilador para processar o arquivo com as regras da gramática.

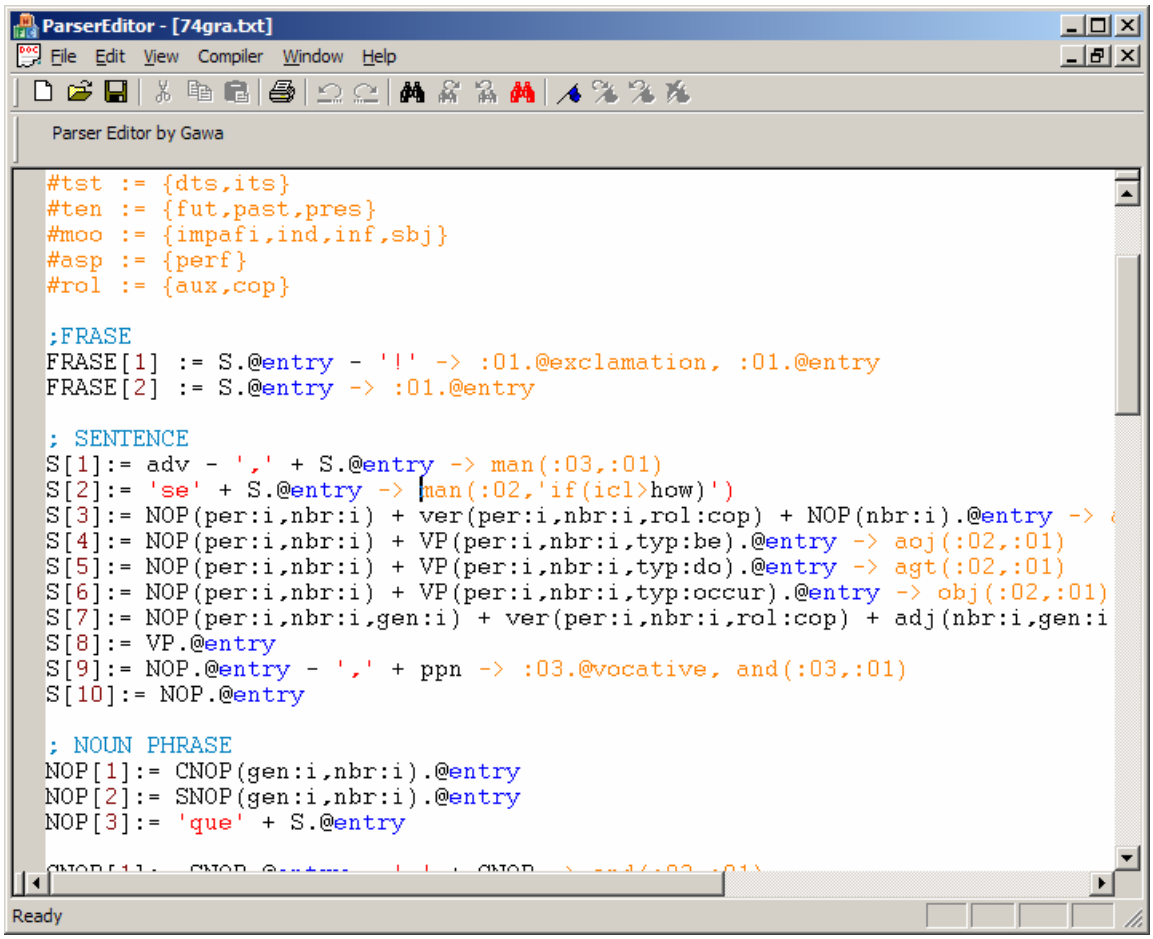


Figura 13 – Janela do Editor de Parser utilizado para edição das regras da gramática

O editor de *parser* é um editor de texto que contém, além dos recursos para edição, recursos para facilitar a visualização dos elementos que compõem as regras da gramática. Durante a edição das regras, o editor destaca automaticamente através das cores os elementos que forem sendo identificados. Por exemplo, as entradas do dicionário são

destacadas em vermelho, as regras de interpretação semântica são destacadas em laranja e as palavras reservadas são destacadas em azul. A figura 13 mostra o editor carregado com a gramática apresentada no Anexo II deste relatório.

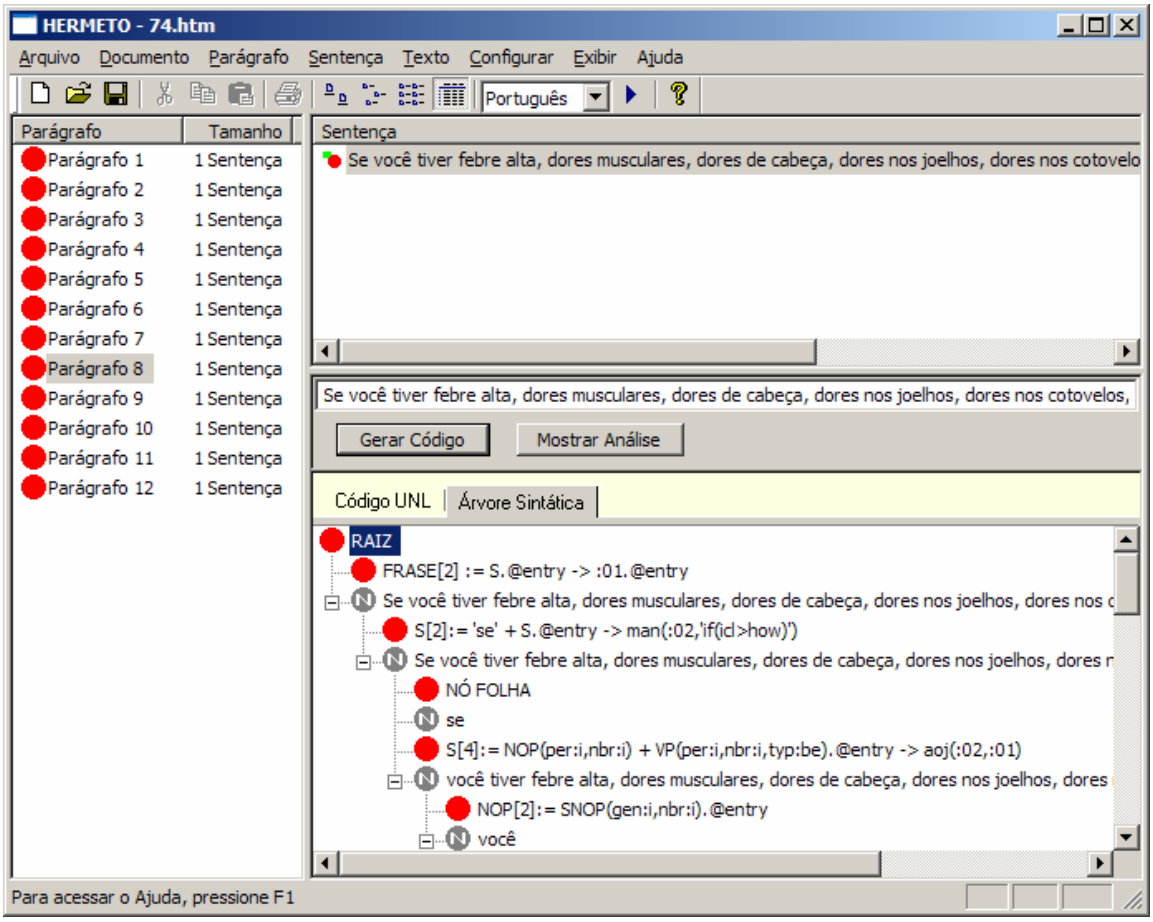


Figura 14 – Ambiente de janelas do Hermeto com a árvore sintática da primeira sentença do oitavo parágrafo

O compilador é um programa implementado em Visual C++ .Net para a plataforma Windows que converte um arquivo texto plano contendo as regras de produção sintática e interpretação semântica em vários outros arquivos contendo classes de objetos em linguagem de programação C++. Uma classe de objetos é formada pela união de todas as regras da gramática que possuam o mesmo símbolo expresso por **a[w]** (sintaxe na Seção 6.2). As regras de produção sintática expressas por P são convertidas em métodos da classe e controladas por um gerenciador de regras que aplica a prioridade dada por W. Este gerenciador de regras também é responsável por parte do controle do mecanismo de *backtracking* e é responsável por aplicar alguns mecanismos de otimização. Por exemplo, se após o gerenciador aplicar uma determinada regra em determinada posição da frase e a regra não for válida para esta posição, ele irá desabilitar esta regra para esta posição da

frase. Este mecanismo permite que regras não candidatas sejam disparadas apenas uma única vez por análise.

Uma das características aproveitadas pelo compilador é o mecanismo de herança. Assim, a classe-base inclui os métodos que todas as classes derivadas subseqüentes devem possuir em comum (ou seja, quase todos os métodos com os mecanismos de gerenciamento das regras); e as classes derivadas incluem, principalmente, os métodos que correspondem às regras de produção sintática e interpretação semântica.

Esse código C++ gerado pelo compilador é então convertido em uma DLL através do Visual C++ chamado internamente pelo próprio Hermeto. Na figura 15, pode-se ver a saída fornecida pelo Visual C++ durante o processo de compilação do código C++.

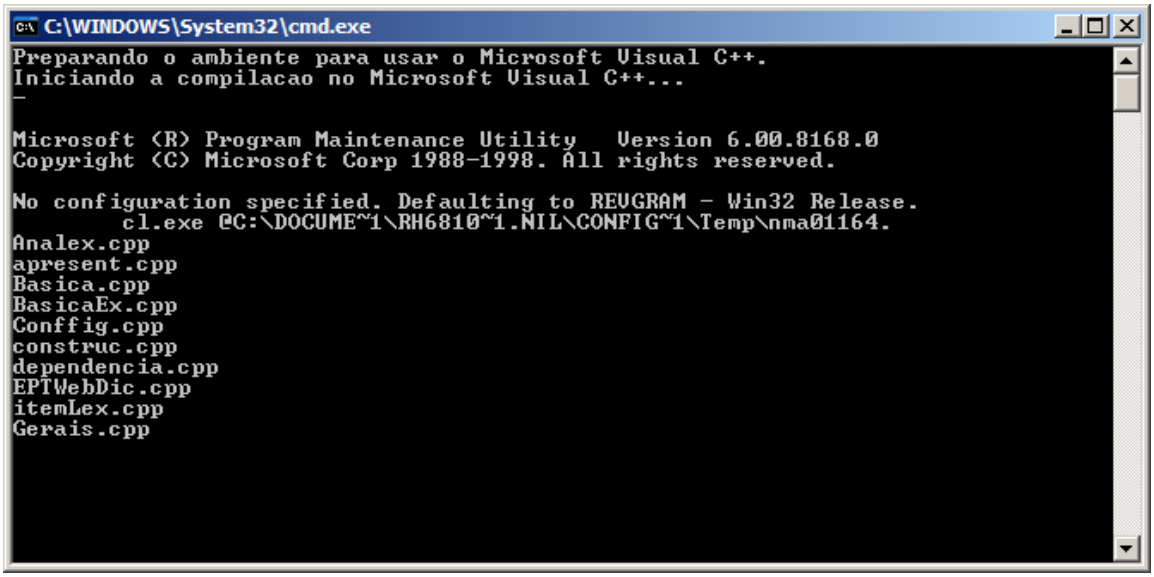


Figura 15 – Janela com a saída do Microsoft Visual C++ durante a conversão do código C++ em DLL

6.5 Resultados parciais

O sistema é capaz de codificar e gerar uma representação em UNL do *corpus* de treinamento. Entretanto, os resultados ainda estão aquém do ideal por diversos fatores lingüísticos e computacionais. A seguir são descritos alguns dos principais problemas/limitações encontrados durante a fase de teste do Hermeto.

Ausência de UWs compostas

A codificação automática das frases em português para UNL, embora seja aceitável, ainda não é a ideal. Por exemplo, na codificação da frase "Falei que água é um perigo!", o *parser* ainda não é capaz de representar o fenômeno de UWs compostas, aplicável já nesta frase e previsto pela UNL. As UWs compostas são um conjunto de relações agrupadas para expressar um conceito. Uma sentença por si só pode ser considerada uma UW composta. A

pertinência de uma relação a uma UW composta é expressa por um identificador especial de UW composta (UW-ID) colocado logo após a RL da relação. Após ser definida, a UW composta pode ser citada ou referenciada ao se usar a UW-ID como uma UW. UWs compostas também podem ser citadas dentro de outras UWs compostas.

Para a frase "falei que água é um perigo!" a codificação ideal com o uso de UW composta seria:

agt(tell(icl>do).@entry.@past, I)
obj(tell(icl>do).@entry.@past, :01)
aoj:01(danger, water(icl>matter))

Entretanto, a codificação gerada pelo *parser* é:
agt(say(icl>do).@exclamation.@entry, I)
obj(say(icl>do).@exclamation.@entry, danger.@indef)
aoj(danger.@indef, water(icl>matter))

Esse tipo de simplificação, pelo menos correntemente e para os casos encontrados no corpus, não prejudica a conversão UNL-LIST. Entretanto, é provável que não funcione para o caso geral, o que torna desejável a extensão do Hermeto para suportar a criação de UWs compostas.

Sem saída em caso de falha

É desejável que uma ferramenta de tradução automática seja capaz de detectar e alertar o usuário de possíveis problemas encontrados nas sentenças de entrada durante o processo de codificação. Entretanto, os problemas de elipse, anáfora e outros fenômenos ainda não foram equacionados de maneira a auxiliar o usuário durante a normalização e codificação das sentenças conforme previsto no início do projeto e já discutido na Seção 5 sobre o corpus de treinamento. Citando um exemplo encontrado no corpus: se o usuário não perceber que deve inserir na sentença "Se você tiver febre alta, dores musculares, de cabeça, nos joelhos, cotovelos, e mal estar geral..." as palavras "dores" e "nos" ("... ***dores*** de cabeça, ***dores*** nos joelhos, ***dores nos*** cotovelos ..."), o Hermeto não conseguirá obter o resultado desejado e simplesmente não haverá qualquer recurso, *feedback* ou ajuda ao usuário para resolver a situação.

Atualmente, todo o processo de tradução é unidirecional, ou seja, os módulos do sistema ativados após a normalização do texto (Hermeto e ManateCo) não conseguem (i) fornecer *feedback* que possa auxiliar o usuário no processo de normalização quando não há êxito na codificação de uma determinada sentença ou (ii) requisitar qualquer informação do usuário para auxiliar decisões em seus respectivos processamentos. Tudo deve estar previsto e ter sido resolvido pelo módulo de normalização. Talvez seja interessante experimentar um novo modelo de processamento em que cada módulo possa requerer dados do usuário conforme necessário, possivelmente por intermédio de um quarto módulo adicional, genérico e único, de interface.

7. O pulo UNL-LIST

O módulo de tradução UNL-LIST, ManateCo, é um dos itens mais prototípicos de PUL∅ exatamente por se basear fortemente no DeConverter (ou carinhosamente DeCo), ferramenta genérica de decodificação UNL-NL fornecida pela *Universal Networking Digital Language Foundation* que apresenta algumas características que limitam ou desaconselham sua aplicação ao problema em vista, pelo menos em sua forma mais geral. O ManateCo propriamente dito não passa de um script Perl que (i) isola expressões UNL de arquivos-texto de entrada (de extensão *.unl*, necessariamente), devidamente marcadas com as etiquetas *{unl}* e *{/unl}*; (ii) submete-as ao DeCo, devidamente alimentado com regras de tradução UNL-LIST e um dicionário UNL-LIST; (iii) procede a uma formatação muito simples da resposta do DeCo; e (iv) gera arquivos-texto de saída (de mesmo *path* que os de entrada, mas com extensão *.list*) idênticos aos de entrada, exceto (iv.i) pela substituição (comportamento *default*) das expressões UNL pelas respectivas versões LIST, marcadas com *{list}* e *{/list}*, ou (iv.ii) pela justaposição (mediante a opção de linha de comando *--unl*) destas em seguida àquelas. O ManateCo é invocado pela seguinte linha de comando:

```
perl manateco.pl [--unl] <arq1.unl> <arq2.unl> ...
```

Apesar de sabermos com antecedência das limitações do DeCo, resolvemos experimentar com essa ferramenta exatamente para levantar requisitos para o projeto de uma ferramenta de decodificação também genérica mas mais compatível com a tradução intermodal especificamente e, em geral, mais amigável, apoiando-se em algum formalismo de mais alto nível, mais declarativo. Ainda assim, além da experiência e requisitos futuros obtidos, alguns dos demais itens produzidos no desenvolvimento do ManateCo foram considerados valiosos na medida em que provavelmente sobreviverão à futura substituição do DeCo, a saber: (i) um conjunto inicial de traços morfossintáticos e morfossemânticos para as entradas lexicais de libras os quais apóiem a (boa) formação de sentenças nessa língua; (ii) uma disciplina inicial de extensão do dicionário UNL-LIST; e (iii) alguns mecanismos de tradução relativamente genéricos. Todos esses resultados são apresentados ao longo desta seção.

7.1 DeCo

Grosso modo o DeCo pode ser descrito como uma Máquina de Turing não-determinística acrescida de alguma funcionalidade para consulta de dicionários. O iniciado em Teoria da Computação já pode adivinhar quão primitivos são os recursos de programação oferecidos por essa ferramenta. Resumidamente, a máquina virtual do DeCo consiste nos seguintes elementos:

- um **dicionário** de correspondências LIST-UNL, de formato idêntico ao exposto na Seção 6.3;

- uma **fita extensível**, cujas **células** representam, a um só tempo, (i) nós do grafo UNL que está sendo traduzido e (ii) as respectivas palavras correspondentes na língua-alvo[7]. Cada célula guarda um **estado** que consiste de um simples conjunto de **traços atômicos**. Inicialmente a fita contém apenas o nó nuclear (com atributo *@entry*) do grafo UNL;
- um **par de cabeças de leitura/gravação sempre contíguas** capazes de sentir e mudar o estado das células, bem como se mover conjuntamente uma posição para a esquerda ou direita sobre a fita, de acordo com
- um **conjunto de regras de tradução**, totalmente ordenado. Cada regra especifica **pré-condições** de aplicação, em termos de traços presentes ou ausentes no estado das células correntemente lidas, a ao menos uma das cabeças. Dada uma configuração da máquina, um de cujos principais fatores é *onde exatamente se localiza o par de cabeças no momento considerado*, o DeCo tenta aplicar as regras na ordem de prioridade. A primeira regra aplicável é efetivada e o processo continua até que as cabeças tentem ultrapassar o limite direito da fita, sinal de sucesso. Em caso de falha (máquina parada por se encontrar numa configuração não tratada por nenhuma regra), ocorre *backtracking*, ou seja, a última aplicação de regra é anulada, a regra aplicável alternativa de maior prioridade é efetivada e o processo é retomado desse ponto. Basicamente há dois tipos de regras:

  i) **regras de modificação de traços**, que simplesmente mudam o estado das células sob as cabeças de l/g, adicionando ou removendo traços, e opcionalmente dão um passo à esquerda ou à direita sobre a fita. O estado das células usualmente codifica não só informações gramaticais, semânticas, etc. das palavras correspondentes, mas muito freqüentemente é usado para simular estados globais da máquina (explícito na Máquina de Turing) e assim programar um rudimento de sub-rotinas;

  ii) **regras de inserção**, que também podem testar e mudar o estado das células envolvidas mas cujo efeito mais relevante é *a transferência de um nó do grafo UNL para a fita* e a conseqüente expansão desta. Essa transferência é um momento muito importante em que o nó a ser inserido é "amalgamado" com uma de suas possíveis traduções no dicionário, gerando uma nova célula cujo estado é inicializado (i) com os traços constantes do dicionário para aquela tradução e (ii) com traços homônimos aos atributos UNL daquele nó. Naturalmente, as regras de inserção especificam pré-condições a serem satisfeitas tanto (i) pelo nó a ser inserido, ao especificar que tipo de relação UNL deve existir entre este e o **nó-âncora** (aquele já pertencente à fita e associado à célula sob a cabeça de l/g direita/esquerda e imediatamente à esquerda/direita da qual a nova célula será inserida), quanto (ii) pelas traduções aceitáveis, ao especificar que traços devem apresentar ou não. Mais uma vez, caso haja mais de uma tradução aplicável, um eventual *backtracking* acabará por experimentar traduções alternativas.

O processo de tradução grosso modo se resume à inserção de nós na hora certa, o que se consegue por meio da ordenação das regras. Por exemplo, se sabemos que um verbo

[7] Por vezes, essas células não correspondem a nós do grafo UNL, mas apenas a palavras (geralmente de classe fechada) inseridas por regras de tradução especiais.

deve ser precedido por seu sujeito e seu objeto nessa ordem, a regra de inserção do sujeito deve ter prioridade sobre a de inserção do objeto, já que o nó-âncora de ambos será o mesmo (o verbo). Assim, haverá um momento em que o (núcleo do) sujeito estará adjacente ao verbo pela esquerda, momento que deve ser aproveitado para resolver qualquer questão de concordância entre eles, por regras que escreverão traços específicos no verbo em função dos traços lidos no sujeito. Quando o (núcleo do) objeto for por fim inserido, o sujeito será deslocado para a esquerda e o contato sujeito-verbo será perdido.

A concordância verbal serve para exemplificar uma das maiores inconveniências do DeCo, qual seja a absoluta simplicidade da linguagem de especificação de pré-condições. Nada de lógica de primeira ordem, variáveis ou análise de traços aqui: cada traço é efetivamente atômico, e a linguagem de especificação de condições equivale à lógica de predicados. Assim, para cada possível pessoa que o sujeito puder assumir, deverá haver uma regra diferente para gerar a informação correspondente no verbo. Simplesmente não é possível expressar algo do tipo: "se na célula da direita há o traço *pers=X*, então escreva *psuj=X* na da esquerda".

Enquanto a linguagen de pré-condições torna o DeCo incômodo, há uma outra limitação que praticamente o desqualifica para a geração LIST, a saber: simplesmente não há como gerar, talvez por não haver como manipular, identificadores únicos, necessários para a co-indexação de sinais, como exemplificado na Seção 3. Outra desvantagem dessa ferramenta, como se pode observar, é a natureza totalmente procedimental – e nisso de baixíssimo nível – de sua programação, num contexto (processamento de língua natural) em que abordagens mais declarativas e de nível mais alto são desejáveis.

O funcionamento do DeCo está completamente descrito no manual *DeConverter Specifications, Version 2.5* do *UNL Centre/UNDL Foundation*.

7.2 Dicionário UNL-LIST

O dicionário UNL-LIST integrante do ManateCo, usado para alimentar diretamente o DeCo, é apresentado na íntegra no Anexo IV. Será abordado aqui o conjunto de traços morfossintáticos e morfossemânticos usado neste dicionário para caracterizar o *potencial sintático* de cada entrada lexical de LIST. A Tabela 3 relaciona esses traços, seguidos de seus respectivos significados e algumas observações pertinentes. Quanto à interpretação dessa tabela, vale ressaltar o seguinte:

i) uma ocorrência em específico de uma entrada lexical não precisa necessariamente realizar todo o potencial sintático previsto. ;

ii) todos os traços em MAIÚSCULAS são eminentemente artificiais e redundantes, concebidos, contudo, para contrabalançar a incapacidade já discutida do DeCo em lidar com lógica de primeira ordem;

iii) traços do tipo *@X* sempre terminam por gerar atributos LIST;

iv) traços do tipo *X~Y* geralmente permitem uma leitura do tipo "*X* se realiza como *Y*";

v) traços do tipo *X_* ou *_X* caracterizam uma entrada (representada pelo "_" do símbolo) como possivelmente localizada à direita ou à esquerda de *X*, respectivamente;

vi) alguns traços apresentam uma variável *Path* que representa um "caminho" de relações semânticas ou sintáticas a ser seguido para se chegar a uma entidade (nó ou sintagma) a partir do nó ou sintagma associado à entrada em questão. Um **path semântico** tem o formato $R_1R_{2...}R_n$, $n \geq 1$, onde cada R_i tem o formato $<RL$ ou $>RL$ e representa a passagem do nó corrente N_i para algum N_{i+1} com que mantenha a relação UNL $RL(N_i, N_{i+1})$ ou $RL(N_{i+1}, N_i)$, respectivamente. Um **path sintático** tem o formato R_1-R_2-...-R_n, $n \geq 1$, onde cada R_i é um rótulo de relação sintática e representa a passagem do núcleo sintático corrente N_i para algum N_{i+1} com que mantenha a relação sintática $R_i(N_i, N_{i+1})$.

		traço	significado e observações
nominais		n	núcleo do sintagma nominal (aplicável e.g. a nomes e pronomes)
		_n	modificador pré-nominal (aplicável e.g. a nomes e adjetivos)
		n_	modificador pós-nominal
		rel	elemento ([+n], advérbio, [+_n], etc.) relativo
		pers=*X*	informação de pessoa requerido de toda entrada [+n], inclusive não-pronominal. O traço [+pers=3?] indica subespecificação de número. • $X \in$ {1s, 1p, 2s, 3s, 3?}, no corpus
		PERSED	requerido de toda entrada [+pers=X]
		gen=*X*	informação de gênero requerido de toda entrada [+n] a que se aplique. • $X \in$ {coisa-redonda, coisa-achatada}, no corpus
		GENED	requerido de toda entrada [+gen=X]
		gentampa=*X*	informação de gênero de tampa (!) requerido de toda entrada [+n] a que se aplique. • $X \in$ {coisa-redonda, coisa-achatada}, no corpus
		GENTAMPAED	requerido de toda entrada [+gentampa=X]
		@gen	[+n] autoconcordante em gênero (vide Seção 3.4)
verbais	**aplicação geral**	v	verbo
		@psuj	[+v] que concorda em pessoa com o sujeito
		@pobj	[+v] que concorda em pessoa com o objeto
		@gensuj	[+v] que concorda em gênero com o sujeito
		@genobj	[+v] que concorda em gênero com o objeto
		@gentampaobj	[+v] que concorda em gênero com a tampa do objeto
		_obj	[+v] que tem objeto posposto
		obj_	[+v] que tem objeto anteposto
		Path~suj	[+v] cujo sujeito deve ser a tradução de um nó atingido por *Path*. • *Path* $\in$ {<agt, <aoj, <obj}, no corpus

		Path~obj	[+v] cujo objeto deve ser a tradução de um nó atingido por *Path*. • *Path* ∈ {<obj}, no corpus
	suporte	backing	[+v] de suporte; traço que dispara um movimento de tradução em que o *verbo suporte* será substituído por um outro verbo, dito *dominante*, associado a um outro nó. • A substituição é realizada de forma transparente: (i) o verbo dominante é caracterizado normalmente, ou seja, sem que se tenha de considerar que poderá vir a ser dominante; (ii) todas as restrições válidas para o verbo dominante, e.g. [+_obj/obj_, @psuj, etc.], são devidamente respeitadas
		Path~pred	[+v, +backing] cujo verbo dominante deve estar associado ao nó atingido por *Path*. • *Path* ∈ {<obj<aoj, <obj}, no corpus; • usado na tradução de "manter fechada" e "tomar cuidado com"
		Prep~obj	[+v, +backing] cujo objeto deve ser a tradução de um nó *O* atingido pelas seguintes relações UNL (seja *BV* o nó associado ao verbo suporte em questão): *man(BV, Prep), obj(Prep, O)* • *Path* ∈ {with}, no corpus; • usado na tradução de "tomar cuidado com"
outros		conj	conjunção
		prep	preposição (geralmente apagada)
		silentreq=*Path*/*C*	elemento que exige a condição *C* de um nó ou sintagma necessariamente existente, dito *implicado*, atingido por *Path*. Adicionalmente, [+silentreq=*Path/C*] *inibe a realização do implicado*. • *Path/C* ∈ {<mod/icl>head, <mod/icl>medical, <mod/icl>general}, no corpus; • usado para gerar as expressões LIST "dor-de-cabeça", "médico" e "MAL corpo todo" a partir de pares de UWs
		SILENTREQ	requerido de todo [+silenreq=*R/C*]
		req=*Path*/*C*	elemento que, se atinge um nó ou sintagma por *Path*, exige deste a condição *C*. • *Path/C* ∈ {obj/GENTAMPAED}, no corpus; • Usado para determinar que a versão LIST de "fechar" em "fechar a caixa d'agua" deve ser "fechar-com-tampa"
		REQ=*Path*	requerido de todo [+req=*Path*/*C*]
		@exclamation	requerido de toda entrada que deve expressar o traço LIST *excl*

Tabela 3 - Relação dos traços possíveis entradas lexicais de LIST.

7.3 Regras de tradução UNL-LIST

O conjunto de regras de tradução UNL-LIST integrante do ManateCo, usado para alimentar diretamente o DeCo, é apresentado na íntegra no Anexo V. Trata-se de um programa numa linguagem simples, mas consideravelmente confusa, o qual corresponde, resumida e abstratamente, às seguintes regras de tradução, em ordem de prioridade:

1. **verificação de *silentreq*:** caso seja inserido um nó com traço *SILENTREQ*, sua condição é imediatamente testada para que uma falha seja gerada o mais cedo possível no processo;
2. **concordância:** dada a adjacência de um sujeito ou objeto ao seu verbo, geram-se neste os atributos pertinentes de acordo com as informações de pessoa, gênero, etc. daquele. Vale notar que qualquer verbo, mesmo que não marcado para atributos de concordância (*@psuj*, *@gensuj*, etc.), carregará toda a informação de concordância disponível, possivelmente de forma apenas subjacente. Isso é importante, por exemplo, para a geração de "vamos" (vide item 15) ou ainda para transferência desses dados de verbos suporte (item 13) para verbos dominantes, possivelmente marcados para atributos de concordância;
3. **autoconcordância:** dada a coexistência, numa célula, do par de traços *@X* e *X=Y*, o primeiro é substituído pelo traço *@X=Y*. Ao final da tradução, traços deste último tipo darão origem a atribuições LIST *{X:Y}*;
4. **verificação de *req*:** caso tenha sido inserido um nó com traço *REQ=Path* e o *Path* esteja acessível, testa-se a condição imposta;
5. **inicialização:** envolve-se o nó inicial num grupo rítmico (sentencial), inserindo duas **células artificiais** (não associadas a nós UNL) à volta daquele nó as quais realizam a abertura e fechamento de colchetes. Eventuais atributos UNL sentenciais (*@exclamation* e *@imperative*) são transferidos para a célula que fecha o grupo de forma a serem expressos na posição correta ao final do processo;
6. **apagamento de sujeito:** apaga-se o sujeito do imperativo ou ainda o sujeito de 1ª pessoa do plural genérico. Perceba que isso só ocorre depois de executadas as regras de concordância, devido à prioridade destas;
7. **coordenação:** elementos coordenados pela *RL and* são justapostos. Cadeias de mais de dois coordenandos são consideradas enumerações longas e já formatadas segundo as determinações da Seção 3.4. Tal é o motivo da alta prioridade da regra: com a formatação prematura, a subseqüente expansão dos coordenandos com seus modificadores e complementos gerará células dentro dos grupos rítmicos corretos;
8. **marcação de vocativos:** o núcleo do vocativo, comumente elemento de uma coordenação, é envolvido por um grupo rítmico próprio a cuja célula de fechamento é transferido o atributo *@vocative* do núcleo em questão. Esse atributo gerará o traço LIST {voc} ao final da tradução;
9. **inserção de modificadores nominais**, à esquerda ou à direita do núcleo de acordo com a especificação dos modificadores para os traços *[_n]* e *[n_]*. Inclui-se aí a inserção de relativos, *[+rel]*, sempre à direita, entretanto. O pronome relativo tratado até o

momento (UW “that”) é realizado por uma palavra vazia, o que equivale ao seu apagamento. Sua função na tradução, apesar de meramente operacional, é essencial, agendando a inserção da oração relativa e propagando as informações de seu referente para efeito de concordância;

10. **inserção de conjunções:** se existe um nó *C* que (i) estabeleça a relação UNL *man(V, C)*, típica de advérbios de modo, com o nó-âncora verbal *V* e (ii) que tenha uma leitura marcada *[+conj]*, então *C* tem essa leitura selecionada e é inserido à esquerda de *V* como uma conjunção. A alta prioridade dessa regra tende a fazer que a conjunção apareça na extremidade das cláusulas;
11. **inserção de advérbios de lugar:** se existe um nó *L* que estabeleça a relação UNL *plc(L, X)* com o nó-âncora *X*, então este será inserido à esquerda de *X*;
12. **inserção de sujeito**, sempre à esquerda e de acordo com o traço *[+X~suj]* do verbo em questão;
13. **verbo suporte:** o verbo suporte é substituído pelo verbo dominante. A prioridade desse grupo de regras, estrategicamente entre a inserção do sujeito e a do objeto, deve-se aos seguintes fatores: (i) o sujeito tem posição sempre à esquerda de qualquer verbo; (ii) o sujeito do verbo suporte sempre será o mesmo do dominante; e (iii) a colocação do objeto é que é delicada e não-padrão, já que depende da situação do verbo dominante, e não do suporte, para os traços *[_obj]* e *[obj_]*;
14. **inserção de objeto**, à esquerda ou à direita, de acordo com o traço *[+X~obj]* do verbo em questão e sua situação para os traços *[obj_]* e *[_obj]*;
15. **inserção de “vamos”**, como célula artificial, à esquerda do verbo imperativo de 1ª pessoa do plural. A baixa prioridade dessa regra tende a fazer que “vamos” apareça adjacente ao verbo em questão;
16. **inserção de “já”**, como célula artificial, à direita de verbo com atributo *@past*. A baixa prioridade dessa regra tende a fazer que “já” apareça adjacente ao verbo em questão;
17. **inserção de “precisar”**, como célula artificial, à esquerda de verbo com atributo *@need*. A baixa prioridade dessa regra tende a fazer que “precisar” apareça adjacente ao verbo em questão;
18. **inserção de atributos LIST:** uma série de atributos @X e todos os atributos @X=Y de cada célula *C* gerarão atributos LIST correspondentes imediatamente à direita de *C*. Um passo simples de pós-processamento da saída do DeCo converterá seqüências contínuas de tais atributos em matrizes atributo-valor.

8. Integração do sistema

Atualmente os módulos do PUL∅ se articulam da seguinte forma: o ManateCo funciona como uma aplicação independente que, dados pré-documentos LIST (i.e., documentos LIST que prescindam exatamente das versões LIST de suas expressões UNL), gera documentos LIST correspondentes completos; o Hermeto está disponível como uma DLL exportando uma função que, dada uma sentença em português normalizado, gera a expressão UNL correspondente; e o Kaapor se apresenta como uma aplicação que, por conveniência, executa todo o procedimento de tradução de forma transparente (i) gerando o arquivo normalizado (extensão *.kpr*), (ii) submetendo cada uma de suas sentenças à DLL Hermeto, (iii) usando o resultado para gerar um pré-documento LIST (*.unl*) e (iv) invocando o ManateCo para o arquivo em questão, o que resulta num documento LIST (*.list*) completo correspondente à historinha editada (como o apresentado no Anexo IX).

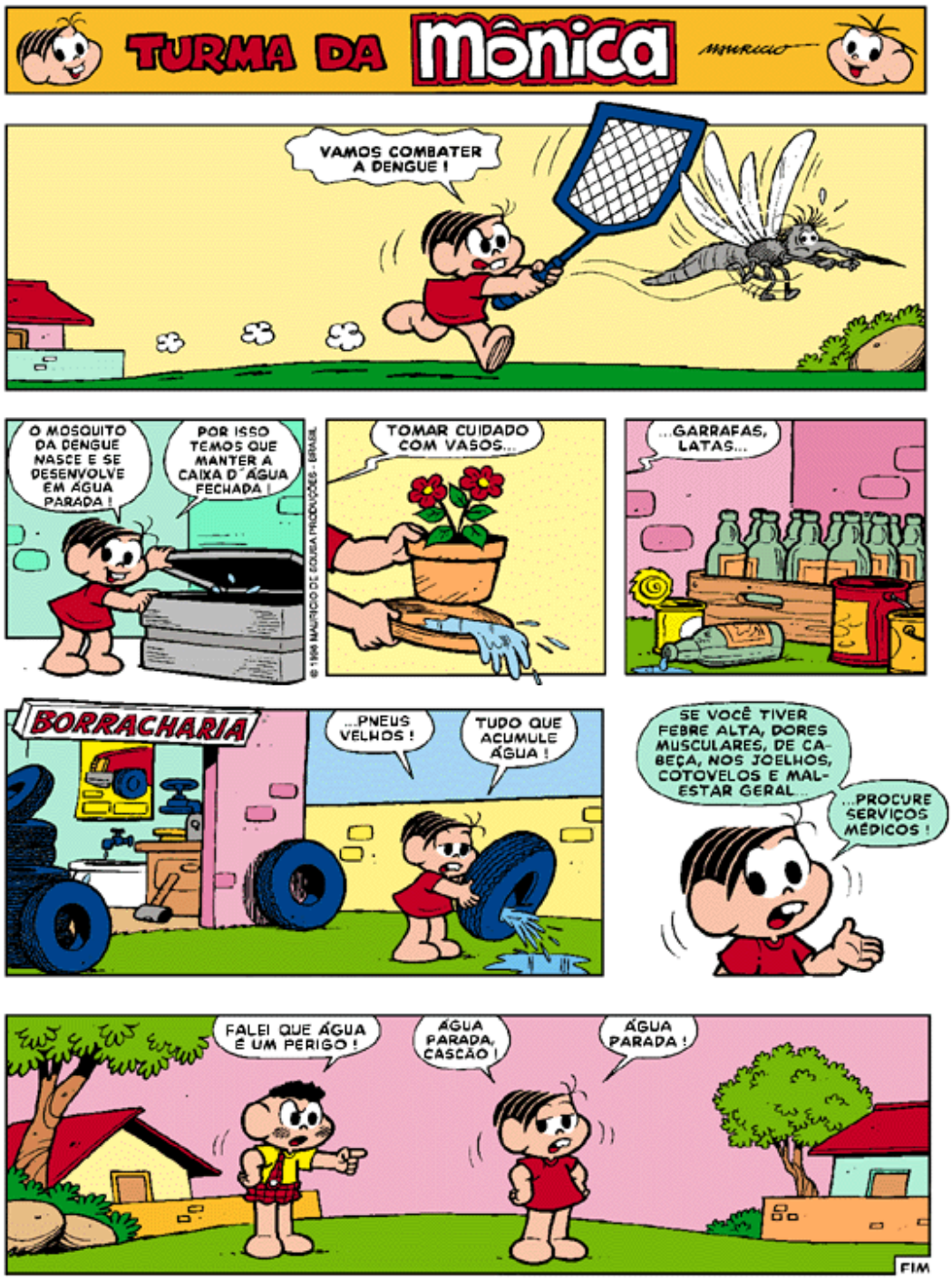
9. Avaliação global e perspectivas de trabalhos futuros

As seções anteriores deixam evidente a complexidade da tarefa executada nessa fase do projeto – a tradução de português para uma transcrição de libras suficiente para alimentar um sintetizador de fala.

A estratégia de prototipação, focando uma única história até o momento, mostrou-se apropriada, embora pareça ser bastante restritiva, uma vez que em um corpus maior de sentenças em português deverão surgir outros problemas a serem atacados. No entanto, tomou-se o devido cuidado de se tratar fenômenos bastante freqüentes, de modo que o modelo proposto, a menos do vocabulário, mostra-se abrangente o suficiente para que sua extensão ocorra com custo mínimo.

Os próximos passos deverão, assim, considerar a extensão do corpus tratado, a superação das limitações já identificadas e, principalmente, a integração do sistema PUL∅ com o módulo de síntese de fala. Há ainda o desenvolvimento de uma alternativa ao DeCo, como agendado na Seção 7, e a possibilidade de revisão do modelo unidirecional de tradução como discutido na Seção 6.5.

Anexo I – História 74



Original	**UNL**	**LIST**
Vamos combater a dengue!	obj(fight(icl>do).@exclamation.@entry.@imperative, dengue(icl>disease).@def) agt(fight(icl>do).@exclamation.@entry.@imperative, we)	[DENGUE vamos vencer{pobj:3s psuj:1p}]{excl imp}
O mosquito da DENGUE nasce e se desenvolve em água parada.	mod(mosquito(icl>insect).@def, dengue(icl>disease).@def) and(develop(icl>occur), be born(icl>occur).@entry) mod(water(icl>matter), stagnant(mod<thing)) plc(be born(icl>occur).@entry,water(icl>matter)) obj(be born(icl>occur).@entry,mosquito(icl>insect).@def)	[água largado DENGUE mosquito surgir desenvolver]
Por isso, temos que manter a caixa d'água fechada.	obj(keep(icl>be).@exclamation.@entry.@need, water tank(icl>container).@def) aoj(closed(mod<thing), water tank(icl>container).@def) aoj(keep(icl>be).@exclamation.@entry.@need, we) man(keep(icl>be).@exclamation.@entry.@need, thus(icl>how))	[por-isso caixa água precisar fechar-com-tampa{gentampaobj: coisa-achatada}]{excl}
Tomar cuidado com vasos...	obj(take(icl>do).@entry, care) obj(with(icl>how), flower pot(icl>container).@pl) man(take(icl>do).@entry, with(icl>how))	[cuidado{excl} vaso flor]
...garrafas, latas...	and(can(icl>container).@pl, bottle(icl>container).@entry.@pl)	[garrafa ferro{gen:coisa-redonda}]
...pneus velhos!	mod(tire(pof>automobile).@exclamation.@entry.@pl, old(mod<thing))	[pneu velho]{excl}
Tudo que acumule água!	obj(gather(icl>do), water(icl>matter)) agt(gather(icl>do), that) mod(all.@exclamation.@entry, that)	[tudo água dentro]{excl}
Se você tiver febre alta, dores musculares, de cabeça, nos joelhos, cotovelos, e mal estar geral ...	mod(fever(icl>disease), high(mod<thing)) mod(pain(icl>disease):01.@pl, muscle) mod(pain(icl>disease):02.@pl, head(pof>body)) plc(pain(icl>disease):03.@pl, knee(pof>body).@def.@pl) plc(pain(icl>disease):04.@pl, elbow(pof>body).@def.@pl) mod(bad feeling(icl>disease), general(mod<thing)) and(bad feeling(icl>disease), pain(icl>disease):04.@pl) and(pain(icl>disease):04.@pl, pain(icl>disease):03.@pl) and(pain(icl>disease):03.@pl, pain(icl>disease):02.@pl) and(pain(icl>disease):02.@pl, pain(icl>disease):01.@pl) and(pain(icl>disease):01.@pl, fever(icl>disease)) obj(feel(icl>be).@entry, fever(icl>disease)) aoj(feel(icl>be).@entry, you) man(feel(icl>be).@entry, if(icl>how))	[SI você [[fiever(icl>disease) alto] [músculo dor] [dor-de-cabeça] [joelho dor] [cotovelo dor] [MAL corpo todo]]{enum}]
procure serviços médicos	mod(service.@pl, medical(mod<thing)) obj(search(icl>do).@entry, service.@pl)	[procurar médico]
eu falei que água é um perigo!	aoj(danger.@indef, water(icl>matter)) obj(say(icl>do).@exclamation.@entry, danger.@indef) agt(say(icl>do).@exclamation.@entry, I)	[eu falar água perigoso]{excl}

água parada, Cascão!	mod(water(icl>matter).@exclamation.@entry, stagnant(mod<thing)) and(Cascão(icl>person).@vocative, water(icl>matter).@exclamation.@entry)	[água largado [Cascão]{voc}]{excl}
água parada!	mod(water(icl>matter).@exclamation.@entry, stagnant(mod<thing))	[água largado]{excl}

Anexo II – Gramática português-UNL

```
;=========================================================================
; DEFINIÇÕES USADAS PARA FACILITAR A INTEGRAÇÃO ENTRE AS REGRAS DO PARSER
E OS ATRIBUTOS DAS PALAVRAS NO LÉXICO.
;=========================================================================
#pos := {art,adj,adv,cjc,nou,ppn,ppr,pre,pro,ver}
#per := {1per,2per,3per}
#nbr := {sgn,pln}
#gen := {fem,mcl}
#typ := {acc,be,def,do,ndf,nom,occur,pro,rel}
#tst := {dts,its}
#ten := {fut,past,pres}
#moo := {impafi,ind,inf,sbj}
#asp := {perf}
#rol := {aux,cop}

;FRASE
FRASE[1] := S.@entry - '!' -> :01.@exclamation, :01.@entry
FRASE[2] := S.@entry -> :01.@entry

; SENTENCE
S[1]:= adv - ',' + S.@entry -> man(:03,:01)
S[2]:= 'se' + S.@entry -> man(:02,'if(icl>how)')
S[3]:= NOP(per:i,nbr:i) + ver(per:i,nbr:i,rol:cop) + NOP(nbr:i).@entry ->
aoj(:03,:01)
S[4]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:be).@entry -> aoj(:02,:01)
S[5]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:do).@entry -> agt(:02,:01)
S[6]:= NOP(per:i,nbr:i) + VP(per:i,nbr:i,typ:occur).@entry -> obj(:02,:01)
S[7]:= NOP(per:i,nbr:i,gen:i) + ver(per:i,nbr:i,rol:cop) +
adj(nbr:i,gen:i).@entry -> aoj(:03,:01)
S[8]:= VP.@entry
S[9]:= NOP.@entry - ',' + ppn -> :03.@vocative, and(:03,:01)
S[10]:= NOP.@entry

; NOUN PHRASE
NOP[1]:= CNOP(gen:i,nbr:i).@entry
NOP[2]:= SNOP(gen:i,nbr:i).@entry
NOP[3]:= 'que' + S.@entry

CNOP[1]:= SNOP.@entry - ',' + CNOP -> and(:03,:01)
CNOP[2]:= SNOP.@entry - ',' + ['e'] + SNOP -> and(:04,:01)
CNOP[3]:= SNOP.@entry + 'e' + SNOP -> and(:03,:01)

SNOP[1]:= art(gen:i,nbr:i,typ:def) + SNOP(gen:i,nbr:i).@entry -> :02.@def
SNOP[2]:= art(gen:i,nbr:i,typ:ndf) + SNOP(gen:i,nbr:i).@entry -
> :02.@indef
SNOP[3]:= NOU(gen:i,nbr:i).@entry + MOD -> mod(:01,:02)
SNOP[4]:= NOU(gen:i,nbr:i).@entry + PLC -> plc(:01,:02)
SNOP[5]:= NOU(gen:i,nbr:i).@entry + adj(gen:i,nbr:i) -> mod(:01,:02)
SNOP[6]:= pro(typ:ndf).@entry + MOD -> mod(:01,:02)
;SNOP[7]:= NOU(gen:i,nbr:pln).@entry -> :01.@pl
SNOP[7]:= NOU(gen:i,nbr:i).@entry
```

```
SNOP[8]:= ppr(gen:i,nbr:i).@entry

; ADJECTIVE PHRASE
AJP[1]:= 'que'.@entry + VP(typ:do) -> agt(:02,:01)
AJP[2]:= 'que'.@entry + VP(typ:be) -> aoj(:02,:01)
AJP[3]:= 'que'.@entry + VP(typ:occur) -> obj(:02,:01)

; VERBAL PHRASE
VP[1]:= CVP(typ:i,per:i,nbr:i).@entry + PLC -> plc(:01,:02)
VP[2]:= CVP(typ:i,per:i,nbr:i).@entry + MAN -> man(:01,:02)
VP[3]:= CVP(typ:i,per:i,nbr:i).@entry

VP[4]:= SVP(typ:i,per:i,nbr:i).@entry + PLC -> plc(:01,:02)
VP[5]:= SVP(typ:i,per:i,nbr:i).@entry + MAN -> man(:01,:02)
VP[6]:= SVP(typ:i,per:i,nbr:i).@entry

CVP[1]:= SVP(typ:i,per:i,nbr:i,ten:i,moo:i).@entry - ',' +
CVP(typ:i,per:i,ten:i,moo:i,nbr:i)  -> and(:03,:01)
CVP[2]:= SVP(typ:i,per:i,nbr:i,ten:i,moo:i).@entry - ',' +
SVP(typ:i,per:i,ten:i,moo:i,nbr:i)  -> and(:03,:01)
CVP[3]:= SVP(typ:i,per:i,nbr:i,ten:i,moo:i).@entry + 'e' +
SVP(typ:i,per:i,ten:i,moo:i,nbr:i)  -> and(:03,:01)

SVP[1]:= <ter> + 'que' + SVP(moo:inf).@entry -> :03.@need
SVP[2]:= 'vamos' + SVP(typ:do,moo:inf).@entry -> :02.@imperative,
agt(:02,'we')
SVP[3]:= ver(per:i,nbr:i,typ:i,rol:cop).@entry + NOP(nbr:i,gen:i) +
adj(nbr:i,gen:i) -> obj(:01,:02), aoj(:03,:02)
SVP[4]:= ver(per:i,nbr:i,typ:be,tst:dts).@entry + NOP -> obj(:01,:02)
SVP[5]:= ver(per:i,nbr:i,typ:do,tst:dts).@entry + NOP -> obj(:01,:02)
SVP[6]:= ver(per:i,nbr:i,typ:do,tst:dts).@entry + NOP -> gol(:01,:02)
SVP[7]:= ppr(typ:acc,per:i,nbr:i) +
ver(per:i,nbr:i,typ:pro,tst:its).@entry
SVP[8]:= ver(per:i,nbr:i,typ:i,tst:its).@entry

; MOD
MOD[1]:= 'd' - SNOP.@entry
MOD[2]:= 'de' + SNOP.@entry
MOD[3]:= AJP.@entry

; PLC
PLC[1] := CPLC.@entry
PLC[2] := SPLC.@entry
CPLC[1]:= SPLC.@entry - ',' + CPLC -> and(:03,:01)
CPLC[2]:= SPLC.@entry - ',' + ['e'] + SPLC -> and(:04,:01)
CPLC[3]:= SPLC.@entry + 'e' + SPLC -> and(:03,:01)
SPLC[1]:= 'n' - SNOP.@entry
SPLC[2]:= 'em' + SNOP.@entry

; MAN
MAN[1]:= 'com'.@entry + SNOP -> obj('with(icl>how)', :02)

; NOU -> Acrescenta a UNL o número quanto este estiver no plural
NOU[1] := nou(gen:i,nbr:pln).@entry -> :01.@pl
```

```
NOU[2] := nou(gen:i,nbr:i).@entry

; ACESSO AO LEXICO
adv[1] := find(pos:adv)
ver[1] := find(pos:ver)
adj[1] := find(pos:adj)
art[1] := find(pos:art)
nou[1] := find(pos:nou)
pro[1] := find(pos:pro)
ppr[1] := find(pos:ppr)
ppn[1] := find(pos:ppn)
```

Anexo III – Dicionário português-UNL

```
[a] {} o "the" (pos:art;typ:def;gen:fem;nbr:sgn) <PT-BR,0,0>;
[acumule] {} acumule "gather(icl>do)"
(pos:ver;typ:do;tst:dts;per:3per;nbr:sgn;ten:pres;moo:sbj) <PT-BR,0,0>;
[alta] {} alta "high(mod<thing)" (pos:adj;gen:fem;nbr:sgn) <PT-BR,0,0>;
[água] {} água "water(icl>matter)" (pos:nou;gen:fem;nbr:sgn) <PT-BR,0,0>;
[cabeça] {} cabeça "head(pof>body)" (pos:nou;gen:fem;nbr:sgn) <PT-BR,0,0>;
[caixa d'água] {} caixa_d'água "water tank(icl>container)"
(pos:nou;gen:fem;nbr:sgn) <PT-BR,0,0>;
[Cascão] {} Cascão "Cascão(icl>person)" (pos:ppn;gen:mcl;nbr:sgn) <PT-
BR,0,0>;
[com] {} com "with(icl>how)" (pos:pre) <PT-BR,0,0>;
[combater] {} combater "fight(icl>do)" (pos:ver;typ:do;tst:dts;moo:inf)
<PT-BR,0,0>;
[cotovelos] {} cotovelo "elbow(pof>body)" (pos:nou;gen:mcl;nbr:pln) <PT-
BR,0,0>;
[cuidado] {} cuidado "care" (pos:nou;gen:mcl;nbr:sgn) <PT-BR,0,0>;
[d] {} de "of" (pos:pre) <PT-BR,0,0>;
[de] {} de "of" (pos:pre) <PT-BR,0,0>;
[dengue] {} dengue "dengue(icl>disease)" (pos:nou;gen:fem;nbr:sgn) <PT-
BR,0,0>;
[desenvolve] {} desenvolve "develop(icl>occur)"
(pos:ver;typ:occur;tst:its;typ:pro;per:3per;nbr:sgn;ten:pres;moo:ind)
<PT-BR,0,0>;
[dores] {} dor "pain(icl>disease)" (pos:nou;gen:fem;nbr:pln) <PT-BR,0,0>;
[e] {} e "and" (pos:cjc) <PT-BR,0,0>;
[em] {} em "in" (pos:pre) <PT-BR,0,0>;
[eu] {} eu "I" (pos:ppr;typ:nom;per:1per;nbr:sgn) <PT-BR,0,0>;
[é] {} ser "be(icl>be)"
(pos:ver;typ:be;tst:dts;rol:cop;per:3per;nbr:sgn;ten:pres;moo:ind) <PT-
BR,0,0>;
[falei] {} falar "say(icl>do)"
(pos:ver;typ:do;tst:dts;per:1per;nbr:sgn;ten:past;asp:perf;moo:ind) <PT-
BR,0,0>;
[febre] {} febre "fever(icl>disease)" (pos:nou;gen:fem;nbr:sgn) <PT-
BR,0,0>;
[fechada] {} fechado "closed(mod<thing)" (pos:adj;gen:fem;nbr:sgn) <PT-
BR,0,0>;
[garrafas] {} garrafa "bottle(icl>container)" (pos:nou;gen:fem;nbr:pln)
<PT-BR,0,0>;
[geral] {} geral "general(mod<thing)" (pos:adj;gen:mcl,fem;nbr:sgn) <PT-
BR,0,0>;
[joelhos] {} joelho "knee(pof>body)" (pos:nou;gen:mcl;nbr:pln) <PT-
BR,0,0>;
[latas] {} lata "can(icl>container)" (pos:nou;gen:fem;nbr:pln) <PT-
BR,0,0>;
[mal estar] {} mal_estar "bad feeling(icl>disease)"
(pos:nou;gen:mcl;nbr:sgn) <PT-BR,0,0>;
[manter] {} manter "keep(icl>be)" (pos:ver;typ:be;tst:dts;rol:cop;moo:inf)
<PT-BR,0,0>;
```

```
[médicos] {} médico "medical(mod<thing)" (pos:adj;gen:mcl;nbr:pln) <PT-
BR,0,0>;
[mosquito] {} mosquito "mosquito(icl>insect)" (pos:nou;gen:mcl;nbr:sgn)
<PT-BR,0,0>;
[musculares] {} muscular "muscle" (pos:adj;gen:mcl;gen:fem;nbr:pln) <PT-
BR,0,0>;
[n] {} em "in" (pos:pre) <PT-BR,0,0>;
[nasce] {} nascer "be born(icl>occur)"
(pos:ver;typ:occur;tst:its;per:3per;nbr:sgn;ten:pres;moo:ind)  <PT-BR,0,0>;
[nascem] {} nascer "be born(icl>occur)"
(pos:ver;typ:occur;tst:its;per:3per;nbr:pln;ten:pres;moo:ind)  <PT-BR,0,0>;
[nós] {} nós "we" (pos:ppr;typ:nom;per:1per;nbr:pln) <PT-BR,0,0>;
[o] {} o "the" (pos:art;typ:def;gen:mcl;nbr:sgn) <PT-BR,0,0>;
[os] {} o "the" (pos:art;typ:def;gen:mcl;nbr:pln) <PT-BR,0,0>;
[parada] {} parado "stagnant(mod<thing)" (pos:adj;gen:fem;nbr:sgn) <PT-
BR,0,0>;
[perigo] {} perigo "danger" (pos:nou;gen:mcl;nbr:sgn) <PT-BR,0,0>;
[pneus] {} pneu "tire(pof>automobile)" (pos:nou;gen:mcl;nbr:pln) <PT-
BR,0,0>;
[por este motivo] {} por_isso "thus(icl>how)" (pos:adv) <PT-BR,0,0>;
[por isso] {} por_isso "thus(icl>how)" (pos:adv) <PT-BR,0,0>;
[procure] {} procurar "search(icl>do)"
(pos:ver;typ:do;tst:dts;per:3per;nbr:sgn;moo:impafi) <PT-BR,0,0>;
[que] {} que "that" (pos:cjc) <PT-BR,0,0>;
[que] {} que "that" (pos:pro;typ:rel) <PT-BR,0,0>;
[se] {} se "him" (pos:ppr;typ:acc;per:3per;nbr:sgn;nbr:pln) <PT-BR,0,0>;
[se] {} se "if" (pos:cjc) <PT-BR,0,0>;
[serviços] {} serviço "service" (pos:nou;gen:mcl;nbr:pln) <PT-BR,0,0>;
[temos] {} ter "need(icl>be)"
(pos:ver;typ:be;tst:dts;rol:aux;per:1per;nbr:pln;ten:pres;moo:ind) <PT-
BR,0,0>;
[tiver] {} ter "feel(icl>be)"
(pos:ver;typ:be;tst:dts;per:3per;nbr:sgn;ten:fut;moo:sbj) <PT-BR,0,0>;
[tomar] {} tomar "take(icl>do)" (pos:ver;typ:do;tst:dts;moo:inf) <PT-
BR,0,0>;
[tudo] {} tudo "all" (pos:pro;typ:ndf;per:3per;nbr:sgn) <PT-BR,0,0>;
[um] {} um "a" (pos:art;typ:ndf;gen:mcl;nbr:sgn) <PT-BR,0,0>;
[vamos] {} ir "shall"
(pos:ver;typ:be;rol:aux;per:1per;nbr:pln;ten:pres;moo:ind) <PT-BR,0,0>;
[vasos] {} vaso "flower pot(icl>container)" (pos:nou;gen:mcl;nbr:pln)
<PT-BR,0,0>;
[velhos] {} velho "old(mod<thing)" (pos:adj;gen:mcl;nbr:pln) <PT-BR,0,0>;
[você] {} você "you" (pos:ppr;typ:nom;per:3per;nbr:sgn) <PT-BR,0,0>;
```

Anexo IV – Dicionário UNL-LIST

```
;    Nomes

[água] {} "water(icl>matter)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[cabeça] {} "head(pof>body)" (n, _n, PERSED, pers=3?, icl>head) <L,0,0> ;
[caixa água] {} "water tank(icl>container)" (n, _n, PERSED, pers=3?,
    GENTAMPAED, gentampa=coisa-achatada) <L,0,0> ;
[cotovelo] {} "elbow(pof>body)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[DENGUE] {} "dengue(icl>disease)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[dor-de-cabeça] {} "pain(icl>disease)" (n, _n, PERSED, pers=3?,
    silentreq=<mod/icl>head, SILENTREQ) <L,0,1> ;
[dor] {} "pain(icl>disease)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[febre] {} "fever(icl>disease)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[ferro] {} "can(icl>container)" (n, _n, @gen, gen=coisa-redonda, GENED,
    PERSED, pers=3?) <L,0,0> ;
[garrafa] {} "bottle(icl>container)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[joelho] {} "knee(pof>body)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[MAL corpo todo] {} "bad feeling(icl>disease)" (n, _n, PERSED, pers=3?,
    silentreq=<mod/icl>general, SILENTREQ) <L,0,1> ;
[médico] {} "service" (n, _n, PERSED, pers=3?,
    silentreq=<mod/icl>medical, SILENTREQ) <L,0,1> ;
[mosquito] {} "mosquito(icl>insect)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[músculo] {} "muscle" (n, _n, PERSED, pers=3?) <L,0,0> ;
[pneu] {} "tire(pof>automobile)" (n, _n, PERSED, pers=3?) <L,0,0> ;
[vaso flor] {} "flower pot(icl>container)" (n, _n, PERSED, pers=3?)
    <L,0,0> ;
[cascão] {} "Cascão(icl>person)" (n, _n, PERSED, pers=3?) <L,0,0> ;

;    Pronomes

[eu] {} "I" (n, PERSED, pers=1s) <L,0,0> ;
[nós] {} "we" (n, PERSED, pers=1p) <L,0,0> ;
[tudo] {} "all" (n, PERSED, pers=3s) <L,0,0> ;
[você] {} "you" (n, PERSED, pers=2s) <L,0,0> ;
[] {} "that" (n, rel) <L,0,0> ;

;    Verbos

[cuidado] {} "care" (@exclamation, v, _obj, <obj~obj, <agt~suj) <L,0,0> ;
[dentro] {} "gather(icl>do)" (v, <obj~obj, <agt~suj, obj_) <L,0,0> ;
[desenvolver] {} "develop(icl>occur)" (v, <obj~suj) <L,0,0> ;
[falar] {} "say(icl>do)" (v, <obj~obj, <agt~suj, _obj) <L,0,0> ;
[fechar-com-tampa] {} "closed(mod<thing)" (v, @gentampaobj,
    req=obj/GENTAMPAED, REQ=OBJ, <obj~obj, <agt~suj, obj_) <L,0,0> ;
[] {} "feel(icl>be)" (v, <obj~obj, <aoj~suj, _obj) <L,0,0> ;
[procurar] {} "search(icl>do)" (v, <obj~obj, _obj, <agt~suj) <L,0,0> ;
[surgir] {} "be born(icl>occur)" (v, <obj~suj) <L,0,0> ;
[vencer] {} "fight(icl>do)" (v, obj_, @psuj, @pobj, <obj~obj, <agt~suj)
    <L,0,0> ;

;    Verbos "suporte"
```

```
[] {} "keep(icl>be)" (v, backing, <obj<aoj~pred, <obj~obj, <aoj~suj)
    <L,0,0> ;
[] {} "take(icl>do)" (v, backing, <obj~pred, with~obj) <L,0,0> ;

;   Adjetivos

[alto] {} "high(mod<thing)" (n_) <L,0,0> ;
[largado] {} "stagnant(mod<thing)" (n_) <L,0,0> ;
[médico] {} "medical(mod<thing)" (n_, icl>medical) <L,0,0> ;
[perigoso] {} "danger" (n_) <L,0,0> ;
[velho] {} "old(mod<thing)" (n_) <L,0,0> ;
[geral] {} "general(mod<thing)" (n_, icl>general) <L,0,0> ;

;   Conjunções

[por-isso] {} "thus(icl>how)" (conj) <L,0,0> ;
[SI] {} "if(icl>how)" (conj) <L,0,0> ;

;   Preposições (a serem apagadas)

[with] {} "with(icl>how)" (prep, _obj) <L,0,0> ;
```

Anexo V – Regras de tradução UNL-LIST

```
;     RSHIFT e LSHIFT

R {RSHIFT:-RSHIFT::} {:::} ;
? {RSHIFT:::} {:::} ;
R {^RSHIFT:::} {RSHIFT:-RSHIFT::} ;
? {^RSHIFT:::} {RSHIFT:::} ;
L {:::} {LSHIFT:-LSHIFT::} ;
? {:::} {LSHIFT:::} ;
L {LSHIFT:-LSHIFT::} {^LSHIFT:::} ;
? {LSHIFT:::} {^LSHIFT:::} ;

;     BYE

DL {BYE:::} {:::} ;
? {BYE:::} {:::} ;
DR {:::} {BYE:::} ;
? {:::} {BYE:::} ;

;     REWIND

DL {REWIND:::} {:REWINDING::} ;
? {REWIND:::} {:::} ;
: {^SHEAD:REWINDING::} {REWINDING:-REWINDING::} ;
: {SHEAD:::} {REWINDING:-REWINDING::} ;
: {^SHEAD,STOP_REWINDING:-STOP_REWINDING::} {REWINDING:-REWINDING::} ;
? {:::} {REWINDING:::} ;

;     SILENTREQ

: {:::} {SILENTREQ:-SILENTREQ,SILENTREQ1,NO_SILENTREQ_RSHIFT::} ;
? {:::} {SILENTREQ:::} ;
: {SILENTREQ:-SILENTREQ,SILENTREQ1::} {:::} ;
? {SILENTREQ:::} {:::} ;
: "icl>medical,^<and:RSHIFT:mod:"
{SILENTREQ1,silentreq=<mod/icl>medical:-SILENTREQ1,SILENTREQ2::} ;
: "icl>general,^<and:RSHIFT:mod:"
{SILENTREQ1,silentreq=<mod/icl>general:-SILENTREQ1,SILENTREQ2::} ;
: "icl>head,^<and:RSHIFT:mod:" {SILENTREQ1,silentreq=<mod/icl>head:-
SILENTREQ1,SILENTREQ2::} ;
?R {:::} {SILENTREQ1:::} ;
: {:[],RSHIFT,SILENTREQ_BYE::} {SILENTREQ2:-SILENTREQ2,SILENTREQ3::} ;
? {:::} {SILENTREQ2:::} ;
: {:::} {SILENTREQ3,NO_SILENTREQ_RSHIFT:-SILENTREQ3,-
NO_SILENTREQ_RSHIFT::} ;
R {:::} {SILENTREQ3,^NO_SILENTREQ_RSHIFT:-SILENTREQ3::} ;
? {:::} {SILENTREQ3:::} ;

DL {SILENTREQ_BYE:::} {:::} ;
? {SILENTREQ_BYE:::} {:::} ;
```

```
;     preparação do imperativo

: {:::} {IMPERATIVE0:-IMPERATIVE0,imperative,PROP_IMPERATIVE::} ;
? {:::} {IMPERATIVE0:::} ;
: {IMPERATIVE0:-IMPERATIVE0,imperative,PROP_IMPERATIVE,RSHIFT::} {:::} ;
? {IMPERATIVE0:::} {:::} ;

;     TAKE_V_ATTRS - transferência de atributos de verbo

: {@need:-@need,RSHIFT::} {TAKE_V_ATTRS:@need::} ;
? {@need:::} {TAKE_V_ATTRS:::} ;
: {imperative:-imperative,RSHIFT::} {TAKE_V_ATTRS:imperative::} ;
? {imperative:::} {TAKE_V_ATTRS:::} ;
: {@past:-@past,RSHIFT::} {TAKE_V_ATTRS:@past::} ;
? {@past:::} {TAKE_V_ATTRS:::} ;
: {:::} {TAKE_V_ATTRS:-TAKE_V_ATTRS::} ;
? {:::} {TAKE_V_ATTRS:::} ;

;     propagação do imperativo para o sujeito

: {suj:imperative,RSHIFT::} {PROP_IMPERATIVE:-PROP_IMPERATIVE::} ;
? {suj:::} {PROP_IMPERATIVE:::} ;

;     resolução de número de 3a pessoa

: {pers=3?,@pl:-pers=3?,pers=3p,RSHIFT::} {:::} ;
: {pers=3?,^@pl:-pers=3?,pers=3s,RSHIFT::} {:::} ;
? {pers=3?:::} {:::} ;
: {:::} {pers=3?,@pl:-pers=3?,pers=3p::} ;
: {:::} {pers=3?,^@pl:-pers=3?,pers=3s::} ;
? {:::} {pers=3?:::} ;

;     propagação da informação de pessoa

: {psuj=1s:::} {GET_PSUJ:-GET_PSUJ,psuj=1s,PSUJED::} ;
: {psuj=1p:::} {GET_PSUJ:-GET_PSUJ,psuj=1p,PSUJED::} ;
: {psuj=2s:::} {GET_PSUJ:-GET_PSUJ,psuj=2s,PSUJED::} ;
: {psuj=3s:::} {GET_PSUJ:-GET_PSUJ,psuj=3s,PSUJED::} ;
: {psuj=3p:::} {GET_PSUJ:-GET_PSUJ,psuj=3p,PSUJED::} ;
? {PSUJED:::} {GET_PSUJ:::} ;
: {pers=1s:::} {GET_PSUJ:-GET_PSUJ,psuj=1s,PSUJED::} ;
: {pers=2s:::} {GET_PSUJ:-GET_PSUJ,psuj=2s,PSUJED::} ;
: {pers=1p:::} {GET_PSUJ:-GET_PSUJ,psuj=1p,PSUJED::} ;
: {pers=3s:::} {GET_PSUJ:-GET_PSUJ,psuj=3s,PSUJED::} ;
: {pers=3p:::} {GET_PSUJ:-GET_PSUJ,psuj=3p,PSUJED::} ;
? {PERSED:::} {GET_PSUJ:::} ;
: {:::} {GET_PSUJ:-GET_PSUJ,psuj=0::} ;
? {:::} {GET_PSUJ:::} ;

: {pobj=3s:::} {GET_POBJ:-GET_POBJ,pobj=3s,POBJED::} ;
: {pobj=3p:::} {GET_POBJ:-GET_POBJ,pobj=3p,POBJED::} ;
? {POBJED:::} {GET_POBJ:::} ;
: {pers=3s:::} {GET_POBJ:-GET_POBJ,pobj=3s,POBJED::} ;
: {pers=3p:::} {GET_POBJ:-GET_POBJ,pobj=3p,POBJED::} ;
```

```
? {PERSED:::} {GET_POBJ:::} ;
: {:::} {GET_POBJ:-GET_POBJ,pobj=0::} ;
? {:::} {GET_POBJ:::} ;

;     propagação da informação de gênero

: {:::} {GET_GENSUJ:-GET_GENSUJ,gensuj=0::} ;
? {:::} {GET_GENSUJ:::} ;

: {:::} {GET_GENOBJ:-GET_GENOBJ,genobj=0::} ;
? {:::} {GET_GENOBJ:::} ;

;     propagação da informação de gênero de tampa ;o)

: {gentampaobj=coisa-achatada:::} {GET_GENTAMPAOBJ:-
GET_GENTAMPAOBJ,gentampaobj=coisa-achatada,GENTAMPAOBJED::} ;
? {GENTAMPAOBJED:::} {GET_GENTAMPAOBJ:::} ;
: {gentampa=coisa-achatada:::} {GET_GENTAMPAOBJ:-
GET_GENTAMPAOBJ,gentampaobj=coisa-achatada,GENTAMPAOBJED::} ;
? {GENTAMPAED:::} {GET_GENTAMPAOBJ:::} ;
: {:::} {GET_GENTAMPAOBJ:-GET_GENTAMPAOBJ,gentampaobj=0::} ;
? {:::} {GET_GENTAMPAOBJ:::} ;

;     propagação da informação do sujeito

: {SUJED:::} {GET_SUJED:-GET_SUJED,SUJED::} ;
? {SUJED:::} {GET_SUJED:::} ;

: {:::} {GET_SUJINFO:-GET_SUJINFO,GET_PSUJ,GET_GENSUJ,GET_SUJED::} ;
? {:::} {GET_SUJINFO:::} ;

;     propagação da informação do objeto

: {OBJED:::} {GET_OBJED:-GET_OBJED,OBJED::} ;
? {OBJED:::} {GET_OBJED:::} ;
: {GET_OBJED:-GET_OBJED,OBJED,RSHIFT::} {OBJED:::} ;
? {GET_OBJED:::} {OBJED:::} ;


: {:::} {GET_OBJINFO:-
GET_OBJINFO,GET_POBJ,GET_GENOBJ,GET_GENTAMPAOBJ,GET_OBJED::} ;
? {:::} {GET_OBJINFO:::} .
: {GET_OBJINFO:-
GET_OBJINFO,RSHIFT,GET_POBJ,GET_GENOBJ,GET_GENTAMPAOBJ,GET_OBJED::} {:::}
;
? {GET_OBJINFO:::} {:::} ;

;     concordância em pessoa com sujeito e objeto

: {:::} {@psuj,psuj=1p:-@psuj,@psuj=1p::} ;
? {:::} {@psuj,PSUJED:::} ;

: {:::} {@pobj,pobj=3s:-@pobj,@pobj=3s::} ;
? {:::} {@pobj,POBJED:::} ;
```

```
;     concordância em gênero com o objeto

: {:::} {@gentampaobj,gentampaobj=coisa-achatada:-
@gentampaobj,@gentampaobj=coisa-achatada::} ;
? {:::} {@gentampaobj,GENTAMPAOBJED:::} ;

;     autoconcordância de gênero

: {:::} {@gen,gen=coisa-redonda:-@gen,@gen=coisa-redonda::} ;
? {:::} {@gen,GENED:::} ;

;     DROP_OBJ_BYE - semeia o objeto de uma preposição e DESLOCA para o
lado certo

: ":OBJ:obj:" {DROP_OBJ_BYE:-DROP_OBJ_BYE,BYE::} ;
? {:::} {DROP_OBJ_BYE:::} ;
: {DROP_OBJ_BYE:-DROP_OBJ_BYE,BYE::} ":RSHIFT,OBJ:obj:" ;
? {DROP_OBJ_BYE:::} {:::} ;

;     criação de blocos

;     BLOCK_OUT

R {:::} {BLOCK_OUT:NO_BLOCK_OUT_RSHIFT::} ;
? {:::} {BLOCK_OUT:::} ;
: {BLOCK_OUT:-BLOCK_OUT,BLOCK_OUT1::} "[}]:block_out,artif::" ; L
? {BLOCK_OUT:::} {:::} ;
: {:::} {BLOCK_OUT1,NO_BLOCK_OUT_RSHIFT:-BLOCK_OUT1,-
NO_BLOCK_OUT_RSHIFT::} ;
R {:::} {BLOCK_OUT1,^NO_BLOCK_OUT_RSHIFT:-BLOCK_OUT1::} ;
? {:::} {BLOCK_OUT1:::} ;

;     BLOCK_IN

L {BLOCK_IN:BLOCK_IN_RSHIFT::} {:::} ;
? {BLOCK_IN:::} {:::} ;
: "[{]:block_in,artif,RSHIFT::" {BLOCK_IN:-BLOCK_IN,BLOCK_IN1::} ;
? {:::} {BLOCK_IN:::} ;
R {:::} {BLOCK_IN1,BLOCK_IN_RSHIFT:-BLOCK_IN1,-BLOCK_IN_RSHIFT::} ;
: {:::} {BLOCK_IN1,^BLOCK_IN_RSHIFT:-BLOCK_IN1::} ;
? {:::} {BLOCK_IN1:::} ;

;     BLOCK

R {:::} {BLOCK:NO_BLOCK_RSHIFT::} ;
? {:::} {BLOCK:::} ;
: {BLOCK:-BLOCK,BLOCK1,BLOCK_IN,BLOCK_OUT,block_entry::} {:::} ; L
; como a regra acima movimenta para a esquerda, "block_entry" termina à
direita
? {BLOCK:::} {:::} ;
: {:::} {BLOCK1,NO_BLOCK_RSHIFT:-BLOCK1,-NO_BLOCK_RSHIFT::} ;
R {:::} {BLOCK1,^NO_BLOCK_RSHIFT:-BLOCK1::} ;
? {:::} {BLOCK1:::} ;
```

```
;     BACKING1: completando verbo suporte - importante que tenha
prioridade sobre OBJ

: "v,_obj:GET_SUJINFO,TAKE_V_ATTRS,RSHIFT_LOWP:aoj:" {BACKING1:-
BACKING1::} ;
: {:::} {BACKING1:-BACKING1,BACKING1_2::} ;
? {:::} {BACKING1:::} ;
: {:::} {BACKING1_2:-BACKING1_2,BACKING1_3,TAKE_V_ATTRS::} ;
? {:::} {BACKING1_2:::} ;
R {:::} {BACKING1_3:::} ;
? {:::} {BACKING1_3:::} ;
: {BACKING1_3:RSHIFT,-BACKING1_3:aoj:}
"v,obj_:GET_SUJINFO,TAKE_V_ATTRS,GET_OBJINFO::" ;
? {BACKING1_3:::} {:::} ;

;     Verbo suporte 2

;     <OBJ~PRED - resolvendo as coisas entre as cópias

: {<OBJ~PRED1:-<OBJ~PRED1,<OBJ~PRED2,PRED_IN,RSHIFT::} {:-v,-
<obj~pred,PRED_OUT::} ;
? {<OBJ~PRED1:::} {:::} ;
: {<OBJ~PRED2:-<OBJ~PRED2,RSHIFT::} "v:GET_SUJINFO,TAKE_V_ATTRS:obj:" ;
?L {<OBJ~PRED2,^>obj:::} {:::} ;
? {<OBJ~PRED2:::} {:::} ;

: {with~obj,PRED_IN:-with~obj,-PRED_IN,RSHIFT,WITH~OBJ::} {v,obj_:::} ;
R {with~obj,PRED_IN:-with~obj,-PRED_IN::} {v,_obj:::} ;
? {with~obj,PRED_IN:::} {:::} ;
: {v,_obj:::} {with~obj,PRED_OUT:-with~obj,-PRED_OUT,WITH~OBJ::} ;
? {v,_obj:::} {with~obj,PRED_OUT:::} ;

: {WITH~OBJ:-WITH~OBJ,RSHIFT::} "[with]:DROP_OBJ_BYE,RSHIFT:man:" ;
? {WITH~OBJ:::} {:::} ;
: "[with]:DROP_OBJ_BYE:man:" {WITH~OBJ:-WITH~OBJ::} ;
? {:::} {WITH~OBJ:::} ;

;     SUJ (resolve as coisas entre sujeito e verbo)

: {SUJ:RSHIFT,-SUJ,suj::} {:GET_SUJINFO,SUJED::} ;
? {SUJ:::} {:::} ;

;     OBJ (resolve as coisas entre objeto e verbo)

: {OBJ:RSHIFT,-OBJ,obj::} {:GET_OBJINFO,OBJED::} ;
? {OBJ:::} {:::} ;
: {:RSHIFT,OBJED::} {OBJ:-OBJ,obj::} ;
? {:::} {OBJ:::} ;

;     verifica requisitos semânticos de objeto

: {obj,GENTAMPAED:::} {req=obj/GENTAMPAED:-req=obj/GENTAMPAED,-REQ=OBJ::}
;
```

```
?R {obj,^GENTAMPAED:::} {req=obj/GENTAMPAED} ;
? {obj:::} {REQ=OBJ:::} ;
: {req=obj/GENTAMPAED:RSHIFT,-req=obj/GENTAMPAED,-REQ=OBJ::}
{obj,GENTAMPAED:::} ;
?L {req=obj/GENTAMPAED} {obj,^GENTAMPAED:::} ;
? {REQ=OBJ:::} {obj:::} ;


;     RSHIFT_LOWP

R {RSHIFT_LOWP:-RSHIFT_LOWP::} {:::} ;
? {RSHIFT_LOWP:::} {:::} ;
R {^RSHIFT_LOWP:::} {RSHIFT_LOWP:-RSHIFT_LOWP::} ;
? {^RSHIFT_LOWP:::} {RSHIFT_LOWP:::} ;
L {:::} {LSHIFT_LOWP:-LSHIFT_LOWP::} ;
? {:::} {LSHIFT_LOWP:::} ;
L {LSHIFT_LOWP:-LSHIFT_LOWP::} {^LSHIFT_LOWP:::} ;
? {LSHIFT_LOWP:::} {^LSHIFT_LOWP:::} ;


;     inicialização da sentença

R {:::} {^SCOPE,@entry,^started_up:STARTING_UP::} ;
? {:::} {^SCOPE,@entry,^started_up:::} ;
: {STARTING_UP,^sblock_entry:sblock_entry,BLOCK,RSHIFT::} {:::} ;
? {STARTING_UP,^sblock_entry:::} {:::} ;
: {STARTING_UP,@imperative:-@imperative,IMPERATIVE0,RSHIFT::}
{:@imperative::} ;
? {STARTING_UP,@imperative:::} {:::} ;
: {STARTING_UP,@exclamation:-@exclamation,RSHIFT::} {:@exclamation::} ;
? {STARTING_UP,@exclamation:::} {:::} ;
: {STARTING_UP:-STARTING_UP,started_up::} {:::}
? {STARTING_UP:::} {:::} ;

;     apagamento do sujeito

DL {suj,imperative:::} {:::} ; do imperativo
DL {suj,pers=1p:::} {:::} ; "nós genérico"

;     tratando ands

L {:LSHIFT::} {AND_LSHIFT:-AND_LSHIFT::} ;

L {AND_OUT2:-AND_OUT2::} {:@enum::} ;
? {AND_OUT2:::} {:::} ;

: {:ENUM,RSHIFT::} {ENUM,^AND_IN:-ENUM,-AND_INNER,BLOCK,AND_LSHIFT::} ;
: {:::} {ENUM,AND_IN:-ENUM,ENUM1,BLOCK_IN::} ;
? {:::} {ENUM:::} ;
: {:::} {ENUM1:-ENUM1,BLOCK::} ;
? {:::} {ENUM1:::} ;

: {^AND_INNER:AND_IN,coorded,-<and,RSHIFT:and:} ":AND_INNER,coorded,-
>and,RSHIFT::" ;
```

```
: {AND_INNER:-<and,RSHIFT:and:} ":AND_INNER,coorded,->and,RSHIFT::" ;
? {<and:::} {:::} ;
L {AND_INNER:-AND_INNER,AND_OUT::} {^AND_INNER,^AND_OUT:::} ;
? {AND_INNER:::} {^AND_INNER,^AND_OUT:::} ;
: {AND_IN:-AND_IN::} {AND_OUT:-AND_OUT::} ;
? {AND_IN:::} {AND_OUT:::} ;
R {^AND_IN:::} {AND_OUT:-AND_OUT,AND_OUT2,BLOCK_OUT,ENUM::} ;
? {^AND_IN:::} {AND_OUT:::} ;

;     vocativos (em bloco)

: {@vocative,^voc_blocked:-@vocative,VOCKING1,RSHIFT,NO_VOCKING_LSHIFT::}
{:::} ;
? {@vocative,^voc_blocked:::} {:::} ;
: {:::} {@vocative,^voc_blocked:-@vocative,VOCKING1,RSHIFT::} ;
? {:::} {@vocative,^voc_blocked:::};
: {VOCKING1:-VOCKING1,VOCKING2,RSHIFT,BLOCK::} {:::} ;
? {VOCKING1:::} {:::} ;
: {VOCKING2:-VOCKING2,VOCKING3,RSHIFT::} {:@vocative,voc_blocked::} ;
? {VOCKING2:::} {:::} ;
R {VOCKING3,NO_VOCKING_LSHIFT:-VOCKING3,-NO_VOCKING_LSHIFT,RSHIFT::}
{:RSHIFT::} ;
L {VOCKING3,^NO_VOCKING_LSHIFT:-VOCKING3,LSHIFT::} {:::} ;
? {VOCKING3:::} {:::} ;

;     mod

: "_n::mod:" {:::} ;
: {:::} "n,rel:GET_SUJINFO,GET_OBJINFO:mod:";
: {:::} "n_::mod:" ;

;     conjunções

: "conj::man:" {v:::} ;

;     termos acessórios

: "::plc:" {:::} ;

;     sujeito (relativo ou não) - sempre antes.

: {rel:RSHIFT,SUJ,-<agt:agt:} "<agt~suj:->agt::" ;
? {rel,<agt:::} {:::} ;
: ":RSHIFT,SUJ:agt:" {<agt~suj:->agt,LSHIFT_LOWP::} ;
? {:::} {>agt,<agt~suj:::} ;
: ":RSHIFT,SUJ:obj:" {<obj~suj:->obj,LSHIFT_LOWP::} ;
? {:::} {>obj,<obj~suj:::} ;
: ":RSHIFT,SUJ:aoj:" {<aoj~suj:->aoj,LSHIFT_LOWP::} ;
? {:::} {>aoj,<aoj~suj:::} ;
: ":suj:aoj:" {:->aoj::} ;
; {:::} {>aoj:::} ;

;     verbo suporte 1 (keep)
```

```
: {v,backing,<obj<aoj~pred,<obj~obj:-<obj<aoj~pred,-<obj~obj,RSHIFT::}
":GET_SUJINFO,OBJ,BACKING1:obj:" ;
?L {v,backing,<obj<aoj~pred,<obj~obj:::} {:::} ;

;     verbo suporte 2 (take PRED with OBJ)

CL {v,backing,<obj~pred:-<obj~pred,<OBJ~PRED1,RSHIFT::} {:::} ;
? {v,backing,<obj~pred:::} {:::} ;

;     objeto (relativo ou não)

: {rel:RSHIFT,OBJ,-<obj:obj:} "<obj~obj:->obj::" ;
? {rel,<obj:::} {:::} ;
: ":RSHIFT,OBJ:obj:" {obj_,<obj~obj:->obj,LSHIFT_LOWP::} ;
? {:::} {>obj,obj_,<obj~obj:::} ;
: {>obj,_obj,<obj~obj,^STOP_REWINDING:RSHIFT,STOP_REWINDING::}
"[*]:REWIND::" ;
? {>obj,_obj,<obj~obj,^STOP_REWINDING:::} {:::} ;
: {_obj,<obj~obj:->obj,RSHIFT::} ":OBJ:obj:" ;
? {>obj,_obj,<obj~obj:::} {:::} ;

;     "vamos"

: "[vamos]:RSHIFT,vaux,artif::"
{imperative,psuj=1p,^backing,^vamos:vamos::} ;

;   expressando o passado

: {@past,^backing:-@past::} "[já]:adv,artif::" ;

;     expressando necessidade

: "[precisar]:RSHIFT,vaux,artif::" {@need,^backing:-@need::} ;

;   MÍNIMA PRECEDÊNCIA - movimento default para a direita.

R {:::} {:::} ;

; PÓS-EDIÇÃO

PR {RSHIFT:-RSHIFT::} {:::} ;
PR {:::} {RSHIFT:-RSHIFT::} ;

;     pós-edição: expressando atributos LIST

P: {@vocative:-@vocative,RSHIFT::} "[@voc]:attr,artif::" ;
P: {@imperative:-@imperative,RSHIFT::} "[@imp]:attr,artif::" ;
P: {@exclamation:-@exclamation,RSHIFT::} "[@excl]:attr,artif::" ;
P: {@gentampaobj=coisa-achatada:-@gentampaobj=coisa-achatada,RSHIFT::}
"[@gentampaobj=coisa-achatada]:attr,artif::" ;
P: {@gen=coisa-redonda:-@gen=coisa-redonda,RSHIFT::} "[@gen=coisa-
redonda]:attr,artif::" ;
P: {@pobj=3s:-@pobj=3s,RSHIFT::} "[@pobj=3s]:attr,artif::" ;
P: {@psuj=1p:-@psuj=1p,RSHIFT::} "[@psuj=1p]:attr,artif::" ;
```

```
P: {@enum:-@enum,RSHIFT::} "[@enum]:attr,artif::" ;

;     pós-edição: inserindo espaços em branco

PR {NEXT_PEND:-NEXT_PEND::} {:PEND::} ;
PL {^SHEAD,^PEND:::} {:::} ;
PR {SHEAD:::} {:PEND::} ;
P: {PEND:-PEND,RSHIFT::} "[ ]:NEXT_PEND,RSHIFT::" ;
```

Anexo VI – Relação de Relation Labels (RLs)

List of Relation Labels

UNL Specifications version 3 Edition 1, December 2002

agt	agent	a thing in focus which initiates an action
and	conjunction	a conjunctive relation between concepts
aoj	thing with attribute	a thing which is in a state or has an attribute
bas	basis	a thing used as the basis (standard) for expressing a degree
ben	beneficiary	an indirectly related beneficiary or victim of an event or state
cag	co-agent	a thing not in focus which initiates an implicit event which is done in parallel
cao	co-thing with attribute	a thing not in focus, as in a state in parallel
cnt	content	an equivalent concept
cob	effected co-thing	a thing which is directly effected by an implicit event done in parallel or an implicit state in parallel
con	condition	a non-focused event or state which conditions a focused event or state
coo	co-occurrence	a co-occurrent event or state for a focused event or state
dur	duration	a period of time during which an event occurs or a state exists
fmt	range	a range between two things
frm	origin	an origin of a thing
gol	goal/final state	the final state of an object or the thing finally associated with the object of an event
ins	instrument	the instrument to carry out an event
man	manner	the way to carry out an event or characteristics of a state
met	method	the means to carry out an event
mod	modification	a thing which restricts a focused thing
nam	name	a name of a thing
obj	effected thing	a thing in focus which is directly effected by an event or state
opl	effected place	a place in focus where an event takes effect
or	disjunction	a disjunctive relation between two concepts
per	proportion, rate of distribution	a basis or unit of proportion, rate of distribution
plc	place	the place where an event occurs, or a state is true, or a thing exists
plf	initial place	the place where an event begins or a state becomes true
plt	final place	the place where an event ends or a state becomes false
pof	part-of	a concept of which a focused thing is a part
pos	possessor	the possessor of a thing
ptn	partner	an indispensable non-focused initiator of an action
pur	purpose or objective	the purpose or objective of an agent of an event or the purpose of a thing

		which exists
qua	quantity	quantity of a thing or unit
rsn	reason	a reason why an event or a state happens
scn	scene	a virtual world where an event occurs, or a state is true, or a thing exists
seq	sequence	a prior event or state of a focused event or state
src	source/initial state	the initial state of an object or thing initially associated with the object of an event
tim	time	the time an event occurs or a state is true
tmf	initial time	the time an event starts or a state becomes true
tmt	final time	the time an event ends or a state becomes false
to	destination	a destination of a thing
via	intermediate place or state	an intermediate place or state of an event

The following relation is used only in the UNL KB or UW definition.

icl	included	a concept of which a focused concept is a proper subset
iof	instance of	an instance of a class
equ	equal	an acronym of an original word

Anexo VII – Relação de Attribute Labels (ALs)

List of Attribute Labels

UNL Specifications version 3 Edition 1, 20 February 2002

@ability	ability, capability of doing something	Ex) The child can 't walk yet. Ex) He can speak English but he can't write it very well.
@admire	admiring feeling of the speaker about something	Ex)
@affirmative	affirmation	Ex)
@although	something follows against [contrary to] or beyond expectation	Ex) Although he didn't speak, I felt a certain warmth in his manner.
@angle_bracket	< > is used	
@begin	beginning of an event or a state	Ex) It began to work again. work.@begin.@past
@blame	blameful feeling of the speaker about something	Ex) A sailor, and afraid of the sea!
@certain	certainty that something is true or happens	Ex) If Peter had the money, he would have bought a car.
@complete	finishing/completion of a (whole) event.	Ex) I've looked through the script. look.@entry.@complete
@conclusion	logical conclusion due to a certain condition	Ex) He is her husband; she is his wife.
@confirmation	confirmation	Ex) You won't say that, will you? Ex) It's red, isn't it? Ex) Then you won't come, right?
@consequence	logical consequence	Ex) He was angry, wherefore I left him alone.
@continue	continuation of an event	Ex) He went on talking. talk.@continue.@past
@contrast	contrasted UW	For instance, "but" in the examples below is used to introduce a word or phrase that contrasts with what was said before. Ex) It wasn't the red one but the blue one. Ex) He's poor but happy.
@custom	customary or repetitious	Ex) I used to visit [I would often go]

	action	there when I was a boy. visit.@custom.@past
@def	already referred	Ex) the book you lost
@discontented	discontented feeling of the speaker about something	Ex) (I'll tip you 10 pence.) But that's not enough!
@dissent	dissenting feeling of the speaker about something	Ex) But that's not true.
@double_parenthesis	(( )) is used	
@double_quote	" " is used	
@emphasis	emphasized UW	Ex) I do like it.
@end	end/termination of an event or a state	Ex) I have done it. do.@end.@present
@entry	entry or main UW of a sentence or a scope	Ex) He promised (entry of the sentence) that he would come(entry of the scope)
@exclamation	feeling of exclamation	Ex) kirei na! ("How beautiful (it is)!" in Japanese) Ex) Oh, look out!
@expectation	expectation of something	Ex) Children ought to be able to read by the age of 7. Ex) If you leave now, you should get there by five o'clock.
@experience	experience	Ex) Have you ever visited Japan? visit.@experience.@interrogation Ex) I have been there. visit.@exterience
@future	will happen in future	Ex) He will arrive tomorrow
@generic	generic concept	Ex) The dog is a faithful animal.
@grant	to give/get consent/permission to do something	Ex) Can I smoke in here? Ex) You may borrow my car if you like.
@grant-not	not to give consent to do something	Ex) You {mustn't/are not allowed to/may not} borrow my car.
@imperative	imperative	Ex) Get up! Ex) You will please leave the room.
@indef	non-specific class	Ex) There is a book on the desk.
@inevitable	logical inevitability that something is true or happens	Ex) There must be a mistake. Ex) They should be home by now.
@insistence	strong will to do something	Ex) He will do it, whatever you say.
@intention	intention about something	Ex) He shall get this money.

	or to do something	(Speaker's intention) Ex) We shall let you know our decision.
@interrogative	interrogation	Ex) Who is it?
@invitation	inducement to do something	Ex) Will / Won't you have some tea? Ex) Let's go, shall we?
@just	expresses an event or a state that has just begun or ended/been completed	Ex) He has just come. come.@complete.@just
@may	practical possibility that something is true of happens	Ex) It may be true. Ex) It could be.
@need	necessity of doing something	Ex) You need to finish thit work today.
@not	complement set	Ex) Don't be late!
@obligation	obligation to do something according to (quasi-) law, contract, or …	Ex) The vendor shall maintain the equipment in good repair.
@obligation-not	obligation not to do something, forbid to do something according to (quasi-) law, contract or …	Ex) Cars must not park in front of the entrance. Ex) No smoking
@ordinal	ordinal number	Ex) the 2nd door
@parenthesis	( ) is used	Ex) UNL (Universal Networking Language) cnt(UNL, Universal Networking Language.@parenthesis)
@past	happened in the past	ex) It was snowing yesterday
@pl	plural	Ex) These (this.@pl) are the wrong size.
@present	happening at present	ex) It's raining hard.
@progress	an event is in progress	Ex) I am working now. work.@progress.@present
@polite	polite feeling. Puts emphasis on a way of talking.	Ex) Could you (please)... Ex) If you could … I would …
@possible	logical possibility that something is true or happens	Ex) Anybody can make mistakes. Ex) If Peter had the money, he would buy a car.
@probable	(practical) probability that	Ex) That would be his mother.

	something is true or happens	Ex) He must be lying.
@qfocus	focused UW of a question	Ex) Are you painting the bathroom blue? To this question, the answer will be "No, I'm painting the LIVING-ROOM blue"
@rare	rare logical possibility that something is true or happens	Ex) If such a thing should happen, what shall we do? Ex) If I should fail, I will [would] try again.
@regret	regretful feeling of the speaker about something	Ex) It's a pity that he should miss such a golden opportunity.
@repeat	repetition of an event	Ex) It is so windy that the tree branches are knocking against the roof. knock.@entry.@present.@repeat
@request	request	Ex) Please don't forget…
@respect	respectful feeling. In many cases, some special words are used.	Ex) o taku ("(your) house" in Japanese) Ex) Good morning, sir.
@should	to do something as a matter of course	Ex) You should do as he says. Ex) You ought to start at once.
@single_quote	' ' is used	
@square_bracket	[ ] is used	
@state	final state or the existence of the object on which an action has been taken	Ex) It is broken. break.@state
@surprised	surprised feeling of the speaker about something	Ex) (He has succeeded!) But that's great!
@theme	instantiates an object from a different class	Ex)
@title	title	Ex)
@topic	topic	Ex) He(@topic) was killed by her. Ex) The girl(@topic) was given a doll. Ex) This doll(@topic) was given to the girl.
@unreal	unreality that something is true or happens	Ex) If we had enough money, we could buy a car. Ex) If Peter had the money, he could buy a car.
@vocative	vocative	Ex) Boys, be ambitious!
@will	will to do something	Ex) I'll write as soon as I can.

		Ex) We won't stay longer than two hours.
@wish	wishful feeling, to wish something is true or has happened	Ex) If only I could remember his name! (~I `do wish I could remember his name!) Ex) You might have just let me know.
@yet	expresses the feeling of something not yet begun, ended or completed, or expresses an event or a state that has not yet started or ended/been completed, together with @not.	Ex) I have not yet done it. do.@complete

Anexo VIII – Relação de palavras suspeitas de anáfora

aquela, aquelas, aquele, aqueles, aquilo, daquela, daquelas, daquele, daqueles, daquilo, dessa, dessas, desse, desses, desta, destas, deste, destes, disso, disto, essa, essas, esse, esses, esta, estas, este, estes, idem, isso, isto, mesma, mesmas, mesmo, mesmos, naquela, naquelas, naquele, naqueles, naquilo, nessa, nessas, nesse, nesses, nesta, nestas, neste, nestes, nisso, nisto, o, os, própria, próprias, próprio, próprios, semelhante, semelhantes, tais, tal, consigo, ela, elas, ele, eles, lha, lhas, lhe, lhes, lho, lhos, se, si, aonde, como, cuja, cujas, cujo, cujos, onde, quais, qual, quando, quanta, quantas, quanto, quantos, que, quem

Anexo IX – Documento LIST gerado pelo PUL∅

```
<xml>
<head>
[D:VAMOS COMBATER A DENGUE!, from C:\HERMETO\SAMPLES\74.TXT, at 17:11:00 24/4/2003, by
RICARDO, upd 27/05/2003]
</head>
<body>

<p>
<s>
{pt-br}
Vamos combater a dengue!
{/pt-br}
{unl}
obj(fight(icl>do).@exclamation.@entry.@imperative,dengue(icl>disease).@def)
agt(fight(icl>do).@exclamation.@entry.@imperative,we)
{/unl}
{list}
[ DENGUE vamos vencer{pobj:3s psuj:1p} ]{excl imp}
{/list}
</s>
</p>

<p>
<s>
{pt-br}
O mosquito da dengue nasce e se desenvolve em água parada
{/pt-br}
{unl}
mod(mosquito(icl>insect).@def,dengue(icl>disease).@def)
and(develop(icl>occur),be born(icl>occur).@entry)
mod(water(icl>matter),stagnant(mod<thing))
plc(be born(icl>occur).@entry,water(icl>matter))
obj(be born(icl>occur).@entry,mosquito(icl>insect).@def)
{/unl}
{list}
[ água largado DENGUE mosquito surgir desenvolver ]
{/list}
</s>
</p>

<p>
<s>
{pt-br}
Por este motivo, nós temos que manter a caixa d´água fechada!
{/pt-br}
{unl}
obj(keep(icl>be).@exclamation.@entry.@need,water tank(icl>container).@def)
aoj(closed(mod<thing),water tank(icl>container).@def)
aoj(keep(icl>be).@exclamation.@entry.@need,we)
man(keep(icl>be).@exclamation.@entry.@need,thus(icl>how))
{/unl}
{list}
[ por-isso caixa água precisar fechar-com-tampa{gentampaobj:coisa-achatada} ]{excl}
{/list}
</s>
```

```
</p>

<p>
<s>
{pt-br}
Tomar cuidado com vasos
{/pt-br}
{unl}
obj(take(icl>do).@entry,care)
obj(with(icl>how),flower pot(icl>container).@pl)
man(take(icl>do).@entry,with(icl>how))
{/unl}
{list}
[ cuidado{excl} vaso flor ]
{/list}
</s>
</p>

<p>
<s>
{pt-br}
garrafas, latas
{/pt-br}
{unl}
and(can(icl>container).@pl,bottle(icl>container).@entry.@pl)
{/unl}
{list}
[ garrafa ferro{gen:coisa-redonda} ]
{/list}
</s>
</p>

<p>
<s>
{pt-br}
pneus velhos!
{/pt-br}
{unl}
mod(tire(pof>automobile).@exclamation.@entry.@pl,old(mod<thing))
{/unl}
{list}
[ pneu velho ]{excl}
{/list}
</s>
</p>

<p>
<s>
{pt-br}
Tudo que acumule água!
{/pt-br}
{unl}
obj(gather(icl>do),water(icl>matter))
agt(gather(icl>do),that)
mod(all.@exclamation.@entry,that)
{/unl}
{list}
[ tudo água dentro ]{excl}
{/list}
```

```
</s>
</p>

<p>
<s>
{pt-br}
Se você tiver febre alta, dores musculares, dores de cabeça, dores nos joelhos, dores
nos cotovelos, e mal estar geral
{/pt-br}
{unl}
mod(fever(icl>disease),high(mod<thing))
mod(pain(icl>disease):01.@pl,muscle)
mod(pain(icl>disease):02.@pl,head(pof>body))
plc(pain(icl>disease):03.@pl,knee(pof>body).@def.@pl)
plc(pain(icl>disease):04.@pl,elbow(pof>body).@def.@pl)
mod(bad feeling(icl>disease),general(mod<thing))
and(bad feeling(icl>disease),pain(icl>disease):04.@pl)
and(pain(icl>disease):04.@pl,pain(icl>disease):03.@pl)
and(pain(icl>disease):03.@pl,pain(icl>disease):02.@pl)
and(pain(icl>disease):02.@pl,pain(icl>disease):01.@pl)
and(pain(icl>disease):01.@pl,fiever(icl>disease))
obj(feel(icl>be).@entry,fiever(icl>disease))
aoj(feel(icl>be).@entry,you)
man(feel(icl>be).@entry,if(icl>how))
{/unl}
{list}
[ SI você [ [ fiever(icl>disease) ] [ músculo dor ] [ dor-de-cabeça ] [ joelho dor ] [
cotovelo dor ] [ MAL corpo todo ] ]{enum} ]
{/list}
</s>
</p>

<p>
<s>
{pt-br}
procure serviços médicos
{/pt-br}
{unl}
mod(service.@pl,medical(mod<thing))
obj(search(icl>do).@entry,service.@pl)
{/unl}
{list}
[ procurar médico ]
{/list}
</s>
</p>

<p>
<s>
{pt-br}
eu falei que água é um perigo!
{/pt-br}
{unl}
aoj(danger.@indef,water(icl>matter))
obj(say(icl>do).@exclamation.@entry,danger.@indef)
aoj(say(icl>do).@exclamation.@entry,I)
{/unl}
{list}
[ eu falar água perigoso ]{excl}
```

```
{/list}
</s>
</p>

<p>
<s>
{pt-br}
água parada, Cascão!
{/pt-br}
{unl}
mod(water(icl>matter).@exclamation.@entry,stagnant(mod<thing))
and(Cascão(icl>person).@vocative,water(icl>matter).@exclamation.@entry)
{/unl}
{list}
[ água largado [ cascão ]{voc} ]{excl}
{/list}
</s>
</p>

<p>
<s>
{pt-br}
água parada!
{/pt-br}
{unl}
mod(water(icl>matter).@exclamation.@entry,stagnant(mod<thing))
{/unl}
{list}
[ água largado ]{excl}
{/list}
</s>
</p>

</body>
</xml>
```