# UNITEX 1.2

# MANUAL DO USUÁRIO


## Université Marne-la-Vallée

http://www-igm.univ-mlv.fr/~unitex
unitex@univ-mlv.fr

## Sebastien Paumier

Tradução para o português realizada em 2007 por :
Amanda Lopes Pietrobom, Ana Amélia Furtado Pinto, Ariadne Cristina Colombo Vocci, Bruno Miranda Andrade, Carolina Benedicto da Gama, Cibele Cristhina Santiago, Fernanda Silva Rando, Letícia Bonora Teles, Luciana Teodoro Minei, Luciane Donizeti Faustino, Melina Talon Mendes, Mirelli Caroline Pinheiro Silva, Miriã Granato Bernardes de Araújo, Renata Holdack,

sob supervisão das professoras :
Claudia Maria Xatara
Maria Cristina Parreira da Silva
Maria Emília Pereira Chanut
IBILCE- UNESP - São José do Rio Preto


N.B. Esta é uma primeira versão da tradução que ainda necessita de revisões. Caso queira divulgá-la, pede-se verificar se não existe uma versão mais recente na página web do Unitex.

# Introdução

O Unitex é um conjunto de softwares que permite processar os textos em línguas naturais utilizando recursos lingüísticos. Esses recursos se apresentam na forma de dicionários eletrônicos, de gramáticas e tabelas de léxico-gramática. É resultado de trabalhos iniciados no francês por Maurice Gross no "Laboratório de Automação Documental e Lingüística" (LADL). Esses trabalhos foram estendidos a outras línguas através da rede de laboratórios RELEX.

Os dicionários eletrônicos descrevem as palavras simples e compostas de uma língua associando um lema tanto a uma entrada quanto a uma série de códigos gramaticais, semânticos e flexionais. A presença desses dicionários constitui uma diferença maior em relação a outras ferramentas usuais de busca por padrões, pois pode-se fazer referências às informações que eles contêm e assim descrever amplas classes de palavras com padrões muito simples. Esses dicionários são reapresentados segundo o formalismo DELA e foram elaborados por equipes de lingüistas para várias línguas (francês, inglês, grego, italiano, espanhol, alemão, tailandês, coreano, polonês, norueguês, português, etc...).

As gramáticas são representações de fenômenos lingüísticos por redes de transições recursivas (RTN), um formalismo semelhante ao dos autômatos de estados finitos. Numerosos estudos evidenciaram a adequação dos autômatos aos problemas lingüísticos, tanto em morfologia quanto em sintaxe ou fonética. As gramáticas manipuladas pelo Unitex retomam este princípio, baseando-se em um formalismo ainda mais potente que os autômatos. Essas gramáticas são representadas por meio de grafos que o usuário pode facilmente criar e atualizar.

As tabelas de léxico-gramática são matrizes que descrevem as propriedades de determinadas palavras. Tais tabelas foram elaboradas para todos os verbos simples do francês, das quais elas descrevem as propriedades sintáticas.

Como a experiência tem mostrado que cada palavra tem um comportamento quase único, essas tabelas permitem informar a gramática de cada elemento do léxico, daí o nome de léxico-gramática. O Unitex permite construir gramáticas a partir de tais tabelas.

O Unitex é um motor que permite explorar esses recursos lingüísticos. Estas características técnicas são a portabilidade, a modularidade, a possibilidade de gerar línguas que possuem os sistemas de escritas particulares, como certas línguas asiáticas, e abertura, graças a uma Filosofia de Software Livre. Suas características lingüísticas são as que motivaram a elaboração dos recursos: a precisão, a exaustividade e a consciência da existência dos fenômenos de fixidez, principalmente no que se refere ao cadastramento das palavras compostas.

O capítulo 1 descreve a instalação e a inicialização do Unitex.
O capítulo 2 apresenta as diferentes etapas do processamento de um texto.
O capítulo 3 descreve o formalismo dos dicionários eletrônicos DELA assim como as diferentes operações que podem ser aplicadas nos mesmos.

Os capítulos 4 e 5 apresentam os diferentes meios de efetuar buscas por padrões nos textos. O capítulo 5 descreve em detalhes a utilização do editor de grafos.
O capítulo 6 é dedicado às diferentes utilizações possíveis das gramáticas. As particularidades de cada tipo de gramática são aqui apresentadas.
O capítulo 7 introduz o conceito de autômato do texto e descreve as particularidades desse objeto. Descreve igualmente as operações que podem ser efetuadas neste objeto, principalmente o levantamento de ambigüidades lexicais por meio do programa ELAG.
O capítulo 8 é constituído de uma introdução às tabelas de léxico-gramática, seguida de uma descrição do método que permite construir gramáticas a partir dessas tabelas.
O capítulo 9 descreve em detalhes os diferentes programas externos que constituem o Unitex.
O capítulo 10 fornece a descrição de todos os formatos de arquivos utilizados pelo sistema.
O leitor encontrará em anexo as licenças GPL e LGPL que protegem os códigos-fonte do Unitex, como a licença LGPLLR que cobre os dados lingüísticos distribuídos com o Unitex.

# Capítulo 1

# Instalação do Unitex

O Unitex é um sistema multi-plataformas capaz de funcionar muito bem tanto no Windows quanto no Linux ou MacOS. Este capítulo descreve a instalação e a inicialização do Unitex para cada um desses sistemas. Apresenta igualmente os procedimentos de introdução de novas línguas e de desinstalação.

## 1.1 Licenças

O Unitex é um software livre. Isso significa que as fontes dos programas são distribuídas com o software, e que qualquer um pode modificá-los e redistribuí-los. O código dos programas do Unitex está sob a licença LGPL ([24]), com exceção da biblioteca de manipulação de expressões regulares TER de Ville Laurikari ([36]), que está sob a licença GPL ([23]). A licença LGPL é mais permissiva que a licença GPL, pois ela permite utilizar o código LGPL nos softwares não-livres. Do ponto de vista do usuário, não há diferença, pois nos dois casos, o software pode ser livremente utilizado e distribuído.

Todos os dados lingüísticos distribuídos com o Unitex são submetidos à licença LGPLLR ([29]).

O texto completo das licenças GPL, LGPL e LGPLLR encontra-se nos anexos no fim deste manual.

## 1.2 Ambiente Java

O Unitex é composto de uma interface gráfica escrita em Java e de programas externos escritos em *C/C++*. Essa mistura de linguagens de programação permite que o aplicativo tenha rapidez de funcionamento e que seja portátil sob diferentes sistemas operacionais. Para poder utilizar a interface gráfica, é preciso instalar previamente um ambiente, comumente chamado de máquina virtual Java ou JRE (Java Runtime Environment).

Para funcionar em modo gráfico, o Unitex necessita de uma versão 1.4 (ou mais recente) do Java. Se tiver uma versão muito antiga do Java, o Unitex se bloqueará após escolher sua língua de trabalho. É possível baixar livremente a máquina virtual correspondente ao seu sistema operacional no site de Sun Microsystems [38] no seguinte endereço: `http://java.sun.com`. Se trabalhar com Linux ou MacOS, ou se utilizar uma versão de Windows que necessita de gerenciamento de usuários, será necessário pedir ao seu administrador do sistema para instalar o Java.

## 1.3 Instalação no Windows

Se desejar instalar o Unitex em uma máquina Windows com gerenciamento de usuários, é preferível pedir ao seu administrador para que o faça. Se você for o único usuário de sua máquina, você mesmo pode efetuar a instalação.

Descompactar o arquivo `Unitex_1.2.zip` (você pode baixar este arquivo no seguinte endereço: http://www-igm.univ-mlv.fr/~unitex) em um diretório Unitex que você previamente criou, de preferência em `Program Files`. Após a descompactação, o diretório `Unitex` conterá vários sub-diretórios, onde um se chamará App. Este último diretório contém um arquivo chamado `Unitex.jar`. Este arquivo é o executável Java, que aciona a interface gráfica. Basta dar duplo clique nesse arquivo para acionar o programa. Para facilitar a execução do programa, é aconselhável criar um atalho para esse arquivo na área de trabalho.

## 1.4 Instalação no Linux e MacOS

Para instalar o Unitex no Linux e no MacOS, é recomendável ser administrador do sistema. Descompactar o arquivo `Unitex_1.2.zip` em uma diretório chamado `Unitex`, por meio do seguinte comando:

```
unzip Unitex_1.2.zip -d Unitex
```

Entrar em seguida no diretório `Unitex/Src/C++`, e executar a compilação dos programas por meio do comando:

```
make install
```

Criar em seguida um alias no seguinte modelo:

```
alias unitex=´cd / ..../Unitex/App/ ; java -jar Unitex.jar´
```

## 1.5 Primeira utilização

Se trabalhar com o Windows, o programa pedirá que escolha uma pasta de trabalho pessoal, que você poderá modificar posteriormente em "Info>Preferences...>Directories". Para criar um diretório, clicar no ícone que representa uma pasta (ver figura 1.3).

No Linux e MacOS, o programa criará automaticamente um diretório `/unitex` na sua pasta `$HOME`. Este diretório lhe permitirá armazenar seus dados pessoais. Para cada língua que utilizar, o programa copiará a arborescência da língua em seu diretório pessoal, com exceção dos dicionários. Será possível assim modificar à vontade sua cópia de dados sem risco de danificar os dados do sistema.


Fig.1.1 – Primeira utilização no Windows


Fig.1.2 – Primeira utilização no Linux

## 1.6 Adição de novas línguas

Há duas maneiras de adicionar as línguas. Se desejar adicionar uma nova língua acessível a todos os usuários, é necessário copiar o diretório correspondente a essa língua no diretório `Unitex` do sistema, que necessita ter os direitos de acesso a este diretório (talvez seja preciso solicitar ao administrador do sistema para fazê-lo). Entretanto, se a língua interessar a apenas um usuário, ele pode copiar o diretório em questão no seu diretório pessoal. Poderá assim trabalhar com esta língua sem que ela seja proposta aos outros usuários.


Fig. 1.3 – Criação da pasta pessoal

## 1.7 Desinstalação

Qualquer que seja o sistema com o qual você trabalha, basta apagar o diretório `Unitex` para apagar todos os arquivos do sistema. No Windows, você deverá apagar em seguida o atalho para o `Unitex.jar` se você tiver criado um; o mesmo para o Linux ou MacOS se você tiver criado um alias.

# Capítulo 2

# Carregando um texto

Uma das principais funcionalidades do Unitex é a busca de expressões nos textos. Para isso, os textos devem passar por várias operações de pré-tratamento tais como a normalização das formas não ambíguas e a segmentação do texto em frases. Após efetuar essas operações, os dicionários eletrônicos são aplicados aos textos. Podem-se efetuar então as buscas nos textos aplicando-lhes as gramáticas.

Este capítulo descreve as diferentes etapas preparação dos textos.

## 2.1 Seleção da língua

Na inicialização do Unitex, o programa pede para escolher a língua com a qual trabalhará (ver figura 2.1). As línguas propostas são aquelas apresentadas no diretório sistema Unitex assim como aquelas eventualmente instaladas no seu diretório pessoal. Se utilizar uma língua pela primeira vez, o Unitex copia uma pasta dessa língua para o seu diretório pessoal, com exceção dos dicionários para economizar espaço em disco. Atenção! Se já possuir um diretório de usuário para uma determinada língua, o Unitex não tentará recopiar os dados do sistema. Assim, se uma atualização modificou um arquivo do dicionário, será necessário ou fazer uma atualização manual do arquivo no seu diretório de usuário, ou apagar seu diretório para a língua em questão e deixar para o Unitex o cuidado de recriá-la.

A escolha da língua permite indicar ao Unitex onde encontrar certos dados, como por exemplo, o arquivo alfabeto. Você pode a qualquer momento mudar a língua clicando no “Change Language...” no menu “Text”. Se você mudar de língua, o programa fechará, se houver, todas as janelas relativas ao texto atual. A barra do título da interface gráfica indicará a língua escolhida.

## 2.2 Formato dos textos

O Unitex manipula os textos Unicode. O Unicode é um padrão que descreve uma codificação universal dos caracteres. A cada caractere é atribuído um número único que permite representar os textos sem ter que considerar a códigos específicos das diferentes máquinas e/ou sistemas operacionais. Unitex utiliza uma representação codificada em dois bytes do padrão Unicode 3.0, chamado Unicode Little-Endian (para mais detalhes, ver [12]).

Fig.2.1 – Seleção da língua na inicialização do Unitex

Os textos fornecidos com o Unitex já estão em caracteres Unicode. Se tentar abrir um texto que não está no formato Unicode, o programa lhe recomendará convertê-lo automaticamente (ver figura 2.2). Esta conversão se baseia na língua corrente: se trabalhar em francês, o Unitex lhe recomendará converter seu texto[1] supondo que ele está codificado com uma página francesa de códigos. Por default, o Unitex recomenda tanto substituir o texto original, quanto renomear o arquivo de origem inserindo `.old` antes de sua extensão. Por exemplo, se há um arquivo ASCII nomeado `balzac.txt`, o processo de conversão criará uma cópia deste arquivo ASCII nomeado `balzac.old.txt`, e substituirá o conteúdo do `balzac.txt` pelo seu equivalente em Unicode.

Se o código proposto como default não for bom, ou se desejar renomear o arquivo de outro modo que não seja com sufixo `.old`, você pode utilizar o comando "Transcode Files" no menu "File Edition". Esse comando permite escolher o código de origem e de destino dos documentos a converter (ver figura 2.3). Por default, o código fonte proposto é aquele que corresponde à língua corrente, e o código de destino é o Unicode Little-Endian. É possível modificar essas escolhas selecionando qualquer código fonte ou de destino. Assim será possível, se desejar, converter os dados em outros códigos, como por exemplo, UTF-8, se desejar fazer páginas da web. O botão "Add Files" lhe permitirá selecionar os arquivos a converter. O botão "Remove Files" permite retirar da lista os arquivos selecionados por engano. O botão "Transcode" iniciará a conversão de todos os arquivos. Se um erro ocorre durante o processamento de um arquivo (por exemplo, um arquivo que já estivesse em Unicode), o processamento continua com o arquivo seguinte.


Fig.2.2 – Conversão automática de um texto não-Unicode

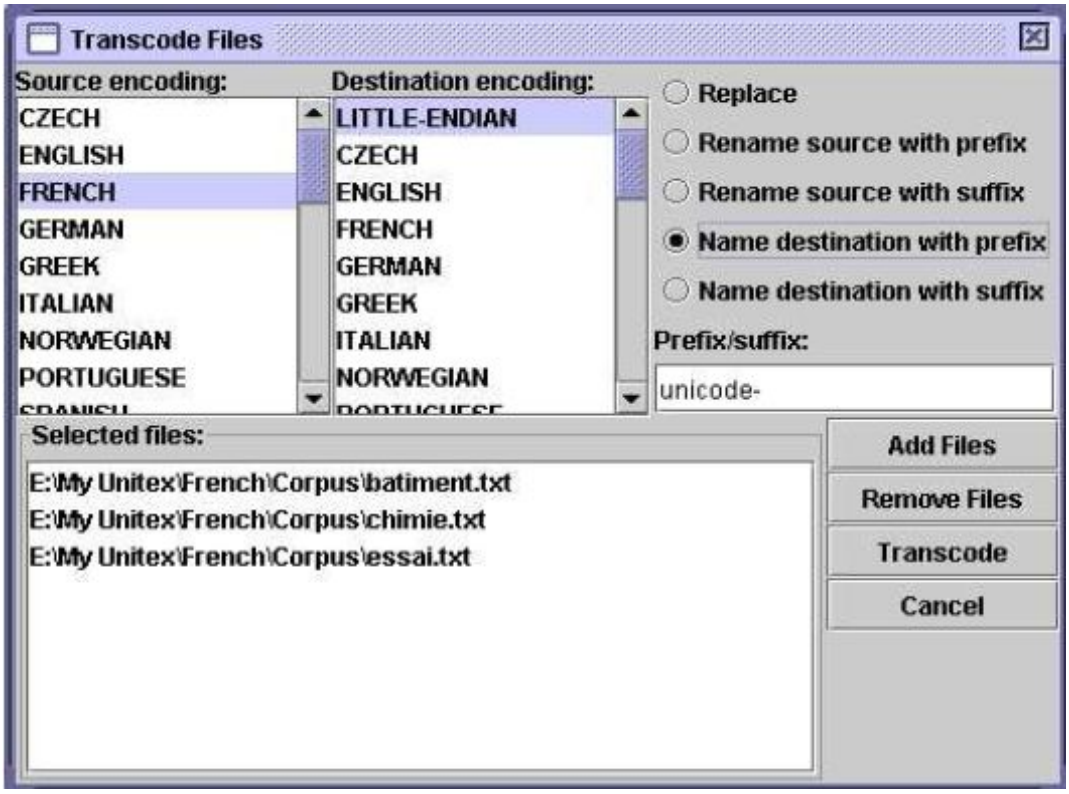



Fig.2.3 – Conversão de arquivos

Para obter um bom formato do texto, pode-se igualmente utilizar um processamento de texto como o software livre OpenOffice.org ([41]) ou Microsoft Word, e fazer um backup do seu documento no formato "Texte unicode". No Office XP, é necessário escolher o formato "Texto bruto (*.txt)" e em seguida selecionar o padrão "Unicode" na janela de configuração apresentada na figura 2.4.

Por default, o padrão proposto em um Pc é sempre Unicode Little-Endian. Os textos assim obtidos não contêm mais informações de formatação (fonte, cores, etc.) e estão prontos para serem utilizados com Unitex.


Fig.2.4 – Backup em Unicode no Office XP

## 2.3 Edição de textos

Há igualmente a possibilidade de utilizar o editor de texto integrado ao Unitex, acessível através do comando "Open..." do menu "File Edition". Este editor lhe oferece as funcionalidades de busca e substituição próprias aos textos e dicionários manipulados pelo Unitex. Para acessá-lo, clicar no ícone "Find" (binóculos). Então, aparecerá uma janela dividida em três abas. A aba "Find" corresponde às operações de busca habituais. Se abrir um texto segmentado em frases, haverá a possibilidade de fazer uma busca pelo número de frase na aba "Find Sentence". Enfim, a aba "Dictionary search", visível na figura 2.5, permite-lhe efetuar as operações próprias aos dicionários eletrônicos. Em particular, será possível efetuar uma busca especificando se ela deve estar na forma flexionada, o lema, nos códigos gramaticais e semânticos e/ou nos códigos flexionais. Assim, se quiser procurar todos os verbos que possuem o traço semântico `t`, marcando a transitividade, basta procurar `t` marcando "Grammatical code". Você obterá assim as entradas desejadas, sem ambigüidades com todas as outras ocorrências da letra `t`.

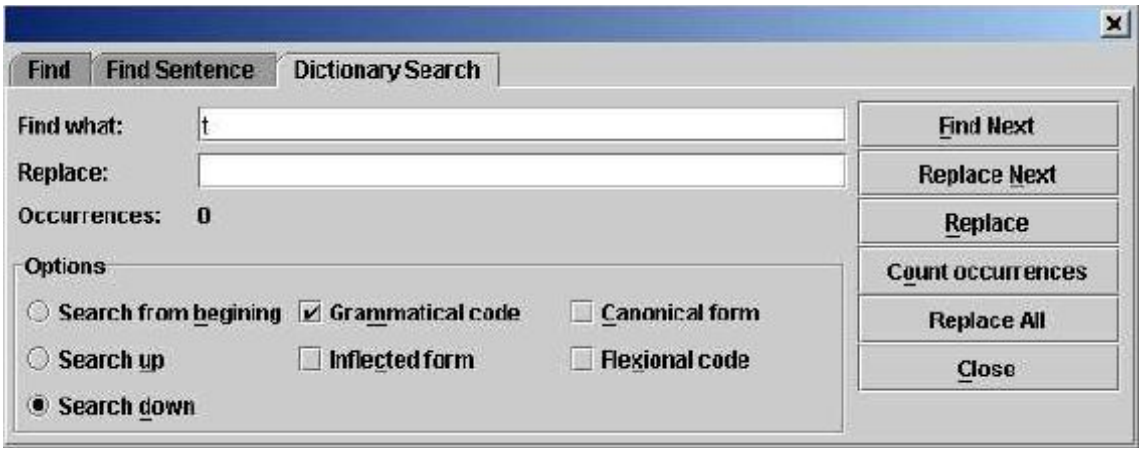



Fig.2.5 – Busca do traço semântico t em um dicionário eletrônico

## 2.4 Abertura de um texto

O Unitex propõe abrir dois tipos de arquivos texto. Os arquivos com a extensão `.snt` são os arquivos textos pré-processados pelo Unitex que estão prontos para serem manuseados pelas diferentes funções do sistema. Os arquivos com a extensão `.txt` são os arquivos de textos brutos. Para utilizar um texto, é necessário começar abrindo o arquivo .txt correspondente clicando em "Open..." no menu "Text".

Escolha o tipo de arquivo "Raw Unicode Texts" e selecione o seu texto. Os arquivos texto que ultrapassam 2 megabytes não são exibidos; a mensagem "`This file is too large to be displayed. Use a wordprocessor to view it.`" é exibida na janela. Esta observação diz respeito a todos os arquivos texto (lista das unidades lexicais, dicionários, etc.). Para modificar este limite, vá ao menu "Info>Preferences", e modifique o valor "Maximum Text File Size" na aba "Text Presentation" (ver figura 4.7, página 62).

## 2.5 Pré-processamento do texto

Uma vez selecionado o texto, o Unitex propõe pré-processá-lo. O pré-processamento do texto consiste em aplicar-lhe as seguintes operações: normalização de separadores, segmentação em unidades lexicais, normalização de formas não ambíguas, segmentação em frases e aplicação dos dicionários. Se recusar o pré-processamento, o texto será sempre normalizado e segmentado em unidades, pois essas operações são indispensáveis para o funcionamento do Unitex. Será possível efetuar o pré-processamento mais tarde, clicando em "Preprocess text..." no menu "Text". Se aceitar o pré-processamento, o Unitex mostrará os parâmetros na janela da figura 2.8.

A opção "Apply FST2 in MERGE mode" serve para efetuar a segmentação do texto em frases. A opção "Apply FST2 in REPLACE mode" é utilizada para efetuar substituições no texto, na maioria das vezes a normalização de formas não ambíguas. A opção "Apply All defaut Dictionaries" permite aplicar ao texto os dicionários no formato DELA (Dicionários Eletrônicos do LADL). A opção "Analyse unknown words as free compound words" é utilizada em norueguês para analisar corretamente as palavras compostas livres formadas por junção de palavras simples. Enfim, a opção

"Construct Text Automaton" é utilizada para construir o autômato do texto. Por default, esta opção é desativada, pois consome muita memória e ocupa muito espaço em disco se o texto for muito grande. A construção do autômato do texto será abordada no capítulo 7.

NOTA: Se clicar em "Cancel but tokenize text", o programa efetuará mesmo assim a normalização dos separadores e a segmentação em unidades lexicais; clicar em "Cancel and close text" para anular completamente a operação.

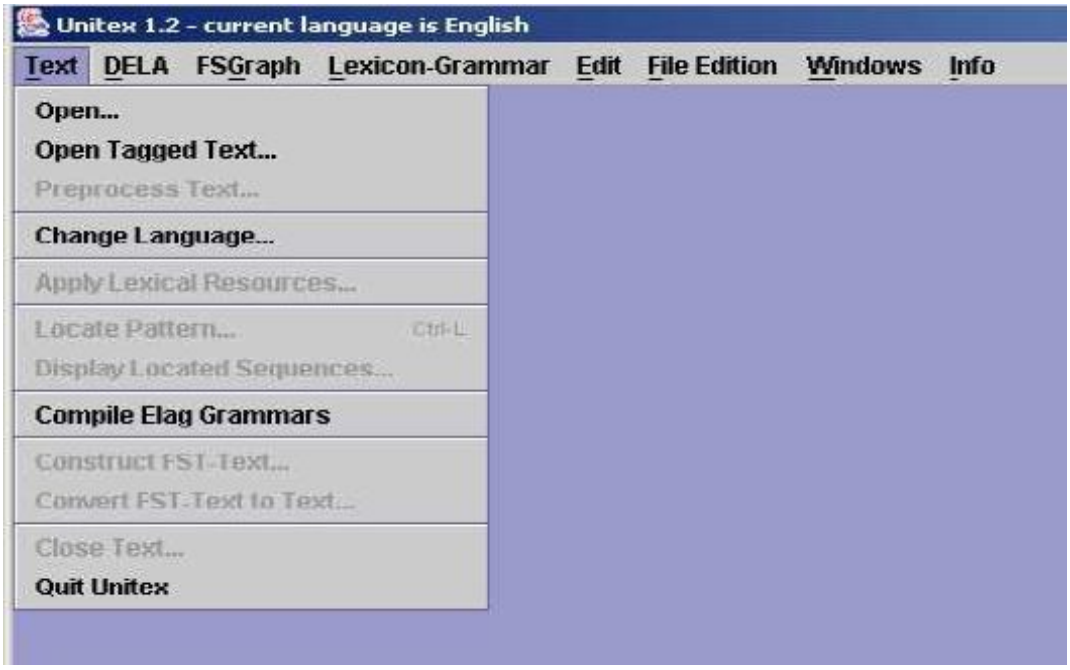



Fig. 2.6 – Menu Text

### 2.5.1 Normalização de separadores

Os separadores usuais são o espaço, a tabulação e o retorno à linha (Enter). Pode-se encontrar vários separadores consecutivos nos textos, mas como isso não tem nenhuma utilidade para uma análise lingüística, normaliza-se os separadores de acordo com as seguintes regras:

- toda seqüência de separadores contendo ao menos um "Enter" será substituída por um único "Enter".

- toda outra seqüência de separadores será substituída por um espaço.

A distinção entre espaço e "Enter" é conservada nesta etapa pois a presença do "Enter" pode influenciar a segmentação do texto em frases. O resultado da normalização de um arquivo `meu_texto.txt` é um arquivo situado no mesmo diretório que o `.txt` e onde o nome é `meu_texto.snt`.

NOTA: quando é feito um pré-processamento de um texto a partir da interface gráfica, um diretório chamado `meu_texto_snt` é criado imediatamente após a normalização. Este diretório, chamado diretório do texto, conterá todos os dados relativos a este texto.

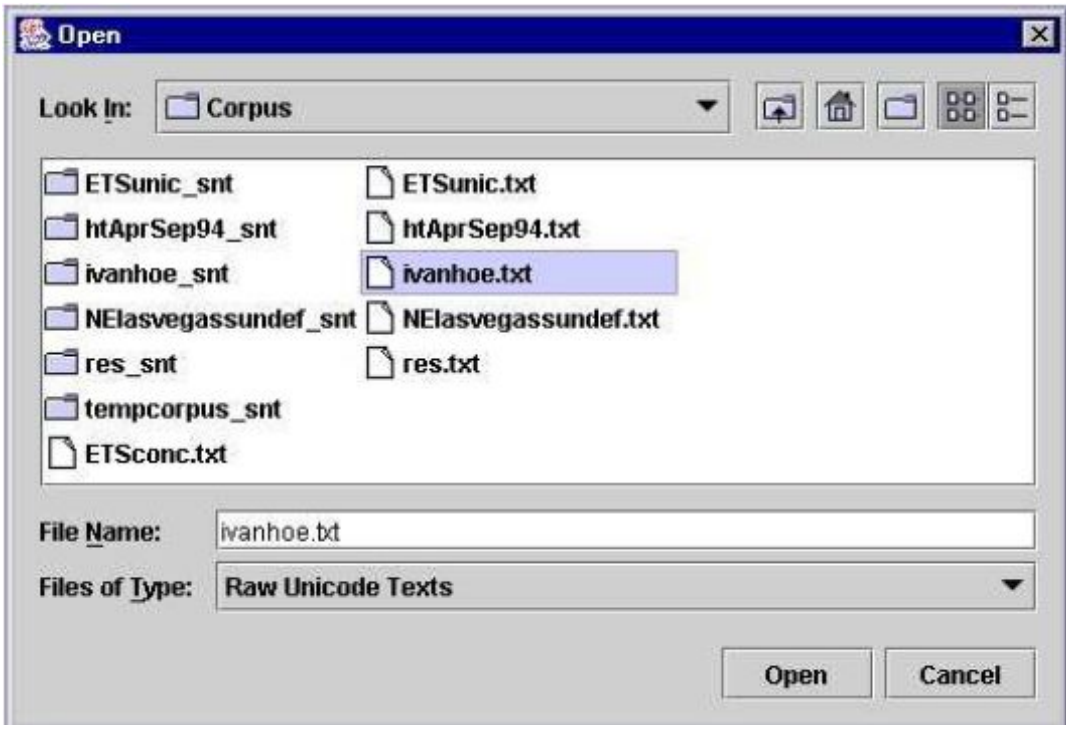



Fig. 2.7 – Abertura de um texto Unicode

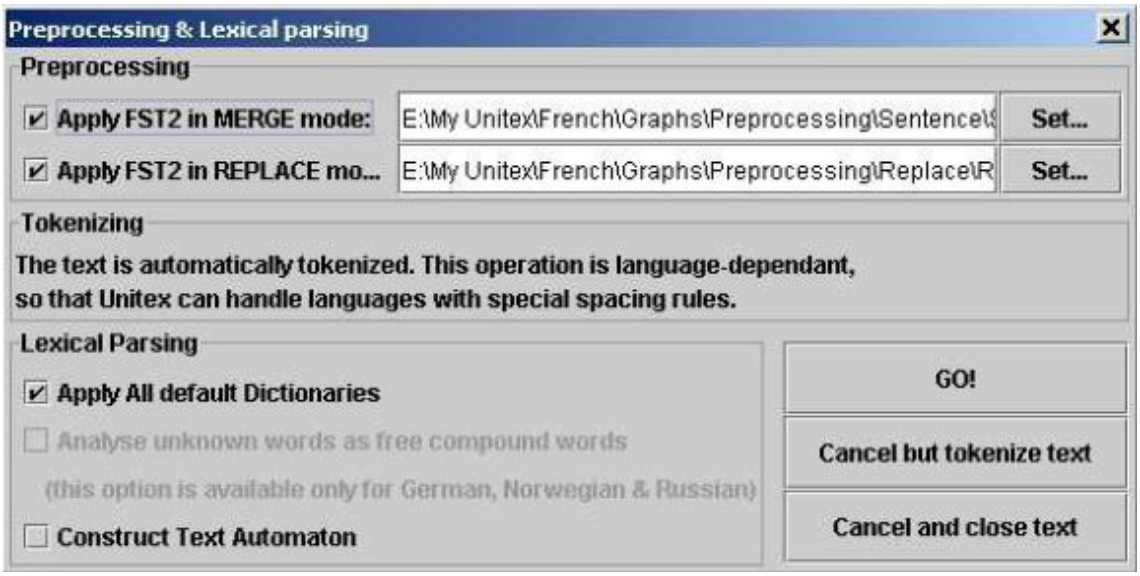



Fig. 2.8 – Janela de pré-processamento

### 2.5.2 Segmentação em frases

A segmentação em frases é uma etapa importante do pré-processamento, pois possibilitará definir as unidades de processamento lingüístico. Essa segmentação será utilizada pelo programa de construção do autômato do texto. Ao contrário do que se possa pensar, a busca dos limites de frases não é um problema trivial. Consideremos o seguinte texto:

*A família chamou o Dr. Martim com urgência.*

O ponto que segue *Dr* é seguido de uma palavra que começa por uma letra maiúscula; ele poderia ser considerado como um ponto de fim de frase, o que seria incorreto. A fim de evitar problemas desse gênero, devidos à ambigüidade dos símbolos de pontuação, utilizam-se as gramáticas que descrevem os diferentes contextos em que podem aparecer os limites da frase. A figura 2.9 mostra um exemplo de gramática de segmentação em frases.

Quando o caminho da gramática reconhece uma seqüência no texto e este caminho produz o símbolo separador de frases `{S}`, este símbolo é inserido no texto. Assim, um caminho da gramática da figura 2.9 reconhece a seqüência composta de um ponto de interrogação e de uma palavra que começa por uma letra maiúscula, e insere o símbolo `{S}` entre o ponto de interrogação e a palavra seguinte. O seguinte texto:

*Que horas são? Oito horas.*

torna-se:

*Que horas são? {S} Oito horas.*

Uma gramática de segmentação pode manipular os seguintes símbolos especiais:

- <E>: palavra vazia, ou epsilon. Reconhece a seqüência vazia;
- <MOT>: reconhece qualquer seqüência de letras;
- <MIN>: reconhece qualquer seqüência de letras minúsculas;
- <MAJ>: reconhece qualquer seqüência de letras maiúsculas;
- <PRE>: reconhece qualquer seqüência de letras que comece por uma letra maiúscula;
- <NB>: reconhece qualquer seqüência de números (1234 é reconhecida mas 1 234, não)
- <PNC>: reconhece os sinais de pontuação **; , ! ? :** assim como os pontos de exclamação e de interrogação invertidos do espanhol e outros sinais de pontuação asiáticos;
- <^>: reconhece um "Enter";
- #: interdita a presença de espaço.

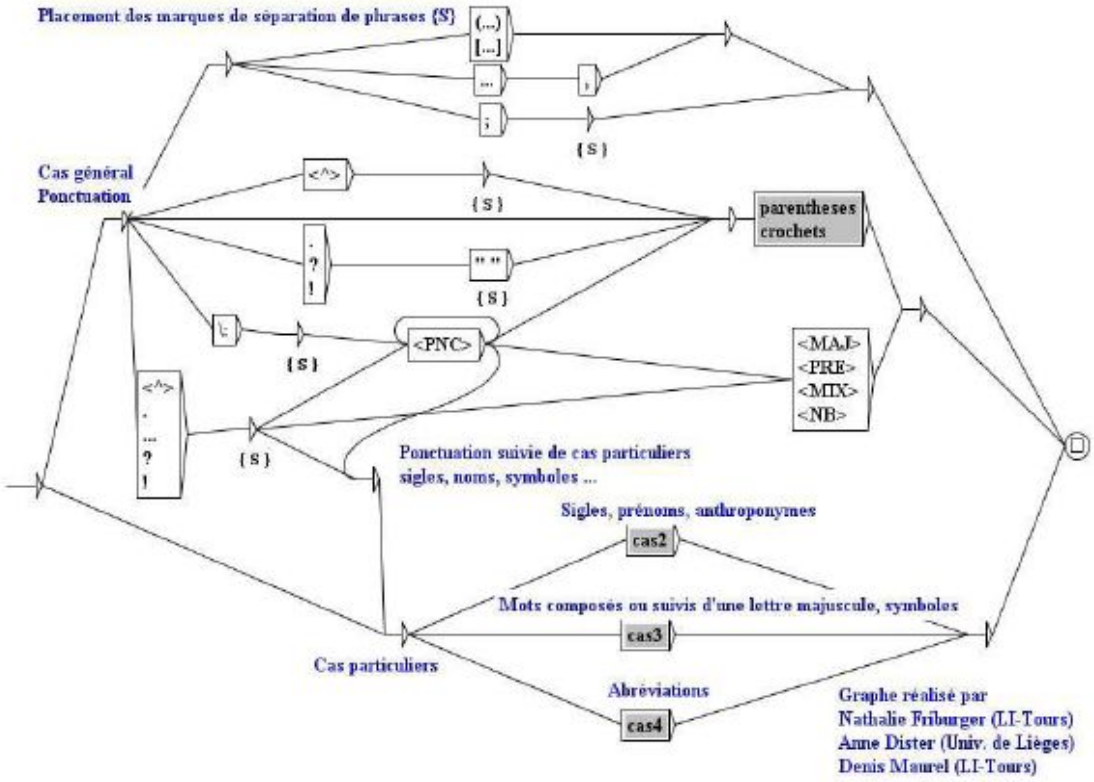



Fig. 2.9 – Gramática de segmentação em frases do francês

Por definição, o espaço é facultativo entre duas caixas. Se quiser interditar a presença desse separador, é necessário utilizar o símbolo especial `#`. Mas se você quiser forçar a presença do espaço, você deve utilizar a seqüência “ ”. As letras minúsculas e maiúsculas são definidas por um arquivo alfabeto (ver capítulo 10). Para mais detalhes sobre os grafos, ver capítulo 5. Para mais detalhes sobre a segmentação de um texto em frases, ver [16]. A gramática utilizada chama-se `Sentence.fst2` e se encontra no seguinte diretório:

```
/(répertoire personnel)/(langue)/Graphs/Preprocessing/Sentence
```

A aplicação dessa gramática a um texto se dá graças ao programa `Fst2Txt` em modo MERGE. Isso significa que as saídas produzidas pela gramática, com a presença do símbolo `{S}`, são inseridas nos textos. Esse programa tem como entrada um arquivo `.snt` e o modifica.

### 2.5.3 Normalização de formas não ambíguas

Certas formas presentes nos textos podem ser normalizadas (por exemplo, a seqüência *l'on* é equivalente à forma *on*). Cada usuário pode, portanto, realizar substituições em função de suas necessidades. No entanto, é preciso estar atento para que as formas normalizadas não sejam ambíguas ou para que o desaparecimento da ambigüidade ocorra sem conseqüências para a aplicação planejada. Se optar por substituir a forma *audit* por *à le-dit*, a frase:

*La cour a procédé à un audit des comptes de cette société.*

será substituída pela frase incorreta :

*La cour a procédé à un à le-dit des comptes de cette société.*

É preciso, pois, ser bastante prudente quando se manipula a gramática de normalização.

É preciso, da mesma forma, ficar atento aos espaços. Pois, se substituir *c'* por *ce* não seguido de espaço, a frase:

*Est-ce que c'était toi ?*

será substituída pela seqüência incorreta:

*Est-ce que ceétait toi ?*

Os símbolos aceitos pelas gramáticas de normalização são os mesmos que os permitidos nas gramáticas de segmentação em frases. A gramática utilizada chama-se `Replace.fst2` e encontra-se no seguinte diretório:

`/(répertoire personnel)/ (langue)/ Graphs/Preprocessing/Replace`

Como na segmentação em frases, essa gramática é utilizada com o programa `Fst2Txt`, porém, nesse caso, no modo REPLACE, o que significa que as entradas reconhecidas pela gramática são substituídas pelas seqüências produzidas por ela própria. Pode-se observar na figura `2.10` uma gramática que normaliza as contrações verbais em inglês.

### 2.5.4 Segmentação do texto em unidades lexicais

Certas línguas, em particular as línguas asiáticas, usam separadores de modo diferente das línguas ocidentais; os espaços podem ser proibidos, opcionais ou obrigatórios. Para melhor gerenciar essas particularidades, o Unitex recorta os textos de acordo com a língua. Assim, línguas como o francês são tratadas de acordo com o seguinte princípio:

Uma unidade lexical pode ser:

- o separador de frases `{S}`;
- o marcador `{STOP}`. Ao contrário do que ocorre com o separador de frases `{S}`, o marcador `{STOP}` JAMAIS pode ser reconhecido, de maneira alguma, por uma gramática. Este marcador particular pode ser utilizado para delimitar os elementos em um corpus. Por exemplo, se um corpus for constituído por notícias separadas por `{STOP}`, essa separação evita que uma gramática possa acidentalmente reconhecer uma seqüência que sobreponha o fim de uma notícia e o começo da seguinte;

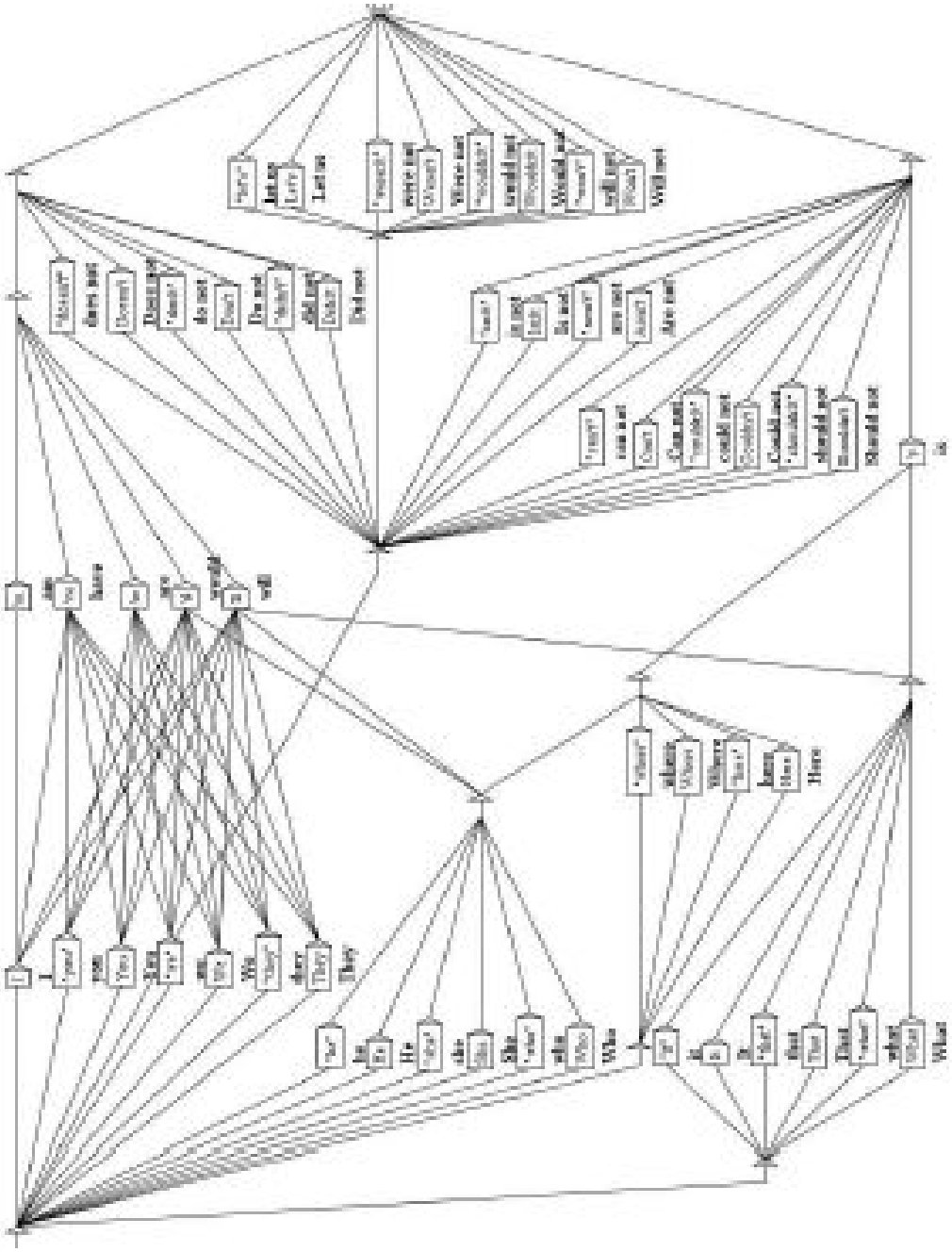


FIG. 2.10 Gramática de normalização de formas verbais em inglês

- uma etiqueta lexical `{aujourd'hui,.ADV}`;

- uma seqüência contígua de letras (sendo as letras definidas pelo arquivo do alfabeto da língua);
- um caractere que não seja uma letra; se esse caractere for uma quebra de página, ele é substituído por um espaço.

Para as outras línguas, o recorte é realizado caractere por caractere, com exceção do separador de frases `{S}`, o marcador `{STOP}` e as etiquetas lexicais. Este recorte básico garante o funcionamento do UNITEX, mas limita a otimização das operações de busca por padrões. Qualquer que seja o modo de recorte, as quebras de páginas presentes em um texto são substituídas por espaços. Esse recorte é realizado pelo programa `Tokenize`. Esse programa produz diversos arquivos, armazenados no diretório do texto:

- `Tokens.txt` contém a lista das unidades lexicais na ordem em que foram encontradas no texto;
- `Text.cod` contém uma tabela de números inteiros; cada número correspondendo ao índice de uma unidade lexical no arquivo `tokens.txt`;
- `Tok_by_freq.txt` contém a lista das unidades lexicais organizada por ordem de freqüência;
- `Tok_by_alph.txt` contém a lista das unidades lexicais organizada por ordem alfabética;
- `Stats.n` contém algumas estatísticas sobre o texto.

O recorte do texto:

*Un sou c'est un sou.*

origina a lista de unidades lexicais seguintes: *Un* ESPAÇO *sou c'est un sou*.

Pode-se observar que se considerou a caixa (*Un* e *un* são duas unidades distintas), mas que cada unidade é codificada somente uma vez. Numerando essas unidades de 0 a 7, esse texto pode ser representado pela seqüência de números descrita na tabela seguinte:

| Índice | 0 | 1 | 2 | 1 | 3 | 4 | 5 | 1 | 6 | 1 | 2 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Unidade lexical correspondente | *Um* | | *sou* | | *c* | *'* | *est* | | *un* | | *so u* | . |

TAB. 2.1 – Representação do texto *Un sou c'est un sou.*

Para mais detalhes, ver o capítulo `10`.

### 2.5.5 Aplicação de dicionários

A aplicação de dicionários consiste na construção do subconjunto dos dicionários que contêm somente as formas presentes no texto. Assim, o resultado da aplicação dos

dicionários do francês no texto *Igor mange une pomme de terre* produz o seguinte dicionário de palavras simples:

```
e,.DET+z1
de,.PREP+z1
de,.XI+z1
mange,manger.V+z1:P1s:P3s:S1s:S3s:Y2s
pomme,.A+z1:ms:fs:mp:fp
pomme,.N+z1:fs
pomme,pommer.V+z3:P1s:P3s:S1s:S3s:Y2s
terre,.N+z1:fs
terre,terrer.V+z1:P1s:P3s:S1s:S3s:Y2s
une,.N+z1:fs
une,un.DET+z1:fs
```

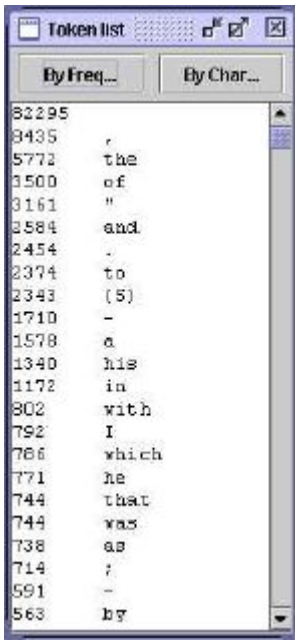



Fig. 2.11 – Unidades lexicais de um texto em inglês ordenadas por freqüência

enquanto que o dicionário de palavras compostas contém a única entrada:

```
pomme de terre,.N+z1:fs
```

A seqüência *Igor*, não sendo nem uma palavra simples do francês, nem uma parte de uma palavra composta, foi considerada uma palavra desconhecida. A aplicação de dicionários é realizada com o programa `Dico`. Os três arquivos produzidos (`dlf` para as palavras simples, `dlc` para as palavras compostas e `err` para as palavras desconhecidas) são colocados no diretório do texto. Chamam-se dicionários do texto os arquivos `dlf` e `dlc`. Tendo sido realizada a aplicação dos dicionários, o Unitex exibe em uma janela, em ordem alfabética, as palavras simples, compostas e desconhecidas encontradas. A figura `2.12` mostra os resultados para um texto francês.

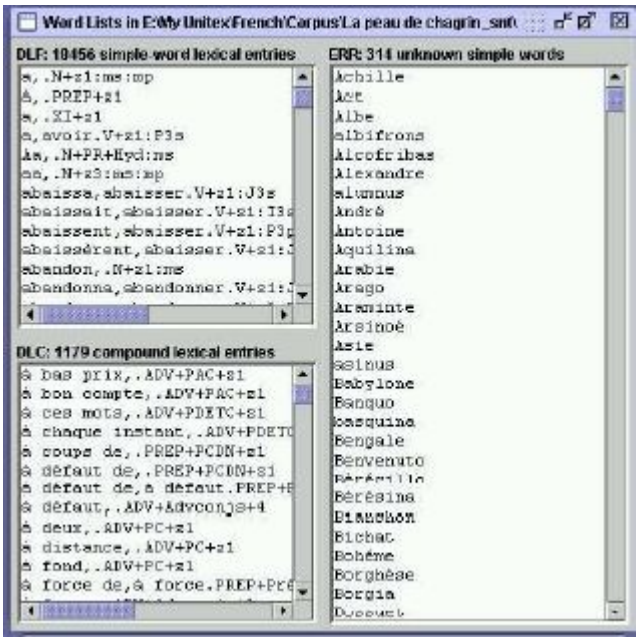



FIG. 2.12 – Resultados da aplicação dos dicionários em um texto francês

É possível, também, aplicar dicionários fora do pré-processamento do texto. Para isso, é necessário clicar em "Apply Lexical Resources..." no menu "Text". O Unitex exibe então uma janela (ver figura 2.13) que permite escolher a lista dos dicionários a serem aplicados.

A lista "User resources" enumera todos os dicionários `.bin` e `.fst2` presentes no diretório `(langue)/Dela` do usuário.

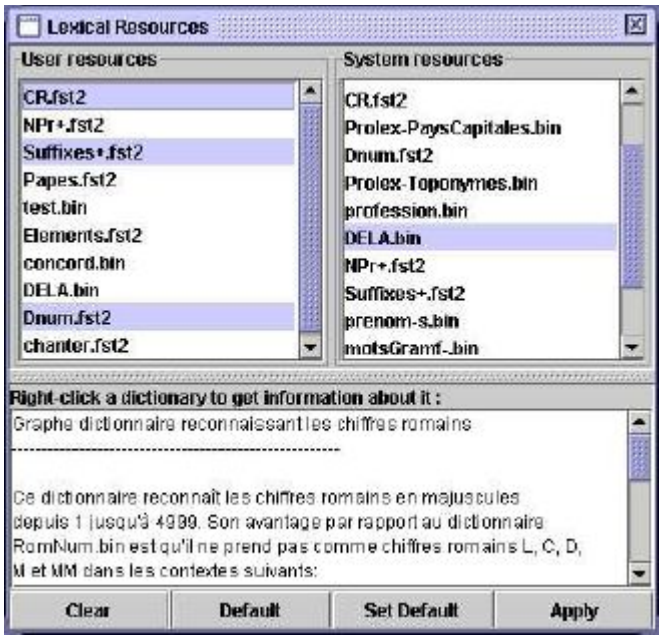



Fig. 2.13 Parâmetro da aplicação dos dicionários

Os dicionários do sistema são listados no quadro chamado "System resources". Utilizar <Ctrl+ clique> para selecionar vários dicionários. A tecla "Set Default" permite definir a seleção atual de dicionários como seleção por definição. Essa seleção por

definição será utilizada na ocasião do pré-processamento se for escolhida a opção "Apply All default Dictionaries". Se clicar com botão direito sobre o nome do dicionário, aparecerá no quadro inferior a sua documentação, caso exista.

### 2.5.6 Análise das palavras compostas livres em alemão, norueguês e russo.

Em certas línguas como o norueguês, é possível formar palavras compostas livres unindo seus elementos. Por exemplo, a palavra *aftenblad* que significa *jornal da noite* é obtida por meio da combinação entre as palavras *aften (noite)* e *blad (jornal)*. O programa `PolyLex` ([44]) explora a lista das palavras desconhecidas após a aplicação dos dicionários no texto e procura analisar cada uma dessas palavras como uma palavra composta. Se uma palavra tiver pelo menos uma análise, é retirada da lista das palavras desconhecidas e as linhas de dicionários produzidas para esta palavra são somadas ao dicionário das palavras simples do texto.

## 2.6 Abertura de um texto etiquetado

Um texto etiquetado é um texto que contém entradas lexicais entre chaves como por exemplo:

*I do not like the {square bracket,. N} sign! {S}*

Esses *tags* permitem retirar as ambigüidades não permitindo qualquer outra interpretação. No exemplo anterior, não será possível reconhecer *square bracket* como uma combinação de duas palavras simples.

Entretanto, a presença desses *tags* pode causar problemas na aplicação dos grafos de pré-processamento. O usuário dispõe, portanto, do comando "Open Tagged Text" no menu "Text", com o auxílio do qual pode abrir um texto que contenha *tags* sem que os grafos de pré-processamento tenham sido aplicados, como pode ser observado na figura `2.14`.

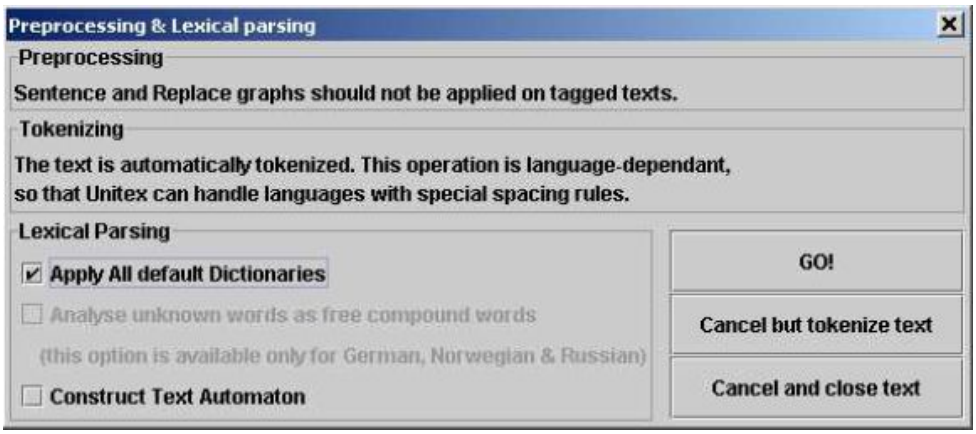



FIG. 2.14 Pré-processamento de um texto etiquetado

# Capítulo 3

# Dicionários

## 3.1 Os dicionários DELA

Os dicionários eletrônicos utilizados pelo Unitex utilizam o formalismo dos DELA (Dicionários Eletrônicos do LADL). Esse formalismo permite descrever as entradas lexicais simples e compostas de uma língua associando-lhes, de modo opcional, informações gramaticais, semânticas e flexionais. Distinguem-se dois tipos de dicionários eletrônicos. O utilizado com maior freqüência é o dicionário de formas flexionadas, chamado DELAF (DELA de formas Flexionadas) ou ainda DELACF (DELA de formas Compostas Flexionadas), quando se tratar de um dicionário de palavras compostas. O segundo tipo é o dicionário de formas não-flexionadas chamado DELAS (DELA de formas Simples) ou DELAC (DELA de formas Compostas). Os programas do Unitex não fazem distinção entre os dicionários de formas simples e compostas. Serão utilizados, portanto, os termos DELAF e DELAS para designar os dois tipos de dicionários, sejam as suas entradas simples, compostas ou mistas.

### 3.1.1 Formato dos DELAF

#### Sintaxe de uma entrada

Uma entrada de um DELAF é uma linha de texto terminada com uma quebra de página que respeita o seguinte esquema:

```
mercantiles,mercantile.A+z1:mp:fp/ceci est un exemple
```

Os diferentes elementos que formam essa linha são os seguintes:

- `Mercantiles` é a forma flexionada da entrada. Essa forma flexionada é obrigatória;
- `Mercantile` é a forma canônica da entrada. Para substantivos e adjetivos trata-se, de um modo geral, da forma no masculino singular; para verbos, a forma canônica é o infinitivo. Essa informação pode ser omitida, como no exemplo seguinte:

  ```
  boîte à merveilles,.N+z1:fs
  ```

Então, isso significa que a forma canônica é idêntica à forma flexionada. A forma canônica fica separada da forma flexionada por uma vírgula;

- `A+z1` é a seqüência de informações gramaticais e semânticas. No exemplo, `A` designa um adjetivo e `z1` indica que se trata de uma palavra comum (ver tabela `3.2`). Todas as entradas devem conter pelo menos um código gramatical ou

semântico, separado da forma canônica por um ponto. Se houver mais códigos, estes devem ser separados pelo caractere verb`+` ;

- `:mp:fp` é a seqüência de informações flexionais. Essas informações descrevem o gênero, o número, os tempos e modos das conjugações, as declinações para as línguas declinativas, etc. Essas informações são opcionais. Um código flexional é composto por um ou mais caracteres, cada um codificando uma informação diferente. Os códigos flexionais devem ser separados pelo caractere : . No exemplo, `m` significa masculino, `p` plural e `f` feminino (ver tabela `3.3`). O caractere : se interpreta como um OU lógico. `:mp:fp` significa, então, "masculino plural" ou "feminino plural". Como cada caractere corresponde a uma informação, é inútil utilizar várias vezes o mesmo caractere. Assim, codificar o particípio passado com o código `:PP` seria estritamente equivalente a utilizar somente `:P`.

- `/ ceci est un exemple` é um comentário. Os comentários são opcionais e devem ser introduzidos pelo caractere `/`. Os comentários são suprimidos quando os dicionários são compactados.

OBSERVAÇÃO IMPORTANTE: é possível utilizar o ponto e a vírgula em uma entrada de dicionário. Para isso, é necessário desabilitá-los com o caractere de escape \:

```
3 \, 1415, PI. NÚMERO
Organization des Nations Unies, O \. N\. U\.. Sigla
```

ATENÇÃO : em uma linha de dicionário, todo caractere é considerado. Por exemplo, se forem introduzidos espaços, eles serão considerados como parte integrante das informações. Na linha seguinte:

```
Gît, gésir. V+z1:P3s/ver ci-gît
```

o espaço que precede o caractere / será considerado como parte de um código flexional de 4 caracteres composto por `P, 3, s` e por um espaço.

É possível inserir linhas de comentários em um dicionário DELAF ou DELAS, fazendo iniciar a linha com o caractere / . Exemplo:

```
/ A entrada nominal por 'par' é um termo do golf em francês
Par, . N+z3:ms
```

**Palavras compostas com espaço ou hífen**

Certas palavras compostas como *grand-mère* podem ser escritas com espaços ou com hífens. Para evitar ter que duplicar todas as entradas, é possível utilizar o caractere =. No momento da compactação do dicionário, o programa `Compress` verifica para cada linha se a forma flexionada ou a forma canônica contém o caractere = não protegido pelo caractere de escape \. Se for o caso, o programa substitui a entrada por duas

entradas: uma em que o caractere = é substituído por um espaço e outra em que ele é substituído por um hífen. Assim, a entrada a seguir:

```
grand=mères,grand=mère.N:fp
```

é substituída pelas duas linhas seguintes :

```
grand mères,grand mère.N:fp
grand-mères,grand-mère.N:fp
```

NOTA : Se desejar escrever uma entrada que contenha o caractere =, é necessário desabilitá-lo com o caractere de escape \, como no exemplo seguinte :

```
E\=mc2, . FÓRMULA
```

Essa operação de substituição ocorre por ocasião da compactação do dicionário. Uma vez o dicionário compactado, os sinais = desabilitados são substituídos por simples =. Assim, se um dicionário contendo as seguintes linhas for compactado:

```
E\=mc2, . FÓRMULA
grand=mère,.N:fs
```

E for aplicado ao texto :

*Ma grand-mère m'a expliqué la formule E=mc2.*

Obtém-se, no dicionário de palavras compostas do texto, as seguintes linhas:

```
E=mc2,.FORMULE
grand-mère,.N:fs
```

**Fatoração das entradas**

Muitas entradas que apresentam as mesmas formas flexionada e canônica podem ser agrupadas em uma única entrada se tiverem os mesmos códigos gramaticais e semânticos. Isso permite, dentre outros, agrupar as conjugações idênticas para um mesmo verbo:

```
glace,glacer.V+z1:P1s:P3s:S1s:S3s:Y2s
```

Se as informações gramaticais e semânticas forem diferentes, é preciso criar entradas distintas:

```
glace,.N+z1:fs
glace,glacer.V+z1:P1s:P3s:S1s:S3s:Y2s
```

Algumas entradas, com códigos gramaticais e semânticos comuns, podem apresentar sentidos diferentes, como é o caso da palavra *poêle* que designa tanto um

aparelho de aquecimento e um véu no masculino quanto um instrumento de cozinha no feminino. Pode-se, então, distinguir as entradas nesse caso:

```
poêle,.N+z1:fs/ intrumento para fritar
poêle,.N+z1:ms/ véu, mortalha; aparelho de aquecimento
```

OBSERVAÇÃO : na prática, essa distinção causa somente o aumento do número de entradas do dicionário. Os diferentes programas que compõem o Unitex fornecerão exatamente os mesmos resultados se reunirmos essas entradas em:

```
poêle,.N+z1:fs:ms
```

O interesse desta distinção fica, então, a cargo dos dicionaristas.

### 3.1.2 Formato dos DELAS

O formato dos DELAS é bastante semelhante ao dos DELAF. A diferença é que somente uma forma canônica seguida por códigos gramaticais e/ou semânticos é citada. A forma canônica é separada dos diferentes códigos por uma vírgula. Eis um exemplo de entrada:

```
cheval,N4+Anl
```

O primeiro código gramatical ou semântico será interpretado pelo programa de flexão como o substantivo da gramática a ser utilizado para flexionar a entrada. A entrada do exemplo acima indica que a palavra *cheval* deve ser flexionada com uma gramática de nome `N4`. É possível acrescentar códigos flexionais às entradas, mas a natureza da operação de flexão limita o interesse dessa possibilidade. Para mais detalhes, ver mais adiante, neste capítulo, a seção `3.4`.

### 3.1.3 Conteúdo dos dicionários

Os dicionários fornecidos juntamente com o Unitex contêm descrições de palavras simples e compostas. Essas descrições indicam a categoria gramatical de cada entrada, seus eventuais códigos de flexão, assim como informações semânticas diversas. As tabelas a seguir fornecem um resumo dos diferentes códigos utilizados nos dicionários fornecidos com o Unitex.

| Código | Significado | Exemplos |
|---|---|---|
| `A` | Adjetivo | Fabuleux |
| `ADV` | Advérbio | Réellement, à la longue |
| `CONJC` | Conjunção de coordenação | Mais |
| `CONJS` | Conjunção de subordinação | Puisque, à moins que |
| `DET` | Determinante | Ses, trente-six |
| `INTJ` | Interjeição | Adieu, mille millions de mille sabords |
| `N` | Substantivo | Prairie, vie sociale |

| | | |
|---|---|---|
| PREP | Preposição | Sans, à la lumière de |
| PRO | Pronome | Tu, elle-même |
| V | verbo | Continuer, copier-coller |

TAB. 3.1 Códigos gramaticais usuais

| Código | Significado | Exemplo |
|---|---|---|
| z1 | Linguagem comum | blague |
| z2 | Linguagem especializada | sépulcre |
| z3 | Linguagem muito especializada | houer |
| Abst | Abstrato | Bon goût |
| Anl | Animal | Cheval de race |
| AnlColl | Animal coletivo | troupeau |
| Conc | Concreto | abbaye |
| Conc Coll | Concreto coletivo | décombres |
| Hum | Humano | Diplomate |
| HumColl | Humano coletivo | Vieille garde |
| t | Verbo transitivo | foudroyer |
| i | Verbo intransitivo | fraterniser |
| en | Partícula pré-verbal (PPV) obrigatória | En imposer[1] |
| se | Verbo pronominal | Se marier |
| ne | Verbo de negação obrigatória | Ne pas cesser de[2] |

TAB. 3.2 – Alguns códigos semânticos

Esses códigos têm o mesmo significado para quase todas as línguas, mesmo se alguns deles forem próprios de certas línguas (*i.e.* marca do neutro, etc.).

Observação: as descrições dos tempos da tabela 3.3 correspondem ao francês. No entanto, a maioria dessas definições é encontrada em várias línguas (infinitivo, presente, particípio, etc.).

Apesar de existir uma base comum à maioria das línguas, os dicionários contêm particularidades de codificação próprias para cada língua. Assim, os códigos de declinações que variam bastante de uma língua para outra não foram aqui descritos. Para uma descrição exaustiva de todos os códigos utilizados em um dicionário, recomenda-se contatar o autor do dicionário.

| Código | Significado |
|---|---|

[1] Particularidade da língua francesa.
[2] Particularidade da língua francesa.

| | |
|---|---|
| m | Masculino |
| f | Feminino |
| n | Neutro |
| s | Singular |
| p | Plural |
| 1,2,3 | 1ª. 2ª, 3ª pessoa |
| P | Presente do indicativo |
| I | Pretérito imperfeito do indicativo |
| S | Presente do subjuntivo |
| T | Imperfeito do subjuntivo |
| Y | Presente do imperativo |
| C | Présent du conditionnel[3] |
| J | Passé simple[4] |
| W | Infinitivo |
| G | Gerúndio |
| K | Particípio |
| F | Futuro |

TAB. 3.3 – Códigos flexionais usuais

Os códigos apresentados não são, de forma alguma, limitativos. Cada usuário pode introduzir seus próprios códigos e criar seus próprios dicionários. Por exemplo, com um objetivo pedagógico, podem ser introduzidas, em dicionários de língua inglesa, marcas que indicam os falsos cognatos em francês:

```
bless,.V+faux-ami/bénir
cask,.N+faux-ami/tonneau
journey,.N+faux-ami/voyage
```

Também é possível utilizar dicionários para armazenar informações particulares. Assim, a forma flexionada de uma entrada pode ser utilizada para descrever uma sigla e a forma canônica para fornecer a forma completa:

```
ADN,Acide DésoxyriboNucléique.SIGLA
LADL,Laboratoire d'Automatique Documentaire et
Linguistique.SIGLA
SAV,Service Après-Vente.SIGLA
```

## 3.2 Verificação do formato de um dicionário

Quando os dicionários são de tamanho considerável, torna-se cansativo verificá-los manualmente. O Unitex contém um programa `CheckDic` que verifica automaticamente os dicionários DELAF e DELAS.

[3] Corresponde ao Futuro do Pretérito do Português.
[4] Corresponde ao Pretérito Perfeito do Indicativo do Português.

Esse programa realiza uma verificação da sintaxe das entradas. Para cada entrada mal formada, o programa exibe o número de linha, o conteúdo dessa linha e a natureza do erro. Os resultados da análise são salvos em um arquivo chamado `CHECK_DIC.TXT` que é exibido assim que a verificação for concluída. Além das eventuais mensagens de erro, esse arquivo contém a lista de todos os caracteres utilizados nas formas flexionadas e canônicas, a lista de códigos gramaticais e semânticos, assim como a lista de códigos flexionais utilizados. A lista de caracteres permite verificar se os caracteres presentes no dicionário são coerentes com aqueles presentes no arquivo do alfabeto da língua. Cada caractere vem acompanhado por seu valor em notação hexadecimal. As listas de códigos podem ser utilizadas para verificar se não existe erro de digitação nos códigos do dicionário.

O programa funciona com dicionários não-compactados, ou seja, sob forma de arquivos texto. A convenção geralmente utilizada é dar a extensão `.dic` a esses dicionários. Para verificar o formato de um dicionário, é preciso, antes de tudo, abri-lo clicando em "Open..." no menu "DELA".

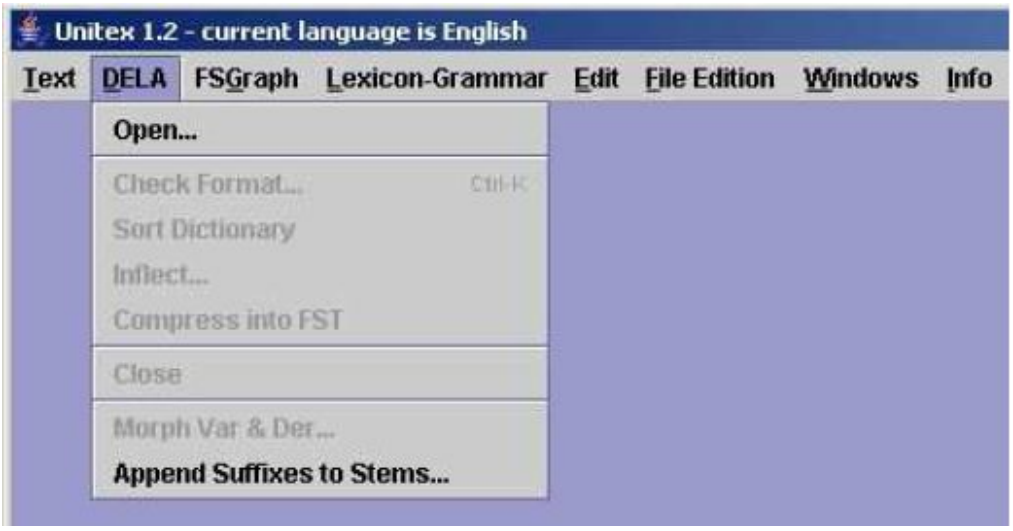



FIG. 3.1 – Menu DELA

É preciso carregar o dicionário da figura `3.2`. Para acionar a verificação automática, clicar em "Check Format" no menu "DELA". E, assim, a janela da figura `3.3` aparecerá. Essa janela permite escolher o tipo de dicionário que se deseja verificar. Os resultados da verificação do dicionário da figura `3.2` estão na figura `3.4`.

O primeiro erro deve-se ao fato de o programa não ter encontrado o ponto. O segundo, ao fato de não ter encontrado a vírgula que marca o fim da forma flexionada. O terceiro erro indica que o programa não encontrou nenhum código gramatical ou semântico.


FIG. 3.2 – Exemplo de dicionário

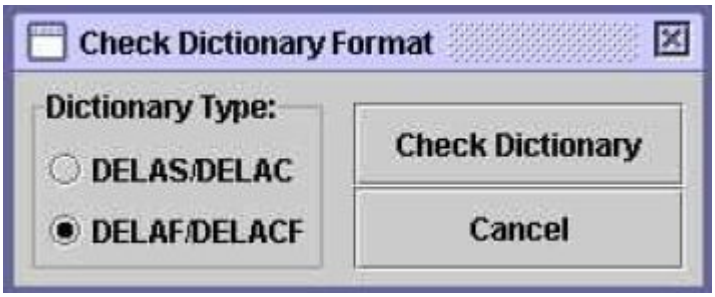



FIG. 3.3 – Verificação automática de um dicionário

### 3.3 Ordenação

O Unitex manipula os dicionários sem se preocupar com a ordem das entradas. Entretanto, para fins de apresentação, muitas vezes é preferível ordenar os dicionários. A operação de ordenação varia de acordo com vários critérios, começando pela língua do texto a ser ordenado. Assim, a ordenação de um dicionário tailandês se realiza de acordo com uma ordem diferente da ordem alfabética, de forma que o Unitex utiliza um modo de ordenação desenvolvido especialmente para o tailandês (ver capítulo 9).

Para as línguas européias, a organização realiza-se geralmente de acordo com a ordem lexicográfica, com, no entanto, algumas variantes. Com efeito, certas línguas como o francês consideram certos caracteres como equivalentes. Por exemplo, a diferença entre os caracteres e e é é ignorado quando se quer comparar as palavras manger e mangés, pois os contextos r e s permitem decidir a ordem. A distinção é feita apenas quando os contextos forem idênticos, que é o caso se comparar pêche e pèche.

A fim de levar em conta esse fenômeno, o programa de ordenação SortTxt utiliza um arquivo que define as equivalências de caracteres. Esse arquivo chama-se Alphabet_sort.txt e encontra-se no diretório da língua comum do usuário. Eis as primeiras linhas do arquivo utilizado como default para o francês:

```
AÀÂÄaàâä
Bb
CÇcç
Dd
```

## 3.4 Flexão automática

Como descrito na seção 3.1.2, uma linha de DELAS é composta, geralmente, por uma forma canônica e por uma seqüência de códigos gramaticais ou semânticos:

```
fênix,N4+Anl
ônix,N4+Conc
remix,N4
```

O primeiro código encontrado é interpretado como o nome da gramática a ser utilizada para flexionar a forma canônica. As gramáticas de flexão devem ter sido compiladas (ver capitulo 5). No exemplo acima, todas as entradas serão flexionadas com uma gramática nomeada N4.

Para ativar a flexão, clicar em "Inflect" no menu "DELA". A janela da figura 3.5 permite indicar ao programa de flexão o repertório no qual se encontram as gramáticas de flexão. Por default, o sub-repertório `Inflection` do repertório da língua comum é utilizado. A opção "Add ´ :´ before inflectional codes if necessary", insere automaticamente o caractere ´:´ antes dos códigos flexionais, casos eles não se iniciem por esse caractere. A opção "Remove class numbers" permite substituir os códigos com números utilizados no DELAS por códigos sem números, prontos para serem utilizados. Exemplo: `V17` e `N4+Hum` serão substituídos respectivamente por `V` e `N+Hum`.

A figura 3.6 apresenta um exemplo de gramática de flexão. Os diretórios descrevem os sufixos a acrescentar ou suprimir para obter a forma flexionada a partir da forma canônica, e as saídas (texto em negrito sob as caixas) oferecem os códigos flexionais a acrescentar à entrada do dicionário. Em nosso exemplo, dois caminhos são possíveis. O primeiro não modifica a forma canônica e acrescenta o código flexional `:s`. O segundo suprime uma letra graças ao operador L, em seguida acrescenta o sufixo ces e acrescenta o código flexional `:p`.

4 operadores são possíveis:

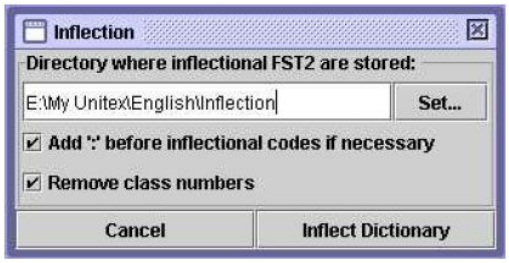



FIG. 3.5 . Configuração da flexão automática

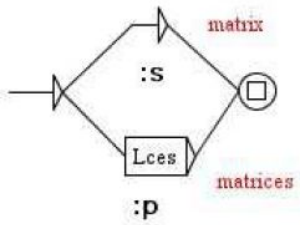
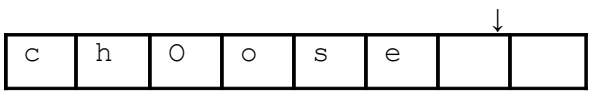



FIG. 3.6 . Gramática de flexão N4

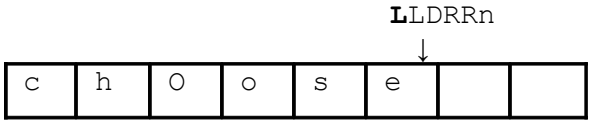
- L (left) retira uma letra da entrada;
- R (right) reestabelece uma letra da entrada. Em francês, muitos verbos do primeiro grupo se conjugam no presente na terceira pessoa do singular retirando-se o `r` do infinitivo e trocando-se a 4ª letra de trás pra frente por `è`: `peler → pèle, acheter → achète, gérer → gère`, etc. Mais do que descrever um sufixo de flexão para cada verbo (LLLLèle, LLLLète et LLLLère), pode-se utilizar o operador R para descrever apenas um: LLLLèRR.
- C (copy) duplica uma letra da palavra, deslocando todas as que se encontram à sua direita. Suponhamos por exemplo que se deseje gerar automaticamente adjetivos em `able`, da língua francesa, a partir de substantivos. Em casos como regrettable ou réquisitionnable, observamos uma duplicação da consoante final do substantivo. Para evitar escrever um grafo de flexão para cada consoante final possível, podemos utilizar o operador C, a fim de duplicar a consoante final, qualquer que seja ela;
- D (delete) suprime uma letra da entrada, deslocando todas as que se encontram à sua direita. Se desejar, por exemplo, flexionar a palavra romena `european` para `europani`, a seqüência utilizada será `LDRi`. O `L` posicionará o cursor sobre a letra `a`, o `D` irá suprimir o `a` deslocando o `n`, depois `Ri` irá restabelecer o `n` e acrescentar um `i`.

Veja um exemplo que descreve a flexão de `choose` em `chosen` graças à seqüência de operadores `LLDRRn`:

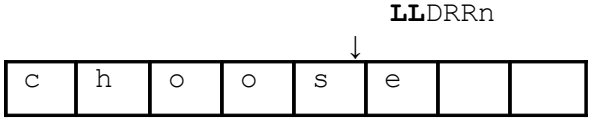
- Etapa 0: iniciação da pilha com a forma canônica; posicionar o cursor após a última letra:




- Etapa 1: deslocar o cursor para a esquerda:

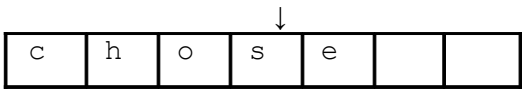



- Etapa 2: deslocar uma segunda vez o cursor para a esquerda:



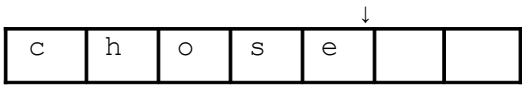
- Etapa 3: deslocar tudo o que está à direita do cursor para a esquerda:

**LLD**RRn



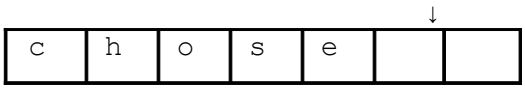

- Etapa 4: deslocar o cursor para a direita:

**LLDR**Rn



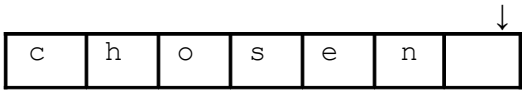

- Etapa 5: deslocar uma vez mais o cursor para a direita:

**LLDRR**n




- Etapa 6: escrever um `n`

**LLDRRn**




Uma vez que a seqüência é utilizada, toma-se o conteúdo da pilha imediatamente anterior ao cursor para se compor a forma flexionada (aqui `chosen`).

O programa de flexão `Inflect` explora todos os caminhos da gramática de flexão engendrando todas as formas flexionadas possíveis. A fim de evitar ter de substituir os nomes das gramáticas de flexão por verdadeiros códigos gramaticais no dicionário obtido, o programa substitui estes nomes por seus mais longos prefixos compostos por letras. Assim, `N4` é substituído por `N`. Ao escolher arbitrariamente os nomes das gramáticas de flexão, pode-se, portanto, produzir diretamente um dicionário pronto para o uso.

Veja o dicionário obtido após a flexão do DELAS do nosso exemplo:

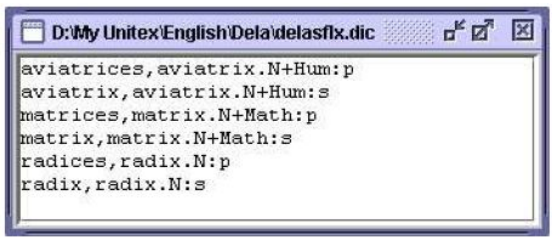



FIG. 3.7 – Resultado da flexão automática

## 3.5 Compactação

O Unitex aplica aos textos dos dicionários compactados. A compactação permite reduzir o tamanho dos dicionários e acelerar sua consulta. Essa operação é efetuada com o programa `Compress` que toma uma entrada de um dicionário sob forma de arquivo texto (por exemplo `meu_dicio.dic`) e produz dois arquivos:

- `meu_dicio.bin` contém o autômato mínimo das formas flexionadas dos dicionários;
- `meu_dicio.inf` contém os códigos que permitem reconstruir o dicionário de origem a partir das formas flexionadas contidas em `meu_dicio.bin`.

O autômato mínimo contido em `meu_dicio`.bin é uma representação das formas flexionadas onde todos os prefixos e sufixos comuns são fatorados. Por exemplo, o autômato mínimo das palavras `me`, `te`, `se`, `ma`, `ta` e `sa` pode ser representado pelo grafo da figura 3.8.

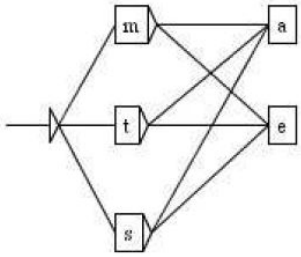



FIG. 3.8 . Representação de um exemplo de autômato mínimo

Para compactar um dicionário é necessário abri-lo e em seguida clicar em "Compress into FST" no menu “DELA”. A compactação é independente da língua e do conteúdo do dicionário. As mensagens produzidas pelo programa são exibidas em uma janela que não se fecha automaticamente. Assim, pode-se ver o tamanho do arquivo `.bin` obtido, o número de linhas lidas, bem como o número de códigos flexionais produzidos. A figura 3.9 mostra o resultado da compactação de um dicionário de palavras simples.

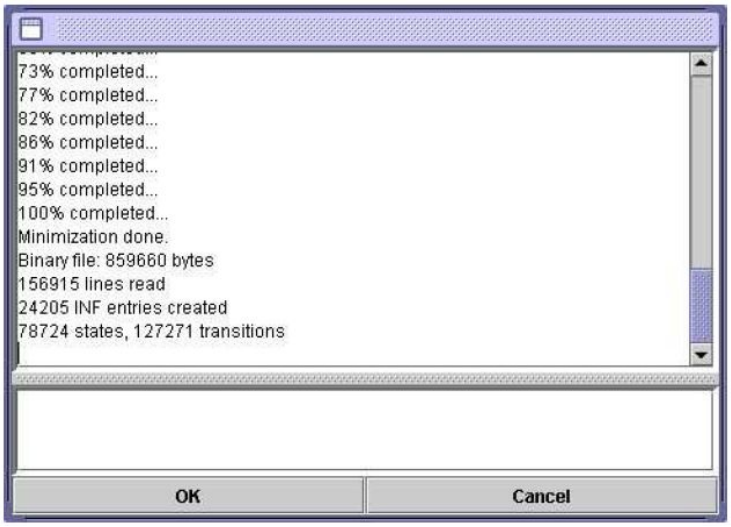


FIG. 3.9 . Resultado de uma compactação

A título indicativo, as taxas de compactação geralmente observadas de aproximadamente 95% para os dicionários de palavras simples e de 50% para os de palavras compostas.

## 3.6 Aplicação dos dicionários

O Unitex pode manipular tanto dicionários compactados (`.bin`), quanto grafos dicionários (`.fst2`). Esses dicionários podem ser aplicados tanto em um pré-processamento, quanto explicitamente clicando em "Apply Lexical Resources..." no menu "Text". Serão detalhados agora as regras de aplicação dos dicionários. O caso dos grafos dicionários será abordado na seção 3.6.3.

### 3.6.1 Prioridades

A regra de prioridade é a seguinte: se uma palavra do texto foi encontrada em um dicionário, esta palavra não mais será levada em conta quando da aplicação de dicionários com uma prioridade inferior.

Isso permite eliminar algumas ambigüidades quando da aplicação dos dicionários. Por exemplo, a palavra `par` tem uma interpretação nominal no domínio do golf. Se não desejar considerar esse emprego, basta criar um dicionário filtro que contenha apenas a entrada `par,.PREP` e salvá-la conferindo-lhe a prioridade mais alta. Desta maneira, mesmo se o dicionário das palavras simples contiver a outra entrada, ela será ignorada graças ao "jeu" das prioridades.

Há três níveis de prioridades. Os dicionários cujos substantivos sem extensão terminam em - têm a maior prioridade; aqueles cujos substantivos terminam em + têm a menor prioridade; os outros dicionários são aplicados com uma prioridade média. A ordem da aplicação de vários dicionários que tenham a mesma prioridade não importa. Em linha de comando, a instrução:

```
Dicio ex.snt alf.txt Estados+.bin Topo-.bin PR.fst2 Regiões-.bin
```

aplicaria portanto os dicionários na seguinte ordem (`ex.snt` é o texto ao qual são aplicados os dicionários, e `alf.txt` é o arquivo do alfabeto utilizado):

1. `Topo-.bin`
2. `Regiões-.bin`
3. `PR.fst2`
4. `Estados.bin`

### 3.6.2 Regras de aplicação dos dicionários

Além da regra de prioridades, a aplicação dos dicionários é efetuada respeitando-se as maiúsculas e os espaços. A regra do respeito às maiúsculas é a seguinte:
. se há uma maiúscula no dicionário, então deve haver uma maiúscula no texto;
. se há uma minúscula no dicionário, pode haver tanto uma minúscula quanto uma maiúscula no texto.

Assim, a entrada `linda,.Adj:fs` reconhecerá as palavras `linda`, `Linda` e `LINDA`, enquanto que `Linda,.N+Nome` reconhecerá apenas `Linda` e `LINDA`. As letras minúsculas e maiúsculas são definidas pelo arquivo alfabeto passado em parâmetro com o programa `Dicio`.

O respeito aos espaçamentos é uma regra bastante simples: para que uma seqüência de texto seja reconhecida por uma entrada do dicionário, ela deve conter exatamente os mesmos espaços. Por exemplo, se o dicionário contém `arco-íris,.N`, a sequência `arco- íris` não será reconhecida por causa do espaço que se segue ao hífen.

### 3.6.3 Grafos dicionários

O programa `Dicio` é capaz de aplicar grafos dicionários. Trata-se de grafos que respeitam a seguinte regra: se eles são aplicados com o programa `Locate` no modo MERGE, eles devem produzir seqüências que correspondam às linhas do DELAF.

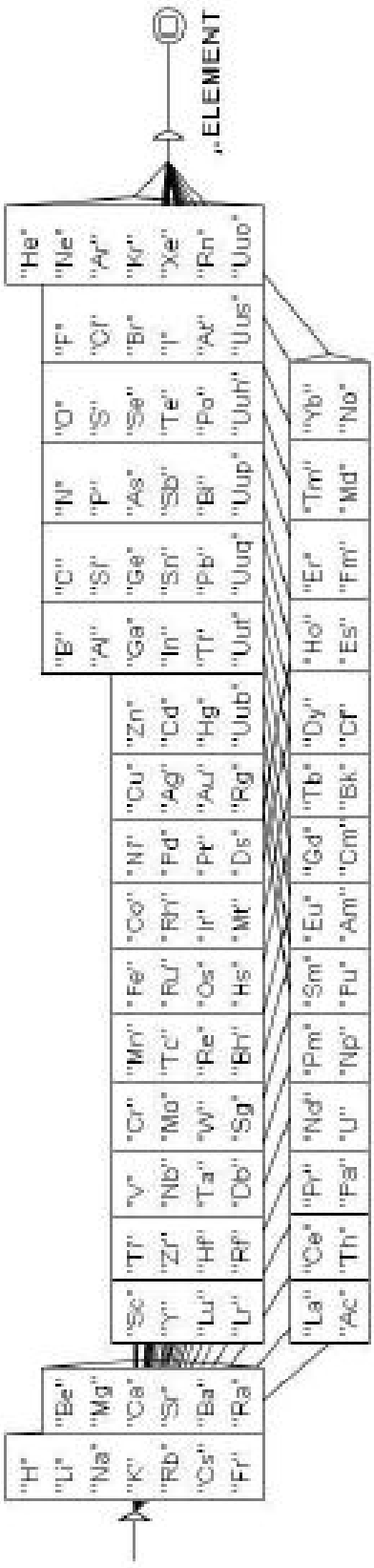



FIG. 3.10 . Grafo dicionário dos elementos químicos

A figura 3.10 mostra um grafo que reconhece os símbolos químicos. Pode-se ver nesta figura uma primeira vantagem em relação aos dicionários compactados: a utilização das aspas permite forçar o respeito à caixa. Assim, esse grafo reconhecerá `Fe`

mas não reconhecerá FE, enquanto que é impossível especificar uma interdição como essa em um DELAF convencional.

A segunda vantagem dos grafos dicionários é que eles podem explorar os resultados fornecidos pelos dicionários aplicados anteriormente. Assim, pode-se aplicar o dicionário geral, depois etiquetar como nomes próprios as palavras desconhecidas começadas por uma maiúscula com a ajuda do grafo NPr+ da figura 3.11. O + no nome do grafo confere-lhe uma baixa prioridade a fim de que ele seja aplicado após o dicionário geral. Para funcionar, esse grafo se baseia nas palavras que nunca são reconhecidas depois da aplicação do dicionário geral. Os colchetes correspondem a uma definição de contexto. Para mais detalhes sobre os contextos ver a seção 6.3.

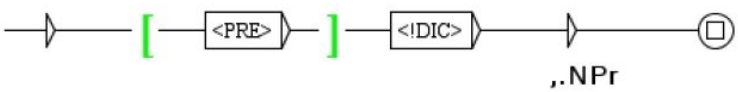



FIG. 3.11 . Grafo dicionário etiquetando como nomes próprios as palavras desconhecidas começadas por letra maiúscula.

Como os grafos dicionários são aplicados pelo motor do programa Locate, eles podem utilizar tudo o que Locate autoriza. De modo particular, é possível utilizar os filtros morfológicos. Assim, o grafo da figura 3.12 utiliza esses filtros para reconhecer os números em algarismos romanos. Note-se que, da mesma forma, ele utiliza contextos a fim de evitar, por exemplo, que D seja tomado apenas como algarismo romano quando ele for seguido de um apóstrofo.

## 3.7 Bibliografia

A tabela 3.4 oferece algumas referências relativas aos dicionários eletrônicos de palavras simples e compostas. Para mais detalhes, consultar a página de referências no site do Unitex: http://www-igm.univ-mlv.fr/~unitex

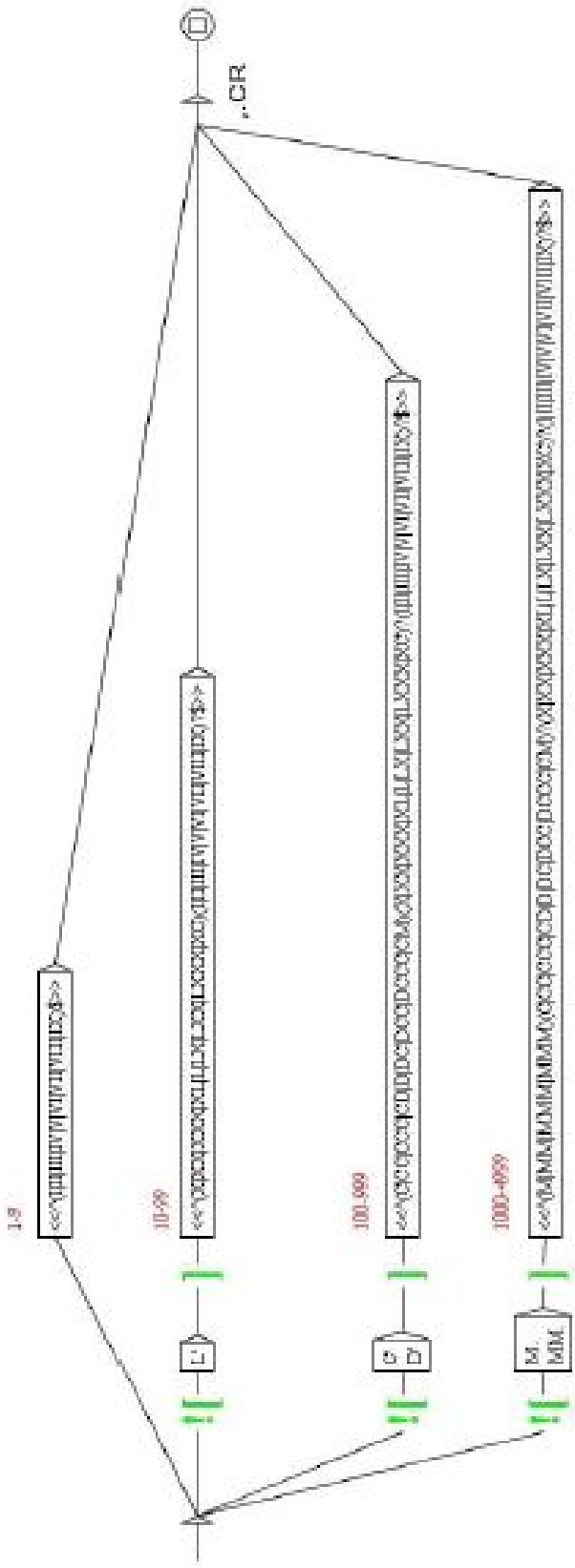



FIG. 3.12 . Grafo dicionário que reconhece os números em algarismos romanos.

| 1.1.1 Língua | Palavras simples | Palavras compostas |
|---|---|---|
| **Inglês** | **[30], [40]** | **[11], [46]** |
| **Francês** | **[14], [15], [34]** | **[15], [25], [47], [27]** |
| **Grego Moderno** | **[2], [13], [31]** | **[32], [33]** |
| **Italiano** | **[19], [20]** | **[51]** |
| **Espanhol** | **[5]** | **[4]** |

TAB. 3.4 . Algumas referências bibliográficas sobre os dicionários eletrônicos

Capítulo 4

# 1.2 Busca de expressões regulares

Este capítulo apresenta como pesquisar padrões simples em um texto utilizando expressões regulares.

## 4.1 Definição

O objetivo deste capítulo não é fazer uma introdução às linguagens formais, mas mostrar como utilizar as expressões racionais no Unitex para pesquisar padrões simples. O leitor interessado em uma apresentação mais formal poderá lançar mão das numerosas obras que tratam do assunto.

Uma expressão regular pode ser:

- uma unidade lexical (`livro`) ou um padrão lexical (`<comer.V>`);
- a concatenação de duas expressões racionais (`eu como`);
- a união de duas expressões racionais (`Pedro+Paulo`);
- o asterisco de Kleene de uma expressão regular (`muito*`).

## 4.2 Unidades lexicais

Em uma expressão regular, a unidade lexical tem a mesma definição mostrada na seção 2.5.4 (página 24). Note-se que os símbolos ponto, adição, asterisco, subtração, bem como os parênteses de abertura e fechamento, têm uma significação específica. É necessário, portanto, desabilita-los com o caractere de escape \ se desejar busca-los. A seguir, alguns exemplos de unidades lexicais válidas:

```
gato
\.
<N:ms>
{S}
```

Por definição, o Unitex tolera que palavras com minúsculas reconheçam palavras escritas com maiúsculas. É possível forçar o respeito à caixa utilizando aspas. Assim, “argentina” reconhece apenas a forma argentina e não Argentina ou ARGENTINA.

NOTA: Se desejar tornar a presença de um espaço obrigatória é necessário colocá-lo entre aspas.

## 4.3 Padrões

### 4.3.1 Símbolos especiais

Há dois tipos de padrões. A primeira categoria reúne todos os símbolos apresentados na seção 2.5.2, com exceção de `<PNC>`, que identifica os sinais de pontuação, e do símbolo `<^>`, que identifica uma quebra de página. Tendo sido todas as quebras de páginas substituídas por espaços, esse símbolo não tem mais nenhuma utilidade quando da busca por padrões. Esses símbolos, também chamados *metas*, são os seguintes:

- <E>: palavra vazia, ou épsilon. Reconhece a seqüência vazia;
- <TOKEN>: Reconhece qualquer unidade lexical;
- <MOT>: Reconhece qualquer unidade lexical formada por letras;
- <MIN>: Reconhece qualquer unidade lexical formada por letras minúsculas;
- <MAJ>: Reconhece qualquer unidade lexical formada por letras maiúsculas;
- <PRE>: Reconhece qualquer unidade lexical formada por letras e iniciada por uma maiúscula;
- <TOKEN>: Reconhece qualquer unidade lexical, exceto o espaço;
- <DIC>: Reconhece qualquer palavra presente nos dicionários do texto;
- <SDIC>: Reconhece qualquer palavra simples presente nos dicionários do texto;
- <CDIC>: Reconhece qualquer palavra composta presente nos dicionários do texto;
- <NB>: Reconhece qualquer seqüência de algarismos consecutivos (1234 é identificado, mas não 1 234);
- # : impede a presença do espaço.

NOTA: Como já havia sido dito na seção 2.5.4, NENHUM dos *meta* pode ser utilizado para reconhecer o marcador `{stop}`, nem mesmo <TOKEN>.

### 4.3.2 Padrões lexicais

O segundo tipo de padrões reúne os que se referem às informações contidas nos dicionários do texto. São os chamadas *padrões lexicais*. As quatro formas possíveis são:

- `<ler>`: reconhece todas as entradas que têm `ler` como forma canônica;
- `<ler.V>`: reconhece todas as entradas que têm `ler` como forma canônica e o código gramatical `V`;
- `<V>` >: reconhece todas as entradas que têm o código gramatical `V`;
- `{leremos, ler.V}` ou `<leremos, ler.V>` : reconhece todas as entradas que têm `leremos` como forma flexionada, `ler` como forma canônica e o código gramatical `V`. Esse tipo de padrão só é interessante quando se trabalha sobre o autômato do texto em que estão explicitadas as ambigüidades das palavras.

Quando se efetua uma busca no texto, esse padrão reconhece o mesmo que a simples unidade lexical `leremos`.

### 4.3.3 Restrições gramaticais e semânticas

Os padrões lexicais dos exemplos acima são simples. É possível exprimir chaves de consulta mais complexas indicando-se mais códigos gramaticais ou semânticos, separados pelo caractere `+`. Uma entrada de dicionário só será reconhecida então se possuir todos os códigos presentes no padrão. O padrão `<N+z1>` reconhece, assim, as entradas:

```
bordados,bordado.N+z1:mp
capitais européias,capital européia.N+NA+Conc+HumColl+z1:fp
```

mas não :

```
Descartes, René Descartes.N+Hum+NProprio:ms
Habituado,.A+z1:ms
```

É possível excluir códigos fazendo com que eles sejam precedidos pelo caractere `-` ao invés do `+`. Para ser reconhecida, uma entrada deve conter todos os códigos autorizados pelo padrão e nenhum dos códigos proibidos. O padrão`<A-z3>` reconhece, portanto, todos os adjetivos que não possuem o código `z3` (ver quadro 3.2). Se desejar fazer referência a um código que contém o caractere `-`, é necessário desabilitar esse caractere precedendo-o do caractere `\`. Assim, o padrão `<N+falso\-cognato>` poderá reconhecer todas as entradas de dicionário que contenham os códigos `N` e `falso-cognato`.

A ordem na qual os códigos aparecem no padrão não tem nenhuma importância. Os três padrões lexicais seguintes são equivalentes:

```
<N-Hum+z1>
<z1+N-Hum>
<-Hum+z1+N>
```

NOTA: não é possível utilizar um padrão que só tenha códigos proibidos. `<-N>` e `<-A-z1>` são portanto padrões incorretos. É possível, todavia, exprimir tais restrições utilizando-se contextos (ver seção 6.3).

### 4.3.4 Restrições flexionais

Pode-se igualmente especificar restrições relativas aos códigos flexionais. Essas restrições devem ser obrigatoriamente precedidas por pelo menos um código

gramatical ou semântico. Elas se apresentam como os códigos flexionais presentes nos dicionários.

Eis alguns exemplos de padrões lexicais que utilizam restrições flexionais:

- `<A:m>` reconhece um adjetivo masculino ;
- `<A:mp:f>` reconhece um adjetivo que seja tanto masculino plural quanto feminino;
- `<V:2:3>` identifica um verbo na 2ª ou 3ª pessoa ; isso exclui todos os tempos que não admitam nem 2ª nem 3ª pessoa (infinitivo impessoal, gerúndio, particípio) assim como os tempos conjugados na 1ª pessoa.

Para que uma entrada de dicionário *E* seja reconhecida por um padrão *M*, é necessário que pelo menos um código flexional de *E* contenha todos os caracteres de um código flexional de *M*. Consideremos o exemplo seguinte:

*E*=`separe,separar.V+z1:S3s:Y3s`
*M*=`<V:P2s:Y3>`

Nenhum código flexional de *E* contém ao mesmo tempo os caracteres `P`, `2` e `s`. No entanto, o código `Y3s` de *E* contém os caracteres `Y` e `3`. O código `Y3` está incluído em pelo menos um código de *E*, o padrão lexical *M* reconhece, portanto, a entrada *E*. A ordem dos caracteres no interior do código flexional não é importante.

### 4.3.5 Negação de um padrão

É possível fazer a negação de um padrão por meio do caractere `!` colocado imediatamente após o caractere <. A negação é possível sobre os *metas* `<PAL>`, `<MIN>`, `<MAI>`,`<PRE>`, `<DIC>`, bem como nos padrões que comportam apenas códigos gramaticais, semânticos ou flexionais (*i.e.* `<!V-z3:P3>`). Os padrões # e " " são a negação um da outro. O *meta* `<!PAL>` pode reconhecer todas as unidades lexicais que não são formadas por letras, exceto o separador de frases e, obviamente, o marcador `{STOP}`. A negação não tem efeito sobre `<NB>`, `<SDIC>`, `<CDIC>` e `<TOKEN>`.

A negação é interpretada de uma maneira particular nos *metas* `<!DIC>`, `<!MIN>`, `<!MAI>` e `<!PRE>`. Ao invés de reconhecer todas as formas que não são reconhecidas pelo *meta* sem a negação, essas chaves de consulta fornecem apenas formas que são seqüências de letras. Assim, o *meta* `<!DIC>` permite obter as palavras desconhecidas do texto. Essas formas desconhecidas são, na maioria das vezes, nomes próprios, neologismos e erros de ortografia.

A negação de um padrão lexical como `<V:G>` reconhece todas as palavras, à exceção daquelas que podem ser reconhecidas por esse padrão. Assim, o padrão `<!V:G>` não reconhecerá a forma inglesa `being`, mesmo que ela exista nos dicionários do texto de entradas não-verbais para essa palavra:

```
being,.A
being,.N+Abst:s
being,.N+Hum:s
```

Seguem abaixo vários exemplos de padrões que misturam os diferentes tipos de restrições:

- `<A-Hum:fs>`: Adjetivo não-humano no feminino singular;
- `<ler.V:P:F>` : O verbo *ler* no presente ou no futuro;

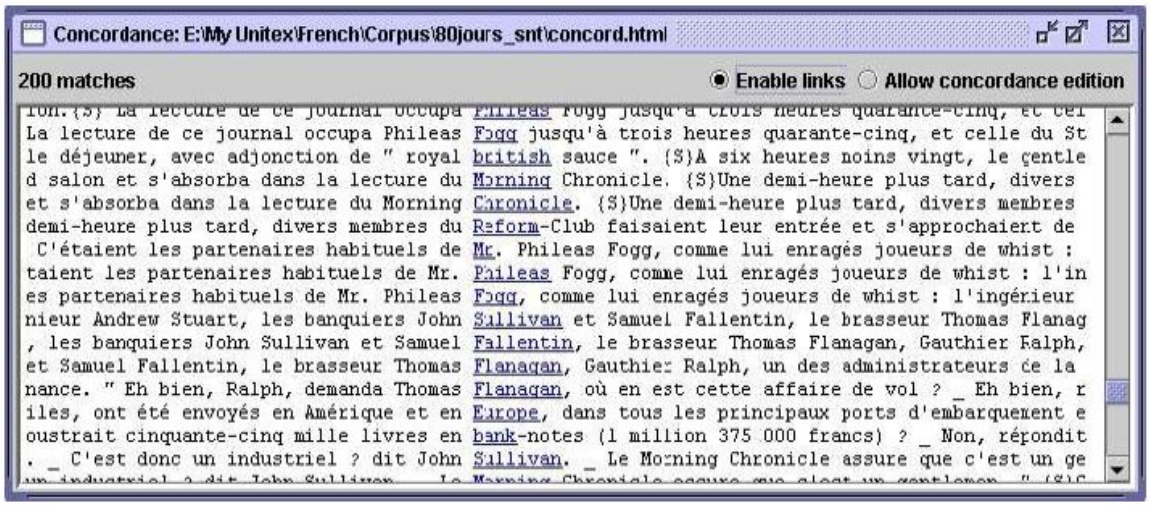



FIG. 4.1 . Resultado da pesquisa do *meta* `<!DIC>`

- `<como,comer.V>` : a palavra *como* enquanto forma conjugada do verbo *comer* e não o advérbio ou a conjunção;
- `<chaveiro.N-Hum>`: todas as entradas nominais que têm *chaveiro* como forma canônica e não têm o código semântico `Hum`;
- `<!ADV>` : todas as palavras que não são advérbios;
- `<!PAL>` : todos os caracteres, que não são letras, exceto o separador de frases (ver figura 4.2).

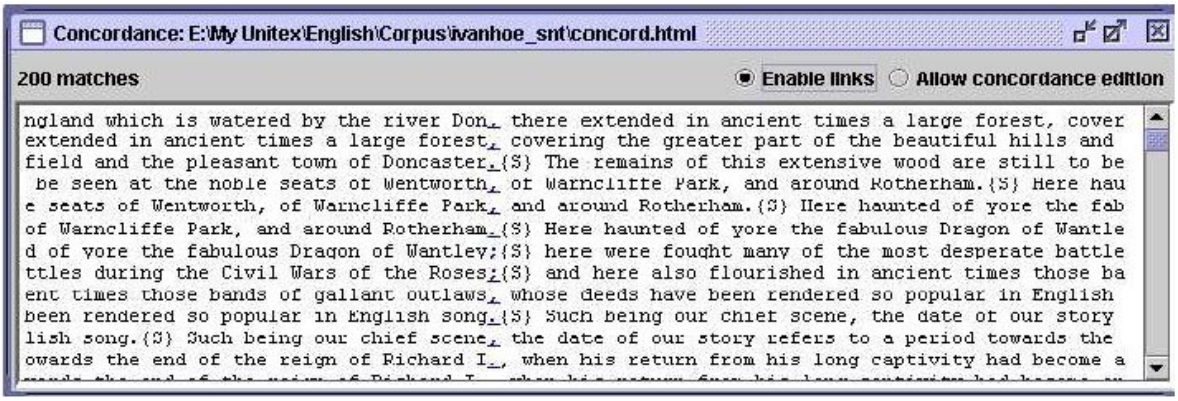



FIG. 4.2 . Resultado da busca do *meta* <!PAL>

## 4.4 Concatenação

Pode-se concatenar expressões regulares de três maneiras. A primeira consiste em utilizar o operador de concatenação representado pelo ponto. Assim, a expressão:

```
<DET>.<N>
```

reconhece um determinante seguido por um substantivo. O espaço pode igualmente servir para concatenar. A expressão do exemplo seguinte:

```
o <A> gato
```

reconhece a unidade lexical *o*, seguida por um adjetivo e da unidade lexical *gato*. Enfim, é possível omitir o ponto e o espaço antes de um parêntese de abertura ou do caractere <, bem como após um parêntese de fechamento ou o caractere >. Os parênteses servem para delimitar uma expressão regular. Todas as expressões seguintes são equivalentes:

```
o <A> gato
(o <A>) gato
o.<A>. gato
(o)<A>. gato
(o (<A>))(gato)
```

## 4.5 União

A união de expressões regulares é feita separando-as pelo caractere +. A expressão:

```
(eu+tu+você+ele+ela+nós+vós+eles+elas) <V>
```

reconhece um pronome seguido por um verbo. Para tornar facultativo um elemento em uma expressão, basta fazer a união desse elemento com a palavra vazia épsilon.

Exemplos:

`o (pequeno+<E>)gato` reconhece as seqüências *o gato* e *o pequeno gato*

`(<E>+franco-)(inglês+belga)` reconhece *inglês, belga, franco-inglês* e *franco-belga*.

## 4.6 Asterisco de Kleene

O asterisco de Kleene, representado pelo caractere *, permite reconhecer zero, uma ou mais ocorrências de uma expressão. O asterisco deve ser colocado à direita do elemento em questão.
A expressão:

```
Faz muito* frio
```

reconhece *faz frio*, *faz muito frio*, *faz muito muito frio*, etc. O asterisco é prioritário sobre os outros operadores. É necessário utilizar os parênteses para aplicar o asterisco a uma expressão complexa. A expressão:

```
0,(0+1+2+3+4+5+6+7+8+9)*
```

reconhece um zero, seguida de uma vírgula e de uma seqüência eventualmente vazia de algarismos.

ATENÇÃO: É proibido pesquisar a palavra vazia com uma expressão regular. Na tentativa de se pesquisar `(0+1+2+3+4+5+6+7+8+9)*`, o programa sinalizará um erro, como mostra a figura 4.3.

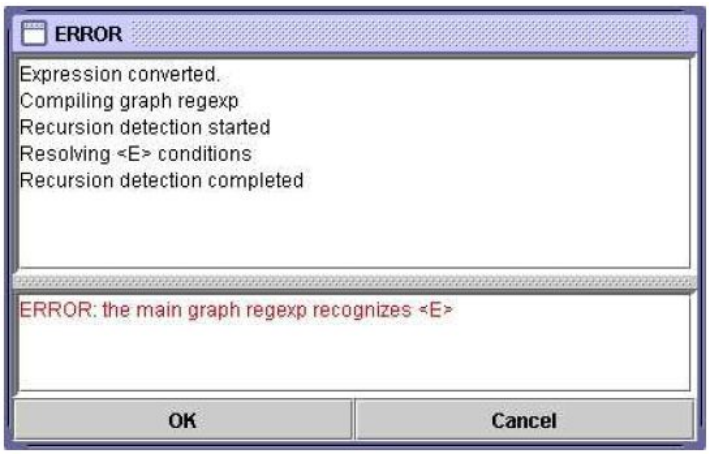


FIG. 4.3 . Erro quando da pesquisa de uma expressão que reconhece a palavra vazia

## 4.7 Filtros Morfológicos

É possível aplicar filtros morfológicos às unidades lexicais procuradas. Para isso, é necessário inserir um filtro, grafado entre aspas angulares duplas, imediatamente após a unidade lexical em questão:

*chave de busca <<chave de busca morfológica>>*

Os filtros morfológicos apresentam-se sob a forma de expressões regulares no formato POSIX (ver [36] para uma sintaxe detalhada). Eis alguns exemplos de filtros elementares:

- `<<ss>>`: contém `ss`
- `<<^a>>`: começa com `a`
- `<<ez$>>`: termina com `ez`
- `<<a.s>>`: contém `a` seguido por um caractere qualquer, seguido por `s`
- `<<a.*s>>`: contém `a` seguido por um número qualquer de caracteres, seguido por `s`
- `<<ss|tt>>`: contém `ss` ou `tt`
- `<<[aeiouy]>>`: contém uma vogal não-acentuada
- `<<[aeiouy]{3,5}>>`: contém uma seqüência de vogais não-acentuadas de extensão entre 3 e 5 caracteres.
- `<<ée?>>`: contém `é` seguido por um e facultativo
- `<<st[^aeiouy]>>`: contém `st` seguido por um caractere que não é uma vogal

É possível combinar esses filtros elementares para formar filtros mais complexos:

- `<<[ai]ble$>>`: termina com `able` ou `ible`
- `<<^(anti|pro)-?>>`: começa com `anti` ou `pro`, seguido por um hífen facultativo
- `<<^([rst][aeiouy]){2,}$>>`: palavra formada por 2 ou mais seqüências, começando com um `r`, `s` ou `t`, seguido por uma vogal não-acentuada
- `<<^([^l]|l[^e])>>`: não começa por `l` ou então a segunda letra não é um `e`, ou seja, qualquer palavra exceto aquelas que começam por `le`. Tais restrições podem ser expressas de modo mais simples utilizando-se os contextos (ver 6.3)

Por definição, um filtro morfológico por si só é considerado como aplicável à meta <TOKEN>, ou seja, a qualquer unidade lexical exceto o espaço e o marcador {STOP}.

Por outro lado, quando um filtro segue imediatamente uma chave de busca, aplica-se àquilo que é reconhecido_pela chave de busca. Seguem abaixo alguns exemplos de tais combinações:

- `<V:K><<i$>>`: particípio passado terminando em `i`
- `<CDIC><<->>`: palavra composta contendo um hífen
- `<CDIC><< .* >>`: palavra composta contendo dois espaços
- `<A:fs><<^pro>>`: adjetivo feminino singular iniciado por `pro`

- `<DET><<^([^u]|(u[^n])|(...+))>>`: determinante diferente de `un`
- `<!DIC><<es$>>`: palavra que não está no dicionário e que termina em `es`
- `<V:S:T><<uiss>>`: verbo em francês no subjuntivo passado ou presente, contendo `uiss`

NOTA: Por definição, os filtros morfológicos são submetidos às mesmas variações de caixa tipográfica que as máscaras lexicais. Assim, o filtro `<<^é>>` vai identificar todas as palavras começando por `é`, mas do mesmo modo aquelas que começam por `E` ou `É`. Para forçar a distinção exata da caixa tipográfica do filtro, é preciso acrescentar `_f_` imediatamente depois dele. Exemplo: `<A><<^é>>_f_`

## 4.8 Busca

### 4.8.1 Configuração da busca

Para poder buscar uma expressão, é necessário em primeiro lugar abrir um texto (ver capítulo 2). Clicar em seguida em "Locate Pattern..." no menu "Text". A janela da figura 4.4 então aparecerá.

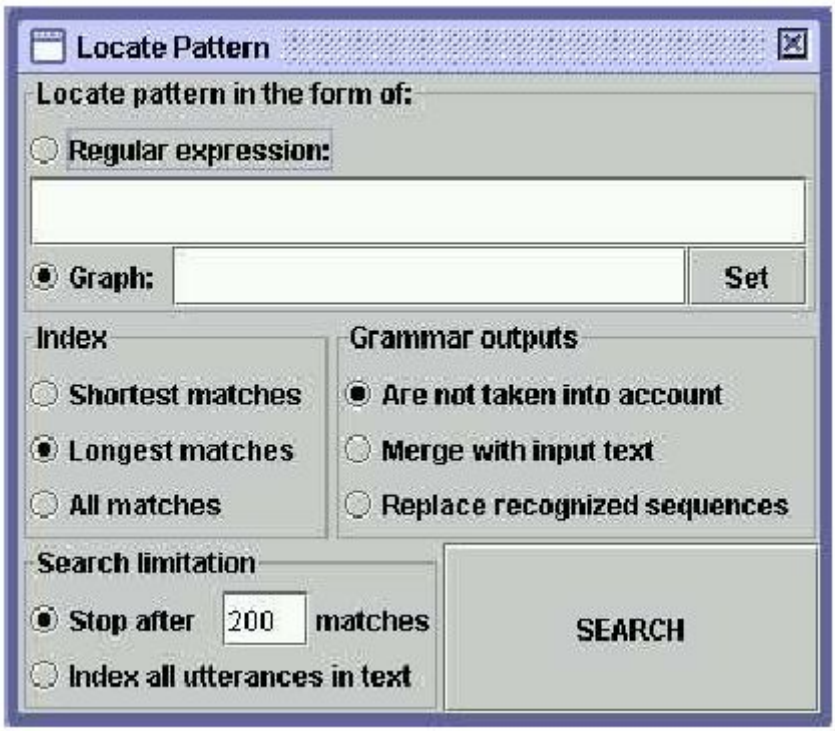


FIG. 4.4 Janela de busca de expressões

A opção “Locate pattern in the form of” permite escolher entre uma expressão racional e uma gramática. Clicar em “Regular expression”.

A opção “Index” permite selecionar o modo de reconhecimento:

- “Shortest matches”: dá prioridade às seqüências mais curtas;
- “Longest matches”: dá prioridade às seqüências mais longas. É o modo utilizado como default;
- “All matches”: fornece todas as seqüências reconhecidas.

A opção “Search limitation” permite limitar ou não a pesquisa a um certo número de ocorrências. Por definição, a busca é limitada às 200 primeiras ocorrências.

As possibilidades dentro da opção “Grammar outputs” não estão relacionadas às expressões racionais. Elas estão descritas na seção 6.7.

Digitar uma expressão e clicar em “Search” para iniciar a busca. O Unitex transformará a expressão em uma gramática no formato `.grf` . Essa gramática será, em seguida, compilada em uma gramática no formato `.fst2` que será utilizada pelo programa de busca.

### 4.8.2 Exibição dos resultados

Uma vez terminada a busca, a janela da figura 4.5 aparecerá, indicando o número de ocorrências encontradas, o número de unidades lexicais reconhecidas, assim como a relação desse número com o número total de unidades lexicais do texto.


FIG. 4.5. Resultados da busca.

Depois de clicar em “OK”, aparecerá a janela da figura 4.6, que permite configurar a exibição da lista das ocorrências encontradas. Pode-se igualmente abrir essa janela clicando em "Display Located Sequences..." no menu "Text".

Chama-se de *concordância* a lista das ocorrências.

A opção “Modify text” oferece a possibilidade de substituir as ocorrências encontradas pelas saídas produzidas. Essa possibilidade será examinada no capítulo 6.

A opção “Extract units” permite construir um arquivo texto com todas as frases, contendo elas ou não ocorrências. O botão “Set File” permite selecionar o arquivo de saída. Clicar em seguida em “Extract matching units” ou “Extract unmatching units” dependendo da escolha entre extrair frases contendo ou não ocorrências.

Na opção “Show Matching Sequences in Context”, pode-se selecionar a quantidade de caracteres dos contextos à esquerda e à direita das ocorrências que serão exibidas na concordância. Se uma ocorrência tiver menos caracteres do que o contexto à sua direita, a linha de concordância será completada com o número de caracteres necessário. Se uma ocorrência tiver maior número de caracteres em relação ao contexto à sua esquerda, ela será exibida por completo.

NOTA: Em tailandês, a extensão dos contextos é medido em caracteres visualizáveis e não em caracteres reais. Isso permite conservar o alinhamento das linhas de concordância apesar da presença de caracteres diacríticos que se combinam com outras letras em vez de serem exibidas como caracteres comuns.

Pode-se selecionar o modo de ordenação a ser aplicado na lista “Sort According to”. O modo “Text Order” exibe as ocorrências na ordem em que elas aparecem no texto. Os seis outros modos permitem a separação em colunas. As três áreas de uma linha são o contexto à esquerda da ocorrência, a ocorrência em si e o contexto à direita dela. As ocorrências e os contextos à direita são ordenados da esquerda para a direita. Os contextos à esquerda são ordenados da direita para a esquerda. O modo utilizado como default é “Center, Left Col.”. A concordância é produzida sob a forma de um arquivo HTML.

Quando as concordâncias atingem vários milhares de ocorrências, é preferível visualizá-las em um navegador web (Firefox [8], Netscape [9], Internet Explorer, etc.).

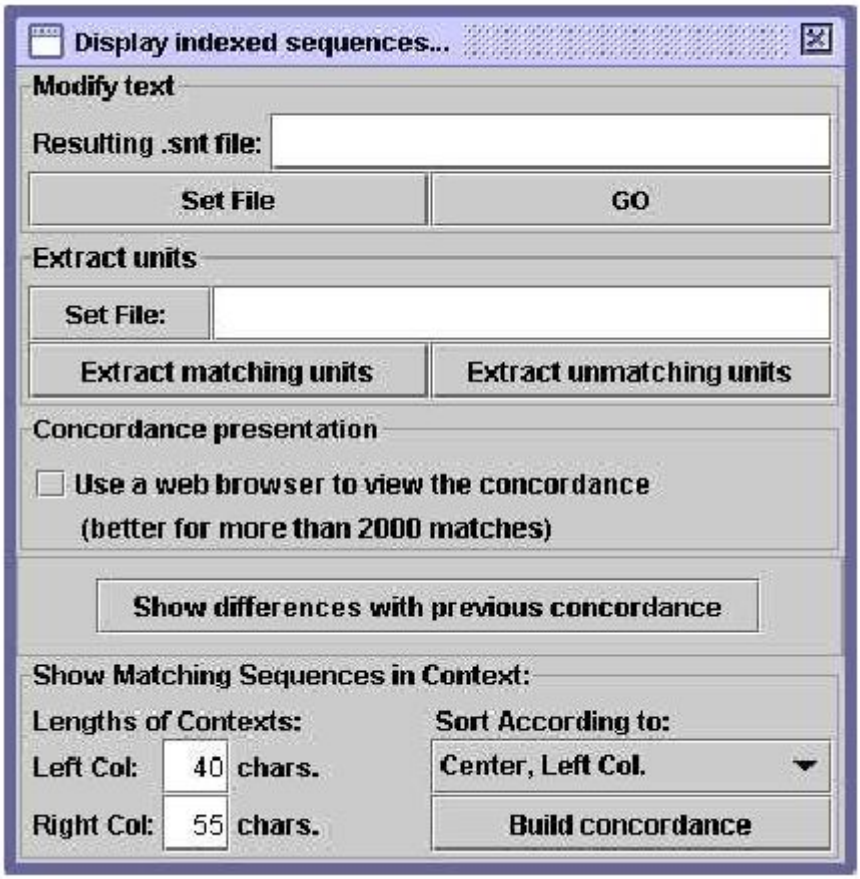



FIG. 4.6 .Configuração da exibição das ocorrências encontradas

Para isso, marcar a opção "Use a web browser to view the concordance" (ver figura 4.6). Essa opção é ativada por definição quando o número de ocorrências é superior a 3000. Para definir o navegador, clicar em "Preferences..." no menu "Info". Clicar sobre a guia "Text Presentation" e selecionar o programa a ser utilizado na opção "HTML Viewer" (ver figura 4.7).

Ao escolher abrir a concordância dentro do Unitex, uma janela como a da figura 4.8 será visualizada. A opção "Enable links", ativada por definição, permite considerar

as ocorrências como hiperlinks. Assim, quando se clica em uma ocorrência, abre-se a janela do texto e a seqüência reconhecida aparece selecionada. Além disso, se o autômato do texto for construído e se a janela não for minimizada em forma de ícone, o autômato da frase que contém a ocorrência clicada será carregado. Se a opção "Allow concordance edition" for selecionada, não se pode, deste modo, clicar nas ocorrências, mas, pode-se editar a concordância como texto. Isso permite, entre outras possibilidades, o deslocamento no texto com um cursor, o que pode ser prático quando se trabalha com uma concordância com grandes contextos.

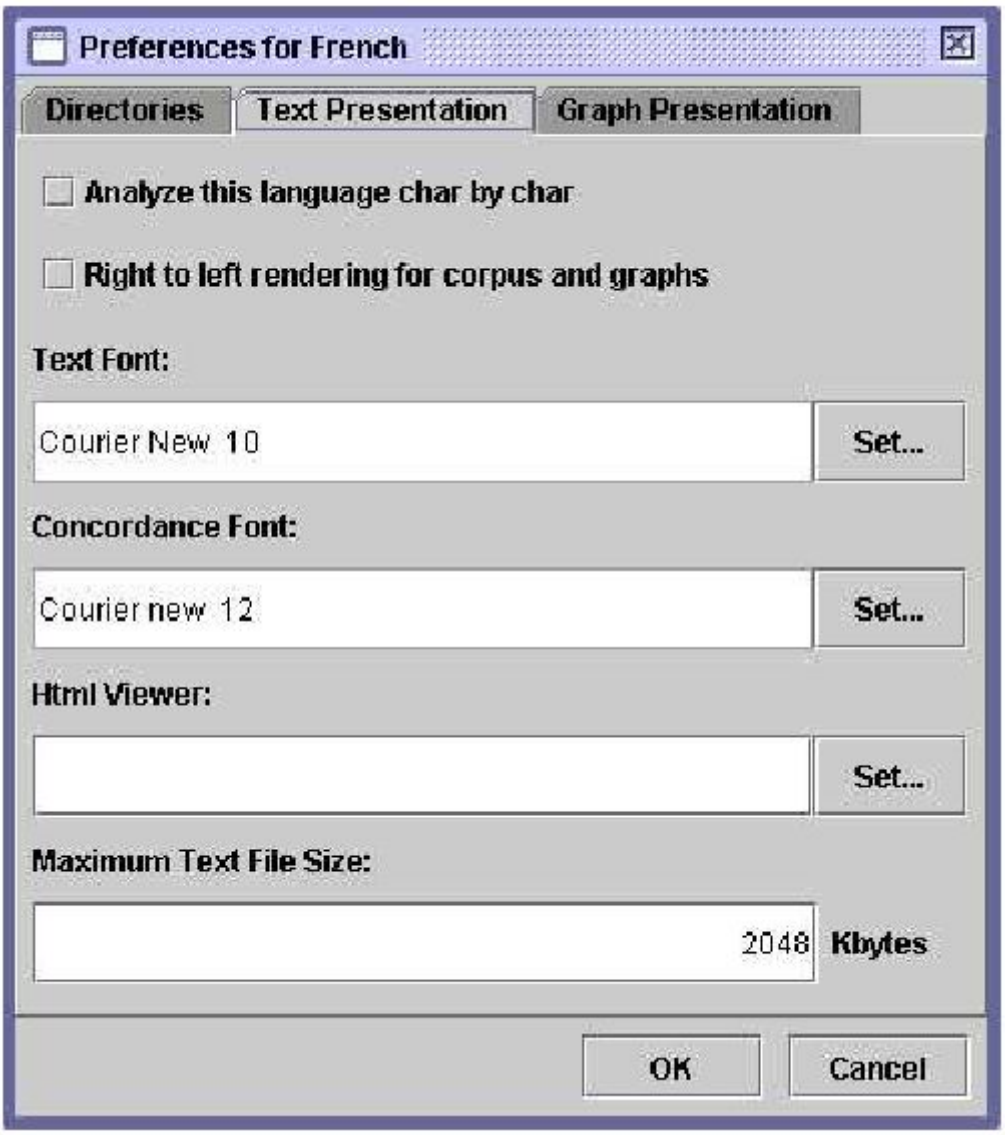


FIG. 4.7. Seleção do navegador para a exibição das concordâncias




FIG. 4.8. Exemplo de concordância

# Capítulo 5

# Gramáticas locais

As gramáticas locais são um meio poderoso de representar a maior parte dos fenômenos lingüísticos. A primeira seção apresentará o formalismo sobre o qual essas gramáticas se fundamentam. Será tratado em seguida de como se construir e se apresentar gramáticas com o Unitex.

## 5.1 Formalismo das gramáticas locais

### 5.1.1 Gramáticas algébricas

As gramáticas Unitex são variantes das gramáticas algébricas, também conhecidas como gramáticas livres de contexto. Uma gramática algébrica é constituída de regras de reescrita. Eis uma gramática que reconhece qualquer número de caracteres `a`:

#### 1.2.1.1 S → aS

$S \rightarrow \varepsilon$

Os símbolos apresentados à esquerda das regras são chamados de *símbolos não-terminais*, pois podem ser reescritos. Os símbolos que não podem ser reescritos por regras são chamados de s*ímbolos terminais*. Os membros à direita das regras são seqüências de símbolos não-terminais e terminais. O símbolo ípsilon notado $\varepsilon$ designa a palavra vazia. Na gramática acima, $S$ é um símbolo não-terminal e `a` um terminal. $S$ pode ser reescrito tanto em um `a` seguido de um $S$, quanto em palavra vazia. A operação de reescrita por meio da aplicação de uma regra é chamada de *derivação*. Diz-se que uma gramática reconhece uma palavra caso exista uma seqüência de derivações que produz essa palavra. O não-terminal que serve de ponto de partida para a primeira derivação é chamado de *axioma.*

A gramática mencionada acima reconhece assim a palavra $aa$, pois se pode obter essa palavra a partir do axioma $S$ efetuando as seguintes derivações:

Derivação 1: reescrita do axioma em $aS$

$\underline{S} \rightarrow aS$

Derivação 2: reescrita do *S* do membro à direita em *aS*

*S → aS → aaS*

Derivação 3: reescrita do *S* em *ε*

*S → aS → aaS → aa*

Denomina-se de linguagem de uma gramática o conjunto de palavras reconhecidas por ela. As linguagens reconhecidas pelas gramáticas algébricas são chamadas *linguagens algébricas*.

### 5.1.2 Gramáticas algébricas estendidas

As gramáticas algébricas estendidas são gramáticas algébricas nas quais os contextos direitos das regras não são mais seqüências de símbolos, mas expressões racionais. Assim, a gramática que identifica uma seqüência qualquer de `a` pode se reescrever em uma gramática estendida de uma só regra:

#### 1.2.1.2 S → a*

Essas gramáticas, também chamadas de *redes de transição recursiva* (RTN em inglês) ou *diagramas de sintaxe*, prestam-se a uma representação gráfica convivial. Com efeito, o membro à direita de uma regra pode ser representado por um grafo cujo nome é o membro à esquerda da regra.

Entretanto, as gramáticas Unitex não são exatamente gramáticas algébricas estendidas, pois elas integram a noção de *transdução*. Essa noção, tomada por empréstimo dos autômatos finitos, significa que uma gramática pode produzir saídas. Por motivo de clareza, serão adotados, apesar de tudo, os termos *gramática* ou *grafo*. Quando uma gramática produzir saídas, o termo *transdutor* será utilizado, por analogia à definição de um transdutor no domínio dos autômatos finitos.

## 5.2 Edição de grafos

### 5.2.1 Importação de um grafo Intex

Para poder utilizar grafos Intex no Unitex, é necessário convertê-los em caracteres Unicode. O processo de conversão é o mesmo realizado nos textos (ver seção 2.2).

ATENÇÃO: Um grafo convertido em caracteres Unicode e utilizado com o Unitex não poderá mais ser usado com o Intex.

Para poder utilizá-lo novamente com o Intex, convertê-lo para texto ASCII, depois abri-lo com um programa de tratamento de texto e substituir a primeira linha:

```
#Unigraph
```

pela linha seguinte :

```
#FSGraph 4.0
```

### 5.2.2 Criação de um grafo

Para criar um grafo, clicar em "New" no menu "FSGraph". Aparecerá, então, a janela da figura 5.2. O símbolo em formato de flecha é o *estado inicial* do grafo e o símbolo redondo contendo um quadrado é o seu *estado final*. A gramática reconhecerá apenas expressões descritas por caminhos que conectem o estado inicial ao estado final.

FIG.5.1 Menu FSGraph

Para criar uma caixa[5], é preciso clicar na janela mantendo a tecla "Ctrl" pressionada. Então, um quadrado azul aparecerá simbolizando a caixa vazia criada (ver figura 5.3). Durante a criação de uma caixa, ela é automaticamente selecionada. Em seguida, o conteúdo da caixa aparecerá na área de texto situada no alto da janela. A caixa criada contém o símbolo <E> que representa a palavra vazia ípsilon. É necessário substituir esse símbolo pelo texto `I+you+he+she+it+we+they` e validar pressionando a tecla "Enter".

Uma caixa contendo sete linhas (ver figura 5.4) será criada. Com efeito, o caractere + serve de separador. A caixa aparece sob a forma de linhas de texto vermelho, pois ela não está, até o momento, conectada a nenhuma outra. Utiliza-se freqüentemente esse tipo de caixa para inserir comentários em um grafo.

Para colocar uma caixa em comunicação com a outra, clicar na *caixa de saída* e depois na *caixa de destino*. Se já houver uma transição entre as duas caixas, a última será apagada. É possível efetuar essa mesma operação clicando primeiramente na caixa de destino e depois na caixa de saída mantendo a tecla "Shift" pressionada. Em nosso exemplo, estando a caixa conectada à etapa inicial e à etapa final do grafo, obtém-se o grafo da figura 5.5:


[5] O termo "caixa(s)" corresponde ao termo "nó(s)" na terminologia geral de grafos.

Fig. 5.2. Grafo vazio

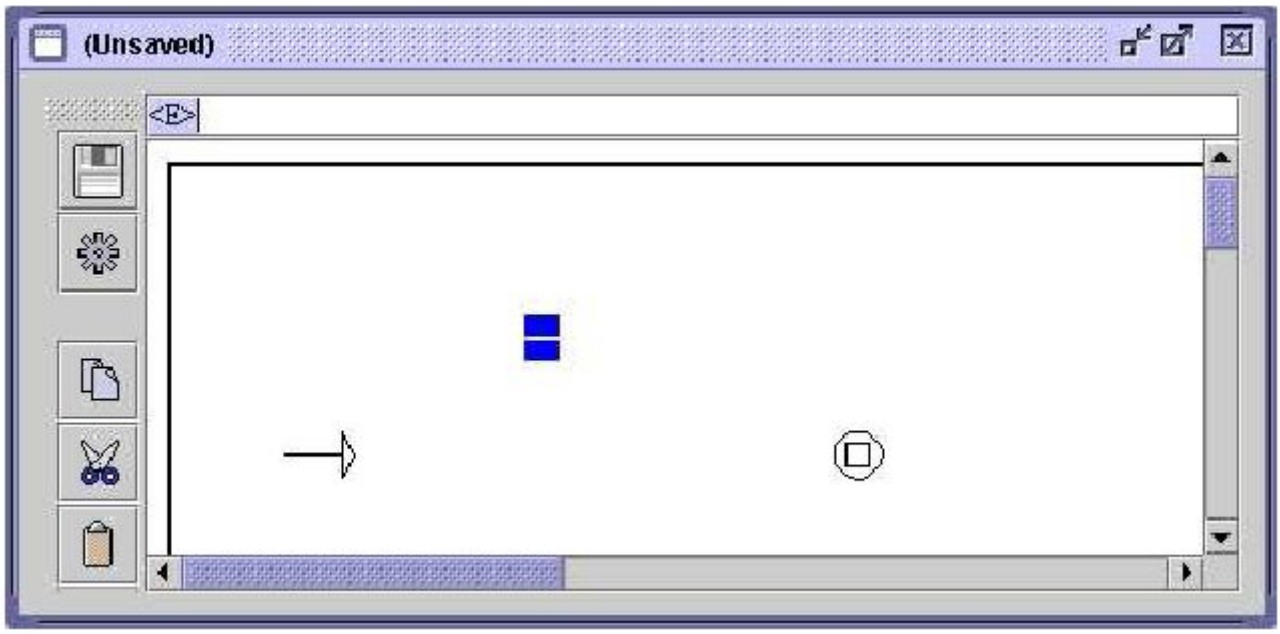



Fig. 5.3. Criação de uma caixa

NOTA: Ao dar um duplo clique em uma caixa, esta entrará em comunicação com ela mesma (ver figura 5.6). Para cancelar, é preciso dar um duplo clique novamente na caixa.

Clicar em "Save as..." no menu "FSGraph" para salvar o grafo. Por definição, o Unitex propõe que se faça a cópia de segurança no subdiretório `Graphs` de seu diretório pessoal. Pode-se verificar se o grafo foi modificado desde o último backup observando se o título da janela contém a notificação `(Unsaved)`.

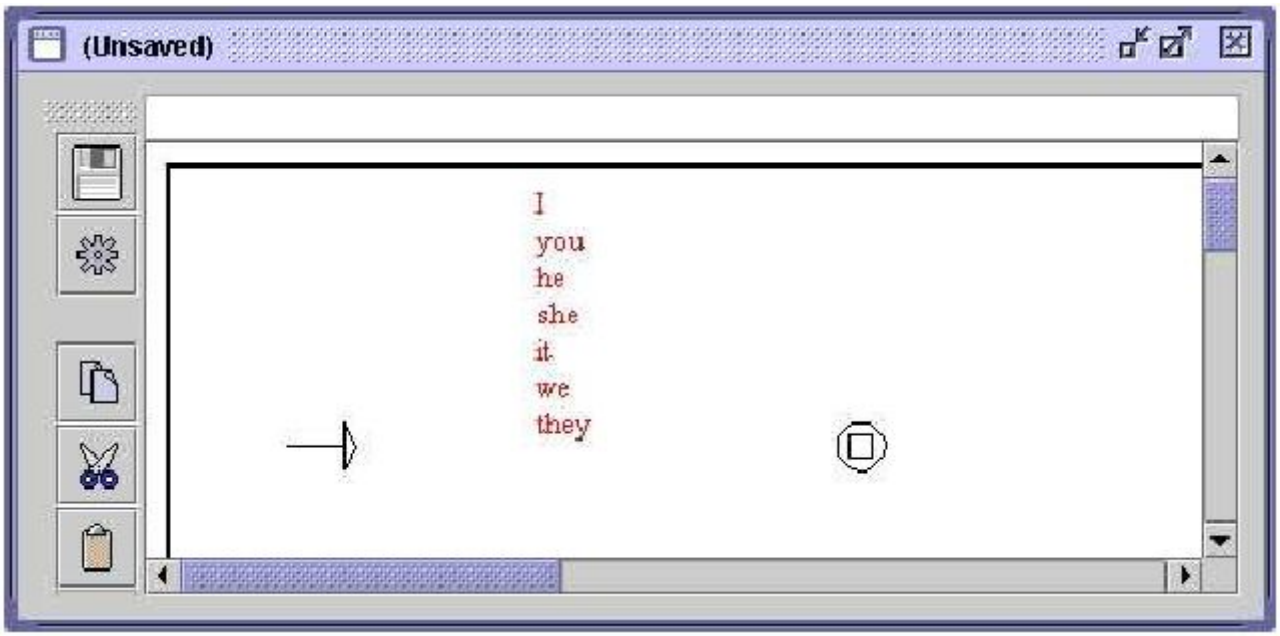



Fig 5.4. Caixa contendo `I+you+he+she+it+we+they`

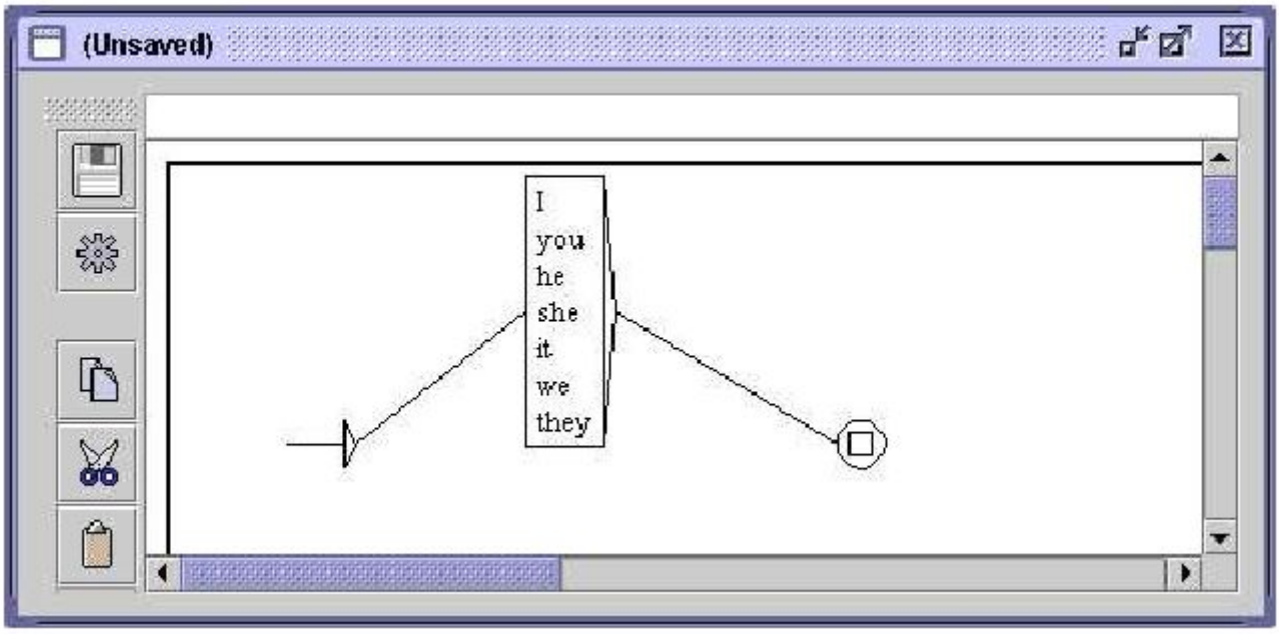


Fig 5.5. Grafo que reconhece pronomes em inglês

### 5.2.3 Subgrafos

Para recorrer a um subgrafo, é necessário indicar em uma caixa o nome dele precedido do caractere `:` . Digitando-se em uma caixa a seguinte entrada:

```
alpha+:beta+gamma+:E:\greek\delta.grf
```

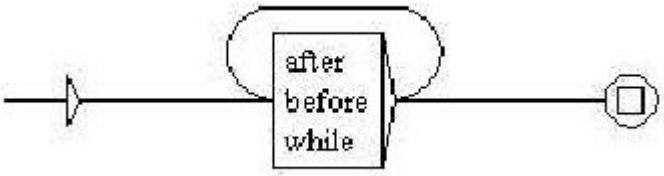



Fig. 5.6 Caixa conectada a ela mesma

Obter-se-á uma caixa similar à da figura 5.7


Fig. 5.7. Grafo recorrendo aos subgrafos `beta` e `delta`

Pode-se indicar o nome completo do grafo (`E:\greek\delta.grf`) ou simplesmente o nome sem o caminho de acesso (`beta`); nesse caso, supõe-se que o subgrafo é se encontre no mesmo diretório que o grafo ao qual faz referência. É desaconselhável utilizar nomes de grafos que comportem caminhos completos, pois isso danifica sua portabilidade. Ocorrendo a utilização de um nome de grafo completo, como é o caso de:

`E:\greek\delta.grf`,

o compilador de grafo emitirá uma advertência (ver figura 5.8)

Ainda em razão da portabilidade, é desaconselhável utilizar \ ou / como separador nos nomes de grafos. Em lugar disso, seria melhor utilizar o caractere `:`, que tem o papel de separador universal, válido para qualquer sistema com o qual se trabalhe.

Pode-se, aliás, observar na figura 5.8 que é esse o separador utilizado internamente pelo compilador de grafo (`E::greek:delta.grf`).


FIG. 5.8 –Aviso para um nome de grafo não transferível

### 5.2.4 Diretório de depósito

Quando se pretende reutilizar uma gramática *X* em uma gramática *Y*, o procedimento usual é recopiar todos os grafos de *X* no diretório onde se situa os grafos de *Y*, o que cria dois problemas:

- o número de gráficos no diretório torna-se rapidamente muito elevado;
- dois grafos não podem ter o mesmo nome.

Para evitar que isto ocorra, é possível estocar a gramática *X* em um diretório específico, chamado *diretório de depósito*. Esse diretório é uma espécie de biblioteca na qual pode-se ordenar os grafos e, em seguida, recorrer a esses mesmos grafos por meio de : : em vez de :. Para utilizar esse mecanismo, é preciso, primeiramente, definir o diretório de depósito no menu "Info>Preferences...>Directories" (ver figura 5.9). Deve-se escolher o diretório no quadro "Graph repository". O diretório de depósito é compatível com a língua de trabalho, portanto não é obrigatória a utilização do mesmo diretório para várias línguas.

FIG. 5.9 – Configuração do diretório de depósito

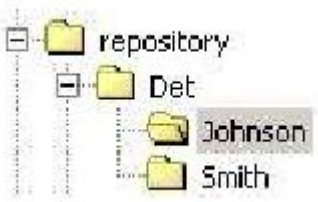



FIG. 5.10 – Exemplo de diretório de depósito


FIG. 5.11 – Referência a um grafo do diretório de depósito

Supõe-se que se tenha uma arborescência como aquela da figura 5.10. Se desejar remeter ao grafo DET que se localiza no subdiretório `Johnson`, será preciso utilizar a nomeação `::Det:Johnson:DET` (ver figura 5.11[6]).

DICA: para evitar colocar nos grafos um caminho complicado como `::Det:Johnson:DET`, é possível criar um grafo nomeado `DET` que poderá ser colocado na raiz do diretório de depósito (no caso: `D:\repository\DET.grf`). Esse grafo conterá simplesmente uma referência ao grafo `::Det:Johnson:DET`. Pode-se, dessa forma, colocar nos grafos um simples referência a `::DET`. Isto permite: 1) não ter nomes complicados e 2) poder modificar os grafos do diretório de depósito sem ter de modificar todos os outros grafos. De fato, bastará colocar em dia o grafo localizado na raiz do diretório de depósito.

As referências aos subgrafos são representadas nas caixas por linhas cujo plano de fundo pode ser tanto cinza quanto marrom no caso de subgrafos a serem buscados no diretório de depósito. No Windows, pode-se abrir um subgrafo clicando sobre a

[6] Por uma questão de clareza, as referências aos grafos do diretório de depósito estão em marrom em vez de cinza.

linha acinzentada e pressionando ao mesmo tempo a tecla Alt. No Linux, a combinação <Alt+Ctrl> é interceptada pelo sistema. Para abrir um subgrafo, é preciso clicar sobre seu nome pressionando simultaneamente o botão esquerdo e direito do *mouse*.

### 5.2.5 Manipulação das caixas

É possível selecionar várias caixas por meio do *mouse*. Para isso, clicar e movimentar o *mouse* sem soltar o botão. Quando soltar o botão todas as caixas compreendidas no retângulo de seleção serão selecionadas e, então, serão apresentadas em branco sobre um fundo azul:

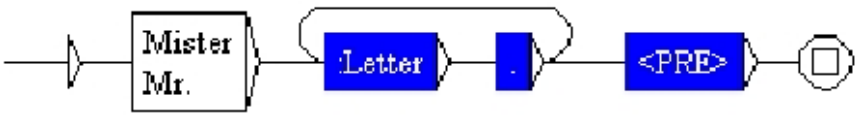



FIG. 5.12 - Seleção de várias caixas

Quando as caixas estiverem selecionadas, é possível deslocá-las clicando e movimentando o cursor sem soltar o botão. Para anular a seleção, clicar sobre uma área vazia do grafo; se clicar sobre uma caixa, todas as caixas da seleção serão associadas àquela.

É possível copiar e colar várias caixas. Para isso, é preciso selecioná-las e pressionar <Ctrl+C> ou clicar sobre "Copy" no menu "Edit". A seleção múltipla está nesse momento na área de transferência do Unitex. Pode-se, então, colar essa seleção pressionando <Ctrl+V> ou clicando sobre "Paste" no menu "Edit".

NOTA: é possível colar uma seleção múltipla em um outro grafo diferente daquele em que foi efetuada a cópia.

Para apagar as caixas, selecioná-las e apagar o texto que elas contêm. Para isso, deve-se apagar o texto presente na área de texto localizada no alto da janela e validar com a tecla Enter. O estado inicial e o estado final não podem ser apagados.

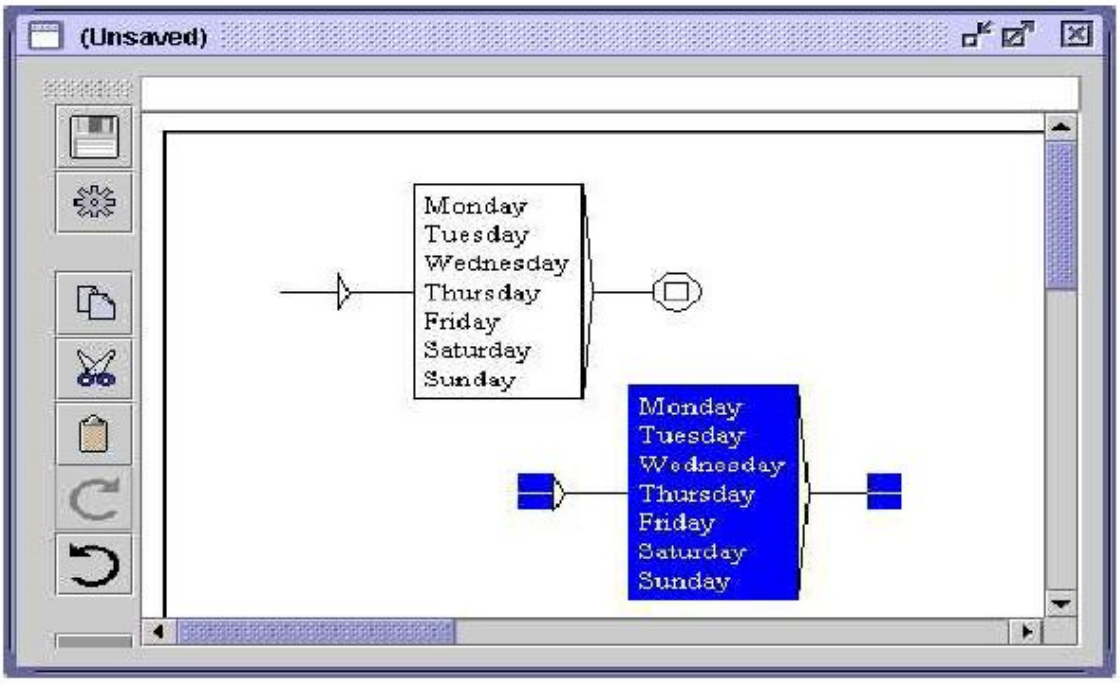



FIG. 5.13 – Copiar e colar de uma seleção múltipla

### 5.2.6 Saída

É possível associar uma saída a uma caixa. Para isso, utilizar o caractere especial /. Todos os caracteres localizados a sua direita serão considerados como fazendo parte da saída. Assim, o texto `one+two+three+/number` origina a caixa da figura 5.14.

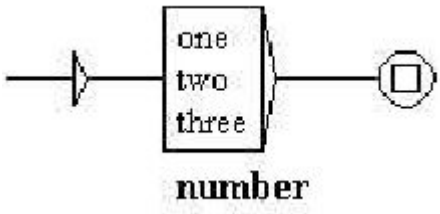


FIG. 5.14 – Exemplo de saída

A saída associada a uma caixa é representada em negrito sob aquela.

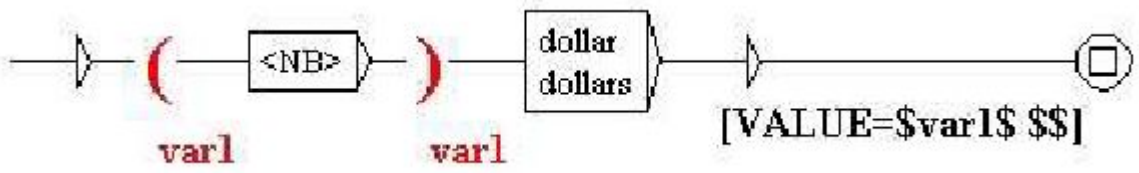



FIG. 5.15 – Utilização de uma variável var1

### 5.2.7 Utilização das variáveis

Pode-se selecionar partes do texto identificado por uma gramática por meio de variáveis. Para associar uma variável var1 a uma parte de uma gramática, utilizar os símbolos especiais `$var1(` (e `$var1)`) para definir o início e o fim, respectivamente, da área para estocar. É preciso criar duas caixas contendo uma `$var1(` (e outra `$var1)`). Essas caixas não devem conter nada além que o nome da variável precedido por `$` e seguido por um parêntese. Associar em seguida essas caixas à área da gramática desejada. No grafo da figura 5.15, pode-se identificar uma seqüência que se inicia por um número que se estoca em uma variável nomeada var1, seguido de `dollar` ou `dollars`.

Os nomes de variáveis podem conter letras latinas não acentuadas, minúsculas ou maiúsculas, assim como dígitos e caractere _ (underline). O Unitex diferencia as letras minúsculas das maiúsculas.

Quando uma variável for assim definida, pode-se utilizá-la nas saídas delimitando seu nome com o caractere `$`. Se desejar escrever em saída o caractere `$`, é preciso repeti-lo como no caso na figura 5.15. A gramática da figura 5.16 identifica uma data formada por um mês e por um ano, e produz em saída a mesma data, mas na ordem ano-mês.

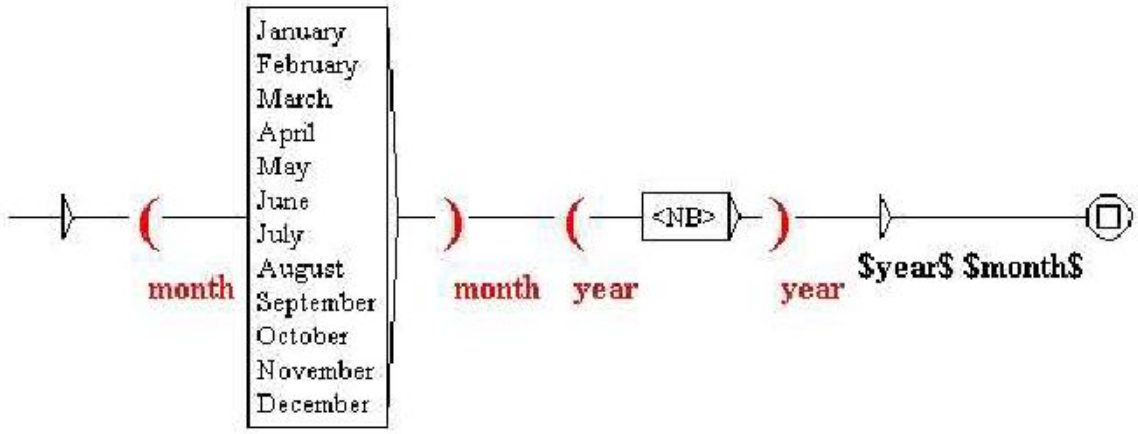



FIG. 5.16 – Inversão do mês e do ano em uma data

### 5.2.8 Cópia de listas

Pode ser prático copiar e colar uma lista de palavras ou de expressões a partir de um editor de texto para uma caixa em um grafo. Para evitar a cópia manual de cada item, o Unitex propõe um mecanismo de cópia de listas. Para utilizá-la, selecionar a lista no editor de texto e copiá-la por meio de <Ctrl+C > ou pela função de cópia integrada ao editor. Deve-se criar, em seguida, uma caixa no grafo, e utilizar <Ctrl+ V> ou o comando "Paste" do menu "Edit" para colá-la na caixa. Dessa forma, aparecerá a janela da figura 5.17.

FIG. 5.17 – Seleção de contexto para a cópia de uma lista

Essa janela permite definir os contextos à esquerda e à direita que serão acrescentados automaticamente a cada item da lista. Por definição, esses contextos são vazios. Se aplicar os contextos <et.V> à lista seguinte:

*eat*

*sleep*

*drink*

*play*

*read*

obtém-se a caixa da figura 5.18:

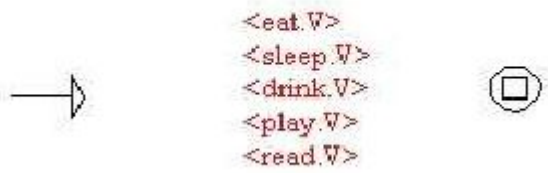



FIG. 5.18 - Caixa obtida pela cópia de uma lista com adição de contextos

### 5.2.9 Símbolos especiais

O editor de grafos do Unitex interpreta de maneira particular os seguintes símbolos:

" + : / < > # \

A tabela 5.1 resume a significação para o Unitex destes símbolos, assim como a ou as formas de identificar estes caracteres nos textos.

| Caractere | Significação | Codificação |
|---|---|---|
| “ | as aspas delimitam seqüências que não devem ser nem interpretadas pelo Unitex, nem sofrer variáveis de caixa | \ “ |
| + | o + separa as diferentes linhas das caixas | “+” |
| : | os : servem para introduzir uma referência a um subgrafo | “:” ou \ : |
| / | a / indica o início da saída em uma caixa | \ / |
| < | o < indica o início de motivo ou de uma meta | “<” ou \< |
| > | o > indica o fim de um motivo ou de uma meta | “>” ou \ > |

| | | |
|---|---|---|
| # | a # serve para impedir a presença de espaço | "#" |
| \ | a \ serve para despecializar a maioria dos caracteres especiais | \\ |

TAB. 5.1 – Codificação de símbolos especiais no editor de grafos

### 5.2.10 Comandos da barra de ícones

As barras de ícones presente à esquerda dos grafos contem atalhos para alguns comandos e permite manipular as caixas de um grafo utilizando as "ferramentas". Essa barra de ícones pode ser deslocada clicando sobre a área "rugosa". Ela pode ser até dissociada do gráfico e, então, aparecer como uma janela separada (ver figura 5.19). Neste caso, ao fechar essa janela a barra de ícones volta novamente a sua posição inicial. Cada grafo tem sua própria barra de ícones.


FIG. 5.19 – Barra de ícones

Os dois primeiros ícones são atalhos que permitem salvar e compilar o grafo. Os três seguintes correspondem às operações "Copiar", "Recortar" e "Colar". Os dois seguintes correspondem às operações "Redo" e "Undo" que permitem refazer ou desfazer as operações. O último ícone em forma de chave é um atalho para a configuração da imagem do grafo.

Os outros seis ícones correspondem a comandos de edição das caixas. O primeiro em forma de seta branca, corresponde ao modo de edição padrão das caixas. Os outros cinco correspondem a ferramentas. Para utilizar uma ferramenta, é preciso clicar sobre o ícone correspondente: o cursor do *mouse* mudará de forma e os cliques do *mouse* serão, dessa maneira, interpretados de maneira particular. Eis a descrição das ferramentas, da esquerda para a direita:

– criar caixas: cria uma caixa vazia no lugar onde for clicado;

– apagar caixas: apaga a caixa sobre a qual for clicada;

– combinar caixas a uma outra caixa: esta ferramenta permite selecionar uma ou várias caixas e, também, a ou as combinar a uma outra. Diferentemente do modo padrão, a ou as transições que vão ser criadas são apresentadas durante o deslocamento do ponteiro do *mouse*;

– combinar caixas a uma outra caixa no sentido inverso: esta ferramenta efetua a mesma ação que a precedente, mas lê no sentido inverso as caixas selecionadas na caixa clicada;

– abrir um subgrafo: abre-se um subgrafo quando clicar sobre a linha acinzentada correspondente em uma caixa.

## 5.3 Opções de apresentação

### 5.3.1 Ordenação das linhas de uma caixa

Pode-se ordenar o conteúdo de uma caixa selecionando-a e clicando sobre "Sort Node Label" no submenu "Tools" do menu "FSGraph". Essa ordenação não faz remissão ao programa `SortTxt`. Trata-se de uma ordenação básica que classifica as linhas segundo a ordem dos caracteres no padrão Unicode.

### 5.3.2 Zoom

O submenu "Zoom" permite escolher a escala na qual será apresentado o grafo.

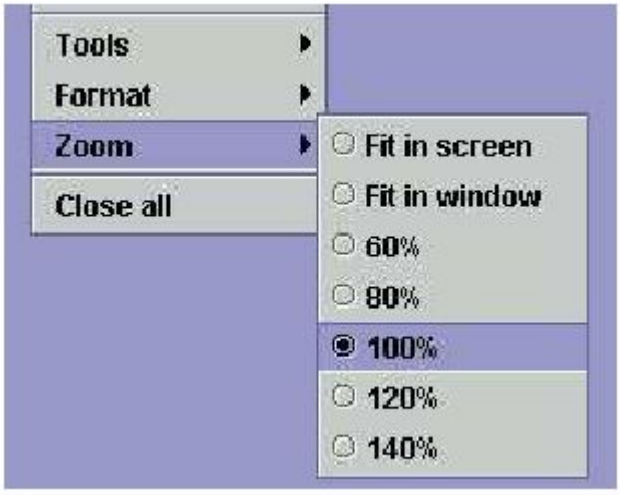



FIG. 5.20 – Submenu Zoom

A opção "Fit in screen" alonga ou estreita o grafo para deixá-lo do tamanho da tela. A opção "Fit in window" ajusta o grafo para que ele seja completamente exibido na janela.

### 5.3.2 Antialiasing

O antialiasing é um efeito gráfico que permite evitar o efeito de pixelização. É possível ativar esse efeito clicando sobre "Antialiasing..." no submenu "Format". A figura 5.21 apresenta dois grafos: um exibido de forma padrão (grafo de cima) e outro com o antialiasing (grafo de baixo).

Esse efeito torna a execução do Unitex lenta. Aconselha-se não utilizá-lo se a máquina for pouco avançada.

### 5.3.3 Alinhamento das caixas

Para obter grafos harmoniosos, é útil alinhar as caixas tanto horizontalmente quanto verticalmente. Para isso, selecionar as caixas a serem alinhadas e clicar sobre "Alignment..." no submenu "Format" do menu "FSGraph" ou pressionar <Ctrl + M>. Então, aparecerá a janela da figura 5.22.

As possibilidades de alinhamento horizontal são:

- Top: as caixas são alinhadas sobre a caixa que está mais alta;
- Center: as caixas são todas centralizadas sobre um mesmo eixo;
- Bottom: as caixas são alinhadas sobre a caixa que está mais baixa.

As possibilidades de alinhamento vertical são:

- Left: as caixas são alinhadas sobre a caixa mais à esquerda;
- Center: as caixas são todas centralizadas sobre um mesmo eixo;
- Right: as caixas são alinhadas sobre a caixa mais à direita.

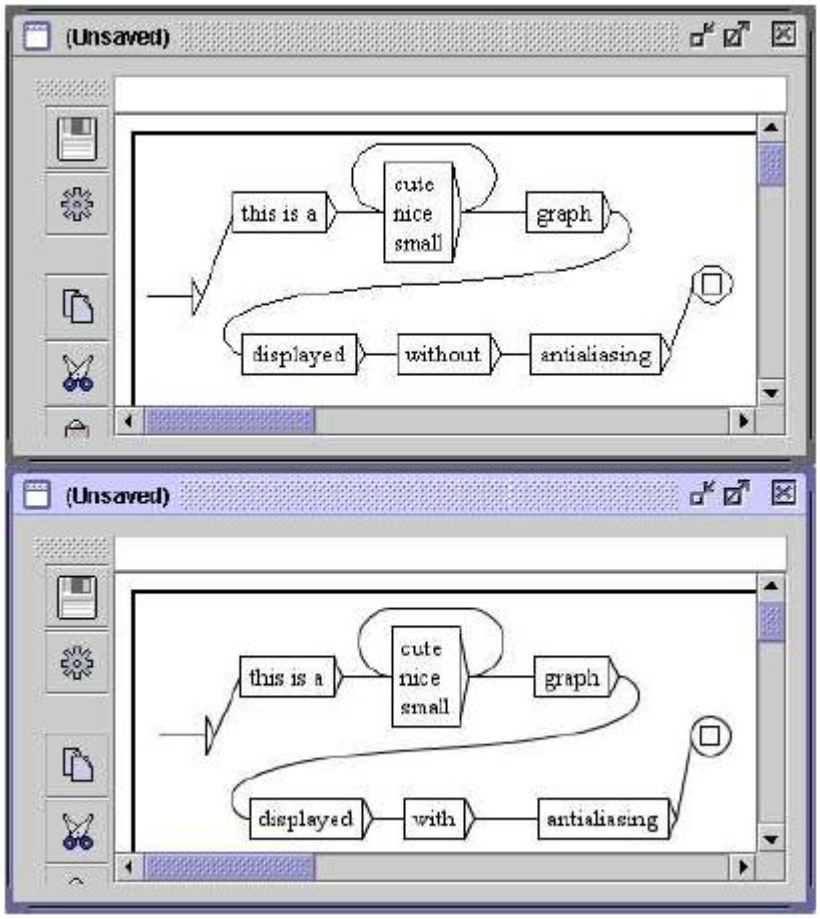



FIG. 5.21 – Exemplo de antialiasing

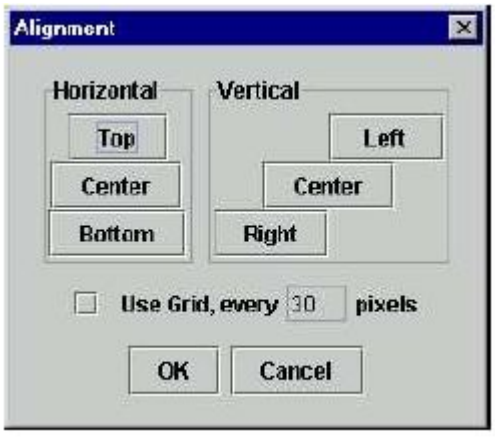


FIG. 5.22 – Janela de alinhamento

A figura 5.23 apresenta um exemplo de alinhamento. O grupo de caixas localizado à direita é uma cópia das caixas da esquerda que foi alinhada verticalmente à esquerda.

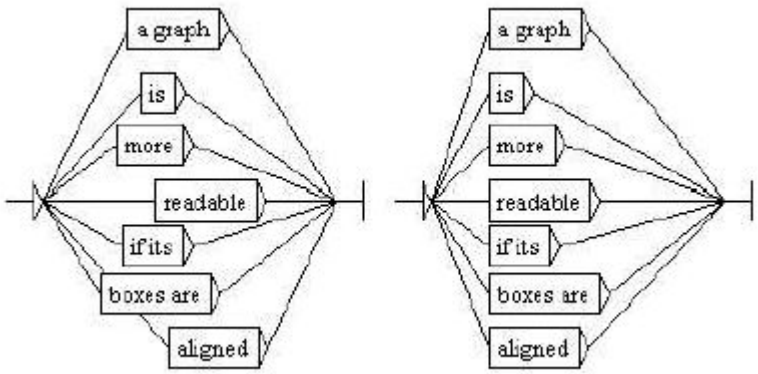



FIG. 5.23 – Exemplo de alinhamento vertical esquerdo

A opção "Use Grid" da janela de alinhamento permite colocar uma grade no plano de fundo do grafo. Isso permite alinhar aproximativamente as caixas.

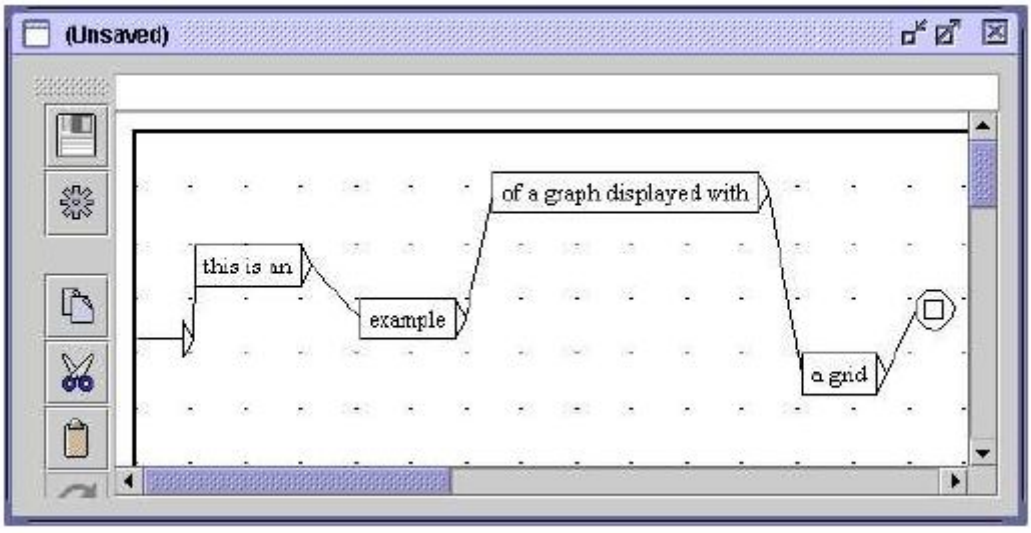


FIG. 5.24 – Exemplo de utilização de uma grade

### 5.3.4 Apresentação, fontes e cores

Pode-se configurar a imagem de um grafo pressionando <Ctrl +R> ou clicando sobre "Presentation..." no submenu "Format" do menu "FSGraph", o que faz com que seja exibida a janela da figura 5.25.

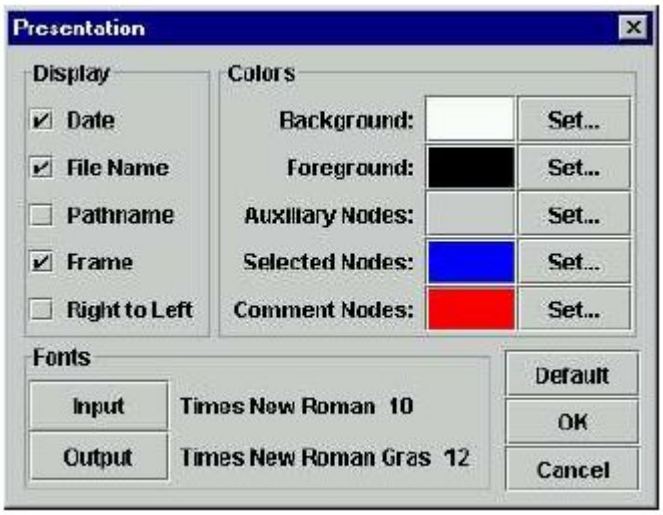



FIG. 5.25 – Configuração da imagem de um grafo

As configurações de fonte são:

- Input: fonte utilizada nas caixas, assim como na área de texto onde se edita o conteúdo das caixas;
- Output: fonte utilizada para exibir as saídas das caixas.

As configurações de cor são:

- Background: cor de fundo;
- Foreground: cor utilizada para o texto e para o desenho das caixas;
- Auxiliary Nodes: cor das caixas remetendo aos subgrafos;
- Slected Nodes: cor utilizada para desenhar as caixas quando elas estiverem selecionadas;
- Comment Nodes: cor utilizada para desenhar as caixas que não estão associadas a nenhuma outra.

As outras configurações são:

- Date: exibição da data atual no canto inferior esquerdo do grafo;
- File Name: exibição do nome do grafo no canto inferior esquerdo do grafo.
- Pathame: exibição do nome do grafo com seu caminho completo no canto inferior esquerdo do grafo. Esta opção só tem efeito se a opção "File Name" estiver selecionada;
- Frame: desenha um quadro em torno do grafo;
- Right ou Left: inverte o sentido de leitura do grafo (ver exemplo da figura 5.26)

Pode-se reconstituir as configurações por definição clicando sobre o botão "Default". Se clicar sobre o botão "OK", só o grafo atual será modificado. Para modificar as preferências por definição de uma língua, clicar sobre "Preferences" no menu "Info" e escolher a aba "Graph Presentation". A janela de configuração das preferências tem uma opção suplementar concernente ao antialiasing (ver figura 5.27). Essa opção permite ativar o antialiasing por definição para todos os grafos da língua atual. É preferível não ativar essa opção se o computador não for muito potente. Há uma outra opção suplementar que permite definir a posição da barra de ícones.


FIG. 5.26 - Grafo que deve ser lido da direita para a esquerda.

NOTA: a opção "Right to Left" não é retomada pela janela de configuração geral dos grafos. De fato, os grafos de uma língua adotam, por definição, a orientação do texto definido por essa língua na aba "Text Presentation" da janela de preferências (ver figura 4.7, página 62).

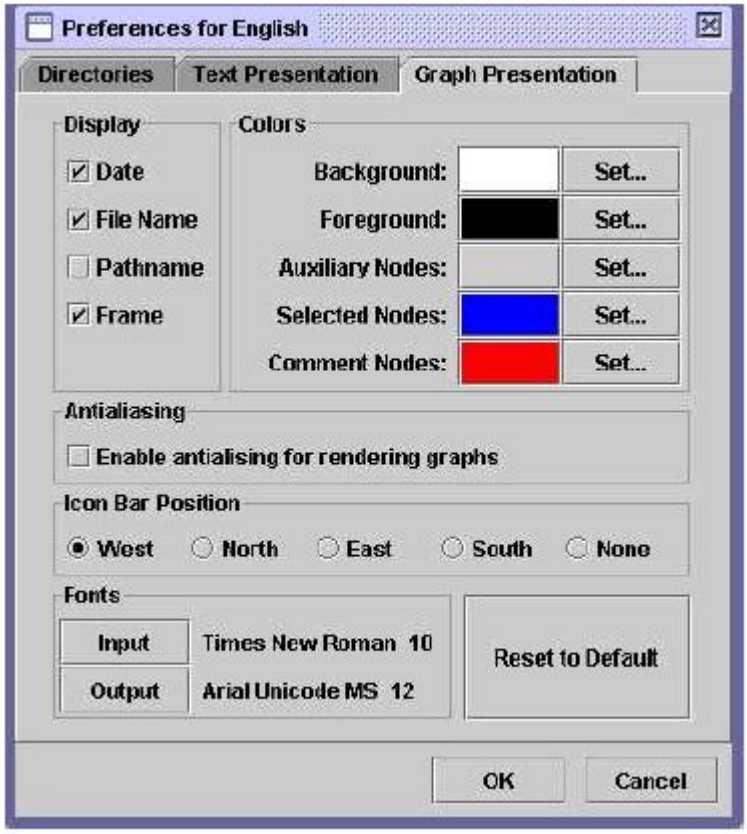



FIG. 5.27 – Configuração das preferências por definição

## 5.4 Os grafos fora do Unitex

### 5.4.1 Inserção de um grafo em um documento

Para inserir um grafo em um documento é preciso transformá-lo em imagem. Para isso, um primeiro método consiste em salvar o grafo como imagem no formato PNG. Para isso, ir até o menu "FSGraph" e clicar em "Save as...". Em seguida, escolher o tipo de arquivo PNG. Deste modo, será obtida uma imagem pronta para ser integrada a um documento ou para ser editada em um software de edição de imagens. Para tornar a imagem mais uniforme, pode-se ativar o antialiasing para o grafo desejado.

O segundo método consiste em fazer uma captura de tela:

No Windows:

Em seguida, pressionar a tecla "Print Screen" do teclado, que deve se encontrar próxima à tecla F12. Abrir o programa Paint no menu "Acessórios" do Windows.

Pressionar <Ctrl+V>. O `Paint` deverá dizer se a imagem contida na área de transferência é muito grande e deverá perguntar se você gostaria de aumentar a imagem[7]. Clicar em "Sim". Agora é possível editar a imagem da tela. Selecionar a área desejada. Para isso, passar para o modo de seleção clicando no retângulo pontilhado que se encontra no canto superior esquerdo da janela. Agora é possível selecionar uma área da imagem com o mouse. Quando a área estiver selecionada, pressionar <Crtl+C>. A seleção ficará na área de transferência e só será preciso ir até o documento e pressionar <Crtl+V> para colar a imagem.

No Linux:

Fazer uma captura de tela (com o programa `xv`, por exemplo). Em seguida, recortar a imagem com um editor gráfico (`TheGimp`, por exemplo) e colar a imagem no documento, do mesmo modo realizado em Windows.

### 5.4.2 Impressão de um grafo

Pode-se imprimir um grafo clicando em "Print..." no menu "FSGraph" ou pressionando <CTRL+P>.

ATENÇÃO: é preciso ter certeza de que a configuração de orientação da página para impressão (retrato ou paisagem) corresponde à orientação do grafo.

É possível definir preferências de impressão clicando em "Page Setup" no menu "FSGraph". É possível, também, imprimir todos os grafos que estão abertos clicando em "Print All...".

# Capítulo 6

# Utilização avançada dos grafos

[7] - A frase parece um pouco estranha, pois se a imagem já está grande, o Paint não deveria perguntar se você gostaria de aumentá-la. Talvez o correto seria se ele perguntasse se você gostaria de aumentar o *bitmap*.

## 6.1 Os tipos de grafos

O Unitex pode manipular vários tipos de grafos que correspondam às seguintes utilizações: flexão automática de dicionários, pré-tratamento de textos, normalização dos autômatos de texto, grafos dicionários, busca por padrões, levantamento de ambigüidades e geração automática de grafos. Esses diferentes tipos de grafos não são interpretados da mesma maneira pelo Unitex. Certas coisas, como as saídas, são permitidas para certos tipos de grafos e proibidas para outros. Além disso, conforme o tipo de grafo, os símbolos especiais não são os mesmos. Esta seção apresenta, então, cada um dos tipos de grafos especificando suas particularidades.

### 6.1.1 Grafos de flexão

Um grafo de flexão descreve as variações morfológicas associadas a uma classe de palavras, associando códigos flexionais a cada variante. Os caminhos de tal grafo descrevem as modificações a serem aplicadas às formas canônicas ao passo que as saídas contêm as informações flexionais que serão produzidas.

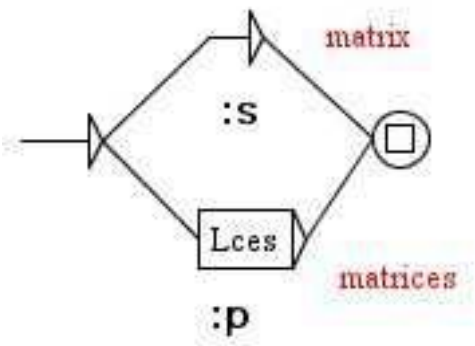



FIG. 6.1. Exemplo de gramática de flexão

Os caminhos podem conter operadores e letras. Os operadores possíveis são representados pelos caracteres `L`, `R`, `C` e `D`. As letras são todos os caracteres que não são operadores. O único símbolo especial autorizado é a palavra vazia <E>. Não é possível fazer remissão aos dicionários em um grafo de flexão. Entretanto, é possível recorrer a subgrafos.

As saídas são concatenadas para produzir uma cadeia de caracteres. Essa cadeia é, em seguida, concatenada à linha de dicionário produzida. As saídas com variáveis não têm sentido em um grafo de flexão.

O conteúdo de um grafo de flexão é manipulado sem nenhuma variação de caixa: as letras minúsculas continuam minúsculas, idem para as maiúsculas. Além disso, a ligação de duas caixas é estritamente equivalente à concatenação de seus conteúdos provida da concatenação de suas saídas (ver figura 6.2).

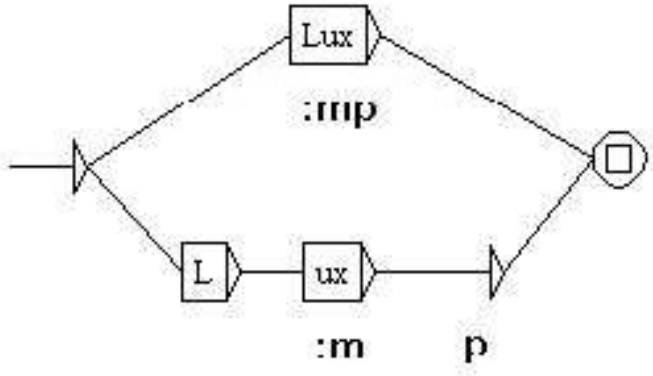



FIG. 6.2. Dois caminhos equivalentes em uma gramática de flexão

Os grafos de flexão devem ser compilados antes de serem utilizados pelo programa de flexão.

Para mais detalhes, ver seção 3.4.

### 6.1.2 Grafos de pré-processamento

Os grafos de pré-processamento são destinados a serem aplicados aos textos antes que estes sejam recortados em unidades lexicais. Esses grafos podem ser utilizados para inserir ou substituir seqüências nos textos. As duas utilizações usuais desses grafos são a normalização de formas não ambíguas[8] e o recorte em frases.

A interpretação desses grafos no Unitex é muito próxima da interpretação dos grafos sintáticos utilizados para a busca por padrões. As diferenças são as seguintes:

- é impossível utilizar o símbolo especial <^>, que reconhece uma quebra de página;
- é impossível fazer remissão aos dicionários;
- é impossível utilizar os filtros morfológicos;
- é impossível utilizar contextos.

As figuras 2.9 (página 23) e 2.10 (página 25) mostram exemplos de grafos de pré-tratamento.

### 6.1.3 Grafos de normalização do autômato do texto

Os grafos de normalização do autômato do texto permitem a normalização das formas ambíguas. De fato, eles podem descrever várias etiquetas para uma mesma forma. Essas etiquetas são, em seguida, inseridas no autômato do texto, explicitando, assim, as ambigüidades. A figura 6.3 mostra um extrato do grafo de normalização utilizado para o francês.

[8] - O correto não seria formalizar formas "ambíguas" em vez de "não ambíguas"?

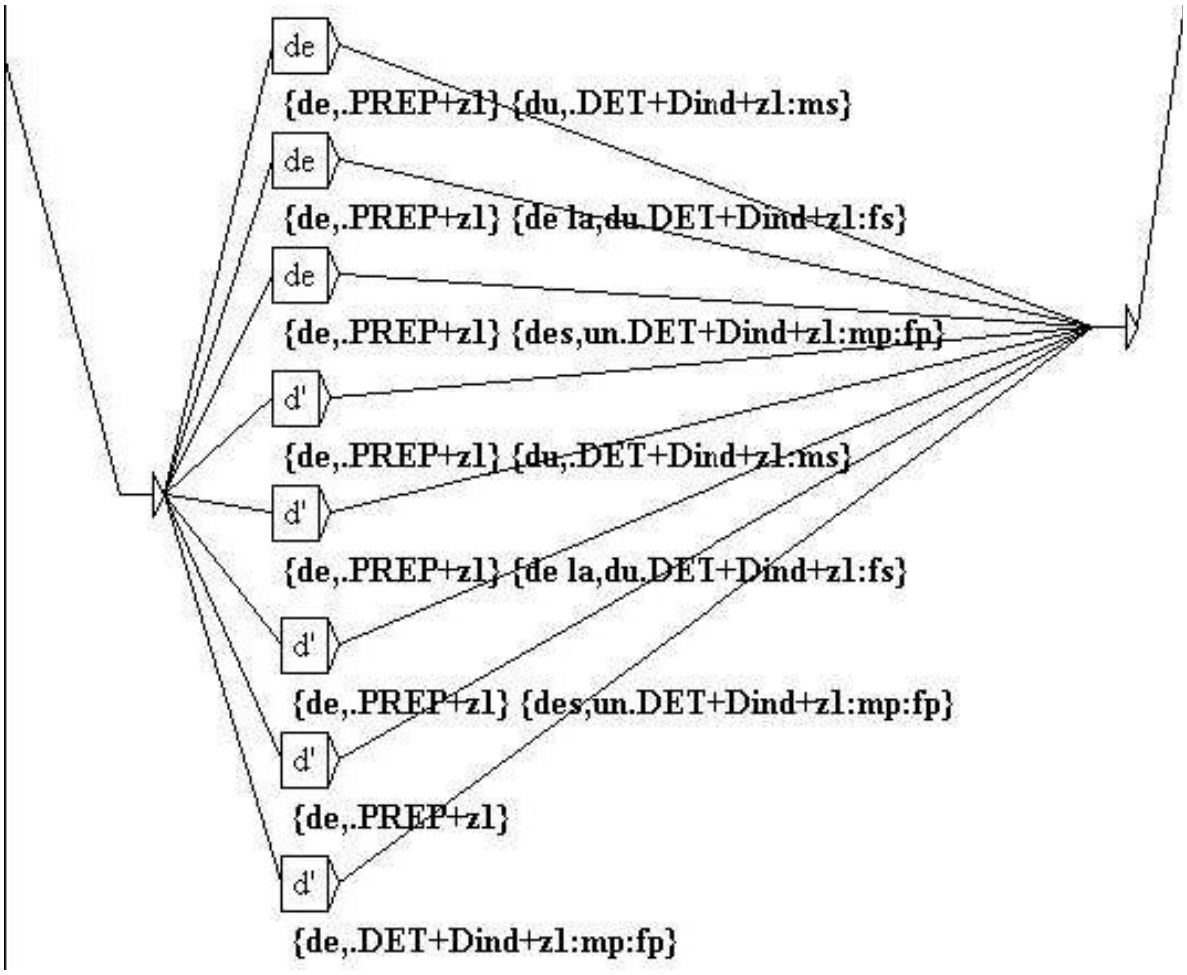



FIG. 6.3. Extrato do grafo de normalização utilizado para o francês

Os caminhos descrevem as formas que devem ser normalizadas. As variantes minúsculas e maiúsculas são levadas em conta conforme o seguinte princípio: as letras maiúsculas no grafo só reconhecem as letras maiúsculas no autômato do texto; as letras minúsculas conseguem reconhecer as letras minúsculas ou maiúsculas.

As saídas representam as seqüências de etiquetas que serão inseridas no autômato do texto. Essas etiquetas podem ser entradas de dicionários ou simples cadeias de caracteres. As etiquetas que representam entradas de dicionário devem respeitar o formato das entradas de um DELAF e estar entre os símbolos `{` e `}`. As saídas com variáveis não têm sentido nesse tipo de grafo.

É possível recorrer aos subgrafos. Não é possível fazer remissão aos dicionários para descrever as formas a serem normalizadas. O único símbolo especial reconhecido nesse tipo de grafo é a palavra vazia `<E>`. Os grafos de normalização de formas ambíguas devem ser compilados antes de poderem ser utilizados.

### 6.1.4 Grafos dicionários

Os grafos dicionários, já apresentados na seção 3.6.3, são grafos sintáticos aplicados pelo programa `Dico` de maneira à gerar entradas de dicionários. Dado que o `Dico` utiliza o mecanismo do programa `Locate` para aplicar esses grafos, eles têm exatamente as mesmas características dos grafos sintáticos.

### 6.1.5 Grafos sintáticos

Os grafos sintáticos, também chamados de gramáticas locais, permitem a descrição das chaves de busca sintáticas que poderão, em seguida, ser procuradas nos textos. De todos os tipos de grafos, estes possuem o maior poder de expressão, pois permitem a referência aos dicionários.

As variantes minúsculas/ maiúsculas são autorizadas conforme o princípio descrito mais acima. Contudo, é possível impor os limites de uma caixa colocando uma expressão entre aspas. O emprego de aspas permite também que se determinem espaçamentos. De fato, o Unitex considera por default que um espaço entre duas caixas é possível. Para impor a presença de um espaço, é preciso colocá-lo entre aspas. Para impedir a presença de um espaço, é preciso utilizar o símbolo especial #.

Os grafos sintáticos podem recorrer a subgrafos (ver seção 5.2.3). Eles geram, do mesmo modo, as saídas, inclusive as saídas com variáveis. As seqüências produzidas são interpretadas como cadeias de caracteres que serão inseridas nas concordâncias ou no texto se desejar modificá-lo (ver seção 6.7.3).

Os símbolos especiais suportados pelos grafos sintáticos são os mesmos que os utilizados nas expressões racionais (ver seção 4.3.1).

Os grafos sintáticos podem utilizar contextos (ver seção 6.3).

Não é obrigatório compilar os grafos sintáticos antes de utilizá-los para a pesquisa de chaves de busca. Se um grafo não estiver compilado, o sistema o compilará automaticamente.

### 6.1.6 Gramáticas ELAG

A sintaxe das gramáticas de levantamento de ambigüidades está apresentada na seção 7.3.1, página 120.

### 6.1.7 Grafos parametrizados

Os grafos parametrizados são metagrafos que permitem a geração de uma família de grafos a partir de uma tabela de léxico-gramática. É possível construir grafos parametrizados para qualquer tipo de grafo. A construção e a utilização dos grafos parametrizados serão desenvolvidas no capítulo 8.

## 6.2 Compilar uma gramática

### 6.2.1 Compilação de um grafo

A compilação é a operação que permite que se passe do formato `.grf` para um formato mais fácil de ser manipulado pelos programas do Unitex. Para compilar um

grafo, é preciso abri-lo, depois clicar em "Compile FST2" no submenu "Tools" do menu "FSGrafh". O Unitex abre, então, o programa `Grf2Fst2`, no qual é possível seguir a execução em uma janela (ver figura 6.4).

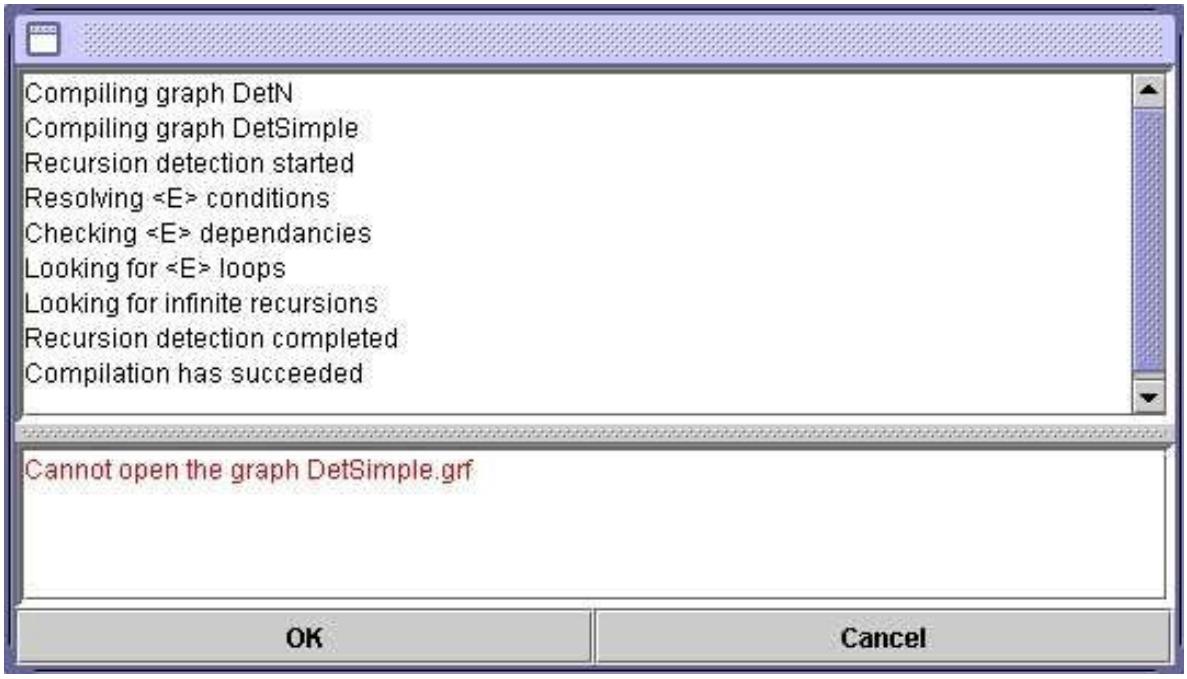



FIG. 6.4 - Janela de compilação

Se o grafo recorre a subgrafos, estes são automaticamente compilados. O resultado é um arquivo `.fst2` que reúne todos os grafos que compõem a gramática. A gramática está, então, pronta para ser utilizada pelos diferentes programas do Unitex.

### 6.2.2 Aproximação por um transdutor de estados finitos

O formato FST2 conserva a arquitetura em subgrafos das gramáticas, o que as diferencia dos estritos transdutores de estados finitos. O programa `Flatten` permite

que se transforme uma gramática FST2 em um transdutor de estados finitos quando isso é possível, ou senão que se construa uma aproximação entre eles. Essa função permite, assim, que se obtenha objetos mais simples de serem manipulados e nos quais podem ser aplicados todos os algoritmos clássicos sobre os autômatos.

Para compilar e transformar, assim, uma gramática, selecionar o comando "Compile & Flatten FST2" no submenu "Tools" do menu "FSGraph". A janela da figura 6.5 permite que se configure a operação de aproximação.

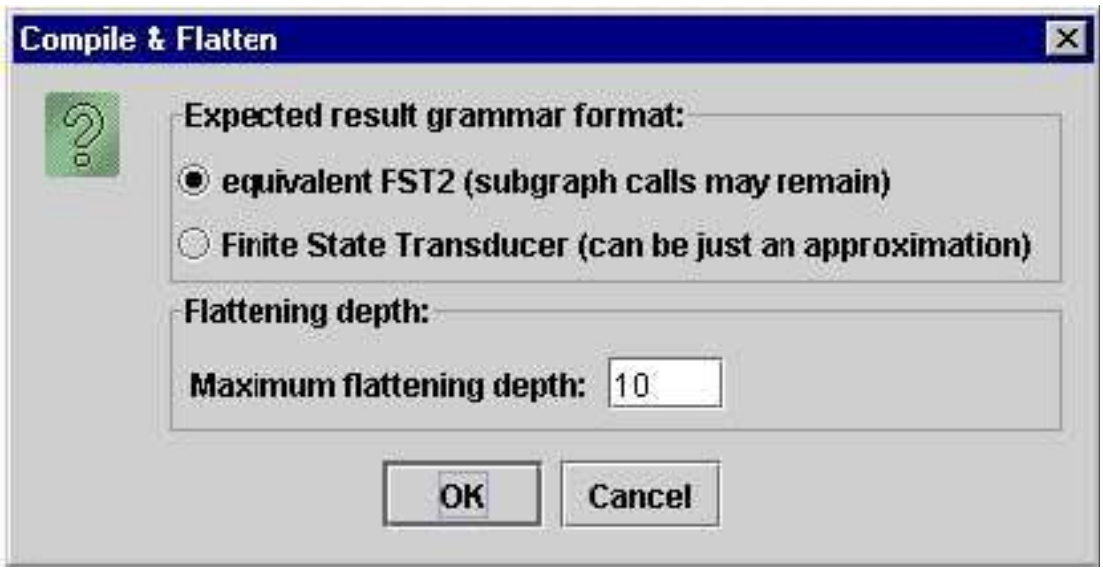



FIG. 6.5 Configuração da aproximação de uma gramática

A opção "Flattening depth" permite que se precise o nível de imbricação dos subgrafos. Esse valor representa a profundidade máxima além da qual as chamadas a subgrafos não serão mais substituídas pelos próprios subgrafos.

A opção "Expected result grammar format" permite que se determine o comportamento do programa além do limite indicado. Se a opção "Finite State

Trasducer" for selecionada, as chamadas aos subgrafos serão ignoradas depois da profundidade máxima. Essa opção garante, assim, a obtenção de um transdutor de estados finitos, eventualmente não equivalente à gramática de partida. Por outro lado, a opção "equivalent FST2" indica que, depois da profundidade limite, o programa deve deixar as chamadas aos subgrafos como estão. Essa opção garante a estreita equivalência do resultado com a gramática de origem, mas não produz, necessariamente, um trasdutor de estados finitos. Essa opção pode ser utilizada para otimizar certas gramáticas.

Uma mensagem indica, no fim do processo de aproximação, se o resultado é um transdutor de estados finitos ou uma gramática FST2 e, no caso de ser um transdutor, se ele é equivalente à gramática de origem (ver figura 6.6).

### 6.2.3 Restrições em relação às gramáticas

Com exceção das gramáticas de flexão, uma gramática não pode ter um caminho vazio. Isso significa que o grafo principal de uma gramática não deve poder reconhecer a palavra vazia, mas isso não impede que um subgrafo dessa gramática reconheça o ípsilon.

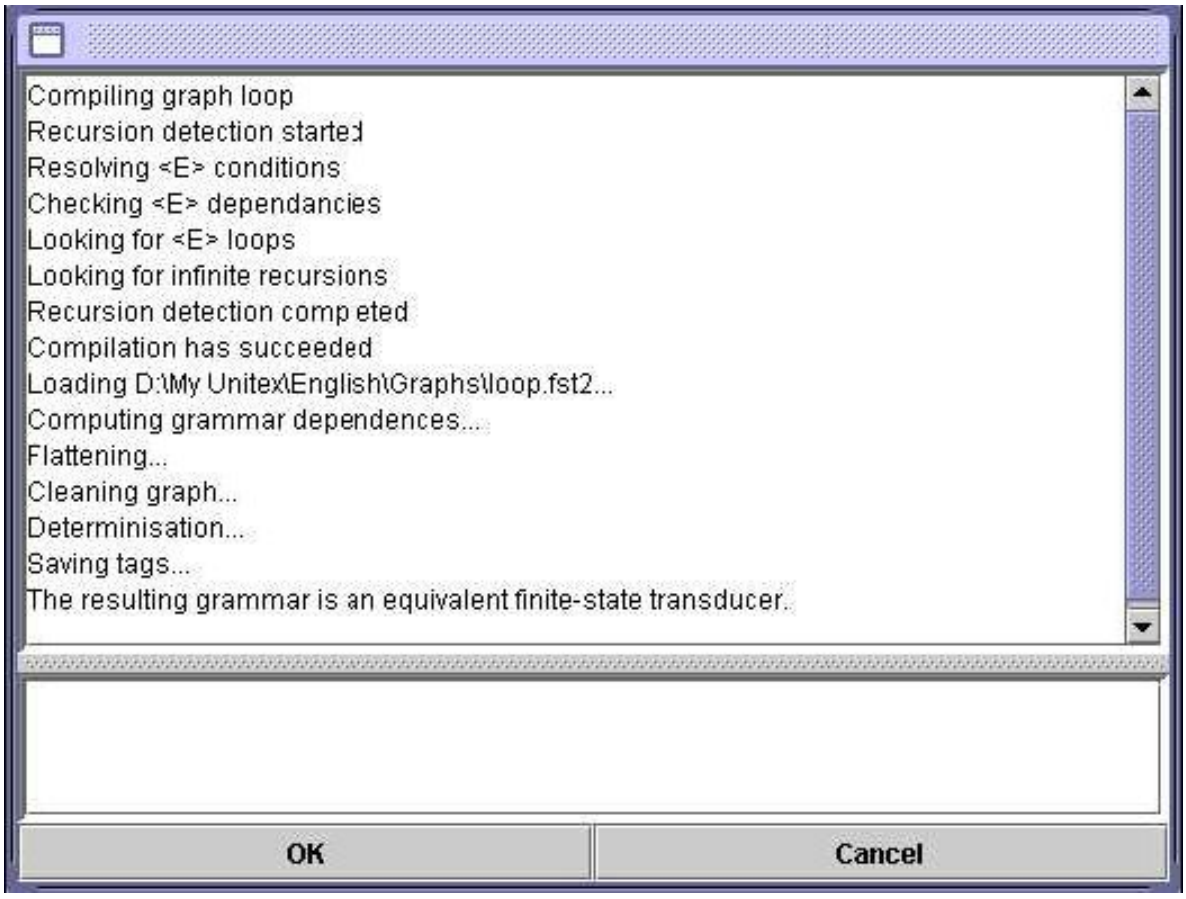



FIG. 6.6. Resultado da aproximação de uma gramática

Não é possível associar uma saída com uma chamada a um subgrafo. Tais saídas são ignoradas pelo Unitex. É preciso, então, utilizar uma caixa vazia situada imediatamente à esquerda da chamada ao subgrafo para obter a saída (ver figura 6.7).

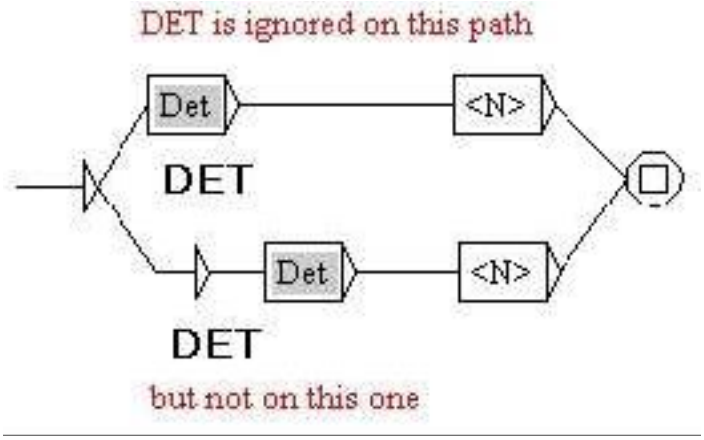



FIG. 6.7 - Como associar uma saída com uma chamada de subgrafo

As gramáticas também não devem comportar laços infinitos, pois os programas do Unitex nunca conseguirão terminar a exploração de tais gramáticas. Esses laços podem existir devido às transições etiquetadas pela palavra vazia ípsilon ou às chamadas de subgrafos recursivos.

Os laços devido à transições pela palavra vazia podem ter duas origens, sendo que a primeira delas está ilustrada na figura 6.8.

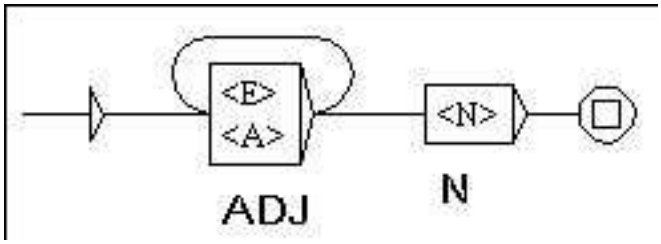



FIG. 6.8 - Laço infinito devido a uma transição pela palavra vazia com saída

Esse tipo de laço se deve ao fato de que uma transição por palavra vazia não pode ser eliminada automaticamente pelo Unitex quando ela possuir uma saída. Assim, a transição pela palavra vazia da figura 6.8 não será suprimida e provocará um laço infinito.

A segunda categoria de laço por ípsilon concerne as chamadas a subgrafos que podem reconhecer a palavra vazia. Esse caso está ilustrado pela figura 6.9: se o subgrafo `Adj` reconhecer o ípsilon, tem-se um laço infinito que o Unitex não pode eliminar.


FIG. 6.9. Laço infinito devido a uma chamada a um subgrafo que reconhece o ípsilon

A terceira possibilidade de laço infinito concerne as chamadas recursivas a subgrafos. Consideremos os grafos `Det` e `DetCompose` da figura 6.10. Cada um desses grafos pode chamar o outro *sem ler nada no texto.* O fato de que nenhum dos dois grafos comporta etiqueta entre o estado inicial e a chamada a um outro grafo é capital. De fato, se houvesse ao menos uma etiqueta diferente de ípsilon entre o começo do grafo `Det` e a chamada ao `DetCompose`, isso significaria que os programas do Unitex que exploram o grafo `Det` deveriam ler a chave de busca descrita por essa etiqueta no texto antes de chamar recursivamente o `DetCompose`. Nesse caso, os programas só poderiam entrar em looping infinito se eles encontrassem uma infinidade de vezes a chave no texto, o que pode não acontecer.

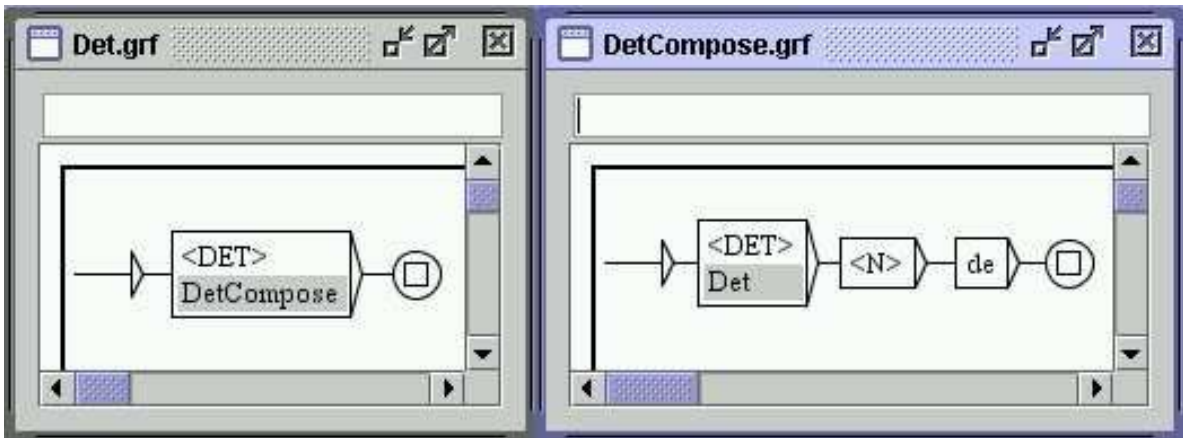



FIG. 6.10. Laço infinito devido aos grafos que recorrem um ao outro

### 6.2.4 Detecção de erros

Para evitar que os programas se bloqueiem ou travem, o Unitex efetua automaticamente uma detecção de erros no momento da compilação dos grafos. O compilador de grafos verifica que o grafo principal não reconhece a palavra vazia e busca todas as formas de laços infinitos. Se um erro é encontrado, uma mensagem de erro aparece na janela de compilação. A figura 6.11 mostra a mensagem obtida quando se tenta compilar o grafo `Det` da figura 6.10.

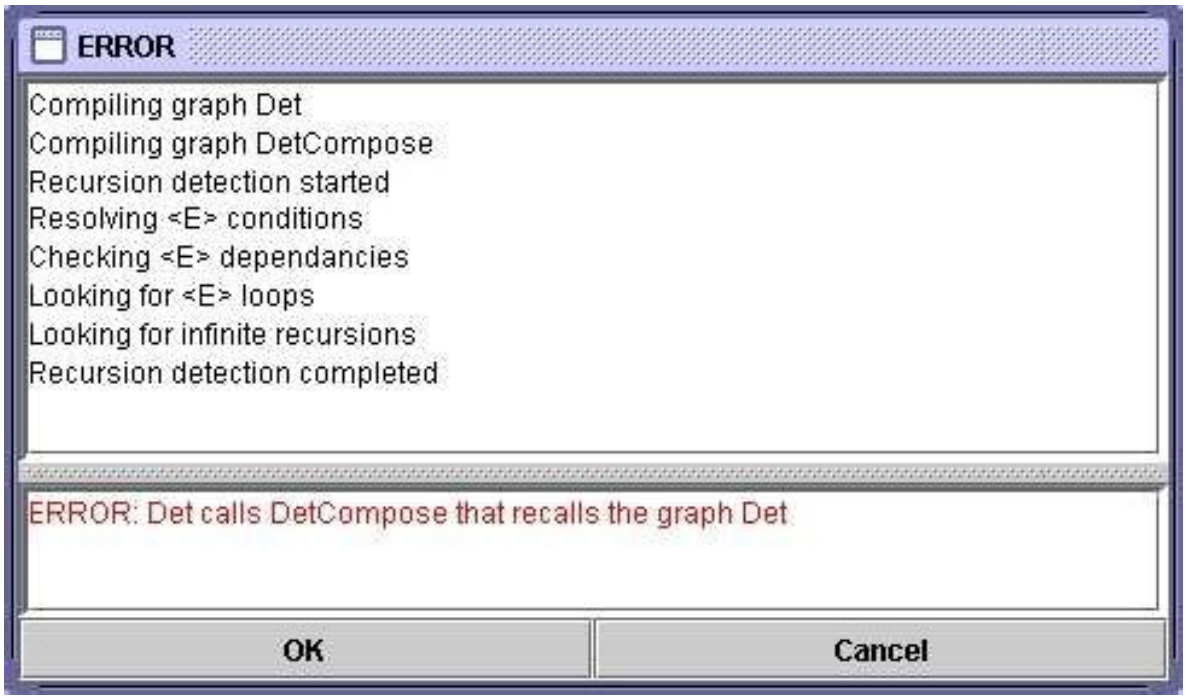



FIG. 11. Mensagem de erro obtida com a compilação do grafo `Det`

Se uma pesquisa de chaves de busca for lançada selecionando um grafo no formato `.grf` e o Unitex detectar um erro, a operação de pesquisa será automaticamente interrompida.

## 6.3 Contextos

Os grafos do Unitex são gramáticas algébricas. Elas são também denominadas de gramáticas livres de contexto, pois quando se deseja reconhecer uma seqüência *A*, não se considera o contexto no qual *A* aparece. Por exemplo, é impossível buscar com um grafo normal todas as ocorrências da palavra `presidente`, salvo aquelas seguidas por `da república`.

Entretanto, é possível considerar o contexto nos grafos sintáticos. Nesse caso, os grafos não são mais gramáticas algébricas, mas sim gramáticas contextuais que não têm as mesmas propriedades teóricas.

Um contexto é definido delimitando-se uma área do grafo com caixas que contenham `$[` e `$]`, que representam, respectivamente, o começo e o fim do contexto que são representados no grafo pelos colchetes verdes. O começo e o fim de um contexto devem aparecer no mesmo grafo.

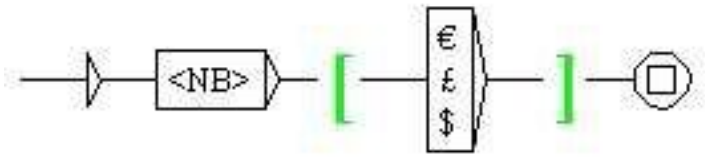



FIG. 6.12. Utilização de um contexto

A figura 6.12 mostra um exemplo simples de contexto. Esse grafo reconhece todos os números seguidos por euro, libra ou dólar, mas sem que o símbolo de unidade apareça nas ocorrências encontradas.

Os contextos são interpretados da seguinte maneira. Suponha-se que um começo de contexto seja encontrado no momento da aplicação de uma gramática em um texto e seja marcado *pos* como posição presente no texto nesse instante. O programa `Locate` vai, em seguida, tentar reconhecer a expressão descrita no contexto. Se ele fracassar, não haverá jogo. Se ele conseguir, ou seja, se ele puder alcançar o fim do contexto, o programa voltará à posição *pos* no texto e continuará a exploração da gramática a partir do fim do contexto.

Pode-se, do mesmo modo, definir contextos negativos, utilizando `$![` como começo de contexto. A figura 6.13 mostra um grafo que reconhece números que não são seguidos por `th`. A diferença em relação aos contextos positivos é que quando o `Locate` tenta reconhecer a expressão descrita no contexto, o fato de alcançar o fim do contexto é considerado um fracasso, pois significa que uma seqüência proibida foi reconhecida. Ao contrário, se o fim do contexto não puder ser alcançado, o programa `Locate` voltará à posição *pos* no texto e continuará a exploração da gramática a partir do fim do contexto.

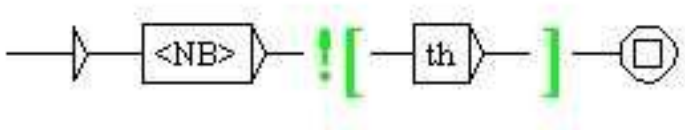



FIG. 6.13. Utilização de um contexto negativo

Os contextos podem ser colocados em qualquer lugar do gráfico, inclusive no começo. A figura 6.14 mostra, deste modo, um gráfico que reconhece um adjetivo no contexto de qualquer coisa que não seja particípio passado. Em outros termos, esse gráfico reconhece todos os adjetivos que não são ambíguos com particípios.

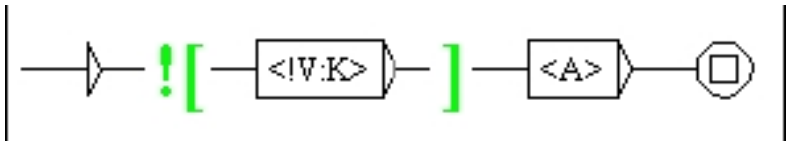



FIG. 6.14 - Busca por um adjetivo não ambíguo com um particípio passado

Graças a esse mecanismo, pode-se formular pedidos complexos. Desse modo, a figura 6.15 mostra um gráfico que reconhece todas as seqüências de dois nomes simples que não são ambíguos com palavras compostas. De fato, a chave de busca `<CDIC><<^([^]+[^]+)$>>` reconhece uma palavra composta que contém exatamente um espaço, e a chave de busca `<N><<^([^]+)$>>` reconhece um nome sem espaço, ou seja, um nome simples. Deste modo, na frase *Black cats should like the town hall*, esse grafo reconhecerá *Black cats*, mas não *town hall*, que é uma palavra composta.

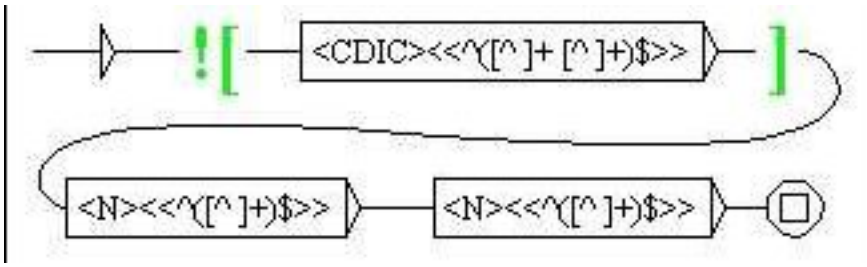



FIG. 6.15 - Utilização avançada dos contextos

É possível imbricar contextos. Por exemplo, o grafo da figura 6.16 reconhece um número que não estiver seguido por um ponto, exceto se esse ponto estiver seguido por um nome. Deste modo, na expressão *5.0+7.=12*, esse grafo reconhecerá *5*, *0* e *12*.


FIG. 6.16 - Imbricação de contextos

As saídas que se encontram nas caixas no interior de um contexto são ignoradas. Por outro lado, é possível utilizar uma variável que tenha sido definida em um contexto, como é o caso apresentado na figura 6.17. Se esse grafo for aplicado no modo MERGE ao texto *the cat is White*, obtém-se como saída:

```
the <pet name="cat" color="branco"/> is white
```

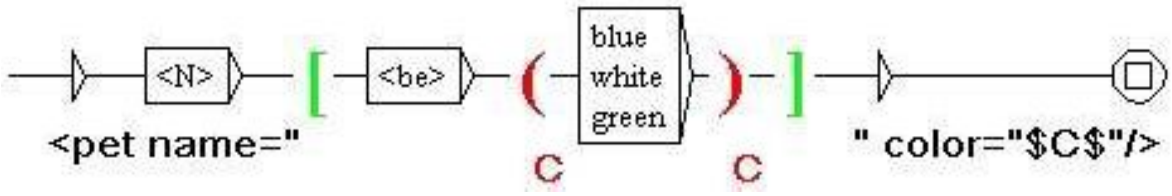



FIG. 6.17 - Variável definida em um contexto

## 6.4 Exploração dos caminhos de uma gramática

É possível criar caminhos reconhecíveis por uma gramática, para verificar, por exemplo, se ela gera corretamente as formas esperadas. Para isso, abrir o grafo principal da gramática e certificar-se de que a janela do grafo é mesmo a janela ativa (a janela ativa possui uma barra de título azul, enquanto as janelas inativas têm uma barra de título cinza). Em seguida, ir até o menu "FSGraph", depois até o submenu "Tools" e clicar em "Explore graph paths". A janela da figura 6.18 aparece, então.

A opção superior contém o nome do grafo principal da gramática a ser explorada. As opções seguintes concernem a gestão das saídas da gramática, assim como o modo de exploração:

- "Ignore outputs": as saídas são ignoradas;
- "Separate inputs and outputs": as saídas, agrupadas, são exibidas depois das entradas (`a b c/ A B C`);
- "Merge inputs and outputs": cada saída é exibida imediatamente depois da entrada que lhe corresponda (a/ A b/B c/C);
- "Only paths": as chamadas aos subgrafos são exploradas recursivamente;
- "Do not explore subgraphs recursively": as chamadas aos subgrafos são exibidas sem serem exploradas recursivamente.

Se a opção “Maximum number of sequences” for marcada, o número especificado será o número máximo de caminhos gerados. Se a opção não for selecionada, todos os caminhos serão gerados.

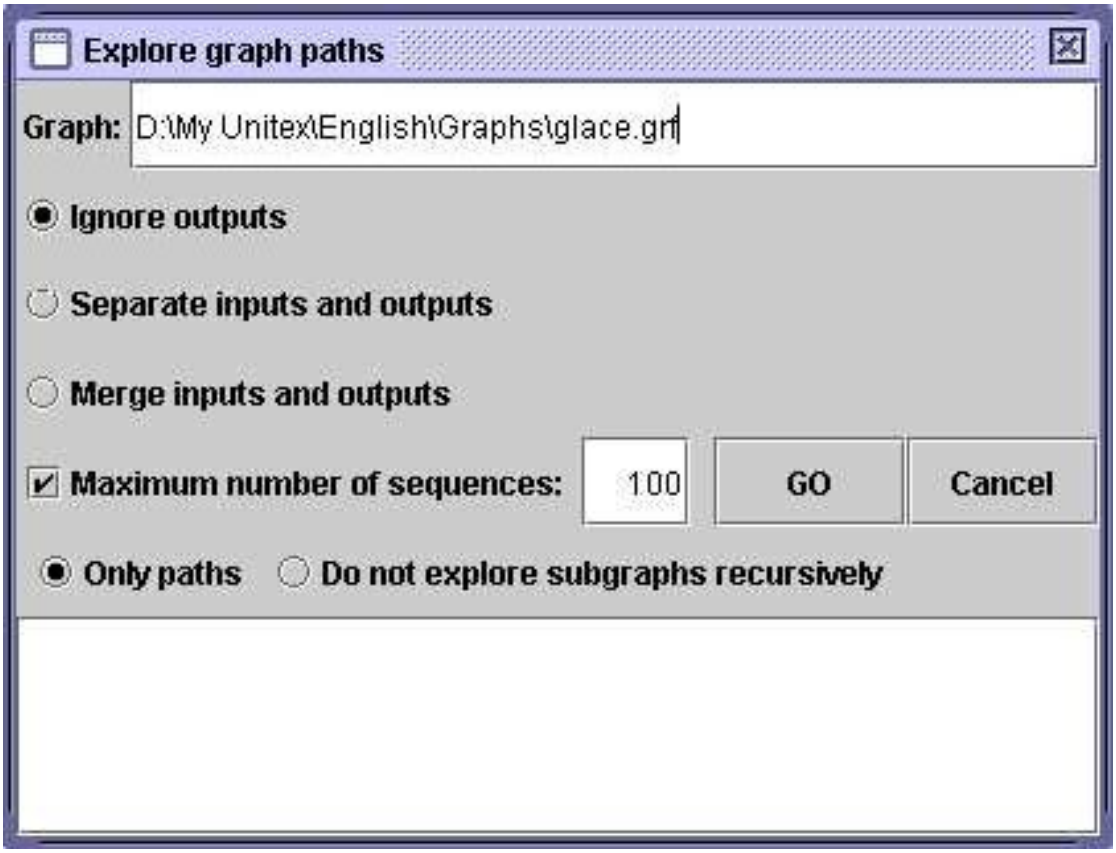



FIG. 6.18 - Exploração dos caminhos de uma gramática

Eis o que se obtém para o grafo da figura 6.19 com os parâmetros pré-definidos (ignorar as saídas, limite = 100 caminhos):

```
<NB> <boule> de glace à la pistache
```

```
<NB> <boule> de glace à la fraise
<NB> <boule> de glace à la vanille
<NB> <boule> de glace vanille
<NB> <boule> de glace fraise
<NB> <boule> de glace pistache
<NB> <boule> de pistache
<NB> <boule> de fraise
<NB> <boule> de vanille
glace à la pistache
glace à la fraise
glace à la vanille
glace vanille
glace fraise
glace pistache
```

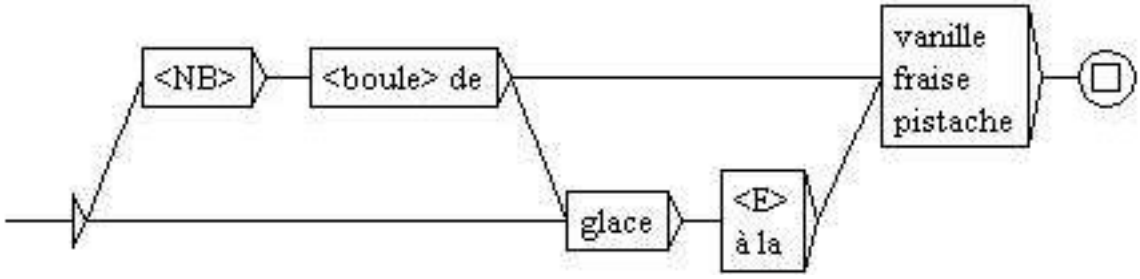



FIG. 6.19 – Exemplo de grafo

## 6.5 Coleção de grafos

Pode acontecer de alguém desejar aplicar várias gramáticas situadas em um mesmo diretório. Por isso, é possível construir automaticamente uma gramática a partir de uma arborescência de arquivos. Suponha-se, por exemplo, que haja a seguinte arborescência:

. *Dicos* :

. *Banque* :

. `carte.grf`

. *Nourriture* :

. `eau.grf`

. `pain.grf`

. `truc.grf`

Se houver o desejo de juntar todas essas gramáticas em uma só, pode-se fazê-lo com o comando "Build Graph Collection" no sub-menu "FSGraph > Tools". Configura-se essa operação por meio da janela da figura 6.20.

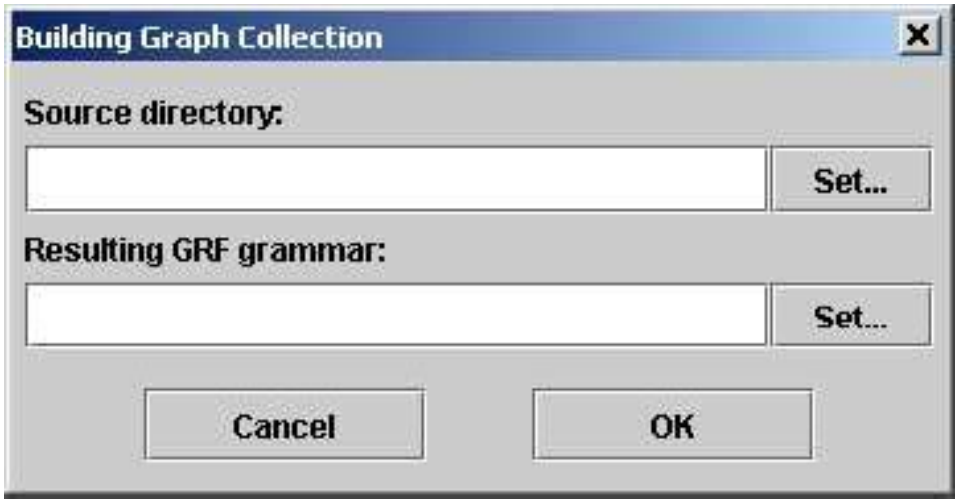



Fig. 6.20. Construção de uma coleção de grafos

No campo "Source directory", selecionar o diretório raíz que deseja explorar (nesse caso, o diretório *Dicos*). No campo "Resulting GRF grammar", indicar o nome da gramática gerada. ATENÇÃO: não colocar a gramática de saída na arborescência que deseja explorar, pois, nesse caso, o programa procurará ler e escrever simultaneamente nesse arquivo, o que provocará um ABEND.

Assim que clicar em "OK", o programa copiará de novo os grafos no diretório da gramática de saída e criará subgrafos correspondentes aos diferentes sub-diretórios, como se pode ver na figura 6.21, que mostra o grafo de saída criado para o nosso exemplo. Pode-se constatar que uma caixa[9] contenha as chamadas a subgrafos correspondentes a sub-diretórios (aqui, os diretórios *Banque* e *Nourriture*) e que a outra caixa faça remissão a todos os outros grafos que se encontravam no diretório (aqui, o grafo truc.grf).

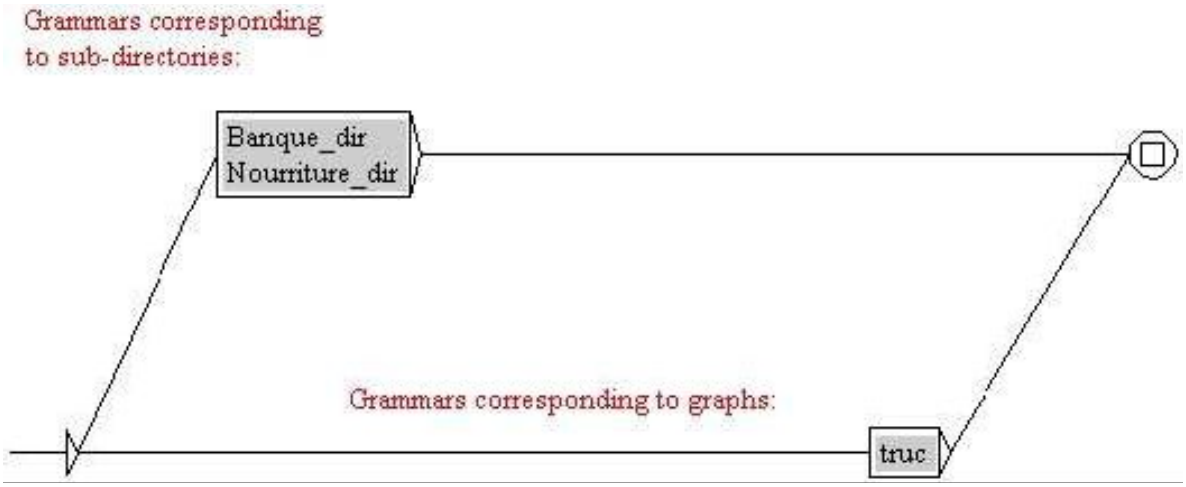



Fig. 6.21. Grafo principal de uma coleção de grafos

## 6.6 Regras de aplicação dos transdutores

Esta seção descreve as regras de aplicação dos transdutores durante operações de pré-tratamento e de busca por padrões. Os grafos de flexão e de normalização de formas ambíguas não se aplicam ao que segue.

[9]Essas caixas correspondem aos 'nós' na terminologia geral de grafos.

### 6.6.1 Inserção à esquerda da chave de busca reconhecida

Assim que um transdutor é aplicado em modo REPLACE, as saídas substituem as seqüências lidas no texto. No modo MERGE, as saídas são inseridas à esquerda das seqüências reconhecidas. Deve-se considerar o transdutor da figura 6.22.

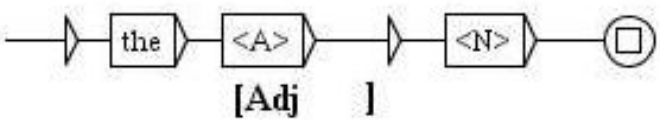



Fig. 6.22 Exemplo de transdutor

Se esse transdutor for aplicado ao romance *Ivanhoe* de Sir Walter Scott no modo MERGE, obtém-se a seguinte concordância da figura 6.23.

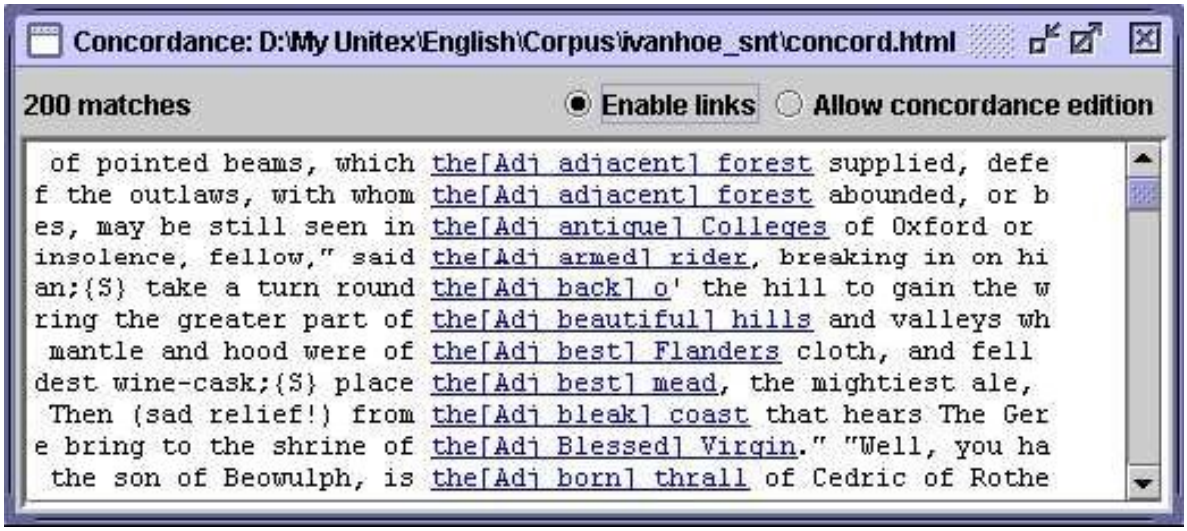



Fig. 6.23 Concordância obtida em modo MERGE com o transdutor da figura 6.22

### 6.6.2 Durante a aplicação

Durante as operações de pré-tratamento, o texto é modificado ao mesmo tempo em que é percorrido. Para evitar o risco de entrar em looping infinito, não é necessário que as seqüências produzidas por um transdutor sejam reanalisadas por ele. Por isso, quando uma seqüência for introduzida no texto, a aplicação do transdutor continua após essa seqüência.

Essa regra só concerne aos transdutores de pré-tratamento, pois durante a aplicação de grafos sintáticos, as saídas não modificam o texto percorrido, mas um arquivo de concordâncias distinto do texto.

### 6.6.3 Prioridade à esquerda

Durante a aplicação de uma gramática local, as ocorrências que se cruzam são todas indexadas. Durante a construção da concordância, todas essas ocorrências são apresentadas (ver figura 6.24).

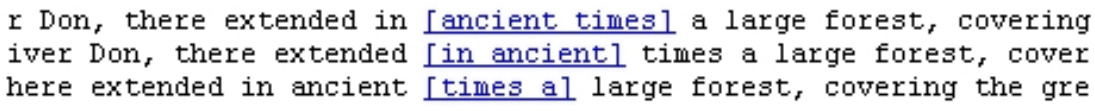



Fig. 6.24 Ocorrências que se cruzam em uma concordância

Em contrapartida, se o texto for modificado ao invés de se construir uma concordância, é necessário escolher dentre essas ocorrências quais serão consideradas. Para isso, o Unitex aplica a seguinte regra de prioridade: a seqüência situada mais à esquerda é preferível.

Se essa regra for aplicada às três ocorrências da concordância precedente, a ocorrência `[in ancient]` é concorrente com `[ancient times]`. A primeira é, portanto, mantida, pois é a ocorrência que está mais à esquerda e `[ancient times]`

é eliminada. A seguinte ocorrência `[times a]` conseqüentemente não está mais em conflito com `[ancient times]` e pode, então, aparecer no resultado:

```
...Don, there extended [in ancient] [times a] large forest...
```

A regra de prioridade à esquerda se aplica unicamente quando o texto for modificado, seja durante o pré-tratamento, seja após a aplicação de um grafo sintático (ver seção 6.7.3).

### 6.6.4 Prioridade para as seqüências mais longas

Durante a aplicação de um grafo sintático, é possível escolher se a prioridade deve ser dada às seqüências mais curtas ou mais longas, ou se todas as seqüências devem ser mantidas. Durante as operações de pré-tratamento, a prioridade é sempre dada às seqüências mais longas.

### 6.6.5 Saídas com variáveis

Como foi visto na seção 5.2.7, é possível utilizar variáveis para estocar o texto que foi analisado por uma gramática. Essas variáveis podem ser utilizadas nos grafos de pré-tratamento e nos grafos sintáticos. Deve-se nomear as variáveis utilizadas. Esses nomes podem conter letras compreendidas entre A e Z, não acentuadas, minúsculas ou maiúsculas, números e o caractere _ (underline).

Para definir o início (ou o fim) da área estocada em uma variável, criar uma caixa contendo o nome da variável delimitado pelos caracteres $ e ( ($ e ) para o fim de uma variável). Para utilizar uma variável em uma saída, colocar antes de seu nome o caractere $ (ver figura 6.25).

As variáveis são globais. Isso significa que se pode definir uma variável em um grafo e se remeter a ela em um outro, como ilustram os grafos da figura 6.25.

Se o grafo `TitleName` for aplicado no modo MERGE ao texto *Ivanhoe*, obtém-se a seguinte concordância da figura 6.26.

As saídas com variáveis podem ser utilizadas para mover grupos de palavras. Na verdade, a aplicação de um transdutor em modo REPLACE só redige no texto as seqüências produzidas por saídas. Para inverter dois grupos de palavras, é preciso apenas estocá-los nas variáveis e produzir uma saída com essas variáveis na ordem desejada. Dessa forma, o transdutor da figura 6.27 aplicado em modo REPLACE ao texto *Ivanhoe* dá a concordância da figura 6.28.

Se o início ou o fim de uma variável está mal definido (fim de uma variável antes de seu início, falta do início ou do fim de uma variável), ela será ignorada durante as saídas.

Não há nenhuma limitação do número de variáveis utilizáveis.

As variáveis podem estar imbricadas e, até mesmo, se cruzar como mostra a figura

## 6.7 Aplicação dos grafos aos textos

Esta seção trata unicamente dos grafos sintáticos.

### 6.7.1 Configuração da busca

Para aplicar um grafo a um texto, abrir o texto e, em seguida, clicar em "Locate Pattern..." no menu "Text" ou dar o comando <Ctrl+L>. Pode-se, então, configurar sua busca de acordo com a janela da figura 6.30.

Na opção intitulada “Locate pattern in the form of”, escolha “Graph” e selecione seu grafo clicando no botão “Set”. Pode-se escolher um grafo no formato .grf (Unicode Graphs) ou um grafo compilado no formato .fst2 (Unicode Compiled Graphs). Se seu grafo estiver no formato .grf, o Unitex o compilará automaticamente antes de lançar a pesquisa.

A opção “Index” permite selecionar o modo de reconhecimento:

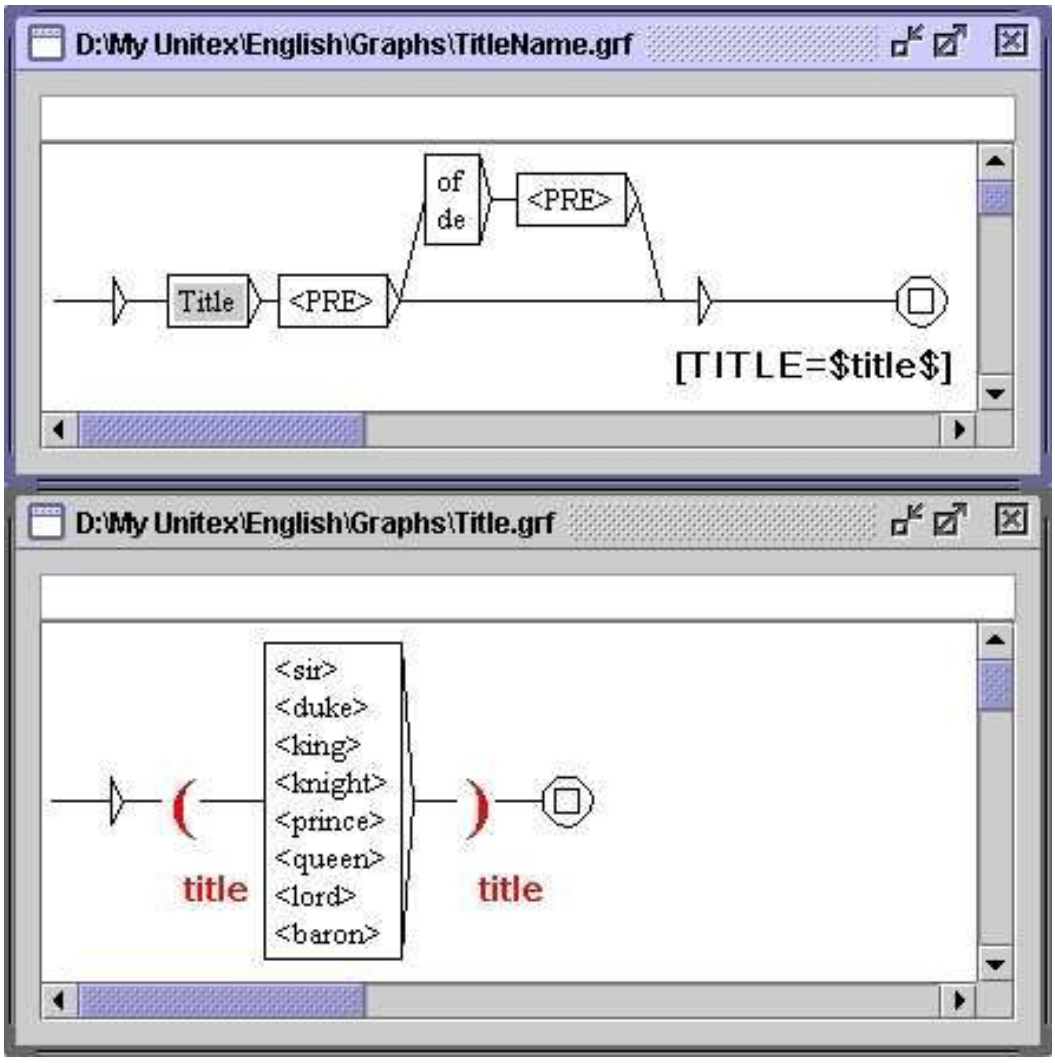


Fig. 6.25 - Definição de uma variável em um subgrafo

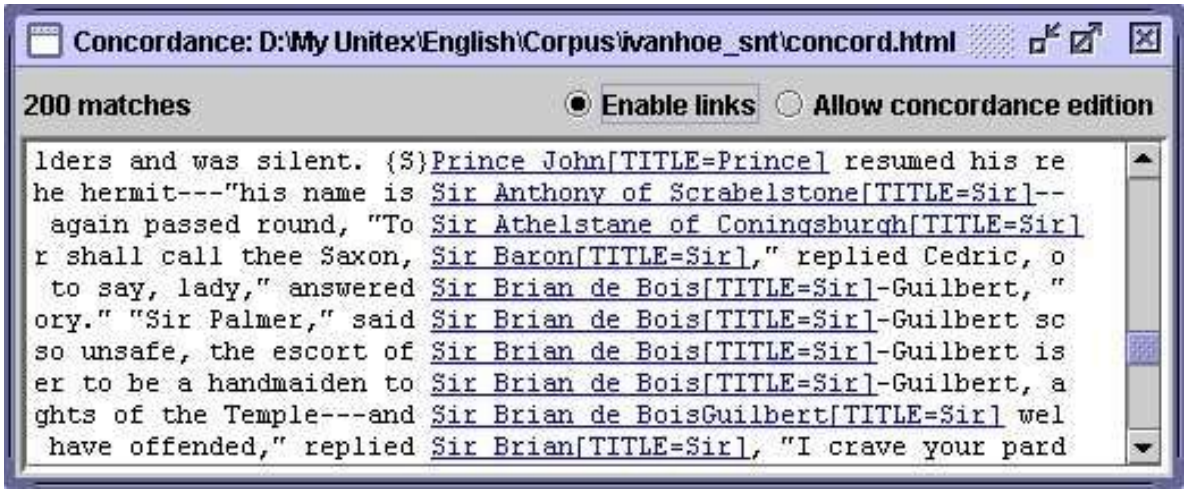



Fig. 6.26 – Concordância obtida pela aplicação do grafo `TitleName`

6.29.

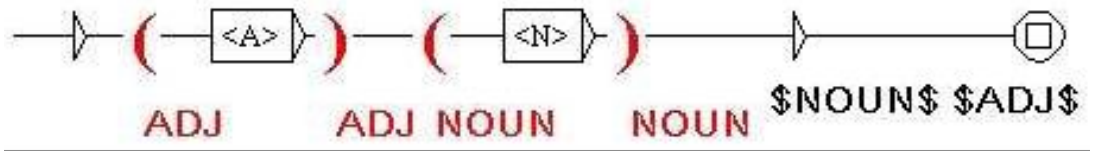



Fig. 6.27 – Inversão de palavras devido à utilização de duas variáveis

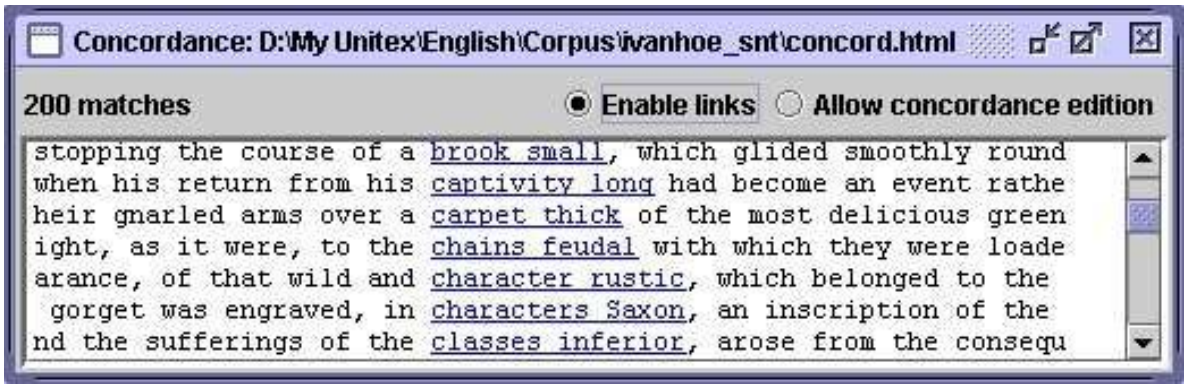



Fig. 6.28 – Resultado da aplicação do transdutor da figura 6.27

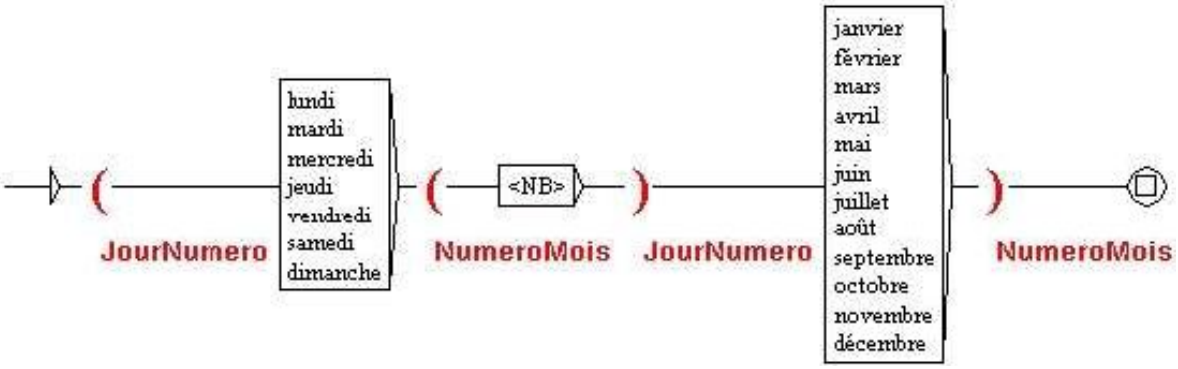



Fig. 6.29 – Cruzamento de variáveis

- "Shortest matches": dá prioridade às seqüências mais curtas;
- "Longest matches": dá prioridade às seqüências mais longas. É o modo utilizado como default;
- "All matches": dá todas as seqüências reconhecidas.

A opção “Search limitation” permite limitar ou não a busca a um certo número de ocorrências. Por definição, a busca é limitada às 200 primeiras ocorrências.

A opção “Grammar outputs” diz respeito ao modo de utilização das saídas. O modo “Merge with input text” permite inserir as seqüências criadas pelas saídas. O Modo “Replace recognized sequences” permite substituir as seqüências reconhecidas pelas seqüências geradas. O terceiro modo ignora as saídas. Este último modo é utilizado como default.

Uma vez que suas configurações foram fixadas, clique em “SEARCH” para lançar a busca.

### 6.7.2 Concordância

O resultado da busca é um arquivo índice contendo as posições de todas as ocorrências encontradas. A janela da figura 6.31 propõe-lhe construir uma concordância, modificar o texto ou comparar o resultado da busca com a busca precedente sobre o mesmo texto.

Para visualizar uma concordância, clicar no botão “Build concordance”. Pode-se configurar o tamanho dos contextos à esquerda e à direita em caracteres. Pode-se igualmente escolher o modo de classificação que será aplicado às linhas da concordância graças ao menu “Sort According to”. Para mais detalhes sobre as configurações de construção da concordância, dirigir-se à seção 4.8.2.

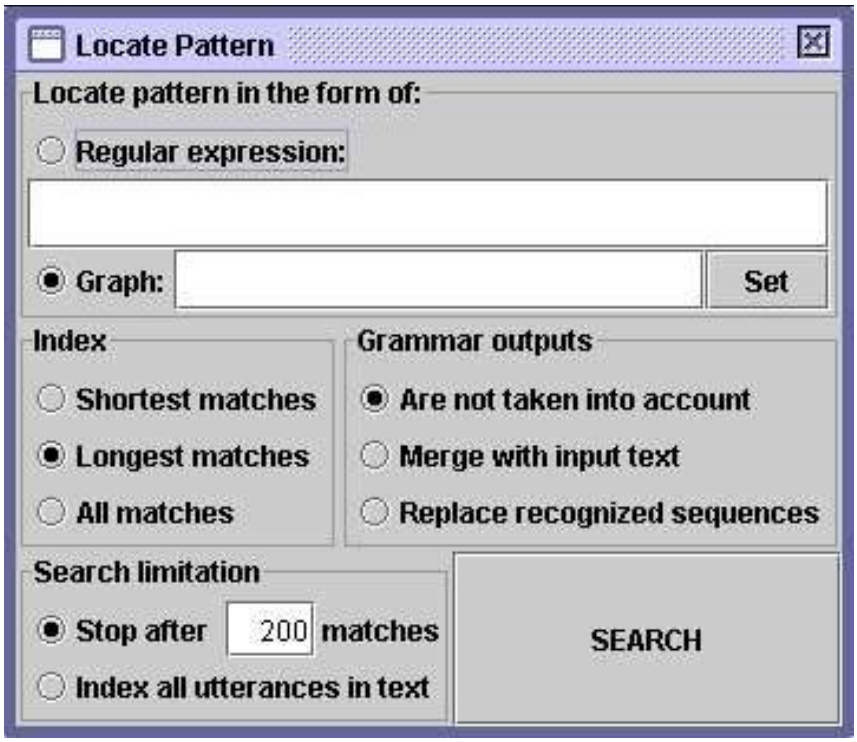



Fig. 6.30 – Janela de busca de expressões

A concordância é produzida sob a forma de um arquivo HTML. Pode-se configurar o Unitex para que as concordâncias sejam lidas com a ajuda de um navegador Web (ver seção 4.8.2).

Se as concordâncias com a janela proposta pelo Unitex forem visualizadas, pode-se acessar a seqüência reconhecida no texto clicando na ocorrência. Se a janela do texto não estiver iconificada e se o texto não estiver muito longo para ser visualizado, a seqüência selecionada aparecerá (ver figura 6.32).

Além disso, se o autômato do texto foi construído e se a janela correspondente não for iconificada, o fato de clicar em uma ocorrência seleciona o autômato da frase que contém essa ocorrência.

### 6.7.3 Modificação do texto

Pode-se escolher modificar o texto ao invés de construir uma concordância. Para isso, selecionar um nome de arquivo no campo “Modify text” da janela da figura 6.31. Esse arquivo deve ter a extensão .txt.

Se desejar modificar o texto atual, é preciso escolher o arquivo .txt correspondente. Se escolher um outro nome de arquivo, o texto atual não será afetado. Clicar no botão “GO” para lançar a modificação do texto. As regras de prioridades aplicadas durante essa operação são detalhadas na seção 3.6.2.

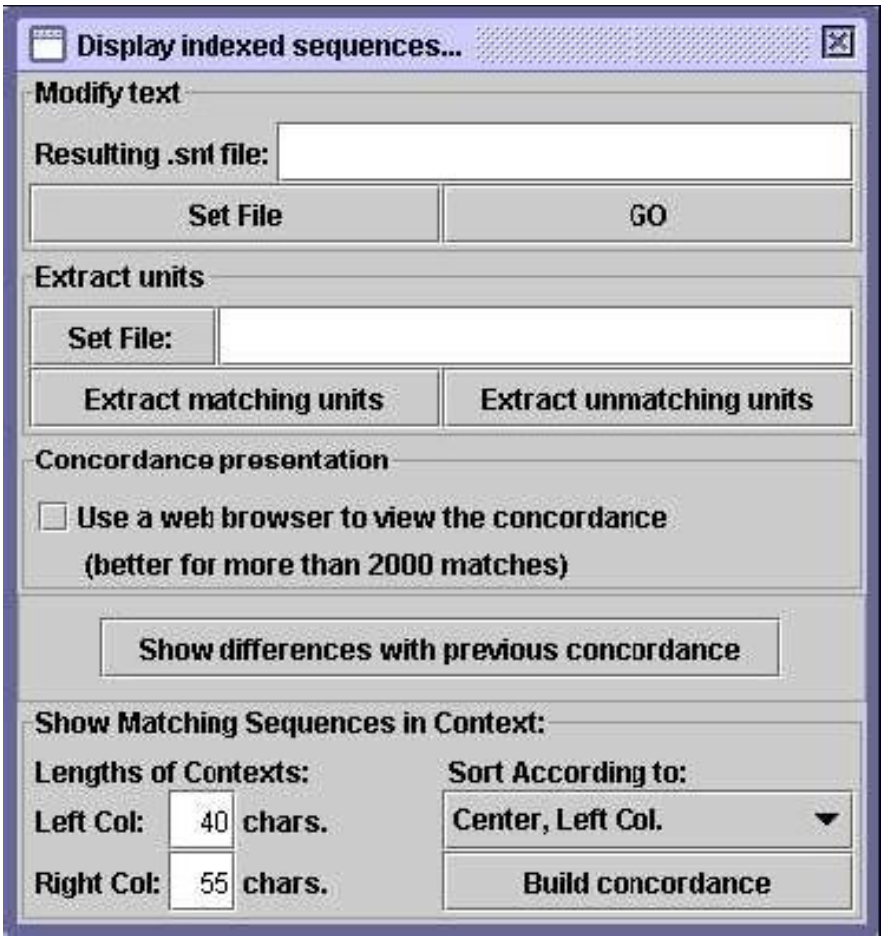



Fig. 6.31 – Configuração da visualização das ocorrências encontradas

Uma vez que essa operação foi efetuada, o arquivo resultante é uma cópia do texto na qual as saídas são consideradas. As operações de normalização e de segmentação em unidades lexicais são automaticamente aplicadas a esse arquivo texto. Os dicionários existentes do texto não são modificados. Desse modo, se escolher por

modificar o texto atual, as modificações são imediatamente efetivas. Pode-se, então, lançar novas buscas no texto.

ATENÇÃO: se escolher por aplicar seu grafo ignorando as saídas, todas as ocorrências serão apagadas do texto.

### 6.7.4 Extração das ocorrências

Pode-se extrair todas as frases do texto que contenham ou não ocorrências. Para isso, escolher um nome de arquivo de saída no botão "Set File" na opção "Extract units" (figura 6.31). Em seguida, clicar em "Extract matching units" ou "Extract unmatching units" se quiser ou não extrair as frases contendo as ocorrências.

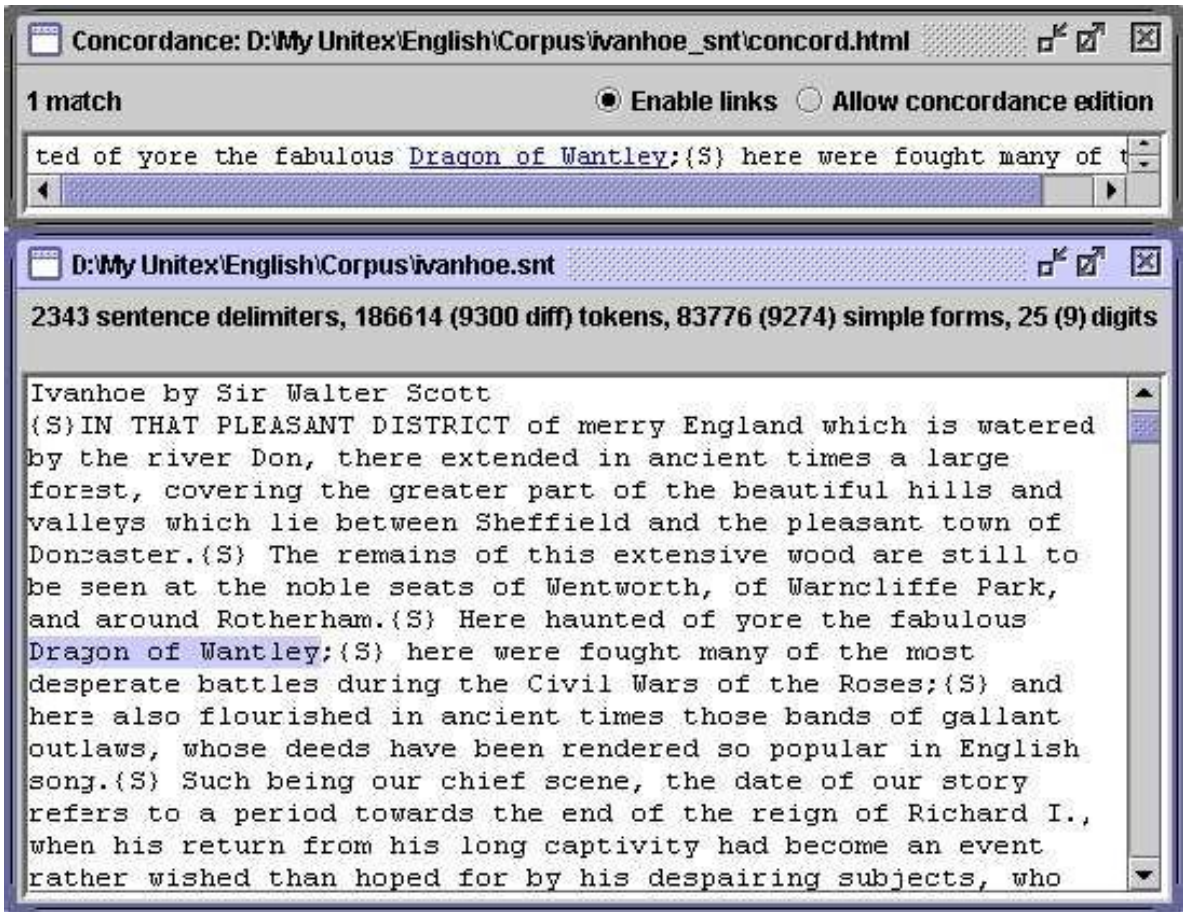


Fig. 6.32 – Seleção de uma ocorrência no texto

### 6.7.5 Comparação de concordâncias

A opção "Show differences with previous concordance" permite comparar a concordância que acabou de ser calculada com a concordância precedente, se ela existir. Para isso, o programa ConcorDiff constrói as duas concordâncias na ordem do texto e depois compara suas linhas. O resultado é uma página HTML que mostra as ocorrências em duas colunas. Uma linha em azul indica que uma mesma ocorrência aparece nas duas concordâncias. Uma linha em vermelho indica que uma ocorrência aparece de forma mais longa em uma concordância do que na outra. Por fim, uma linha em verde indica que uma ocorrência pertence à somente uma concordância. A figura 6.33 mostra um exemplo de comparação de concordâncias.

NOTA: contrariamente a uma concordância normal, não se pode clicar nas ocorrências em uma comparação de concordâncias.

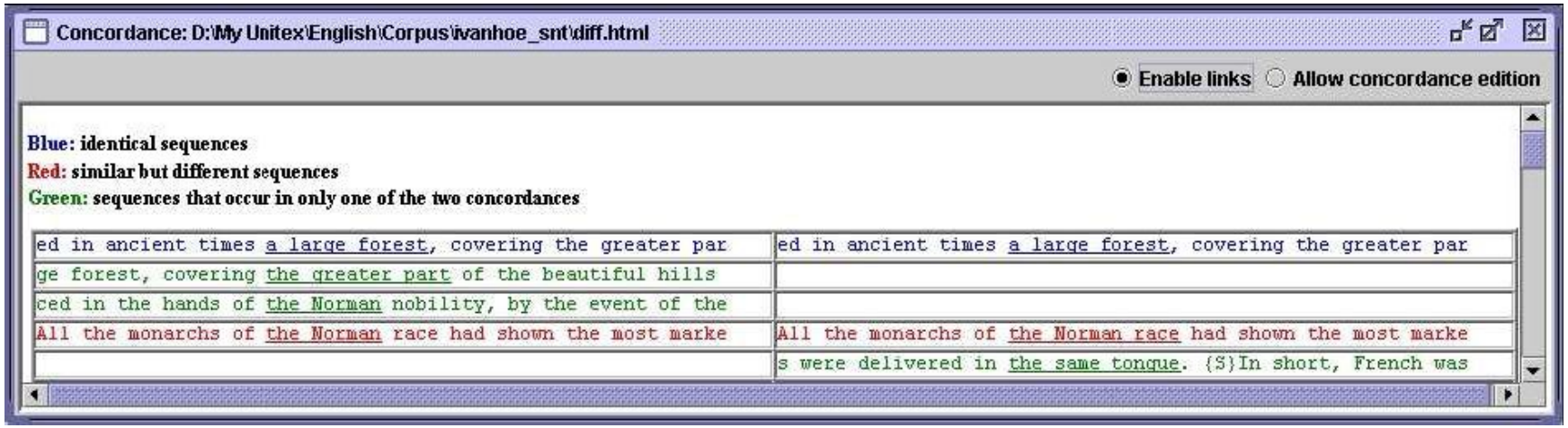

Concordance: D:\My Unitex\English\Corpus\ivanhoe_snt\diff.html
Enable links
Allow concordance edition
Blue: identical sequences
Red: similar but different sequences
Green: sequences that occur in only one of the two concordances
ed in ancient times a large forest, covering the greater par
ge forest, covering the greater part of the beautiful hills
ced in the hands of the Norman nobility, by the event of the
All the monarchs of the Norman race had shown the most marke
ed in ancient times a large forest, covering the greater par
All the monarchs of the Norman race had shown the most marke
s were delivered in the same tongue. {S}In short, French was

Fig. 6.33 – Exemplo de comparação de concordâncias

# Capítulo 7

# Autômato do texto

As línguas naturais contêm muitas ambigüidades lexicais. O autômato do texto é um meio visual e eficaz de representar essas ambigüidades. Cada frase do texto é representada por um autômato cujos caminhos exprimem todas as interpretações possíveis.

Esse capítulo apresenta os autômatos de texto, o pormenor de sua construção, assim como as operações que lhes podem ser aplicadas, em particular o levantamento de ambigüidades por meio do programa ELAG ([35]). Por enquanto não é possível efetuar a pesquisa de padrões sobre o autômato do texto.

## 7.1 Apresentação

O autômato do texto permite exprimir todas as interpretações lexicais possíveis das palavras. Essas diferentes interpretações são as diferentes entradas presentes nos dicionários do texto. A figura 7.1 mostra o autômato da quarta frase do texto *Ivanhoé*.

Pode-se ver na figura 7.1 que a palavra `Here` tem aqui três interpretações (adjetivo, advérbio e substantivo), `haunted` duas (adjetivo e verbo), etc. Todas as combinações possíveis são expressas, pois cada interpretação de cada palavra está ligada a todas as interpretações da palavra seguinte e da anterior.

Em caso de concorrência entre uma palavra composta e uma seqüência de palavras simples, o autômato contém um caminho rotulado pela palavra composta, paralela aos caminhos que exprimem as combinações de palavras simples. Isso é ilustrado pela figura 7.2, onde a palavra composta `courts of law` concorre com uma combinação de palavras simples.

Por construção, o autômato do texto não contém laço. Diz-se que o autômato do texto é *acíclico*.

NOTA: o termo autômato do texto é um abuso de linguagem. Com efeito, há na realidade um autômato para cada frase do texto. Contudo, a concatenação de todos esses autômatos corresponderia ao autômato de todo o texto. Utiliza-se, portanto, o termo autômato do texto mesmo se este objeto não for realmente manipulado por razões práticas.

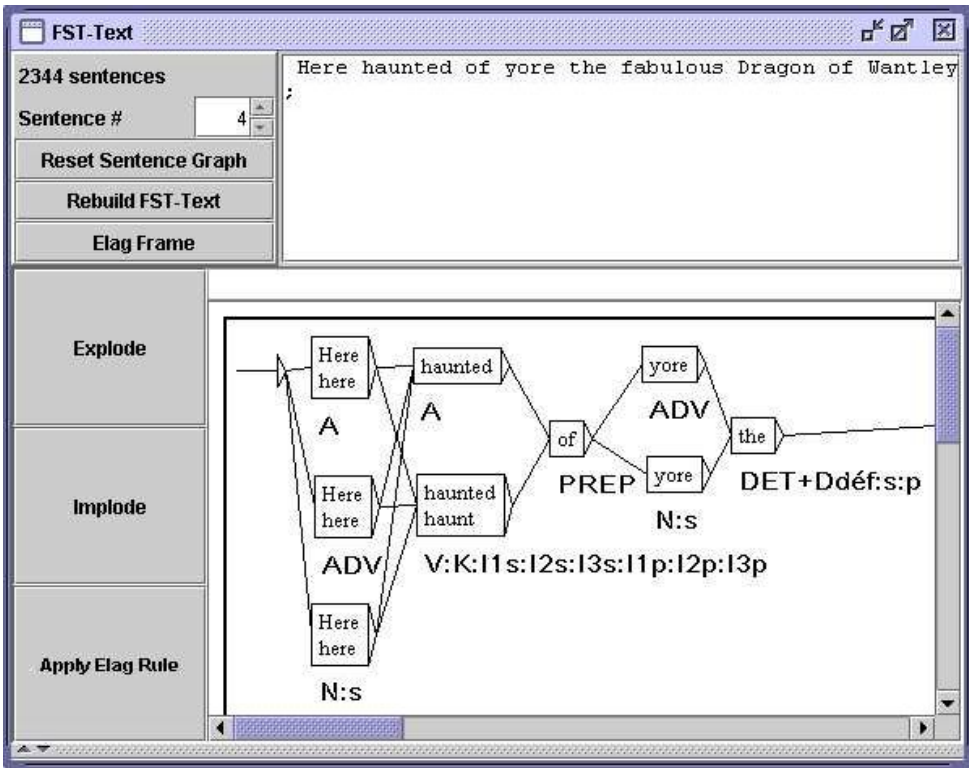



Fig. 7.1 Exemplo de autômato de frase

## 7.2 Construção

Para construir o autômato de um texto, deve-se abrir o texto, depois clicar em "Construct FST-Text..." no menu "Text". É recomendável segmentar o texto em frases e aplicar nele os dicionários. Se o texto não tiver sido segmentado em frases, o programa de construção segmentará arbitrariamente o texto em seqüências de 2000 unidades lexicais em vez de construir um autômato por frase. Se você não tiver aplicado os dicionários, os autômatos de frase obtidos serão constituídos apenas de um único caminho contendo somente palavras desconhecidas.

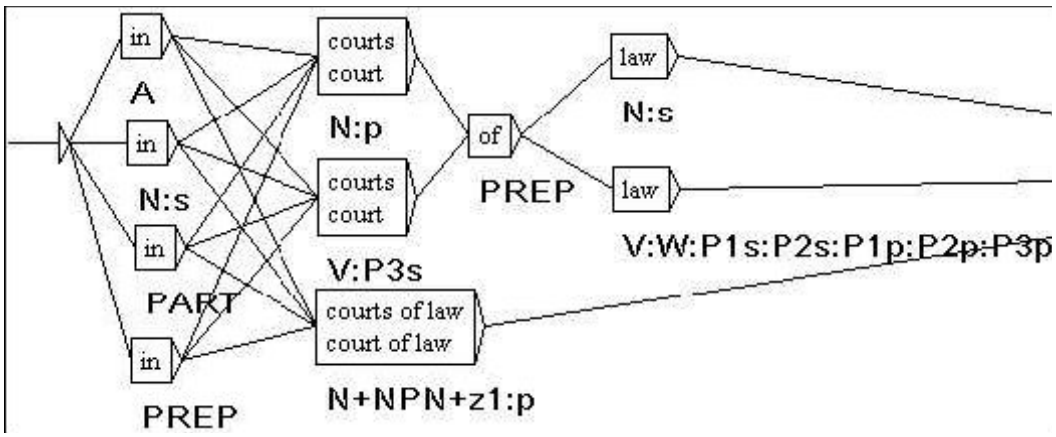



FIG. 7.2 Concorrência entre uma palavra composta e uma combinação de palavras simples.

### 7.2.1 Regras de construção do autômato do texto

Os autômatos de frase são construídos a partir dos dicionários do texto. Logo, o grau de ambigüidade obtido está diretamente ligado à acuidade de descrição dos dicionários utilizados. Sobre o autômato de frase da figura 7.3, pode-se ver que a palavra `which` foi codificada duas vezes como determinante em duas subcategorias da categoria DET. Essa acuidade de descrição não será de nenhuma utilidade se somente interessar a categoria gramatical dessa palavra. É necessário, portanto, adaptar a acuidade dos dicionários à utilização buscada.

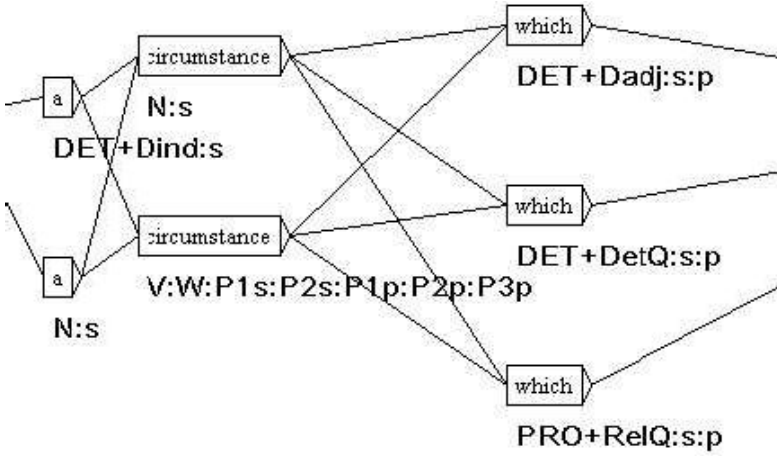



FIG. 7.3 Dupla entrada para `which` na qualidade de determinante

Para cada unidade lexical da frase, o Unitex busca todas as suas interpretações possíveis no dicionário de palavras simples do texto. Buscam-se, em seguida, todas as seqüências de unidades lexicais que têm uma interpretação no dicionário de palavras compostas do texto. Todas as combinações dessas interpretações formam o autômato da frase.

NOTA: quando o texto apresenta etiquetas lexicais (*i.e.* {*hoje*: *.ADV*}), essas etiquetas são reproduzidas de forma idêntica no autômato, sem que o programa tente decompor as seqüências que elas representam.

Em cada caixa, a 1ª linha tem a forma flexionada encontrada no texto, e a 2ª linha se for diferente tem a forma canônica. As outras informações são codificadas sob a caixa (ver seção 7.4.1).

Os espaços que separam as unidades lexicais não são transcritos novamente no autômato, com exceção dos espaços no interior de palavras compostas.

A caixa das unidades lexicais é conservada. Por exemplo, se for encontrada a palavra `Here`, conserva-se a maiúscula (ver figura 7.1). Essa escolha permite que não seja perdida tal informação no momento da passagem ao autômato do texto, o que poderá ser útil para aplicações onde a caixa é importante, como o reconhecimento dos nomes próprios.

### 7.2.2. Normalização de formas ambíguas

No momento da construção do autômato, é possível efetuar uma normalização de formas ambíguas aplicando a gramática de normalização. Essa gramática deve ser nomeada Norm. fst2 e deve ser colocada em seu diretório pessoal, no sub-diretório `/Graphs Normalization` da língua requerida. As gramáticas de normalização de formas ambíguas são descritas na seção 6.1.3.

Se uma seqüência do texto for reconhecida pela gramática de normalização, todas as interpretações descritas pela gramática são inseridas no autômato do texto. A figura 7.4 mostra o trecho da gramática utilizado pelo francês que explicita a ambigüidade da seqüência 1'.

Se essa gramática for aplicada a uma frase francesa que contiver a seqüência 1', obtém-se um autômato de frase similar àquele da figura 7.5.

No autômato obtido, pode-se ver que as quatro regras de reescritura da seqüência 1' foram aplicadas, o que adicionou quatro etiquetas no autômato. Essas etiquetas concorrem com os dois caminhos preexistentes para a seqüência 1'. A normalização à construção do autômato do texto permite adicionar caminhos ao autômato, e não suprimi-lo. Quando a funcionalidade de levantamento de ambigüidades estiver disponível, ela permitirá que sejam eliminados os caminhos que se tornarem supérfluos.

### 7.2.3. Normalização dos pronomes clíticos em português

Em português, os verbos no futuro e no condicional podem ser modificados pela inserção de um ou dois pronomes clíticos entre o radical e o sufixo do verbo. Por exemplo, a seqüência *dir-me-ão* corresponde à forma verbal completa *dirão*, associadas ao pronome *me*. Com vistas à possibilidade de efetuar manipulações nessa forma reescrita, é necessário introduzi-la no autômato do texto, paralelamente à seqüência de origem.

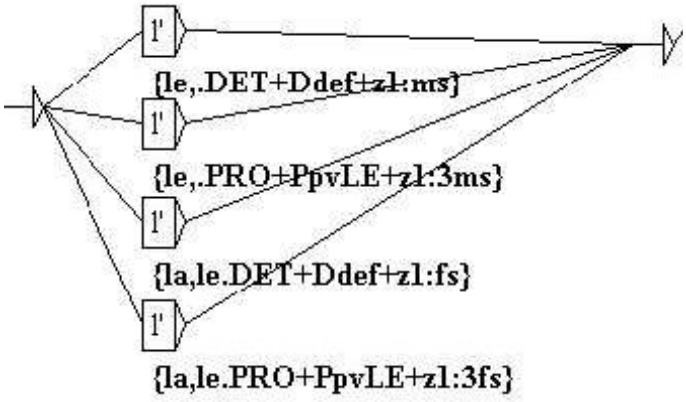



FIG. 7.4 Normalização da seqüência 1'

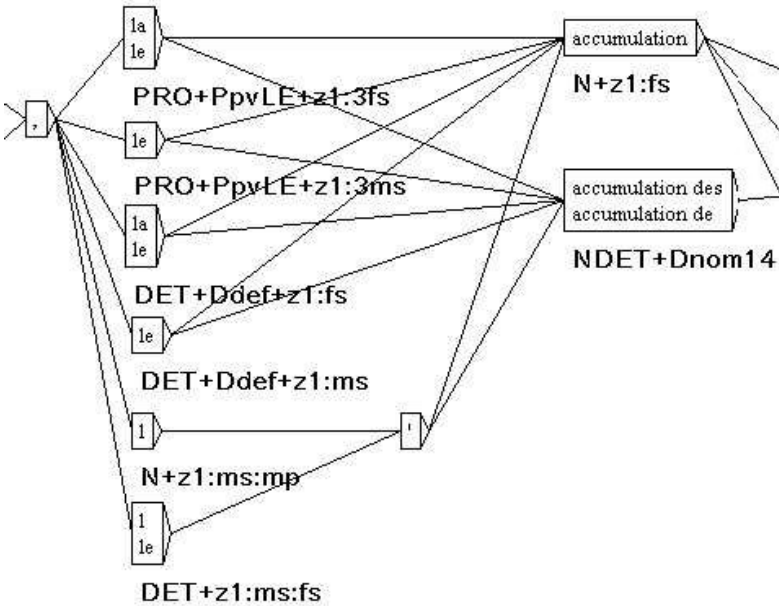


FIG. 7.5 Autômato normalizado com a gramática da figura 7.4

Desse modo, o usuário poderá pesquisar uma ou outra forma de acordo com suas necessidades. As figuras 7.6 e 7.7 mostram o autômato de uma frase antes e depois da normalização dos clíticos

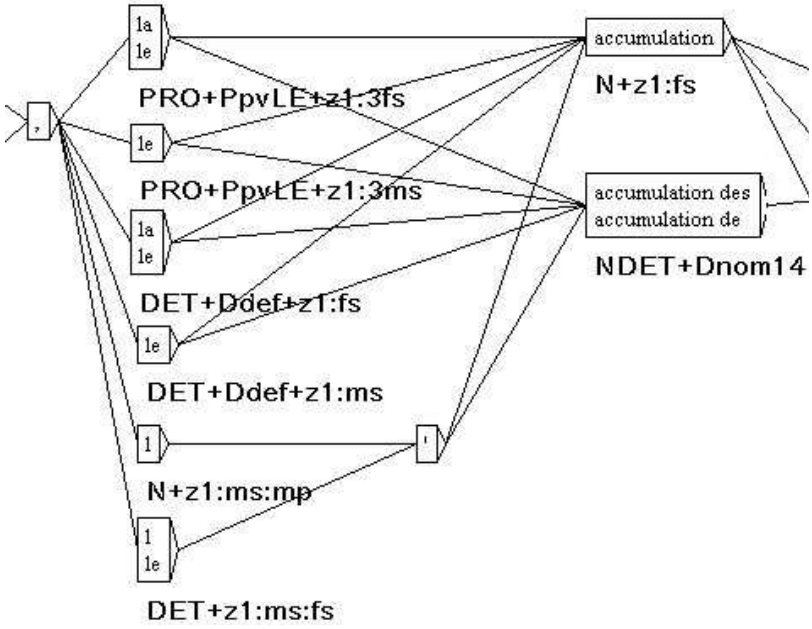



FIG. 7.6 Autômato de frase não normalizada

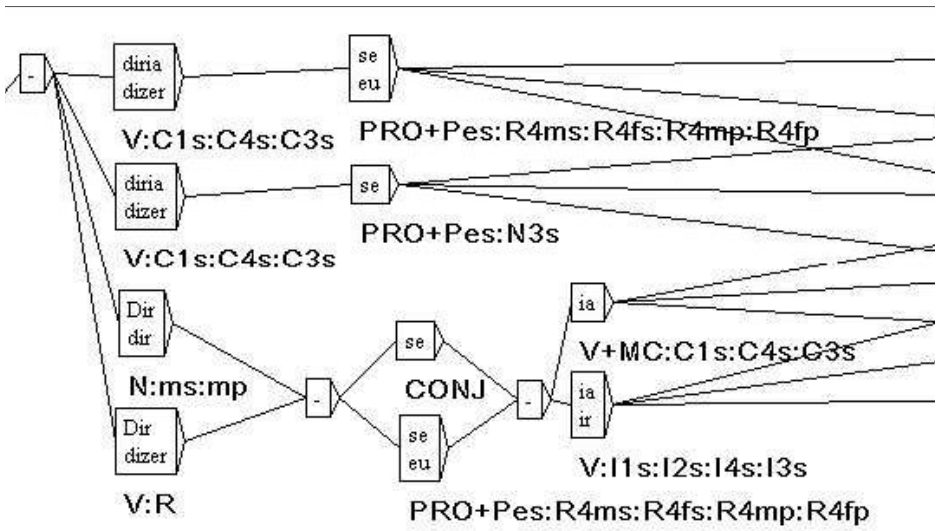



FIG. 7.7 Autômato de frase normalizada

O programa `Reconstrução` permite que se construa dinamicamente, para cada texto, uma gramática de normalização dessas formas. A gramática assim produzida pode então ser utilizada para normalizar o autômato do texto. A janela de configuração de construção do autômato propõe a opção "Build clitic normalization grammar" (ver figura 7.10). Esta opção aciona automaticamente a construção da gramática de normalização, que em seguida é utilizada para construir o autômato do texto, se você tiver selecionado a opção "Apply the Normalization grammar".

### 7.2.4. Conservação dos melhores caminhos

Pode acontecer de uma palavra desconhecida vir parasitar o autômato do texto, ao estar em concorrência com uma seqüência completamente separada em unidades. Dessa maneira, no autômato de frase da figura 7.8, pode-se ver que o advérbio `aujourd'hui` é concorrenciado pela palavra desconhecida `aujourd`, seguido de um apóstrofo e do particípio passado do verbo `huir`.

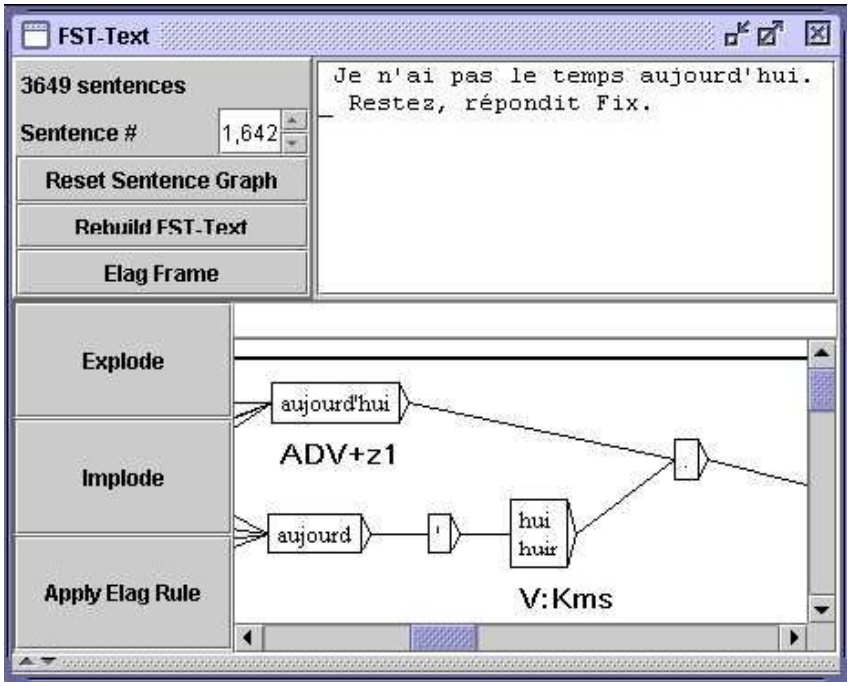



FIG. 7.8 Ambigüidade devido a uma seqüência contendo uma palavra desconhecida.

Observa-se igualmente esse fenômeno no tratamento de algumas línguas asiáticas como o *thai*. Quando as palavras não estão delimitadas, não há outra solução além de considerar todas as combinações possíveis, o que acarreta a criação de numerosos caminhos que comportam palavras desconhecidas que se entrecruzam com os caminhos etiquetados. A figura 7.9 mostra um exemplo de um autômato como esse de uma frase em *thai*.

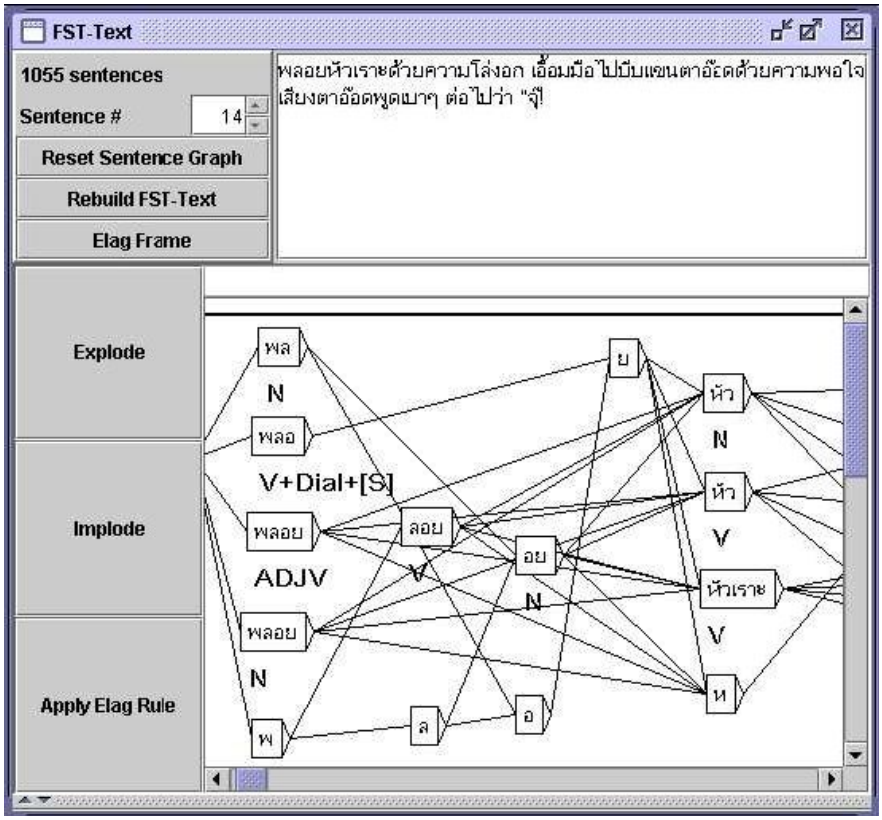



FIG. 7.9 Autômato de uma frase *thai*.

É possível suprimir esses caminhos parasitas. Para tanto, é preciso selecionar a opção “Clean Text FST” na janela de configuração da construção do autômato do texto (ver figura 7.10). Essa opção indica ao programa de construção do autômato que ele deve limpar cada autômato de frase.

Essa limpeza é realizada segundo o seguinte princípio: se vários caminhos estão em concorrência no autômato, o programa conserva aqueles que têm o mínimo de palavras desconhecidas. Por exemplo, a seqüência `aujourd'hui` enquanto advérbio composto suplanta a decomposição em `aujourd` seguido de um apóstrofo e de `hui`, pois `aujourd` é uma palavra desconhecida, o que faz uma forma não etiquetada contra zero no caso do advérbio composto.

A figura 7.11 mostra o autômato da figura 7.9 após limpeza

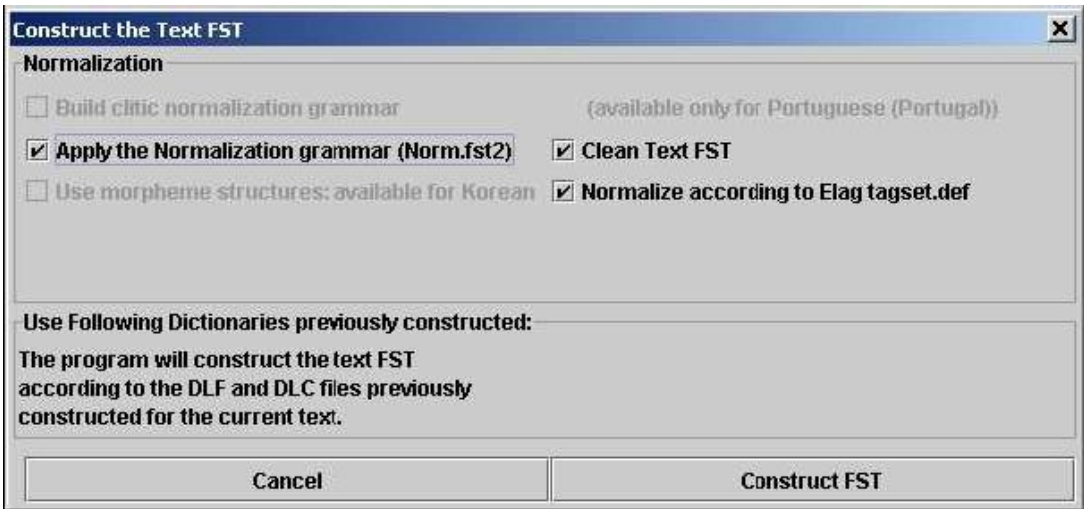



FIG. 7.10 Configuração da construção do autômato do texto

## 7.3 Levantamento de ambigüidades lexicais com ELAG

O programa ELAG permite aplicar gramáticas de levantamento de ambigüidades sobre o autômato do texto. É um mecanismo poderoso que permite a cada um escrever suas próprias regras de modo independente das regras já existentes. Esta seção apresenta rapidamente o formalismo das gramáticas utilizadas por ELAG assim como o funcionamento do programa.

Maiores detalhes, o leitor poderá encontrar em [3] e [35].

### 7.3.1 Gramáticas de levantamento de ambigüidades

As gramáticas manipuladas por ELAG têm uma sintaxe particular. Elas comportam duas partes, que chamaremos parte *se* e parte *então*. A parte *se* de uma gramática ELAG divide-se em duas zonas delimitadas por caixas contendo o símbolo < ! >. A parte *então* é dividida do mesmo modo por meio do símbolo < = >. O significado de uma gramática é o seguinte: no autômato do texto, ser for encontrada uma seqüência reconhecida pela parte *se*, então ela deve também ser reconhecida pela parte *então* da gramática, sem o que ela será retirada do autômato do texto.

A figura 7.12 mostra um exemplo de gramática. A parte *se* reconhece um verbo na segunda pessoa do singular seguido por um travessão e *tu*, seja como pronome, seja como particípio passado do verbo *calar*. A parte *então* impõe que *tu* seja então considerado como pronome. A figura 7.13 mostra o resultado da aplicação dessa gramática na frase "*Feras-tu cela bientôt*?". Pode-se ver sobre o autômato de baixo que o caminho correspondente a *tu* como particípio passado foi eliminado.

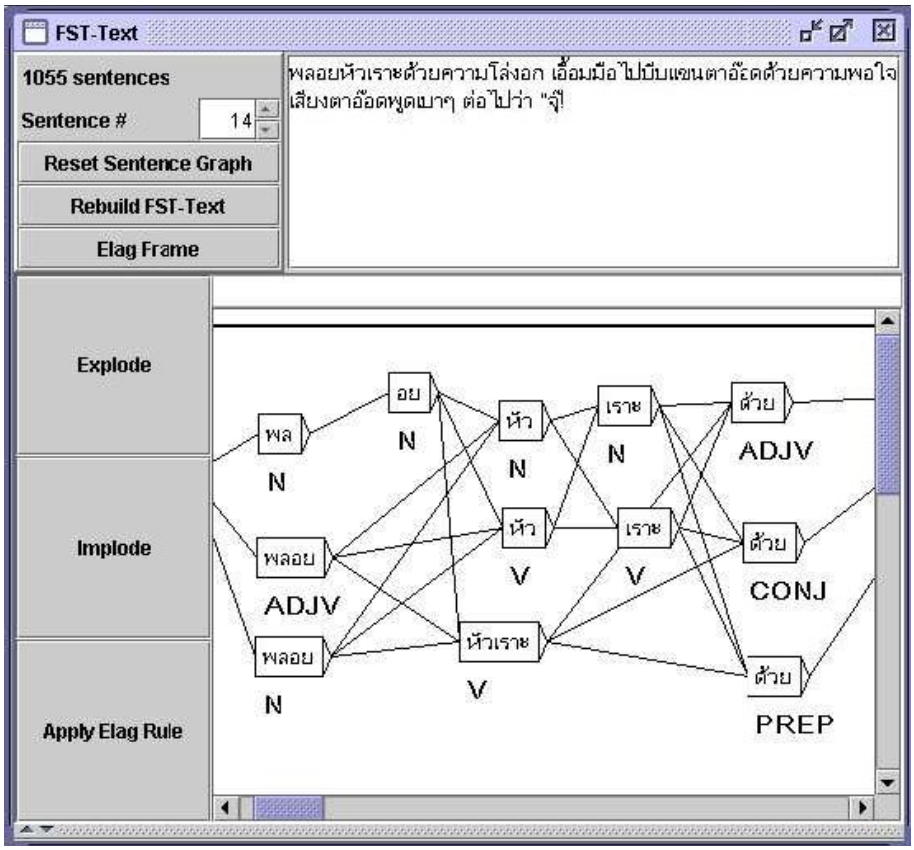



FIG. 7.11 Autômato da figura 7.9 após limpeza

**Ponto de sincronização**

As partes *se* e *então* de uma gramática ELAG são divididas em duas pelo segundo símbolo < ! > na parte *se*, e pelo segundo símbolo < = > na parte *então*. Esses símbolos formam um *ponto de sincronização*. Isso permite escrever regras nas quais as delimitações *se* e *então* não estão necessariamente alinhadas, como estão, por exemplo, no caso sobre a figura 7.14 .Essa gramática é interpretada da seguinte maneira: se for encontrado um travessão seguido por *il*, *elle* ou *on*, então esse travessão deve ser precedido por um verbo, eventualmente seguido de travessão. Desse modo, se for considerada a frase da figura 7.15 começando por *Est-il*, pode-se ver que todas as interpretações não-verbais de *Est* foram suprimidas.

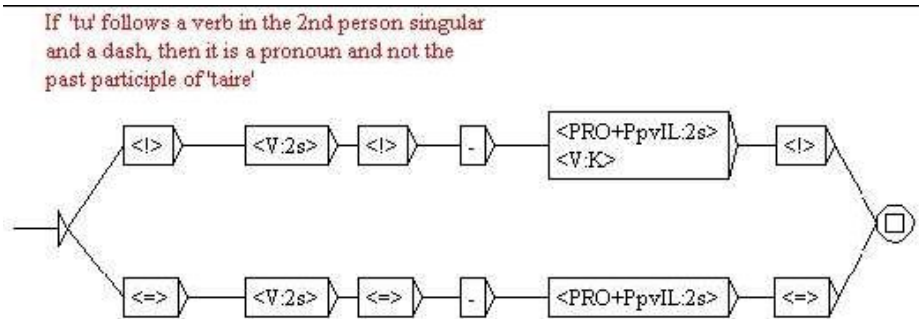


FIG. 7.12 Exemplo de gramática ELAG

### 7.3.2 Compilação das gramáticas ELAG

Antes de poder ser aplicada a um autômato de texto, uma gramática ELAG deve ser compilada em um arquivo `.rul`. Essa operação é efetuada via comando "Elag Ruler", no menu "Text", que faz com que apareça a janela da figura 7.16.

Se o quadro à direita já contiver gramáticas que você não deseje usar, você pode retirá-las por meio do botão <<. Selecione em seguida sua gramática no buscador de arquivos situado no quadro esquerdo, e clique sobre o botão >> para acrescentá-la à lista do quadro direito. Clique, então, no botão *compile*. Isso acionará o programa `ElagComp` que vai compilar a gramática selecionada para criar um arquivo nomeado como *elag.rul*.

Se você selecionou sua gramática no quadro direito, você pode pesquisar os padrões que ela reconhece clicando no botão *locate*. Isso fará com que seja aberta a janela "Locate Pattern", especificando automaticamente um nome de grafia que termine por `–conc.fst2`. Essa grafia corresponde à parte *se* da gramática. Você pode obter, assim, as ocorrências do texto sobre as quais a gramática será aplicada.

NOTA: o arquivo `– conc.fst2`, utilizado para localizar a parte *então* de uma gramática, é gerado durante a compilação das gramáticas ELAG, por meio do botão *compile*. É preciso, então, já haver compilado sua gramática antes de utilizar a função de busca do botão *locate*.

### 7.3.3 Levantamento de ambigüidades

Assim que você tiver compilado sua gramática em um arquivo `elag.rul`, você pode aplicá-la no autômato do texto. Na janela do autômato do texto, clique sobre o botão *elag*. Uma caixa de diálogo aparecerá para perguntar a você o nome do arquivo `.rul` a ser utilizado (ver figura 7.17). Como o arquivo default é mesmo `elag.rul`, simplesmente clique em "OK". Isso acionará o programa `Elag`, que vai efetuar o levantamento de ambigüidades.

Uma vez que o programa for terminado, você pode consultar o autômato resultante clicando sobre o botão *Elag Frame*. Como é possível ver a figura 7.18, a janela é separada em duas: o autômato de origem é exibido no alto, e o autômato resultante em baixo.

Não se espante se o autômato de baixo parecer mais complicado. Isso é explicado pelo fato das entradas lexicais fatorizadas[10] terem sido *explodidas*, de modo a tratar separadamente cada interpretação flexional. Para refatorizar essas entradas,

[10] Essas são as entradas que reagrupam várias interpretações flexionais diferentes, como por exemplo: {se, . PRO+PpvLE :3ms:3fs :3mp :3fp}.

clique no botão *implode*. Um clique no botão *explode* oferece uma vista explodida do autômato do texto.

Se você clicar sobre o botão *replace*, o autômato resultante se tornará o novo autômato do texto. Assim, se você utilizar outras gramáticas, elas serão aplicadas sobre o autômato já parcialmente desambigüizado, o que permite acumular os efeitos de várias gramáticas.

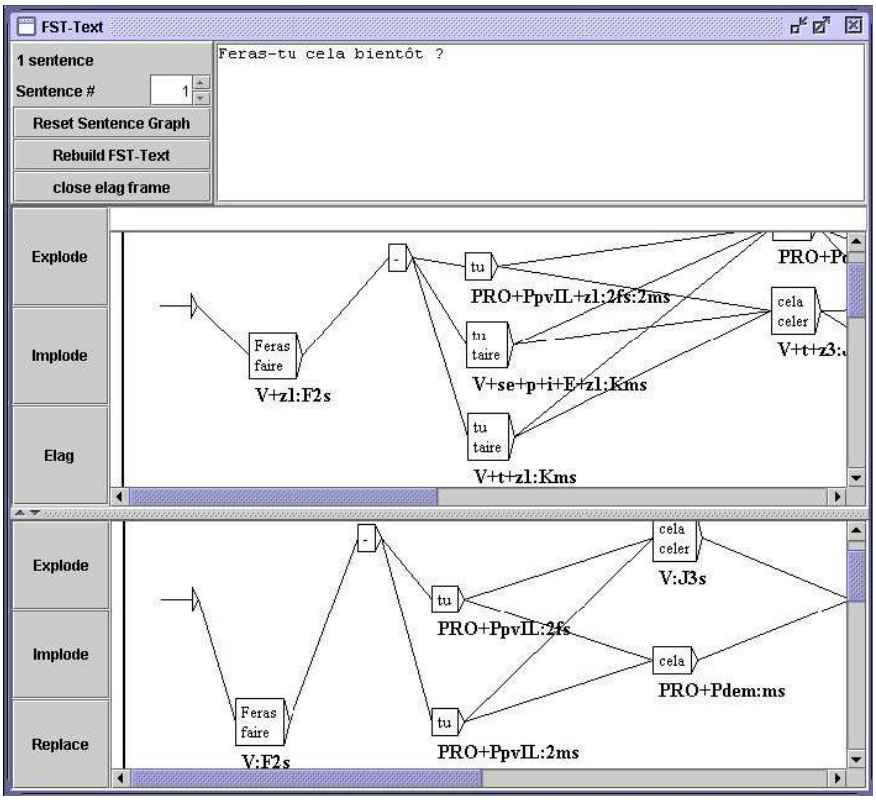



FIG. 7.13 Resultado da aplicação da gramática da figura 7.12

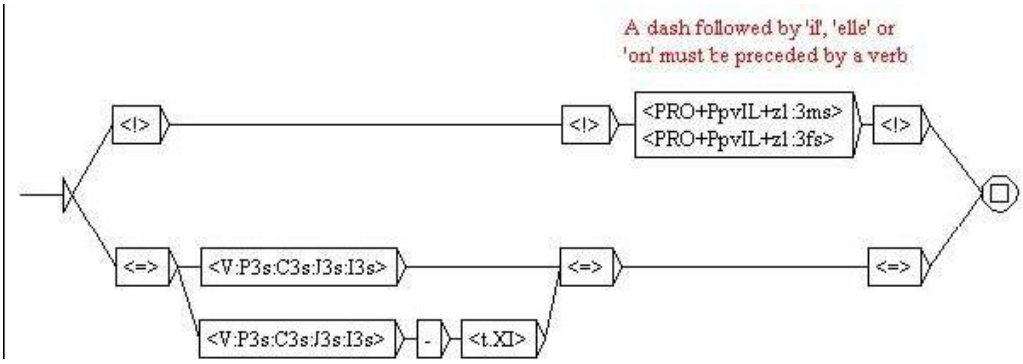



FIG. 7.14 Utilização do ponto de sincronização

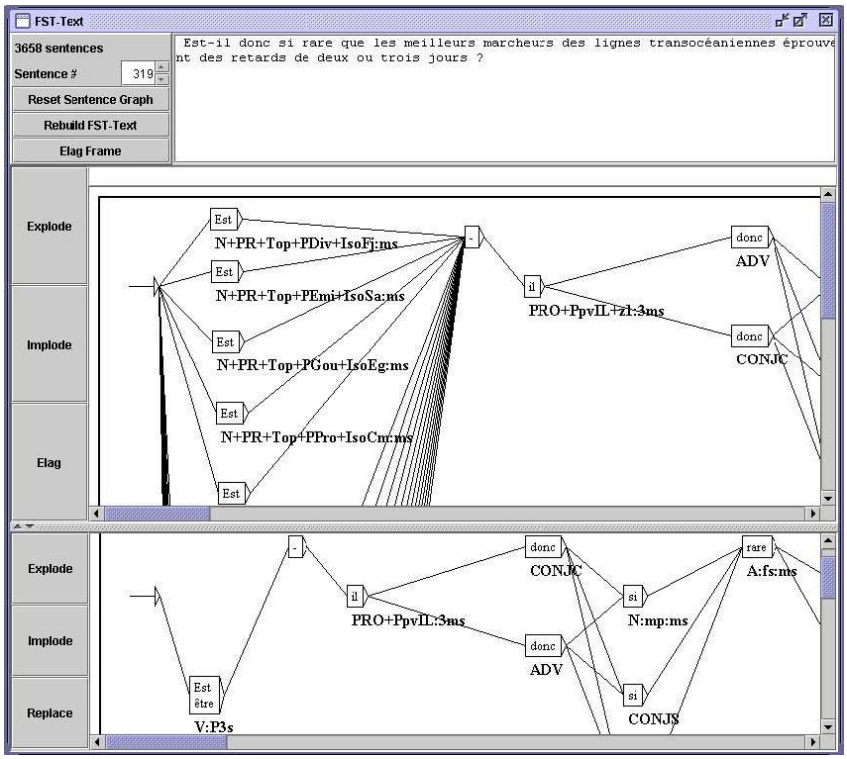



FIG. 7.15 Resultado da aplicação da gramática da figura 7.14

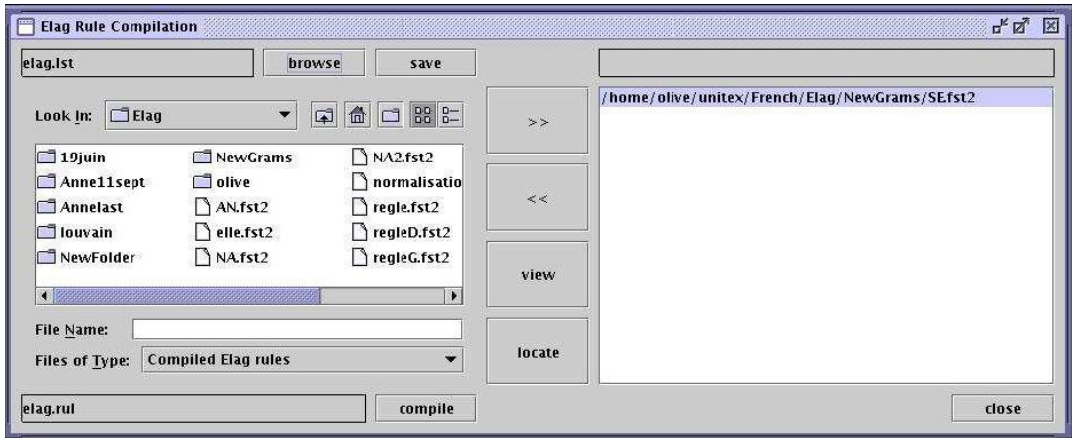



FIG. 7.16 Janela de compilação das gramáticas ELAG

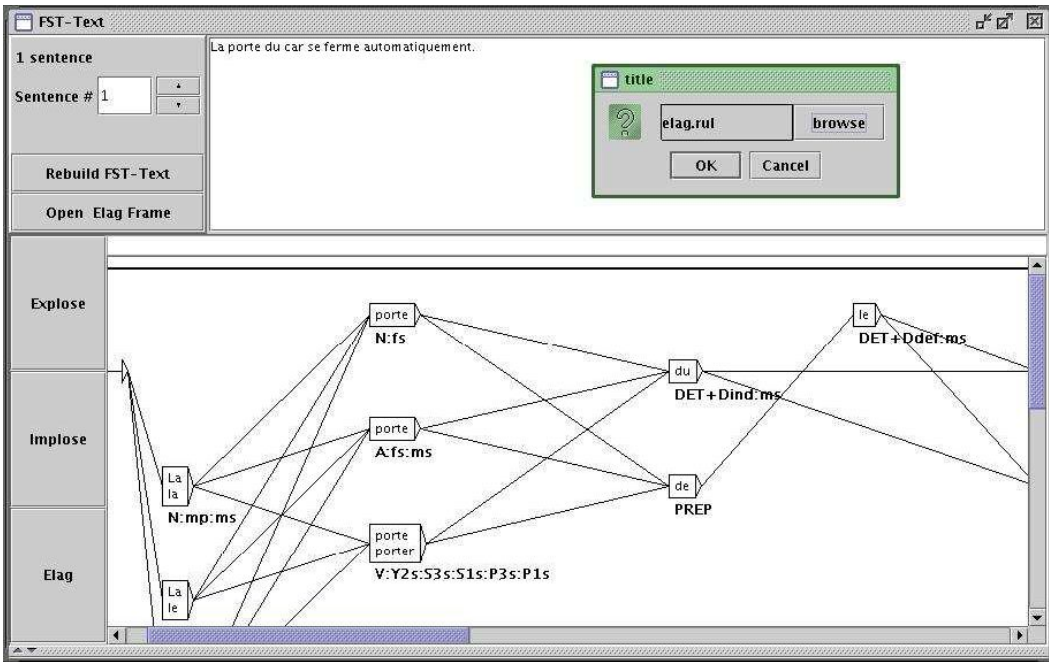



FIG. 7.17 Janela do autômato do texto

### 7.3.4 Conjuntos de Gramáticas

É possível reagrupar várias gramáticas ELAG em um conjunto de gramáticas, a fim de aplicá-las de uma só vez. Os conjuntos de gramáticas ELAG são descritos nos arquivos. 1st. Eles são administrados a partir da janela de compilação das gramáticas ELAG (figura 7.16). A marca na parte superior à esquerda indica o nome do conjunto corrente, por definição *elag. 1st*. É o conteúdo desse conjunto que está fixado no quadro direito da janela.

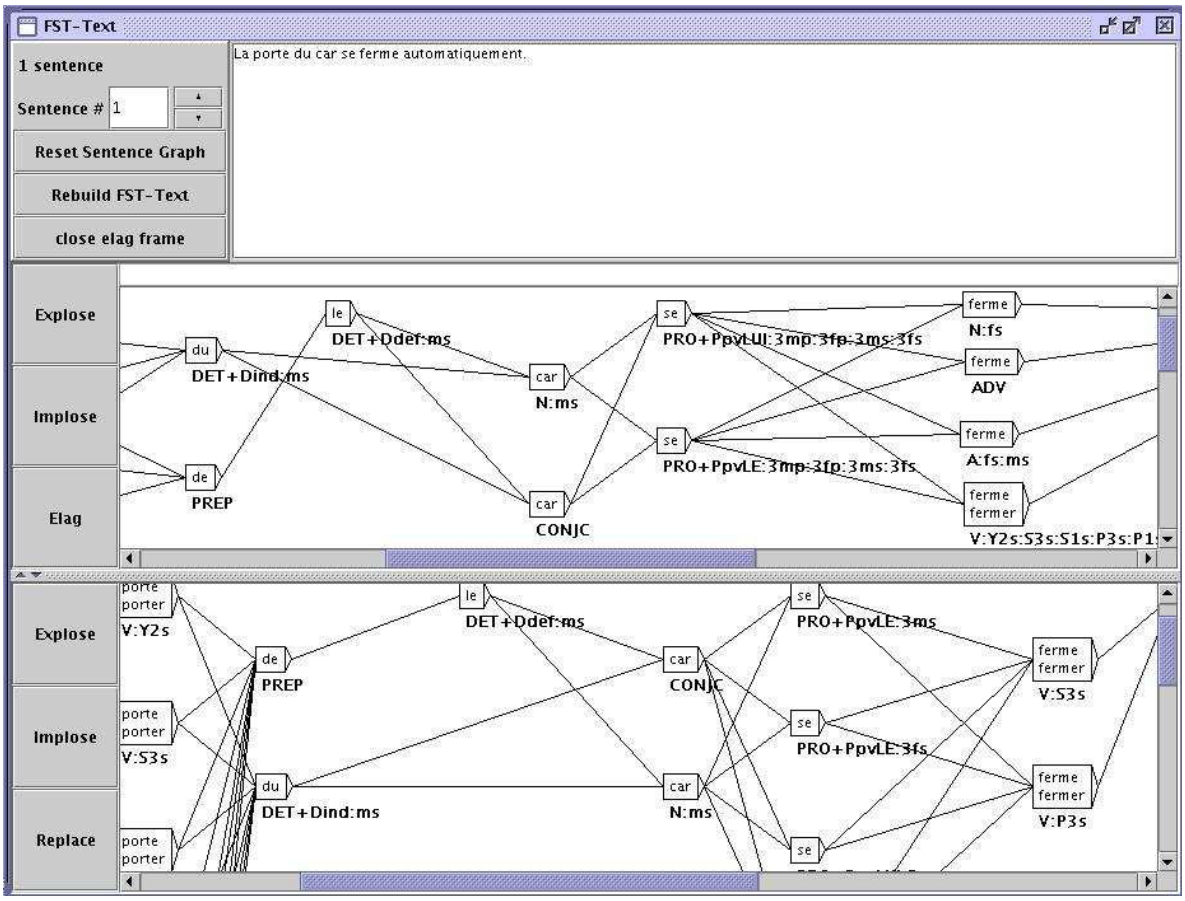



Fig. 7.18 – Janela do autômato do texto separado em duas partes

Para modificar o nome do conjunto, clique sobre o botão *browse.* Na caixa de diálogo que aparece nesse caso, escolha o nome do arquivo. *1st* que você quiser dar ao seu conjunto.

Para adicionar uma gramática ao conjunto, selecione-a no explorador de arquivos do quadro esquerdo e clique sobre o botão >>.

Para retirar uma gramática do conjunto, selecione-a no quadro direito e clique sobre o botão <<.

Uma vez selecionadas todas as suas gramáticas, compile-as clicando sobre o botão *compile.* Isso criará um arquivo. `rul`, contendo o nome indicado na parte inferior à direita (o nome do arquivo é obtido substituindo a extensão *.1st* pela extensão *.rul*).

Agora você pode aplicar seu conjunto de gramáticas. Como foi explicado mais acima, clique sobre o botão *elag* na janela do autômato do texto. Quando a caixa de diálogo lhe pedir o nome do arquivo. *rul* a ser utilizado, clique sobre o botão *browse* e selecione o seu conjunto. O autômato resultante é idêntico àquele que teria sido obtido aplicando sucessivamente cada uma das gramáticas.

### 7.3.5 **Janela de *processing* do ELAG**

No momento da desambigüização, o programa Elag é lançado em uma janela de *processing* que permite ver as mensagens emitidas pelo programa durante sua execução.

Por exemplo, quando o autômato do texto contém os símbolos que não correspondem ao conjunto de etiquetas do ELAG (ver sessão seguinte), uma mensagem indica a natureza do erro encontrado. Do mesmo modo, quando uma frase é rejeitada (todas as análises possíveis foram eliminadas pelas gramáticas), uma mensagem indica o número da frase. Isso permite localizar rapidamente a origem dos problemas.

# Avaliação do levantamento de ambigüidades

A avaliação da taxa de ambigüidade não se baseia unicamente no número médio de interpretações por palavra. A fim de ter uma medida mais representativa, o sistema leva igualmente em conta as diferentes combinações de palavras.

Durante o levantamento de ambigüidades, o programa `Elag` calcula o número de análises possíveis no autômato do texto antes e após modificação (isso corresponde ao número de caminhos possíveis no autômato). Baseando-se nessa estimativa, o programa calcula a ambigüidade média por frase e por palavra. Essa é última medida utilizada para representar a taxa de ambigüidades do texto, pois ela não varia com o tamanho do corpus, nem com o número de frases que ele possui. A fórmula aplicada é:

$$\text{Taxa de ambigüidades} = \exp \frac{\log(\text{número de caminhos})}{\text{tamanho do texto}}$$

A relação entre a taxa de ambigüidades antes e após a aplicação das gramáticas mostra o tamanho de sua eficácia.

Todas essas informações são fixadas na janela de *processing* do ELAG.

### 7.3.6 Descrição do conjunto de etiquetas

Os programas `Elag` e `ElagComp` pedem uma descrição formal do conjunto de etiquetas dos dicionários utilizados. Essa descrição consiste *grosso modo* em uma enumeração de todas as categorias gramaticais presentes nos dicionários, e em relação a cada uma delas, a lista dos códigos sintáticos e flexionais que lhes são associados, além de uma descrição de suas possíveis combinações. Essas informações são descritas no arquivo chamado *`tagset. def`*.

**Arquivo tagset.def**

Abaixo, um trecho do arquivo `tagset.def` utilizado para o francês.

```
NAME        francês
```

```
POS ADV

.

POS PRO
inflex:
     pes  = 1 2 3
     gênero     = m f
     número = s p
discr:
     subcat     = Pind Pdem PpvIL PpvLUI PpvLE Ton PpvPR PronQ
Dnom Pposs1s...completo:
Pind    “gênero”      “número”
Pdem    “gênero”      “número”
Pposs1s   “gênero”    “número”
Pposs1p   “gênero”    “número”
Pposs2s   “gênero”    “número”
Pposs2p   “gênero”    “número”
Pposs3s   “gênero”    “número”
PpvIL     “gênero”    “número”   “pes”
PpvLE     “gênero”    “número”   “pes”
PpvLUI    “gênero”    “número”   “pes” #
Ton       “gênero”    “número”   “pes” # lui, elle, moi
PpvPR                            # en y
PronQ                            # où qui que quoi
Dnom                             # rien
.

POS A # # adjetivos
inflex:
     gênero     = m f
     número = s p
cat:
     esquerda   = e
     direita    = d
complete:
“gênero” “número”
_  # pour {de bonne humeur, .A}, {au bord des larmes, .A}, por
exemplo
.

POS V
inflex:
     tempo = C F I J K P S T W Y G X
     pes  = 1 2 3
     gênero     = m f
     número = s p
complete:
W
G
C “pes” “número”
F “pes” “número”
I “pes” “número”
J “pes” “número”
P “pes” “número”
S “pes” “número”
```

```
T "pes" "número"
X 1 s # eussé dussé puissé fussé (-je)
Y 1 p
Y 2 "número"
K "gênero" "número"
```

O símbolo # indica que o resto da linha é um comentário. Um comentário pode aparecer em qualquer lugar do arquivo. O arquivo começa sempre pela palavra `NAME`, seguida de um significante (`francês`, no exemplo). A seqüência do arquivo é constituída de seções POS (de *Part-of-Speech*), uma para cada categoria gramatical. Cada seção descreve a estrutura das etiquetas das entradas lexicais pertencentes à categoria gramatical concernente. Cada seção compõe-se de quatro partes que são todas opcionais:

— inflex: essa parte enumera os códigos flexionais relativos à categoria gramatical. Por exemplo, os códigos `1, 2, 3` que denotam a pessoa da entrada são os códigos pertinentes aos pronomes, mas não aos adjetivos. Cada linha descreve um atributo flexional (gênero, tempo, etc.) e é composto do nome do atributo, seguido do sinal = e dos valores que ele pode tomar;

Por exemplo, a linha seguinte declara um atributo `pes` podendo tomar os valores 1, 2 ou 3:

```
pes = 1 2 3
```

— `cat`:

essa parte declara os atributos sintáticos e semânticos que podem ser atribuídos às entradas pertencentes às respectivas categorias gramaticais. Cada linha descreve um atributo e os valores que ele pode tomar. Os códigos declarados para um mesmo atributo devem ser exclusivos uns dos outros. Em outras palavras, uma entrada não pode ter mais de um valor para um mesmo atributo. Em contrapartida, podem existir etiquetas que não tomam nenhum valor para determinado atributo. Por exemplo, para definir o atributo `nível_de_língua` que pode ter os valores `z1, z2 e z3`, aparecerá a seguinte linha:

```
nível_de_língua = z1 z2 z3
```

— *`discr`*:

essa parte é constituída da declaração de um único atributo. A sintaxe é a mesma que na parte `cat` e o atributo descrito aqui não deve ser repetido. Essa parte permite dividir a categoria gramatical em sub-categorias *discriminantes,* nas quais as entradas têm os atributos flexionais equivalentes. Para os pronomes, por exemplo, uma indicação de pessoa é atribuída às entradas pertencendo à sub-categoria dos pronomes pessoais, mas não dos pronomes relativos. Essas dependências são descritas na parte *`complete`*;

— *`complete`*:

Nessa parte é explicitada a etiquetagem morfológica da palavra que pertence à categoria gramatical corrente. Cada linha descreve uma combinação válida de códigos flexionais em função de sua sub-categoria discriminante (se uma certa categoria foi declarada). Quando um nome de atributo aparece entre aspas ("e"), isso significa que qualquer valor desse atributo pode convir. É igualmente possível declarar que uma entrada não tem nenhum traço flexional no meio de uma linha que contenha apenas o caractere _ (underscore). Assim por exemplo, se considerarmos as linhas seguintes como trechos da seção concernente à descrição dos verbos:

```
W
```

```
K     "gênero"   "número"
```

Elas permitem declarar que os verbos no infinitivo (denotado pelo código W) não possuem outros traços flexionais posicionados, enquanto um gênero e de um número são igualmente atribuídos às formas no particípio passado (código K).

## 1.3 Descrição dos códigos flexionais

A principal função da parte *discr* é dividir as etiquetas em sub-categorias que possuam um comportamento morfológico similar. Essas sub-categorias são em seguida utilizadas para facilitar a descrição da parte complete. Para a legibilidade das gramáticas ELAG, é desejável que os elementos de uma mesma sub-categoria tenham todos o mesmo comportamento flexional; nesse caso a parte *complete* é composta de uma única linha por sub-categoria.

Consideramos o exemplo das linhas seguintes, trecho da descrição dos pronomes:

```
Pdem "gênero"   "número"
PpvIL"gênero"   "número"   "pes"
PpvPr
```

Essas linhas significam:

- todos os pronomes demonstrativos (<PRO+Pdem>) têm as indicações de gênero e número, e nenhuma outra;
- os pronomes pessoais nominativos (<PRO+PpvIL>) são etiquetados morfologicamente por uma pessoa, um gênero e um número;
- os pronomes preposicionais (*en, y*) não possuem traço flexional.

Todas as combinações de traços flexionais e discriminantes que aparecem nos dicionários devem ser descritos no arquivo `tagset.def`, senão as entradas correspondentes serão rejeitadas pelo ELAG.

No caso em que as palavras de uma mesma sub-categoria difiram pelos seus traços flexionais, é necessário descrever várias linhas na parte *complete*. O inconveniente desse método de descrição é que fica difícil fazer a distinção entre certas palavras de uma gramática ELAG.

Se considerarmos a descrição precedente dada como exemplo, alguns adjetivos do francês possuem um gênero e um número, enquanto que outros não possuem nenhum traço flexional. É por exemplo o caso de seqüências fixas como *de bonne humeur*, que têm um comportamento sintático muito próximo daquele dos adjetivos. Tais seqüências foram assim integradas no dicionário do francês na qualidade de adjetivos invariáveis e, portanto sem traço flexional. O problema é que se quisermos fazer referência exclusivamente a esse tipo de adjetivos em uma gramática de desambigüização, o símbolo "A" não convém, já que ele dará todos os adjetivos.

Para contornar essa dificuldade, é possível negar um atributo flexional, escrevendo o caractere @ exatamente antes de um dos valores possíveis para esse atributo. Dessa maneira, o símbolo <A: @m@p> reconhece todos os adjetivos que não

possuem gênero, nem número. Com a ajuda desse operador, agora é possível descrever as gramáticas como aquelas da figura 7.19, que impõem a concordância em gênero e número entre um nome e um adjetivo que o precede[2]. Essa gramática conservará a análise correta de frases como: *Les personnes de bonne humeur m'insupportent.*

Todavia é recomendável limitar o uso do operador @, pois isso nega a legibilidade das gramáticas. É preferível distinguir as etiquetas que aceitam diferentes combinações flexionais no meio de sub-categorias discriminantes, definidas na parte `discr`.

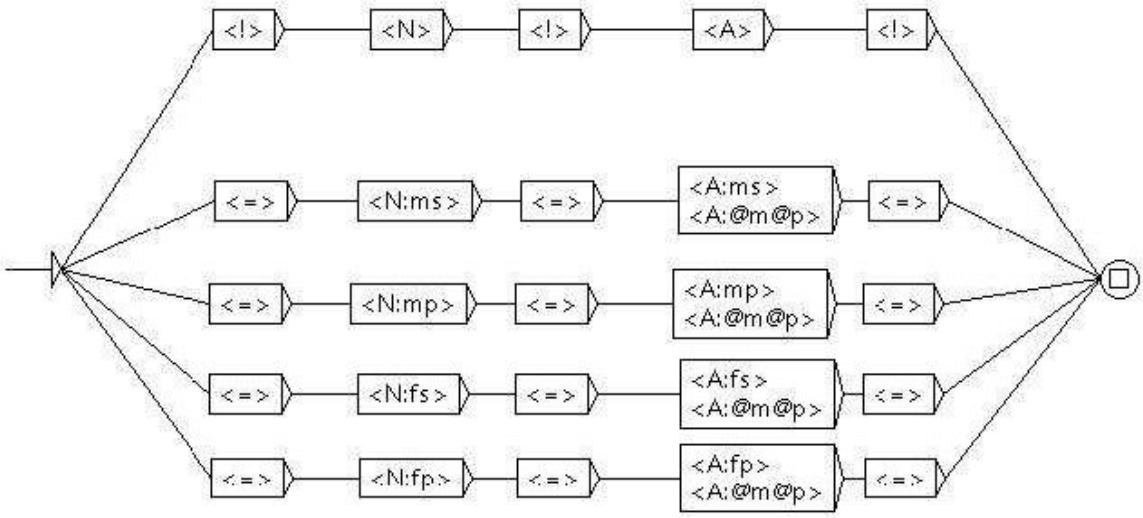



Fig. 7.19 – Gramática ELAG verificando a concordância em gênero e número entre um nome e um adjetivo subseqüente

## Códigos opcionais

Os códigos sintáticos e semânticos opcionais são declarados na parte *cat*. Eles podem ser utilizados nas gramáticas ELAG como os outros códigos. A diferença é que esses códigos não intervêem para decidir se uma etiqueta deve ser rejeitada como inválida ou não, no momento do carregamento do autômato do texto. Esses são os códigos facultativos, que são independentes dos outros códigos, como por exemplo, o atributo do nível de língua (z1, z2 ou z3). Da mesma maneira que para os códigos flexionais, é igualmente possível negar um atributo flexional escrevendo o caractere! exatamente antes do nome do atributo. Assim, com o nosso arquivo de exemplo, o símbolo `<A ! esquerda : f>` reconhece todos os adjetivos no feminino que não possuem o código `g`[3].

[2] Essa gramática não está completamente correta, pois elimina, por exemplo, a análise correta da frase: *J'ai reçu des coups de fil de ma mère hallucinants.*

Todos os códigos que não são declarados no arquivo *tagset.def* são ignorados pelo ELAG. Se uma entrada de dicionário contém certo código, ELAG produzirá um aviso e retirará o código da entrada. Por conseqüência, se duas entradas concorrentes só diferirem do autômato do texto de origem por códigos não declarados, essas entradas deverão se tornar indistinguíveis pelos programas e serão, portanto, unificadas em uma única entrada no autômato resultado. Desse modo, o conjunto de etiquetas descrito no arquivo *tagset.def* pode chegar a reduzir a ambigüidade, produzindo palavras que difiram apenas pelos códigos não declarados e isso independentemente das gramáticas aplicadas.

Por exemplo, na versão mais completa do dicionário do francês, cada emprego distinto de um verbo é caracterizado por uma referência para a tabela do léxico-gramática que a caracteriza. Consideramos até agora que essas informações ressaltam mais a sintaxe que a análise lexical e, portanto, não integramos na descrição do conjunto de etiquetas. Estas são, portanto, automaticamente eliminadas no momento do carregamento do autômato do texto, o que reduz a taxa de ambigüidades.

A fim de distinguir bem os efeitos ligados ao conjunto de etiquetas das gramáticas ELAG, é aconselhável proceder a uma etapa prévia de normalização do autômato do texto antes de aplicar-lhe as gramáticas de desambigüização. Essa normalização efetua-se aplicando ao autômato do texto uma gramática que não imponha nenhum incômodo, como aquela da figura 7.20. Note que essa gramática está normalmente presente na distribuição do Unitex e pré-compilada no arquivo `norm.rul`.

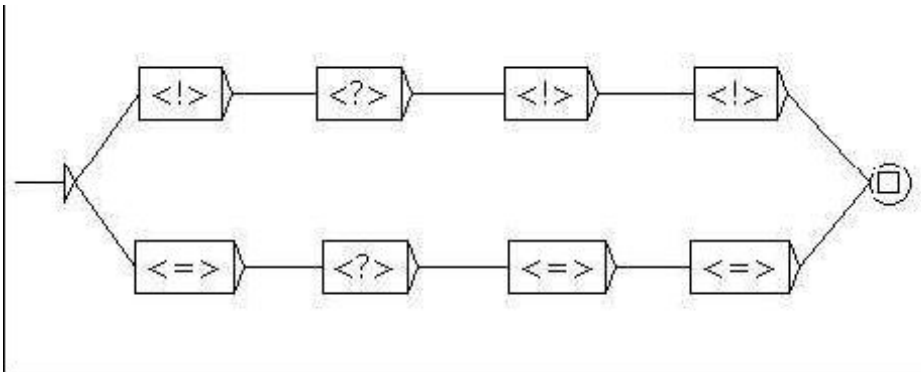



Fig. 7.20 – Gramática ELAG sem nenhuma restrição

O resultado da aplicação dessa gramática é que o autômato de origem está limpo de todos os códigos que ou não estão descritos no arquivo *tagset.def*, ou não estão de acordo com essa descrição (por causa de categorias gramaticais desconhecidas ou de combinações inválidas de traços flexionais). Trocando, então, o autômato do texto pelo autômato assim normalizado, pode-se estar certo de que as modificações posteriores do autômato serão devidas unicamente aos efeitos das gramáticas ELAG.

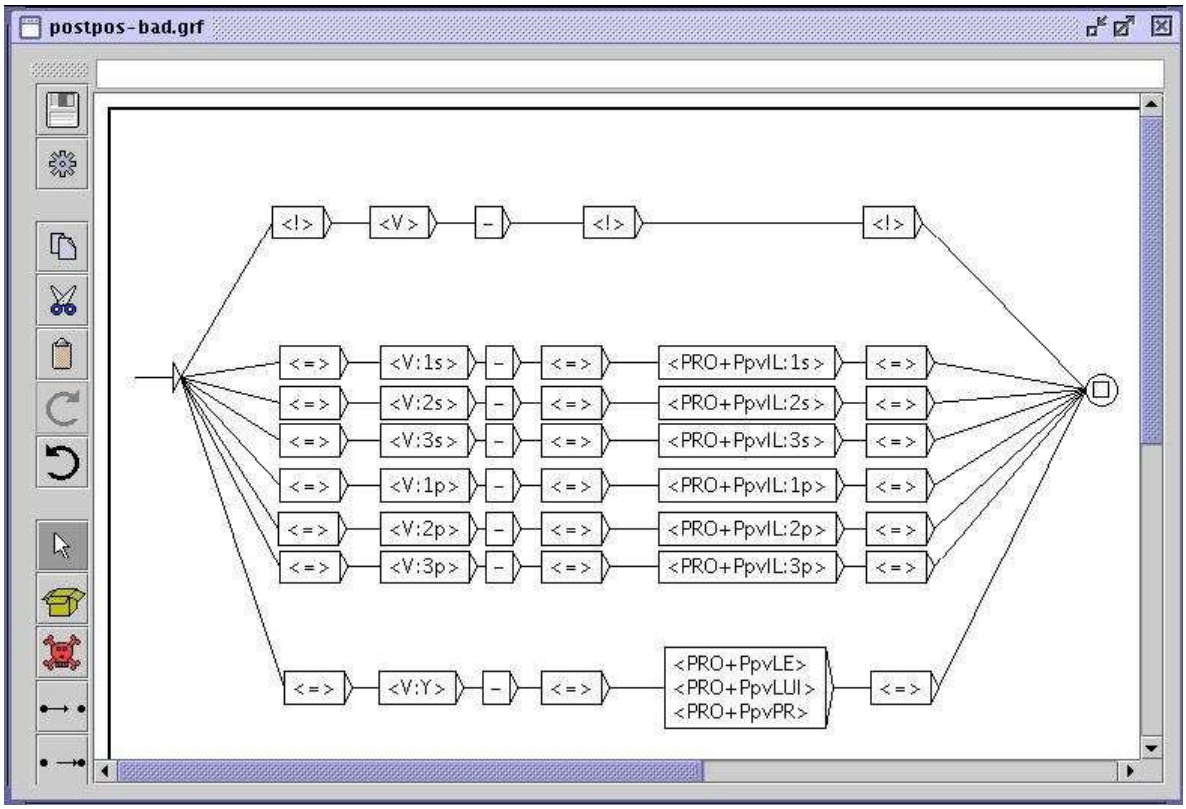
 Esse código indica que o adjetivo deve aparecer à esquerda do nome ao qual ele se refere, como é o caso para *bel*.

### 7.3.7 **Otimizar as gramáticas**

A compilação das gramáticas efetuadas pelo programa `ElagComp` consiste em construir um autômato cuja linguagem é o conjunto das seqüências de entradas lexicais (ou interpretação lexical de uma frase) que não são rejeitadas pelas gramáticas. Essa tarefa é complexa e pode levar muito tempo; é, contudo possível acelerá-la sensivelmente observando certos princípios no momento em que as gramáticas são escritas.

#### Limitar o número de ramificações *então*

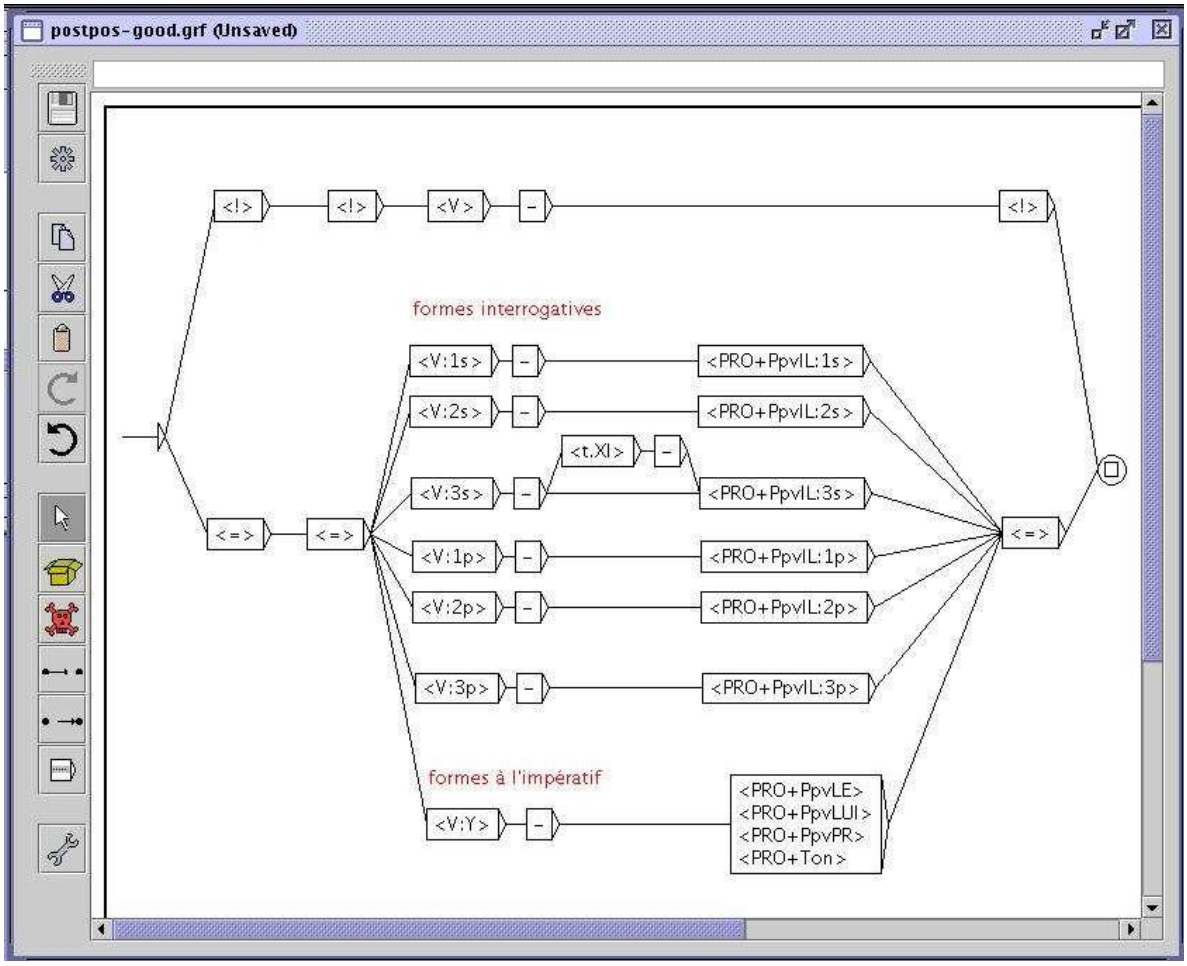
É recomendável reduzir ao máximo o número de partes *então* de uma gramática. Isso pode reduzir consideravelmente o tempo de compilação das gramáticas. Habitualmente, uma gramática que possui muitas partes *então* pode ser reescrita com uma ou duas partes *então*, sem perda de legibilidade. É por exemplo o caso da gramática da figura 7.21, que impõe um contrato entre um verbo e o pronome subseqüente.




7.21 – Gramática ELAG verificando a concordância entre verbo e pronome

Como se pode ver na figura 7.22 pode-se escrever uma gramática equivalente fatorando todas as partes *então* em uma única parte. As duas gramáticas terão

exatamente o mesmo efeito sobre o autômato do texto, mas a segunda será compilada muito mais rapidamente.




7.22 – Gramática ELAG otimizada verificando a concordância entre verbo e pronome

## Utilização dos símbolos lexicais

É melhor utilizar os lemas apenas quando é absolutamente necessário. Isso é particularmente verdadeiro para as palavras gramaticais, quando suas subcategorias possuem tanta informação quanto os próprios lemas. Se utilizar apesar de tudo um lema em um símbolo, recomenda-se explicitar da melhor maneira possível os traços sintáticos, semânticos e flexionais.

Por exemplo, com os dicionários fornecidos pelo francês, é preferível substituir os símbolos como `<je.PRO:1s>`, `<je.PRO+PpvIL:1s>` e `<je.PRO>` pelo símbolo `<PRO+PpvIL:1s>`. De fato, todos os símbolos são idênticos na medida em que podem reconhecer apenas a última entrada do dicionário `{je, .PRO+PpvIL:1ms:1fs}`. Entretanto, como o programa não pode deduzir automaticamente essa informação, se

não explicíta-se todos esses traços, o programa considerará em vão as etiquetas não existentes, tais como <je. PRO:3p>, <je.PRO+PronQ>, etc.

## 7.4 Manipulação do autômato do texto

### 7.4.1 Visualização dos autômatos de frases

Como vimos anteriormente, o autômato de um texto é na realidade um conjunto dos autômatos das frases desse texto. Essa estrutura pode ser representada graças ao formato. *fst2*, utilizado para representar as gramáticas compiladas.

Entretanto, esse formato não permite mostrar diretamente os autômatos de frases. É preciso, portanto utilizar um programa (Fst2Grf) para converter um autômato de frase em um gráfico para que ele possa ser visualizado. Esse programa aparece automaticamente quando você seleciona uma frase para gerar o arquivo *.grf* correspondente.

Os arquivos .grf gerados não são interpretados da mesma maneira que os arquivos .grf que representam os gráficos construídos pelo usuário. De fato, em um gráfico normal, as linhas de uma caixa são separadas pelo símbolo +. Em um gráfico de frase, cada caixa é ora uma unidade lexical sem etiqueta, ora uma entrada de dicionário entre colchetes. Se a caixa contiver apenas uma unidade sem etiqueta, essa aparece somente na caixa. Se a caixa contiver uma entrada de dicionário, a forma flexionada é mostrada, seguida de sua forma canônica se essa for diferente. As informações gramaticais e flexionais são mostradas sob a caixa, como nas transducções.

A figura 7.23 mostra o gráfico obtido pela primeira frase de *Ivanhoe.* As palavras *Ivanhoe, Walter e Scott* são consideradas como palavras desconhecidas. A palavra *by* corresponde a duas entradas no dicionário. A palavra *Sir* corresponde igualmente a duas entradas do dicionário, mas como a forma canônica dessas entradas é *sir*, ela é mostrada, já que difere da forma flexiva por uma minúscula.

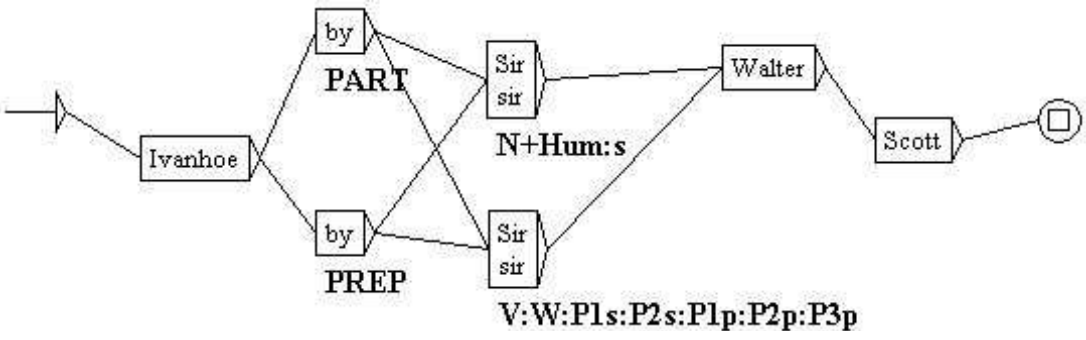



Fig. 7.23 – Autômato da primeira frase de *Ivanhoe*

### 7.4.2 Modificar manualmente o autômato do texto

É possível modificar manualmente os autômatos de frase, salvo aqueles que aparecem no quadro reservado para a ELAG (quadro de baixo). Você pode adicionar

ou suprimir as caixas ou as transições. Quando um gráfico é modificado, ele é protegido na pasta do texto sob o nome de `sentenceN.grf`, onde *N* representa o número da frase.

Quando você seleciona uma frase, se um gráfico modificado existir para essa frase, está é mostrada. Você pode então reinicializar o autômato dessa frase clicando sobre o botão "*Reset Sentence Graph*" (ver figura 7.24).

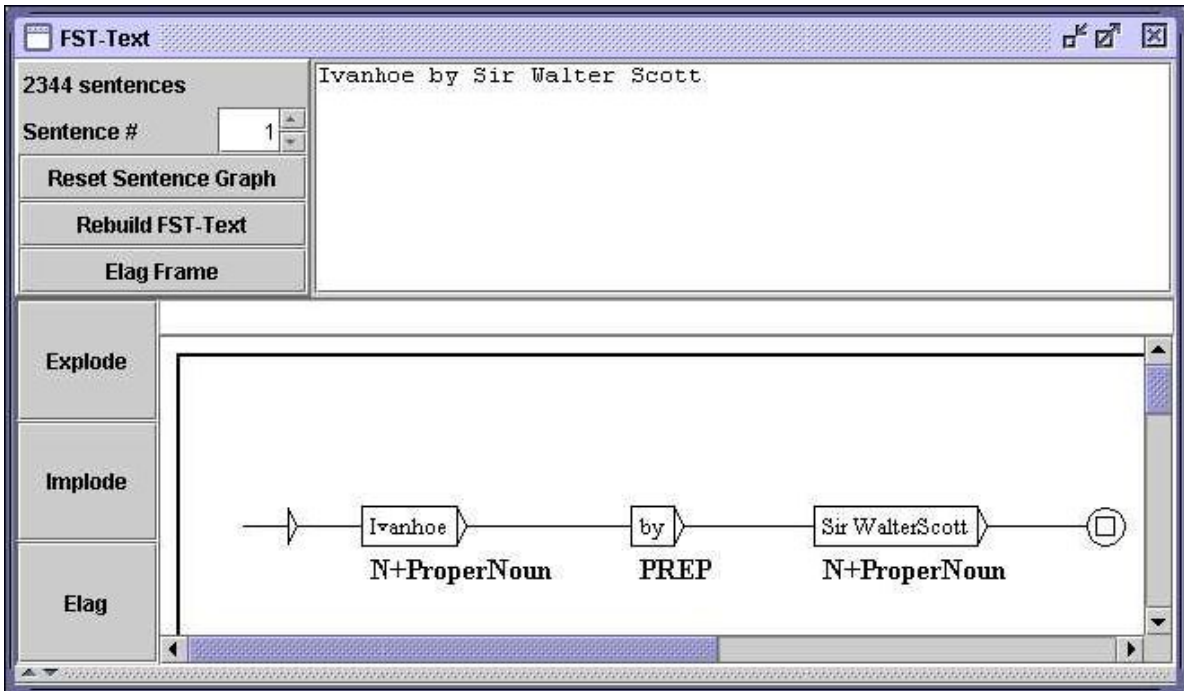



Fig. 7.24 – Autômato de frase modificado

No momento da construção do autômato de um texto, todos os gráficos de frase modificados presentes na pasta do texto são cancelados.

NOTA: você pode reconstruir o autômato do texto levando em consideração suas modificações manuais. Para isso, clique no botão "Rebuild FST-Text". Todas as frases para as quais as modificações foram feitas são agora substituídas no autômato do texto pela sua versão modificada. O novo autômato do texto é em seguida recarregado automaticamente.

### 7.4.3 Parâmetros de apresentação

Os autômatos de frase são submetidos às mesmas opções de apresentação que os gráficos. Eles compartilham as mesmas cores e fontes, do mesmo modo que a utilização do efeito de *antialiasing*. Para configurar a aparência dos autômatos de frase, você deve modificar a configuração geral clicando em "Preferences..." no menu "Info". Para mais detalhes, veja a seção 5.3.5.

Você pode igualmente imprimir um autômato de frase clicando em "Print..." no menu "FSGraph" ou clicando em "Ctrl+P". Assegure-se que o parâmetro de orientação

da impressora esteja bem acertado no modo paisagem. Para acertar esse parâmetro, clique em *"Page Setup" no menu "FSGraph"*.

## 7.5 Converter o autômato do texto em texto linear

Se o autômato do texto não contiver mais a menor ambigüidade, é possível construir um arquivo texto correspondente ao único caminho representado por esse autômato. Para isso, vá ao menu "Texte" e clique em "Convert FST-Texte to Texte...". A janela da figura 7.25 permite dessa maneira que você defina o arquivo texto de saída.

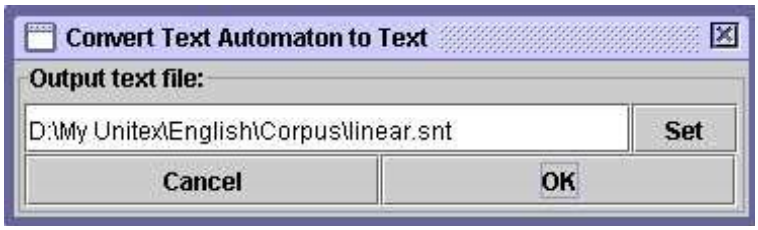



Fig. 7.25 – Escolha do arquivo de saída para a linearização do autômato do texto

Se o autômato não estiver completamente linear, uma mensagem de erro lhe indicará o número da primeira frase contendo uma ambigüidade. Senão, o programa `Fst2Unambig` construirá o arquivo de saída segundo os princípios seguintes:

— o arquivo de saída contém uma linha por frase;
— todas as frases com exceção da última são terminadas por {S};
— para cada caixa, o programa escreve seu conteúdo seguido por um espaço.

NOTA: a gestão dos espaços é deixada inteiramente para o usuário. Assim, se o texto de origem é o do autômato de frase da figura 7.26, o texto produzido será:

```
2 3 {cats, cat . N+Anl: p} {are, be . V: p2s: p1p: P2p: P3p}
{white, white . A} .
```

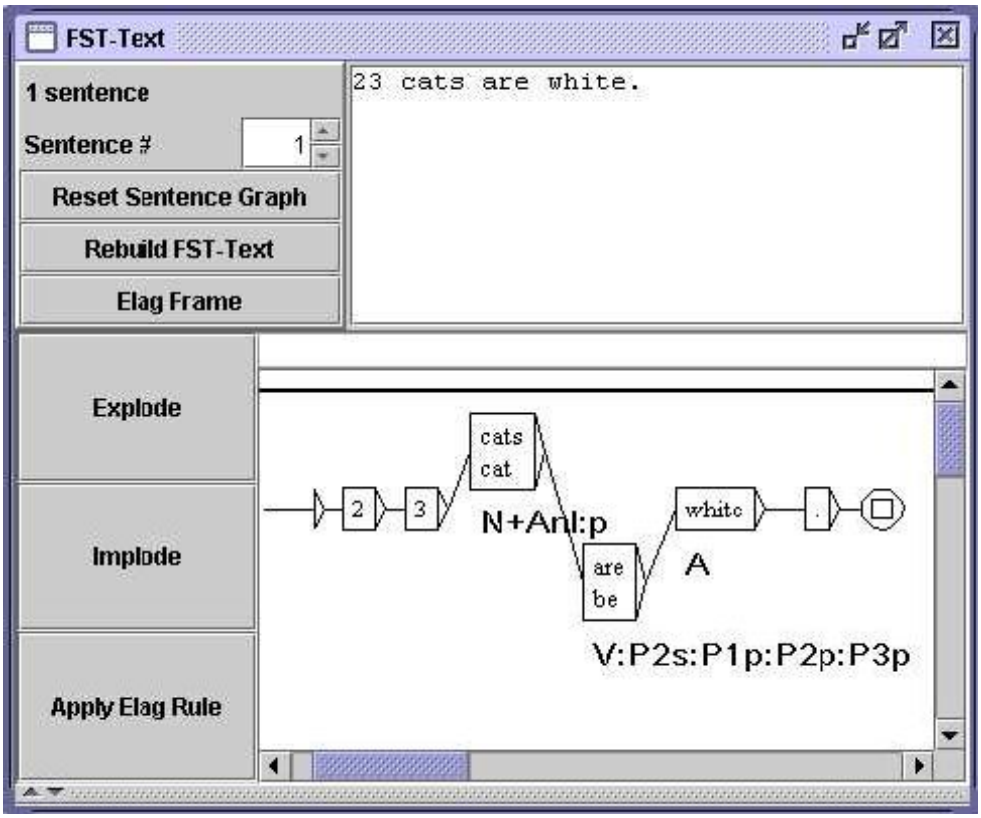



Fig. 7.26 – Exemplo do autômato de texto linear

# Capítulo 8

# Léxico-gramática

As tabelas de léxico-gramática são um meio compacto de representar as propriedades sintáticas dos elementos de uma língua. É possível construir gramáticas locais automaticamente a partir dessas tabelas, graças a um mecanismo de grafos parametrizados.

A primeira parte deste capítulo apresenta a formalização dessas tabelas. A segunda parte descreve os grafos parametrizados e o mecanismo de geração automática de grafos a partir de uma tabela de léxico-gramática.

## 8.1 As tabelas de léxico-gramática

A léxico-gramática é uma metodologia que foi desenvolvida por Maurice Gross e sua equipe do LADL ([6], [7], [26], [28]) com o seguinte princípio: cada verbo possui propriedades sintáticas quase únicas. Sendo assim, essas propriedades têm de ser sistematicamente descritas, pois é impossível prever o comportamento preciso de um

verbo. Essas descrições sistemáticas são representadas por meio de matrizes nas quais as linhas correspondem aos verbos, e as colunas, às propriedades sintáticas. As propriedades consideradas são propriedades formais tais como o número e a natureza dos complementos admitidos pelo verbo e as diferentes transformações que esse verbo pode sofrer (passividade, nominalização, extraposição, etc.). As matrizes, mais comumente denominadas tabelas, são binárias: um sinal + aparece na intersecção de uma linha e de uma coluna de uma propriedade se o verbo verificar essa propriedade, caso contrário, aparece um sinal - .

Esse tipo de descrição foi igualmente aplicada aos adjetivos ([37]), aos substantivos predicativos ([21], [22]), aos advérbios ([27], [39]), assim como às expressões cristalizadas, sendo este procedimento feito em várias línguas ([10], [17], [18], [42], [43], [45], [48], [49], [50]).

A figura 8.1 mostra um exemplo de tabela de léxico-gramática. Essa tabela aplica-se aos verbos que admitem um complemento numérico.

Table32NM.xls

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N0 = Npc | N0 = N-hum | N0 = Nnr | N0 = V-n | Aux | 32NM | N1 V | N1 = Nhum | N1 = N-hum | N1 = le fait Qu P | N1 = V-n | N1 = Dnum Nmes | Ppv = le | N-1 V N0 | N0 V A-dj | N0 V Dnum V-n | N0 V à N1 | N-1 V N0 (<E>=à) N1 | N1 = V-n | Exemple |
| 2 | - | + | - | - | avoir | accepter | - | - | + | - | - | - | + | - | - | - | - | - | - | Ce salon§accepte§vingt personnes |
| 3 | - | + | - | - | avoir | accueillir | - | - | + | - | - | - | + | - | - | - | - | - | - | Ce salon§accueille§vingt personnes |
| 4 | - | + | - | - | avoir | accuser | - | - | + | - | - | + | + | - | - | - | - | - | - | Max§accuse§80 kilos |
| 5 | - | + | - | - | avoir | accuser | - | - | + | + | - | - | - | - | - | - | - | - | - | Max§accuse§ses trente ans |
| 6 | - | + | - | - | avoir | admettre | - | - | + | - | - | - | + | - | - | - | - | - | - | On§admet§50 personnes dans cette salle |
| 7 | - | + | - | - | avoir | affecter | - | - | + | - | - | - | - | - | - | - | - | - | - | Ces cristaux§affectent§une forme géométrique |
| 8 | - | + | - | - | avoir | afficher | - | - | + | - | - | + | - | - | - | - | - | - | - | Les valeurs ont§affiché§un repli |
| 9 | - | + | - | - | avoir | aimer | - | - | + | + | - | - | + | - | - | - | - | - | - | La plante§aime§l'eau |
| 10 | - | + | - | - | avoir | approcher | + | - | + | - | - | + | + | - | - | - | - | - | - | Cette maison§approche§les deux millions |
| 11 | - | + | - | - | avoir | arpenter | - | - | - | - | - | + | + | - | - | + | + | + | - | Ce terrain§arpente§30 arpents |
| 12 | - | + | - | - | avoir | atteindre | - | - | + | - | - | + | + | - | - | - | - | - | - | Max§atteint§80 kilos |
| 13 | + | + | + | - | avoir | avoir | - | - | + | - | - | + | + | - | - | - | - | - | - | Max§a§(une soeur+une voiture+des sous) |
| 14 | - | + | - | - | avoir | avoisiner | - | - | + | - | - | + | + | - | - | - | - | - | - | Ce sac§avoisine§les 20 kg. |
| 15 | - | + | - | - | avoir | battre | + | - | + | - | - | - | + | - | - | - | - | - | - | La montre§bat§les secondes |
| 16 | - | + | - | - | avoir | cacher | - | - | + | - | - | - | - | - | - | - | - | - | - | Son calme§cache§(son+une grande)angoisse |
| 17 | - | + | - | - | avoir | caler | - | - | + | - | - | + | + | - | + | - | - | - | - | Ce bateau§cale§80 cm |

c32NM

FIG. 8.1 - Tabela de léxico-gramática 32NM

## 8.2 Conversão de uma tabela em grafos

### 8.2.1 Princípio dos grafos parametrizados

A conversão de uma tabela em grafos é efetuada por meio de um mecanismo de grafos parametrizados. O princípio é o seguinte: constrói-se um grafo que descreve as construções possíveis. Esse grafo faz referência às colunas da tabela por meio de variáveis. Gera-se, em seguida, para cada linha da tabela, uma cópia desse grafo, na qual as variáveis são substituídas em função do conteúdo das células situadas na intersecção das colunas correspondentes e da linha tratada. Se uma célula da tabela contiver o sinal +, a variável correspondente será substituída por <E>. Se a célula contiver o sinal -, a caixa contendo a variável correspondente é suprimida, o que destrói, ao mesmo tempo, os caminhos que passam por essa caixa. Em todos os outros casos, a variável é substituída pelo conteúdo da célula.

### 8.2.2 Formato da tabela

As tabelas de léxico-gramática são geralmente codificadas com a ajuda de uma planilha eletrônica, tal como o OpenOffice.org Calc ([41]). Para poderem ser utilizadas pelo Unitex, as tabelas devem ser codificadas em caracteres Unicode de acordo com a seguinte convenção: as colunas devem ser separadas por tabulação e as linhas por quebra de página.

Para converter uma tabela com o OpenOffice.org Calc, deve-se salvá-la em formato texto (extension.csv). O programa irá propor, em seguida, parametrar o backup por meio de uma janela como a da figura 8.2. Deve-se, então, escolher o padrão "Unicode", selecionar a tabulação como separador de colunas e não especificar o delimitador de texto.


FIG. 8.2 - Configuração do backup de uma tabela com o OpenOffice.org Calc

Durante a geração dos grafos, o Unitex salta a primeira linha, considerada como cabeçalho das colunas. Deve-se, então, assegurar-se de que o cabeçalho das colunas ocupe exatamente uma linha. Se não houver cabeçalho, a primeira linha da tabela será ignorada e, se houver vários cabeçalhos, eles serão interpretados a partir do segundo como linhas da tabela.

### 8.2.3 Os grafos parametrizados

Os grafos parametrizados são grafos nos quais aparecem variáveis referentes às colunas de uma tabela de léxico-gramática. Geralmente se utiliza esse mecanismo com grafos sintáticos, mas nada impediria de construir grafos parametrizados de flexão, de pré-tratamento ou de normalização.

As variáveis referentes às colunas são formadas pelo caractere `@` (arroba) seguido de um nome de coluna em letras maiúsculas (as colunas são numeradas a partir da letra A).

Exemplo: `@C` refere-se à terceira coluna da tabela.

Quando uma variável deve ser substituída por um + ou por um -, o sinal – corresponde à supressão do caminho que passa por essa variável. É possível realizar a operação contrária colocando um ponto de exclamação antes do caractere `@`. Nesse caso, quando a variável remete ao sinal + o caminho é suprimido. Se a variável não remeter nem ao sinal + e nem ao sinal -, ela será substituída pelo conteúdo da célula.

Existe, igualmente, uma variável especial `@%`, que é substituída pelo número da linha na tabela. O fato de seu valor ser diferente para cada linha permite utilizá-la para caracterizar facilmente uma linha. Essa variável não é afetada pela presença da um ponto de exclamação à sua esquerda.

A figura 8.3 mostra um exemplo de grafo parametrizado concebido para ser aplicado à tabela de léxico-gramática 31H, apresentada na figura 8.4.

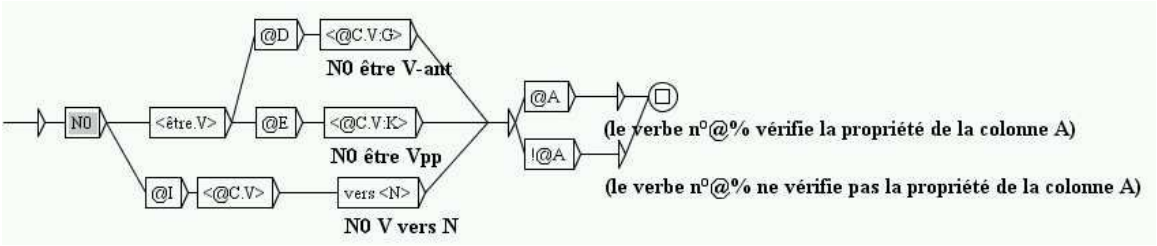



FIG. 8.3 – Exemplo de grafo parametrizado
(o verbo n°@% verifica a propriedade da coluna A)
(o verbo n°@% não verifica a propriedade da coluna A)

Table31H.xls

| 1 | N0 =: V-n | Aux | 31H | N0 est V-ant | N0 est Vpp | N0pc lui V | N0 V de N0pc | Nhum V sur ce point | N0 V vers N | il V N0 W | idée Loc esprit | Nhum Loc Nabs | N0 =: N-hum | N0 =: V-n | Exemple |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | - | avoir | abandonner | - | - | - | - | - | - | - | - | - | - | - | Paul a§abandonné§ |
| 3 | - | avoir | abuser | - | - | - | - | + | - | - | - | - | - | - | Max§abuse§ |
| 4 | - | avoir | acquiescer | - | - | - | + | + | - | - | - | - | - | - | Max a§acquiescé§(E+de la tête) |
| 5 | - | avoir | adouber | - | - | - | - | - | - | - | - | - | - | - | Paul§adoube§\|échecs\| |
| 6 | - | avoir | agioter | - | - | - | - | - | - | + | - | - | - | - | Max§agiote§sur les changes |
| 7 | - | avoir | agoniser | + | - | - | - | - | - | + | - | - | + | - | Max§agonise§ |
| 8 | - | avoir | archaïser | + | - | - | - | + | - | - | - | - | - | - | Cet auteur§archaïse§volontiers |
| 9 | - | avoir | arquer | - | - | - | + | - | + | - | - | - | - | - | Max a§arqué§toute la journée |
| 10 | - | être | arriver | - | + | - | - | - | - | + | - | + | - | - | Max est§arrivé§ |
| 11 | - | avoir | atermoyer | - | - | - | - | + | - | + | - | + | - | - | Max§atermoie§ |
| 12 | + | avoir | badauder | - | - | - | - | - | + | - | + | - | - | badaud | Max§badaude§ |

c31H

FIG. 8.4 - Tabela de léxico-gramática 31H

### 8.2.4 Geração automática de grafos

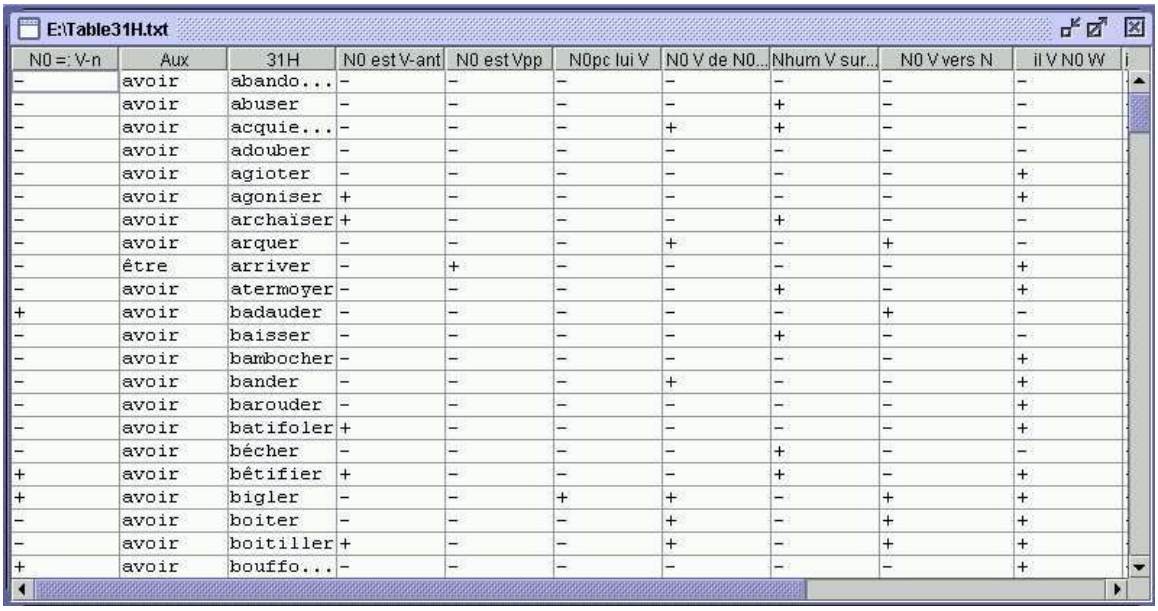
Para poder gerar grafos a partir de um grafo parametrizado e de uma tabela, é preciso, primeiramente, abrir a tabela clicando em "Open..." no menu "Lexicon-Grammar" (ver figura 8.5) . A tabela deve ter sido previamente convertida em caracteres Unicode.

A tabela selecionada é então mostrada em uma janela (ver figura 8.6).

Para gerar grafos automaticamente a partir de grafos parametrizados, clicar sobre "Compile to GRF..." no menu "Lexicon-Grammar". Aparecerá, então, a janela da figura 8.7.

Na opção "Reference Graph (in GRF format)", indicar o nome do grafo parametrizado a ser utilizado. Na opção "Resulting GRF Grammar", indicar o nome do grafo principal a ser gerado. Esse grafo principal é um grafo referente a todos os grafos que terão sido gerados. Ao lançar uma pesquisa em um texto com esse grafo, todos os grafos gerados serão assim aplicados simultaneamente.


FIG. 8.5 – Menu "Léxico-Gramática"




| N0 =: V-n | Aux | 31H | N0 est V-ant | N0 est Vpp | N0pc lui V | N0 V de N0... | Nhum V sur... | N0 V vers N | il V N0 W |
|---|---|---|---|---|---|---|---|---|---|
| - | avoir | abando... | - | - | - | - | - | - | - |
| - | avoir | abuser | - | - | - | - | + | - | - |
| - | avoir | acquie... | - | - | - | + | + | - | - |
| - | avoir | adouber | - | - | - | - | - | - | - |
| - | avoir | agioter | - | - | - | - | - | - | + |
| - | avoir | agoniser | + | - | - | - | - | - | + |
| - | avoir | archaiser | + | - | - | - | + | - | - |
| - | avoir | arquer | - | - | - | + | - | + | - |
| - | être | arriver | - | + | - | - | - | - | + |
| - | avoir | atermoyer | - | - | - | - | + | - | + |
| + | avoir | badauder | - | - | - | - | - | + | - |
| - | avoir | baisser | - | - | - | - | + | - | - |
| - | avoir | bambocher | - | - | - | - | - | - | + |
| - | avoir | bander | - | - | - | + | - | - | + |
| - | avoir | barouder | - | - | - | - | - | - | + |
| - | avoir | batifoler | + | - | - | - | - | - | + |
| - | avoir | bécher | - | - | - | - | + | - | - |
| + | avoir | bêtifier | + | - | - | - | + | - | + |
| + | avoir | bigler | - | - | + | + | - | + | + |
| - | avoir | boiter | - | - | - | + | - | + | + |
| - | avoir | boitiller | + | - | - | + | - | + | + |
| + | avoir | bouffo... | - | - | - | - | - | - | + |

Fig.8.6 - Exibição de uma tabela

O quadro "Name of produced subgraphs" permite especificar o nome dos grafos que serão gerados. A fim de assegurar-se de que todos os grafos terão nomes distintos, aconselha-se utilizar a variável `@%`; essa variável será substituída em cada entrada pelo número da mesma, garantindo assim que todos os grafos tenham um nome diferente. Por exemplo, se preenchermos esse quadro com o nome `TestGraph_@%.grf`, o grafo gerado a partir da 16ª linha será nomeado `TestGraph_0016.grf`.

As figuras 8.8 e 8.9 mostram dois grafos gerados ao aplicar o grafo paramétrico da figura 8.3 à tabela 31H. A figura 8.10 mostra o grafo principal obtido.

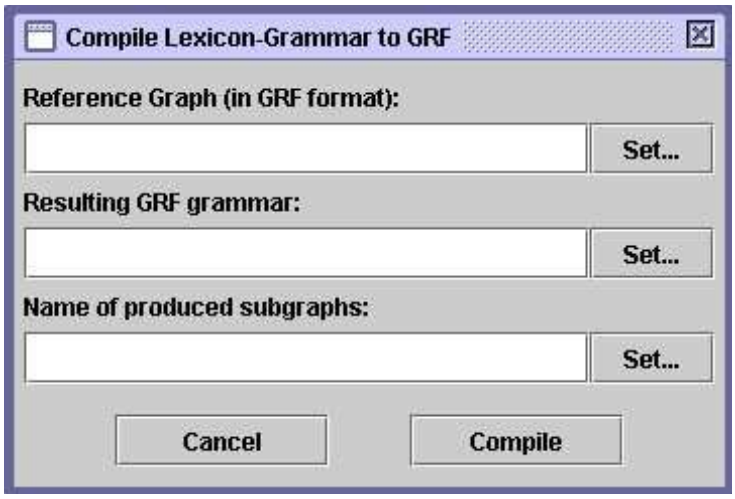



Fig. 8.7 - Configuração da geração automática de grafos

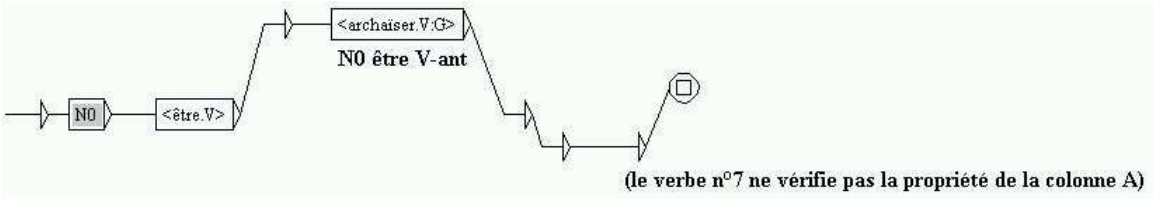



Fig. 8.8 - Grafo gerado para o verbo `archaïser`
(o verbo n°7 não verifica a propriedade da coluna A)

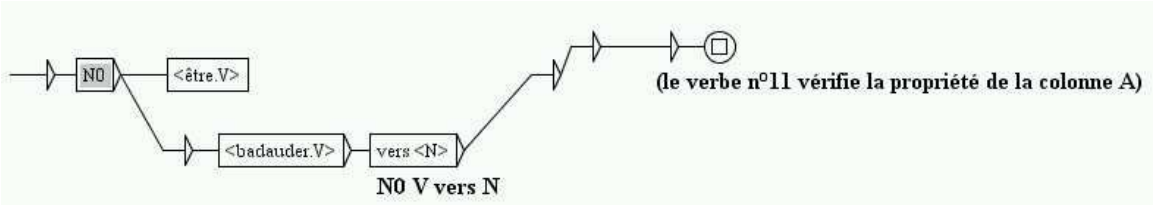



Fig. 8.9 - Grafo gerado para o verbo `badauder`
(o verbo n°11 verifica a propriedade da coluna A)

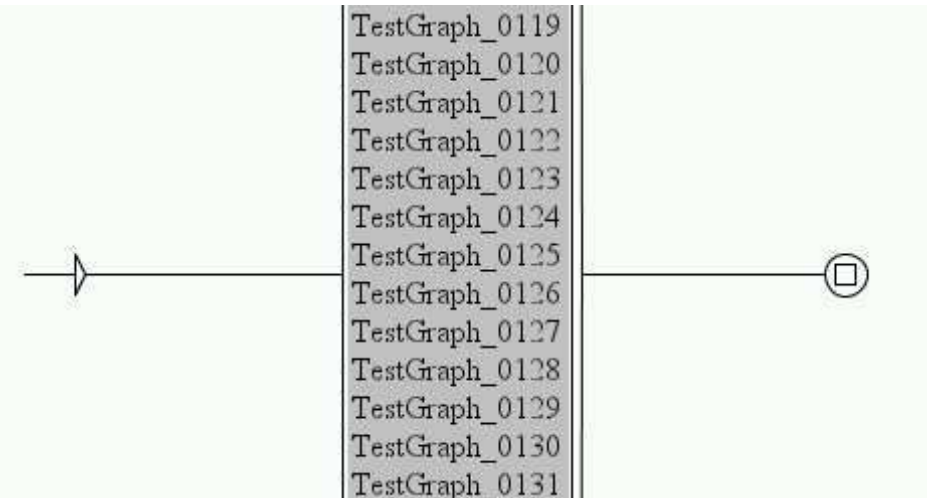


Fig. 8.10 - Grafo principal referente a todos os grafos gerados

**Capítulo 9**

# Utilização dos programas externos

Este capítulo apresenta a utilização dos diferentes programas que compõem o Unitex. Esses programas, que se encontram no diretório `Unitex/App`, são carregados automaticamente pela interface. Pode-se ver os comandos que foram executados clicando em "Console" no menu "Info". Pode-se igualmente ver as opções dos diferentes programas selecionando-os no submenu "Help on commands" do menu "Info".

IMPORTANTE: Muitos programas utilizam o diretório do texto (`mon_texte_snt`). Esse diretório é criado pela interface gráfica após a normalização do texto. Caso se trabalhe em linha de comando, deve-se criar esse diretório manualmente após a execução do programa `Normalize`.

IMPORTANTE (2): quando um parâmetro contiver espaços, deve-se colocá-lo entre aspas para que o mesmo não seja considerado como muitos parâmetros.

## 9.1 CheckDic

CheckDic dictionnaire type

Esse programa efetua a verificação do formato de um dicionário de tipo DELAS ou DELAF. O parâmetro `dictionnaire` corresponde ao nome do dicionário a ser verificado. O parâmetro `type` pode passar a ter o valor de `DELAS` ou `DELAF` caso se queira visualizar o dicionário em um ou outro formato.

O programa testa a sintaxe das linhas do dicionário. Ele verifica igualmente a lista dos caracteres presentes nas formas flexionadas e canônicas, a lista dos códigos gramaticais e sintáticos, assim como a lista dos códigos flexionais utilizados. Os resultados da verificação são armazenados em um arquivo chamado `CHECK_DIC.TXT`.

### 9.2 Compress

`Compress dictionnaire (-flip)`

Esse programa toma como parâmetro um dicionário DELAF e o compacta. A compactação de um dicionário `dico.dic` produz dois arquivos:

- `dico.bin`: arquivo binário contendo o autômato mínimo das formas flexionadas do dicionário;
- `dico.inf`: arquivo texto contendo formas compactadas que permitem reconstruir as linhas do dicionário a partir das formas flexionadas contidas no autômato.

Para mais detalhes sobre os formatos desses arquivos, ver o capítulo 10. O parâmetro opcional `-flip` indica que as formas flexionadas e canônicas serão invertidas no dicionário compactado. Esta opção é utilizada para construir o dicionário reverso necessário para o programa de `Reconstrução`.

### 9.3 Concord

```
Concord index font fontsize left right order mode alph (- thai)
```

Esse programa toma como parâmetro um arquivo de índice de concordância produzido pelo programa `Locate` e produz uma concordância. Ele pode igualmente produzir uma versão do texto modificado baseando-se nas transduções associadas às ocorrências. A seguir a descrição dos parâmetros:
- `index`: nome do arquivo de concordância. Deve-se indicar o caminho de acesso completo a esse arquivo, pois o Unitex se utiliza do mesmo para determinar sobre qual texto a concordância deve ser calculada.
- `font`: nome da fonte de caracteres a ser utilizada quando a concordância dever ser produzida em formato HTML. Caso a concordância não esteja em formato HTML, este parâmetro é ignorado.
- `fontsize`: tamanho da fonte se a concordância estiver no formato HTML. Como no parâmetro `font`, este é ignorado se a concordância não estiver no formato HTML.
- `left:` número de caracteres do contexto esquerdo das ocorrências. Em modo thaï, trata-se do número de caracteres não diacríticos.
- `right`: número de caracteres do contexto direito (não diacríticos, no caso do thaï). Caso a ocorrência tenha uma extensão inferior a este valor, a linha de concordância é completada para que o contexto direito tenha a mesma extensão de `right`. Se a ocorrência tiver uma extensão maior que os caracteres de `right`, mesmo assim será salva por inteiro.
- `order`: indica o modo de ordenação a ser utilizado para ordenar as linhas de concordância. Os valores possíveis são:
    - `TO`: ordem na qual as ocorrências aparecem no texto;
    - `LC`: contexto esquerdo, ocorrência;
    - `LR`: contexto esquerdo, contexto direito;
    - `CL`: ocorrência, contexto esquerdo;
    - `CR`: ocorrência, contexto direito;
    - `RL`: contexto direito, contexto esquerdo;
    - `RC`: contexto direito, ocorrência;
    - `NULL`: não especifica nenhuma seqüência de ordenação. Esta opção deve ser utilizada quando se deseja modificar o texto ao invés de construir uma concordância.

    Para mais detalhes sobre os modelos de ordenação, ver a seção 4.8.2.

- `mode`: indica sob qual formato a concordância deve ser produzida. Os quatro modos possíveis são:
    - `html`: produz uma concordância no formato HTML codificada em UTF-8;
    - `texte`: produz uma concordância no formato texto unicode;
    - `glossanet`: produz uma concordância para o GlossaNet no formato HTML. O arquivo HTML produzido é codificado em UTF-8;

- `nom_de_fichier`: indica ao programa que ele deve produzir uma versão modificada do texto e salvá-la em um arquivo nomeado `nom_de_fichier` (ver seção 6.7.3).
- `alph`: arquivo alfabético utilizado para a seleção. O valor NULL indica a ausência de arquivo de alfabeto.
- `-thai`: esse parâmetro é facultativo. Ele indica ao programa que ele manipula textos thaï. Essa opção é necessária para o bom funcionamento do programa em textos thaï.

O resultado da aplicação desse programa é um arquivo `concord.txt` se a concordância tiver sido construída em modo texto, um arquivo `concord.html` para os modos `html` e `glossanet`, e um arquivo texto cujo nome tenha sido definido pelo usuário, caso o programa tenha construído uma versão modificada do texto.

Em modo `html`, a ocorrência é codificada como um link. A referência associada a esse link é de formato (`a href = "x y z"`). x e y representam as posições de início e fim da ocorrência em caracteres no arquivo `nom_du_texte.snt`. `z` representa o número da frase na qual aparece a ocorrência.

### 9.4 ConcorDiff

```
ConcorDiff concor1 concor2 out font size
```

Esse programa utiliza-se de 2 arquivos de concordância para produzir uma página HTML, mostrando as diferenças entre essas duas concordâncias (ver seção 6.7.5, página 108). Os parâmetros são os seguintes:

- `concor1` e `concor2`: arquivos de concordância (`.ind`). Os nomes dos arquivos devem ser absolutos, pois o Unitex deduz daí o texto sobre o qual foram calculadas;
- `out`: página HTML de saída;
- `font`: fonte a ser utilizada na página HTML de saída;
- `size`: tamanho da fonte a ser utilizada na página HTML de saída.

### 9.5 Convert

```
Convert src [dest] mode text_1 [text_2, text_3, ...]
```

Esse programa permite modificar a codificação dos arquivos texto. O parâmetro `src` indica a codificação de entrada. O parâmetro opcional `dest` indica o código de saída. Por definição, a codificação de saída será `LITTLE-ENDIAN`. Os valores possíveis para esses parâmetros são:

```
FRENCH
ENGLISH
GREEK
THAI
CZECH
```

GERMAN
SPANISH
PORTUGUESE
ITALIAN
NORWEGIAN
LATIN (página de códigos latinos por definição)
windows-1252: página de códigos Microsoft Windows 1252 - Latim I (Europa Ocidental e EUA)
windows-1250: página de códigos Microsoft Windows 1250 – Europa Central
windows-1257: página de códigos Microsoft Windows 1257 – Báltico
windows-1251: página de códigos Microsoft Windows 1251 – Cirílico
windows-1254: página de códigos Microsoft Windows 1254 – Turquia
windows-1258: página de códigos Microsoft Windows 1258 – Vietnã
iso-8859-1: página de códigos ISO 8859-1 – Latim 1 (Europa Ocidental e EUA)
iso-8859-15: página de códigos ISO 8859-15 – Latim 9 (Europa Ocidental e EUA)
iso-8859-2: página de códigos ISO 8859-2 – Latim 2 (Europa Oriental e Central)
iso-8859-3: página de códigos ISO 8859-3 – Latim 3 (Europa do Sul)
iso-8859-4: página de códigos ISO 8859-4 – Latim 4 (Europa do Norte)
iso-8859-5: página de códigos ISO 8859-5 – Cirílico
iso-8859-7: página de códigos ISO 8859-7 – Grego
iso-8859-9: página de códigos ISO 8859-9 – Latim 5 (Turco)
iso-8859-10: página de códigos ISO 8859-10 – Latim 6 (Nórdico)
next-step: página de códigos NextStep
LITTLE-ENDIAN
BIG-ENDIAN

NOTA: existe um modo suplementar para o parâmetro dest com o valor UTF-8, que indica ao programa que ele deve converter os arquivos Unicode Little-Endian em arquivos UTF-8.

O parâmetro mode especifica como gerar os nomes dos arquivos fonte e destino. Os valores possíveis são:

- r: a conversão danifica os arquivos fonte
- ps=PFX: os arquivos fonte são renomeados com o prefixo PFX (toto.txt – PFXtoto.txt)
- pd=PFX: os arquivos destino são renomeados com o prefixo PFX
- ss=SFX: os arquivos fonte são renomeados com o sufixo SFX (toto.txt – totoSFX.txt)
- sd=SFX: os arquivos destino são renomeados com o sufixo SFX

Os parâmetros text_i são os nomes dos arquivos a serem convertidos.

# Capítulo 10

# Formatos de arquivos

Este capítulo apresenta os formatos de diferentes arquivos reconhecidos ou criados pelo Unitex. A formatação dos dicionários DELAS e DELAF já foram apresentadas nas seções 3.1.1 e 3.1.2.

OBS.: Neste capítulo, o símbolo ¶ representará quebra de linha. Salvo indicação contrária, todos os arquivos texto deste capítulo são codificados em Unicode Little-Endian.

## 10.1 Padrão Unicode Little-Endian

Todos os arquivos texto utilizados pelo Unitex devem estar em Unicode Little-Endian. Esse padrão permite representar 65536 caracteres, codificando um a cada dois octetos (bytes). No Little-Endian, os octetos estão na ordem peso fraco-peso forte. Quando essa ordem é invertida, fala-se em padrão Big-Endian. Um arquivo texto codificado em Unicode Little-Endian começa pelo caractere especial de valor hexadecimal `FEFF`.
As quebras de linha devem ser codificadas pelos dois caracteres `000D` e `000A`.

Consideremos o texto seguinte:

```
Unitex¶
ß-version¶
```

Veja a representação em Unicode Little-Endian desse texto:

| início | U | n | i | t | e | x | ¶ | ß |
|---|---|---|---|---|---|---|---|---|
| FFFE | 5500 | 6E00 | 6900 | 7400 | 6500 | 7800 | 0D000A00 | B203 |

| - | V | e | r | s | i | o | n | ¶ |
|---|---|---|---|---|---|---|---|---|
| 2D00 | 7600 | 6500 | 7200 | 7300 | 6900 | 6F00 | 6E00 | 0D000A00 |

Tab. 10.1 – Representação hexadecimal de um texto Unicode

Os octetos de peso forte e peso fraco foram invertidos, o que explica que o caractere inicial é codificado por `FFFE` ao invés de `FEFF`, o mesmo ocorre com `000D` e `000A`, que se transformam em `0D00` e `0A00`.

## 10.2 Arquivos de alfabeto

Há dois tipos de arquivos de alfabeto: um arquivo que define os caracteres de uma língua e um arquivo que indica as preferências pela ordenação. O primeiro é designado *alfabeto* e o segundo, *alfabeto de ordenação.*

### 10.2.1 Alfabeto

O arquivo de alfabeto é um arquivo texto que descreve todos os caracteres de uma língua, como as correspondências entre letras minúsculas e maiúsculas. Esse arquivo deve se chamar Alfabet.txt e deve se localizar na origem do diretório da língua concernida. Sua presença é obrigatória para que o Unitex possa funcionar.

Exemplo: o arquivo de alfabeto do inglês deve se localizar na pasta .../English/

Cada linha do arquivo alfabeto deve ter uma das três formas seguintes, seguidas de uma quebra de linha:

- #가힣 : uma libra seguida de dois caracteres *X* e *Y* indica que todos os caracteres compreendidos entre os caracteres *X* e *Y* são letras. Todos os caracteres são considerados como sendo maiúsculas e minúsculas ao mesmo tempo. Esse modo é útil para definir os alfabetos de línguas asiáticas, como o coreano, o chinês ou o japonês, nos quais não há distinção de quebra e nos quais o número de caracteres tornaria muito trabalhosa uma enumeração completa;
- Ëë: dois caracteres *X* e *Y* indicam que *X* e *Y* são letras e que *X* equivale à maiúscula da letra *Y*;
- 최: um único caractere *X* define *X* como uma letra, ao mesmo tempo maiúscula e minúscula. Esse modo é útil para definir um caractere asiático de maneira precisa.

Para algumas línguas como o francês, ocorre de muitas letras maiúsculas corresponderem a uma minúscula, como é o caso de é, que pode ter como maiúscula E ou É. Para expressar isso, basta utilizar muitas linhas. O inverso é igualmente válido: a uma maiúscula podem corresponder muitas minúsculas. Assim o E pode ser a maiúscula de e,é,è,ê,ë. Veja o fragmento do arquivo alfabeto do francês que define as diferentes letras e:

```
Ee¶
Eé¶
Éé¶
Eè¶
Èè¶
Eê¶
Êê¶
Eë¶
Ëë¶
```

### 10.2.2 Alfabeto de ordenação

O alfabeto de ordenação é um arquivo texto que define as prioridades de letras de uma língua no momento da ordenação, graças ao programa Sort.txt. Cada linha desse arquivo define um conjunto de letras. Se um conjunto de letras *A* é definido antes de um conjunto de letras *B*, não importa que a letra de *A* seja inferior a qualquer letra de *B*.

As letras de um mesmo conjunto só serão distintas se necessário. Por exemplo, caso seja definido o conjunto de letra eéèêë, a palavra ébahi será considerada menor que estuaire, ela própria menor que *été*. Como as letras que precedem e e é permitiam classificar as palavras, não se buscou comparar as letras e e é, pois elas pertencem ao mesmo grupo.

Em contrapartida, se compararmos as palavras *chantés* e *chantes*, *chantes* será considerada a menor. É necessário, pois, comparar as letras e e é para distinguir essas palavras. Como a letra e aparece primeiro no conjunto eéèêë, ela é considerada inferior a é. A palavra *chantes* será, portanto, considerada menor que a palavra *chantés*.

O arquivo de alfabeto de ordenação permite definir equivalências de caracteres. Pode-se, assim, ignorar as diferenças de quebra e de acento. Por exemplo, caso se queira ordenar as letras b, c e d sem se levar em conta a quebra nem a cedilha, pode-se escrever as linhas seguintes:

```
Bb¶
CcÇç¶
Dd¶
```

Esse arquivo é facultativo. Quando nenhum alfabeto de ordenação é especificado no programa SortTxt, este efetua uma ordenação na ordem em que aparecem os caracteres no padrão Unicode.

## 10.3 Grafos

Esta seção apresenta os dois formatos de grafos: o formato gráfico .grf e o formato compilado .fst2.

### 10.3.1 Formato .grf

Um arquivo .grf é um arquivo texto que contém informações de apresentação, além de informações que representam os conteúdos de caixas e as transições do grafo. Um arquivo .grf começa pelas linhas seguintes:

```
#Unigraph¶
SIZE 1313 950¶
FONT Times New Roman: 12¶
OFONT Times New Roman: B 12¶
BCOLOR 16777215¶
FCOLOR 0¶
ACOLOR 12632256¶
SCOLOR 16711680¶
CCOLOR 255¶
DBOXES y¶
DFRAME y¶
DDATE y¶
DFILE y¶
DDIR y¶
DRIG n¶
DRST n¶
```

```
FITS 100¶
PORIENT L¶
#¶
```

A primeira linha `#Unigraph¶` é uma linha de comentário. As linhas seguintes definem os valores de parâmetros de apresentação do grafo:

- `SIZE x y`: define a largura x e a altura y do grafo em pixels;
- `FONT name: xyz`: define a fonte utilizada para exibir o conteúdo das caixas. name representa o nome da fonte. `x` indica se a fonte deve ser em negrito ou não. Se `x` equivale a `B`, isso representa que a fonte deve ser em negrito. Para uma fonte normal, x deve ser um espaço. Da mesma maneira, `y` equivale a `I` se a fonte tiver que ser em itálico, senão apenas um espaço. `z` representa o tamanho da fonte;
- `OFONT name: xyz`: define a fonte utilizada para exibir transduções. Os parâmetros `name, x, y e z` são definidos da mesma forma que para `FONT;`
- `BCOLOR x`: define a cor do segundo plano do grafo. `x` representa a cor em formato RGB;
- `FCOLOR x`: define a cor do traço do grafo. `x` representa a cor em formato RGB;
- `ACOLOR x`: define a cor utilizada para traçar as linhas de caixas que correspondem a chamadas aos sub-grafos. `x` representa a cor em formato RGB;
- `SCOLOR x`: define a cor utilizada para traçar o conteúdo de caixas de comentários (ou seja, as caixas que não estão ligadas a nenhuma outra). `x` representa a cor em formato RGB;
- `CCOLOR x`: define a cor utilizada para traçar as caixas selecionadas. `x` representa a cor em formato RGB;
- `DBOXES x`: esta linha é ignorada pelo Unitex. Ela é conservada pela preocupação com a compatibilidade com os grafos Intex;
- `DFRAME x`: traça ou não um quadro em torno do grafo, segundo o qual `x` equivale a `y` ou `n`;
- `DDATE x`: exibe ou não o dado embaixo do grafo, segundo o qual `x` equivale a `y` ou `n`;
- `DFILE x`: exibe ou não o nome do arquivo embaixo do grafo, segundo o qual `x` equivale a `y` ou `n`;
- `DDIR x`: exibe ou não o caminho completo de acesso ao arquivo embaixo do grafo segundo o qual `x` equivale a `y` ou `n`. Essa opção só é levada em conta se o parâmetro `DFILE` tem o valor de `y`;
- `DRIG x`: traça o grafo da direita para a esquerda ou da esquerda para a direita segundo o qual `x` equivale a `y` ou `n`;
- `DRST x`: esta linha é ignorada pelo Unitex. Ela é conservada pela preocupação com a compatibilidade com os grafos Intex;
- `FITS x`: esta linha é ignorada pelo Unitex. Ela é conservada pela preocupação com a compatibilidade com os grafos Intex;
- `PORIENT x`: esta linha é ignorada pelo Unitex. Ela é conservada pela preocupação com a compatibilidade com os grafos Intex;
- `#`: esta linha é ignorada pelo Unitex. Ela serve para indicar o fim de informações iniciais.

As linhas seguintes dão o conteúdo e a posição de caixas do grafo. As linhas seguintes correspondem a um grafo que reconhece um algarismo:

```
3¶
"<E>" 84 248 1 2 ¶
"" 272 248 0 ¶
s"1+2+3+4+5+6+7+8+9+0" 172 248 1 1 ¶
```

A primeira linha indica o número de caixas do grafo, imediatamente seguido de uma quebra de linha. Esse número jamais deve ser inferior a 2, pois é sensato que um grafo deva sempre possuir um estado inicial e um estado final.

As linhas seguintes definem as caixas do grafo. As caixas são numeradas a partir de 0. Por convenção, o estado 0 é o estado inicial e o estado 1 é o estado final. O conteúdo do estado final deve ser sempre vazio.

Cada caixa do grafo é definida por uma linha que deve ter o seguinte formato

*contenu X Y N transitions* ¶

*contenu* é uma cadeia de caracteres entre aspas que representa o conteúdo da caixa. Essa cadeia pode eventualmente ser precedida de um s no caso de um grafo Intex importado, assim esse caractere é ignorado pelo Unitex. O conteúdo da cadeia é o texto que foi examinado no controle de texto do editor de grafos. A tabela seguinte dá a codificação de duas seqüências especiais que não são codificadas do mesmo modo nos arquivos `.grf`:

| Seqüência no editor de grafo | Seqüência no arquivo `.grf` |
|---|---|
| " | \" |
| \" | \\\" |

Tab. 10.2 – Codificação de seqüências especiais

OBS.: os caracteres compreendidos entre `<` e `>` e `{` e `}` não são interpretados. Assim, o caractere + conteúdo na cadeia "`le<A+Conc>`" não é interpretado como um divisor de linhas, pois o padrão `<A+Conc>` é interpretado prioritariamente.

*X* e *Y* representam as coordenadas da caixa em pixels. A figura 10.1 mostra como essas coordenadas são interpretadas pelo Unitex.

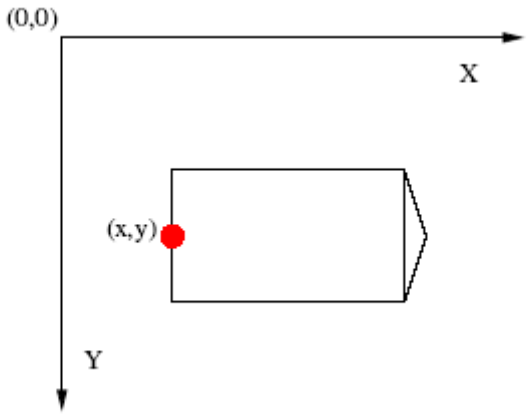



Fig. 10.1 – Interpretação de coordenadas de caixas

*N* representa o número de transições que saem da caixa. Esse número sempre deve valer 0 para o estado final.

As transições são definidas pelos números de caixas em torno das quais são mostradas.

Cada linha de definição de caixa deve terminar com um espaço seguido de uma quebra de linha.

### 10.3.2 Formato .fst2

Um arquivo `.fst2` é um arquivo texto que descreve um conjunto de grafos. Veja um exemplo de arquivo `.fst2`.

```
0000000002¶
-1 GN¶
: 1 1 ¶
: 2 2 -2 2 ¶
: 3 3 ¶
t ¶
f ¶
-2 Adj¶
: 6 1 5 1 4 1 ¶
t ¶
f ¶
%<E>¶
%le/DET¶
%<A>/ADJ¶
%<N>¶
%beau¶
@joli¶
%petit¶
f¶
```

A primeira linha representa o número de grafos codificados no arquivo. O início de cada grafo é identificado por uma linha que indica o número e o nome do grafo (`-1 GN e -2 Adj` no arquivo acima).

As linhas seguintes descrevem os estados de grafo. Se o estado é terminal, a linha é iniciada pelo caractere `t`, senão, pelo caractere: `sinon` . Para cada estado, a lista de transição é uma seqüência eventualmente vazia de binário de forças de inteiros:

- o primeiro inteiro indica o número de unidades ou de sub-grafo correspondente à transição. As unidades são numeradas a partir de 0. Os sub-grafos são representados por inteiros negativos, o que explica os números que precedem os nomes dos grafos serem negativos.
- o segundo inteiro representa o número de estado de chegada da transição. Em cada grafo, os estados são numerados a partir de 0. Por convenção, o estado 0 de um grafo é seu estado inicial.

Cada linha de definição de estado deve terminar com um espaço. O final de cada grafo é marcado por uma linha que contenha um f seguido de um espaço.

As unidades são definidas após o último grafo. Se a linha começa pelo caractere @, significa que o conteúdo da unidade deve ser procurado sem variante de quebra. Essa informação só é útil quando a unidade é uma palavra. Se a linha começa pelo caractere %, as variantes de quebra são autorizadas. Se uma unidade porta transdução, as seqüências de entrada e de saída são separadas pelo caractere / (exemplo: `le/DET`). Por convenção, a primeira unidade sempre deve ser a palavra vazia (`<E>`), mesmo se esta unidade não é utilizada em nenhuma transição.

O final do arquivo é indicado por uma linha contendo o caractere f seguido de uma quebra de linha.

## 10.4 Textos

Esta seção apresenta os diferentes arquivos utilizados para representar os textos.

### 10.4.1 Arquivos .txt

Os arquivos .txt devem ser arquivos texto codificados em Unicode Little-Endian. Esses arquivos não devem conter chave aberta ou fechada, a menos que sejam utilizadas para traçar um divisor de frase (`{S}`) ou uma unidade lexical válida (`{aujourd'hui, .ADV}`). As quebras de linha devem ser codificados pelos dois caracteres especiais de valores hexadecimais `000D e 000A`.

### 10.4.2 Arquivos .snt

Os arquivos .snt são os arquivos texto que foram pré-processados pelo Unitex. Esses arquivos não devem conter tabulação, espaços ou quebras de linha consecutivos. As

únicas chaves autorizadas nos arquivos `.snt` são as do divisor de frases `{S}` e as das unidades lexicais `({aujourd'hui, .ADV})`.

### 10.4.3 Arquivo text.cod

O arquivo text.cod é um arquivo binário que contém uma seqüência de inteiros que representam o texto. Cada inteiro *i* reenvia ao token de índice *i* no arquivo `tokens.txt`. Esses inteiros são codificados sobre 4 octetos.

OBS.: os tokens são numerados a partir de 0.

#### 10.4.4 Arquivo tokens.txt

O arquivo `tokens.txt` é um arquivo texto que contém a lista de todas as unidades lexicais do texto. A primeira linha desse arquivo indica o número de unidades contidas no arquivo. As unidades são divididas por quebras de linha. Quando uma seqüência é encontrada em um texto com variantes de quebra, cada variante é codificada por uma unidade distinta.

OBS.: as quebras de linha eventualmente presentes no arquivo `.snt` são codificados como espaços. Nunca há, então, unidades que codifiquem a quebra de linha.

#### 10.4.5 Arquivos tok_by_alph.txt e tok_by_freq.txt

Estes dois arquivos são arquivos de texto que contêm a lista de unidades lexicais selecionada por ordem alfabética ou por ordem de freqüência.

No arquivo tok_by_alph.txt, cada linha é composta de uma unidade, seguida do caractere tabulação e do número de ocorrências dessa unidade o texto.

As linhas do arquivo `tok_by_freq.txt` são formadas sob o mesmo princípio, mas o número de ocorrências aparece antes do caractere tabulação e da unidade.

### 10.4.6 Arquivo enter.pos

Este é um arquivo binário que contém a lista de posições de quebras de linha no arquivo `.snt`. Cada posição é o índice no arquivo text.cod de uma quebra de linha, substituído por um espaço. Essas posições são de inteiros codificados sobre 4 octetos.

## 10.5 Autômato do texto

### 10.5.1 Arquivo text.fst2

O arquivo `text.fst2` é um arquivo `.fst2` especial que representa um autômato de frase. Nesse arquivo, cada sub-grafo representa um autômato de frase. As localizações reservadas aos nomes de sub-grafos são utilizadas para armazenar as frases a partir das quais foram construídos os autômatos de frases.

Com exceção da primeira unidade, que deve ser sempre epsilon (<E>), as unidades devem ser ou unidades lexicais ou entradas de DELAF enquadradas pelas chaves.

Exemplo: Veja o arquivo correspondente ao texto *Il mange une pomme de terre*.

```
0000000001¶
-1 Il mange une pomme de terre. ¶
: 1 1 ¶
: 2 2 ¶
: 3 3 4 3 ¶
: 5 4 6 4 7 4 ¶
: 8 5 9 5 10 5 ¶
: 11 6 12 6 ¶
: 13 7 ¶
t ¶
f ¶
%<E>¶
%{Il,il.PRO+z1:3ms}¶
%{mange,manger.V+z1:P1s:P3s:S1s:S3s:Y2s}¶
%{une,une.N+z1:fs}¶
%{une,un.DET+z1:fs}¶
%{pomme,pomme.A+z1:ms:fs:mp:fp}¶
%{pomme,pomme.N+z1:fs}¶
%{pomme,pommer.V+z3:P1s:P3s:S1s:S3s:Y2s}¶
%{de,de.DET+z1}¶
%{de,de.PREP+z1}¶
%{terre,terre.N+z1:fs}¶
%{terre,terrer.V+z1:P1s:P3s:S1s:S3s:Y2s}¶
%.¶
f¶
```

### 10.5.2 Arquivo cursentence.grf

O arquivo `cursentence.grf` é gerado pelo Unitex no momento da visualização de um arquivo autômato de uma frase a partir do arquivo `text.fst2`.

### 10.5.3 Arquivo sentenceN.grf

Quando o usuário modifica o autômato de uma frase, esse autômato é copiado com o nome de sentence.grf, no qual N representa o número da frase.

### 10.5.4 Arquivo cursentence.txt

No momento da extração do autômato de frase, o texto da frase é copiado no arquivo texto cursentence.txt. Esse arquivo é utilizado pelo Unitex para exibir o texto da frase acima do autômato. Esse arquivo contém o texto da frase, seguido de uma quebra de linha.

## 10.6 Concordâncias

### 10.6.1 Arquivo concord.ind

O arquivo `concord.ind` é o índice de ocorrências encontradas pelo programa Locate no momento da aplicação de uma gramática. É um arquivo texto que contém as posições de início e fim de cada ocorrência, eventualmente acompanhadas de uma cadeia de caracteres se a concordância foi obtida levando-se em conta as eventuais transduções da gramática. Veja um exemplo de arquivo:

```
#M{
3036 3040 le[ADJ= petit] salon¶
3071 3075 Le nouveau domestique¶
5600 5604 le jeune Lord¶
6052 6056 le second étage¶
6123 6127 le premier étage¶
6181 6185 le même instant¶
6461 6465 le méthodique gentleman¶
7468 7472 le grand salon¶
7520 7524 le laborieux dépliage¶
7675 7679 le grand salon¶
8590 8594 le fait plus¶
10990 10994 le mauvais temps¶
13719 13723 le brave garçon¶
13896 13900 le modeste sac¶
15063 15067 le même compartiment¶
```

A primeira linha indica em qual modo ou transdução a concordância foi calculada. Os três valores possíveis são:

- #I: as transduções foram ignoradas;
- #M: as transduções foram inseridas nas seqüências reconhecidas (modo MERGE);
- #R: as transduções substituíram as seqüências reconhecidas (modo REPLACE).

Cada ocorrência é descrita por uma linha. As linhas começam pelas posições de início e de fim da ocorrência. As posições são dadas em unidades lexicais.

Se o arquivo comporta a linha inicial #I, a posição final de cada ocorrência é imediatamente seguida de uma quebra de linha. Caso contrário, ela é seguida de um espaço e de uma cadeia de caracteres. Em modo REPLACE, essa cadeia corresponde à transdução produzida pela seqüência reconhecida naquela em que foram inseridas as transduções. Em modo MERGE ou REPLACE, é essa cadeia que é exibida na concordância. Se as transduções foram ignoradas, o conteúdo da ocorrência é extraído do arquivo texto.

### 10.6.2 Arquivo concord.txt

O arquivo concord.txt é um arquivo texto que representa uma concordância. Cada ocorrência é codificada por uma linha composta de 3 cadeias de caracteres separados pelo caractere de tabulação e que representa o contexto esquerdo, a ocorrência (eventualmente modificada por transduções) e o contexto direito.

### 10.6.3 Arquivo Concord.html

O arquivo `concord.html` é um arquivo `.html` que representa uma concordância. Esse arquivo é codificado em UTF-8.

O título da página é o número de ocorrências que ela descreve. As linhas das concordâncias são codificadas por linhas em que as ocorrências são consideradas como links de hipertexto. A referência associada a cada um desses links é da forma `<a href= "X Y Z">`. `X` e `Y` representam a posição de início e fim da ocorrência em caracteres no arquivo `nom_du_texte.snt`. `Z` representa o número da frase na qual aparece cada ocorrência.

Todos os espaços são codificados como espaços indivisíveis (& nbsp; em HTML), o que permite conservar o alinhamento das ocorrências mesmo se uma delas, encontrando-se no começo do arquivo, tenha um contexto esquerdo completado com espaços.

Nota: no caso de uma concordância construída com o parâmetro `glossanet`, o arquivo HTML obtém a mesma estrutura, exceto no que se refere aos links. Nessas concordâncias, as ocorrências são links reais reconduzindo de volta ao servidor da *web* da aplicação GlossaNet. Para mais informações sobre GlossaNet, consultar os links sobre o *website* de Unitex (http://www-igm.univ-mlv.fr/~unitex) .

Veja um exemplo de arquivo:

```
<html lang=en>¶
<head>¶
</head>¶
<body>¶
<font face="Courier new" size=3>¶
MAÃŐTRE, L'<a href="104 109 2">AUTRE</a> COMM<br>¶
TRE COMME <a href="116 126 2">DOMESTIQUE</a><br>¶
- , Ãl'tait <a href="270 277 3">habitÃl'e</a> pa<br>¶
'UN COMME <a href="94 100 2">MAÃŐTRE</a>, L'<br>¶
un de les <a href="314 321 3">membres</a> le<br>¶
la maison <a href="158 165 3">portant</a> le<br>¶
</font>¶
</body>¶
</html>¶
```

A figura 10.2 mostra a página correspondente ao arquivo acima.


Fig. 10.2 – Exemplo de concordância.

### 10.6.4 O arquivo diff.html

O arquivo `diff.htlm` é uma página HTML que mostra as diferenças entre duas concordâncias. Esse arquivo está em código UTF-8. Veja um exemplo de arquivo (quebras de linha foram introduzidas pela diagramação).

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html;
         charset=UTF-8">
   <style type="text/css">
   a.blue {color:blue; text-decoration:underline;}
   a.red {color:red; text-decoration:underline;}
   a.green {color:green; text-decoration:underline;}
   </style>
</head>
<body>                                                    ces<br>
<font color="green">Green:</font> sequences that occur in only
one of the two concordances<br>
<table border="1" cellpadding="0" style="font-family: Courier new;
  font-size: 12">
```

## 10.7 Dicionários

A compreensão dos dicionários DELAF pelo programa Compress produz dois arquivos: um arquivo `.bin` que representa o autômato mínimo das formas flexionadas do dicionário, e um arquivo `.inf` que contém as formas compactadas permitindo reconstruir as linhas do dicionário a partir das formas flexionadas. Essa seção descreve o formato de seus dois tipos de arquivos, como o formato do arquivo `CHECK_DIC.TXT` que contém o resultado da verificação de um dicionário.

### 10.7.1 Arquivos .bin

Um arquivo `.bin` é um arquivo binário representando um autômato. Os 4 primeiros octetos do arquivo representam um inteiro mostrando o tamanho do arquivo em octetos. Os estados dos autômatos são em seguida codificados da seguinte maneira:
- os 2 primeiros octetos indicam se o estado é terminal assim como o número de transições que saem daí. O bit mais forte vale 0 se o estado é terminal e 1 se não for. Os outros 15 bits codificam o número de transições.

Exemplo: um estado não terminal, com 17 transições, é codificado pela seqüência hexadecimal `8011`.

- se o estado é terminal, os 3 octetos seguintes codificam o índice no arquivo `.inf` da forma compactada, a ser utilizada para reconstruir as linhas do dicionário para essa forma de arquivo.

Exemplo: se o estado reconduzir a forma compactada do índice `25133`, a seqüência hexadecimal correspondente é `00622D`.

- cada transição que sai é em seguida codificada em 5 octetos. Os 2 primeiros octetos codificam o caractere que etiqueta a transição, e os 3 seguintes codificam as posições em octetos de um arquivo `.bin` do estado de chegada. As transições de um estado são codificadas umas após as outras.

Exemplo: uma transição etiquetada pelo caractere A apontando para o estado cuja descrição inicial é o octeto `50106` será representada pela seqüência hexadecimal `004100C3BA`.

Por convenção, o primeiro estado do autômato é o estado inicial.

### 10.7.2 Arquivo .inf

Um arquivo `.inf` é um a arquivo de texto descrevendo as formas compactadas associadas a um arquivo `.bin`. Veja um exemplo de arquivo `.inf`:

```
0000000006¶
_10\0\0\7.N¶
.PREP¶
_3.PREP¶
.PREP,_3.PREP¶
1-1.N+Hum:mp¶
3er 1.N+AN+Hum:fs¶
```

A primeira linha do arquivo indica o número de formas compactadas que ele contém. Cada linha pode conter uma ou mais formas compactadas. Se houver várias formas, essas devem ser separadas por uma vírgula. Cada forma compactada é formada de uma seqüência, permitindo encontrar uma forma canônica a partir de uma forma flexionada, seguida pela seqüência de códigos gramaticais, semânticos e flexionais associados à entrada.

O modo de compactação da forma canônica varia em função da forma flexionada. Se as duas formas forem exatamente idênticas, a forma compactada resume-se às informações gramaticais, semânticas e flexionais, como é o caso da linha seguinte:

```
.N+Hum:ms
```

Se as formas forem diferentes, o programa de compactação segmenta as duas formas em unidades. Essas unidades podem ser ou um espaço, ou um travessão, ou uma seqüência de caracteres que não contém nem espaço nem travessão. Esse modo de segmentação permite levar eficazmente em conta a flexão de palavras compostas.

Se as formas flexionadas e canônicas não comportam o mesmo número de unidades, o programa codifica a forma canônica pelo número de caracteres da forma flexionada que devem ser segmentados, depois pelos caracteres a serem adicionados. Assim, a primeira linha do arquivo abaixo corresponde à linha de dicionário:

```
James Bond,007.N
```

Como a seqüência `James Bond` contém três unidades e `007` somente uma, a forma canônica é codificada por `_0\0\7`. O caractere _ indica que as duas formas não têm o mesmo número de unidades. O número que segue (aqui 10) indica o número de caracteres segmentados. A seqüência \0\0\7 que segue esse número indica que devemos em seguida acrescentar a seqüência 007. Os dígitos são antecedidos de caracteres \ para não serem confundidos com o número de caracteres a serem segmentados.

Quando as duas formas têm o mesmo número de unidades, as unidades são compactadas duas a duas. Se as duas unidades são compostas de um espaço ou de um travessão, a forma compactada da unidade é a própria unidade, como é o caso na linha seguinte:

```
1-1.N+Hum:mp
```

Isso permite conservar certa visibilidade no arquivo `.inf` quando o dicionário contém palavras compostas.

Quando pelo menos uma das unidades não é nem um espaço nem um travessão, a forma compactada é composta pelo número de caracteres a serem segmentados, seguido da seqüência de caracteres a serem adicionados. Assim, a linha de dicionário:

```
première partie,premier parti.N+AN+Hum:fs
```

é codificada pela linha:

```
3er 1.N+AN+Hum:fs
```

O código 3er indica que devemos separar 3 caracteres da seqüência `première` e acrescentar-lhe os caracteres `er` para obter `premier`. O `1` indica que devemos simplesmente retirar um caractere de `partie` para obter a seqüência `parti`. O número 0 é utilizado quando queremos identificar que não devemos suprimir nenhum caractere.

### 10.7.3 Arquivo CHECK_DIC.TXT

Esse arquivo é produzido pelo programa de verificação do dicionário `Check_Dic`. Trata-se de um arquivo de texto que dá informações sobre o dicionário analisado e se decompõe em 4 partes.

A primeira parte dá a lista, eventualmente vazia, de todos os erros de sintaxe encontrados no dicionário: ausência da forma flexionada ou da forma canônica, ausência do código gramatical, linha em branco, etc. Cada erro é descrito pelo número da linha referida, com uma mensagem descrevendo a natureza do erro, e também o conteúdo da linha. Veja um exemplo de mensagem:

```
Line 12451: no point found
jardin,N:ms
```

A segunda e a terceira partes dão, respectivamente, a lista de códigos gramaticais e/ou semânticos e flexionais. A fim de prevenir erros de codificação, o programa assinala os códigos que contêm espaços, tabulações ou caracteres não ASCIL. Assim, se um dicionário grego contiver o código ADV onde o caractere A é o A grego em vez do A latino, o programa dará o seguinte aviso:

```
ADV warning: 1 suspect char (1 non ASCII char): (0391 D V)
```

Os caracteres não ASCIL são indicados por seus números de caractere hexadecimal. No exemplo abaixo, o código 0391 representa o A grego. Os espaços são indicados pela seqüência SPACE:

```
Km s warning: 1 suspect char (1 space): (K m SPACE s)
```

Quando verificamos o dicionário seguinte:

```
1,2 et 3!,.INTJ
abracadrabra,INTJ
saperlipopette,.INTJ
zut,. INTJ
```

Obtemos o arquivo `CHECK_DIC.TXT` seguinte:

```
------------------------------------¶
----  All chars used in forms  ----¶
------------------------------------¶
  (0020)¶
! (0021)¶
, (002C)¶
1 (0031)¶
2 (0032)¶
3 (0033)¶
I (0049)¶
J (004A)¶
N (004E)¶
T (0054)¶
a (0061)¶
e (0065)¶
h (0068)¶
p (0070)¶
r (0072)¶


  Line 1: unprotected comma in lemma¶
  1,2 et 3!,.INTJ¶
  Line 2: no point found¶
  ah,INTJ¶
```

```
s (0073)¶
t (0074)¶
u (0075)¶
z (007A)¶
-------------------------------------------------------------¶
----    2 grammatical/semantic codes used in dictionary  ----¶
-------------------------------------------------------------¶
INTJ¶
 INTJ warning: 1 suspect char (1 space): (SPACE I N T J)¶
-----------------------------------------------------¶
----    0 inflectional code used in dictionary  -----¶
-----------------------------------------------------¶
```

## 10.8 Arquivo do ELAG

### 10.8.1 Arquivo Tagset.def.

Ver seção 7.3.6 na pagina 127

### 10.8.2 Arquivo .Lst

OS ARQUIVOS .LST NÃO SÃO CODIFICADOS EM UNICODE.

O arquivo `.lst` contém uma lista de nomes de arquivo `.grf`, localizados em relação à pasta ELAG da língua usual. A seguir o arquivo `elag.lst` fornecido para o francês:

```
PPVs/PpvIL.grf¶
PPVs/PpvLE.grf¶
PPVs/PpvLUI.grf¶
PPVs/PpvPR.grf¶
PPVs/PpvSeq.grf¶
PPVs/SE.grf¶
PPVs/postpos.grf¶
```

### 10.8.3 Arquivo .elg

O arquivo `.elg` contem as regras ELAG compiladas. Esses arquivos estão no formato `.fst2`.

### 10.8.4 Arquivo .rul

OS ARQUIVOS .RUL NÃO SÃO CODIFICADOS EM UNICODE.

Esses arquivos listam os diferentes arquivos .elg que compõem um conjunto de regras ELAG. Um arquivo `.rul` é constituído de tantas partes quantos forem os arquivos .elg. Cada parte é composta pela lista de gramática ELAG que corresponde a um arquivo .elg, onde cada nome de arquivo é precedido por uma tabulação seguido por uma linha contendo o nome do arquivo `.elg` entre ângulos. As linhas começam por uma tabulação tendo o valor de comentário e são ignoradas pelo programa `Elag`. Segue o arquivo `elag.rul` fornecido por default para o francês:

```
    PPVs/PpvIL.elg¶
    PPVs/PpvLE.elg¶
    PPVs/PpvLUI.elg¶
<elag.rul-0.elg>¶
    PPVs/PpvPR.elg¶
    PPVs/PpvSeq.elg¶
    PPVs/SE.elg¶
    PPVs/postpos.elg¶
<elag.rul-1.elg>¶
```

## 10.9 Arquivos de Configurações

### 10.9.1 Arquivo Config

Quando o usuário modifica suas preferências por uma língua determinada, essas são salvas em um arquivo de texto nomeado `Config` que se encontra na pasta da língua usual. Esse arquivo tem a seguinte sintaxe (a ordem das linhas pode variar):

```
#Unitex configuration file of 'paumier' for 'English'¶
#Tue Jan 31 11:21:32 CET 2006¶
TEXT\ FONT\ NAME=Courier New¶
TEXT\ FONT\ STYLE=0¶
TEXT\ FONT\ SIZE=10¶
CONCORDANCE\ FONT\ NAME=Courier new¶
CONCORDANCE\ FONT\ HTML\ SIZE=12¶
INPUT\ FONT\ NAME=Times New Roman¶
INPUT\ FONT\ STYLE=0¶
INPUT\ FONT\ SIZE=10¶
OUTPUT\ FONT\ NAME=Arial Unicode MS¶
OUTPUT\ FONT\ STYLE=1¶
OUTPUT\ FONT\ SIZE=12¶
DATE=true¶
FILE\ NAME=true¶
PATH\ NAME=false¶
FRAME=true¶
RIGHT\ TO\ LEFT=false¶
BACKGROUND\ COLOR=-1¶
FOREGROUND\ COLOR=-16777216¶
AUXILIARY\ NODES\ COLOR=-3289651¶
COMMENT\ NODES\ COLOR=-65536¶
```

As duas primeiras linhas são linhas de comentário. As três linhas seguintes indicam o nome, o estilo e o corpo da fonte utilizada para arquivar os textos, os dicionários, as unidades lexicais, as frases do autômato do texto, etc.

O parâmetro `CONCORDANCE FONT NAME` e `CONCORDANCE FONT HTML SIZE` definem o nome e o corpo da fonte utilizada para arquivar as concordâncias em HTML, e deve ser compactada entre 1 e 7.

Os parâmetros `INPUT FONT`...e `OUTPUT FONT`... definem o nome, o estilo e o corpo das fontes utilizadas para arquivar as atribuições e as transcrições gráficas.

Os 10 parâmetros seguintes correspondem aos parâmetros precisos nos cabeçalhos dos gráficos. A tabela 10.3 descreve essas correspondências.

(Parâmetros no arquivo `config`) (Parâmetros no arquivo `.grf`)

| Paramètres dans le fichier `Config` | Paramètres dans un fichier `.grf` |
|---|---|
| DATE | DDATE |
| FILE NAME | DFILE |
| PATH NAME | DDIR |
| FRAME | DFRAME |
| RIGHT TO LEFT | DRIG |
| BACKGROUND COLOR | BCOLOR |
| FOREGROUND COLOR | FCOLOR |
| AUXILIARY NODES COLOR | ACOLOR |
| COMMENT NODES COLOR | SCOLOR |
| SELECTED NODES COLOR | CCOLOR |

Tab.10.3 – Significação de parâmetros.

O parâmetro `PACKAGE NODES` define a cor das chamadas a sub-gráficos da área de armazenamento.

O parâmetro `CONTEXT NODES` define a cor das caixas correspondentes a inícios ou fins de contextos.

O parâmetro `CHABY CHAR` indica se a língua usual deve ser tratada caractere a caractere ou não.

O parâmetro `ANTIALIASING` indica se o gráfico e também os autômatos da frase devem ser mostrados por default com o efeito de antialiasing.

O parâmetro `HTML VIEWER` indica o nome do navegador utilizado para mostrar as concordâncias. Se nenhum nome de navegador for estabelecido, as concordâncias serão mostradas em uma janela Unitex.

O parâmetro `MAXTEXT FILE SIZE` define o tamanho máximo dos arquivos de textos que o Unitex abre na interface gráfica. Se um arquivo tiver um número superior a esse limite, o usuário verá a seguinte mensagem: *" This file is too large to be displayed. Use a wordprocessor to view it"*. O valor por default é 2048ko.

O parâmetro `ICON BAR POSITION` define a posição da barra de ícones nas janelas de gráficos.

O parâmetro `PACKAGE PATH` define a área de armazenagem a ser utilizada para essa língua.

### 10.9.2 Arquivo System_dic.def

O arquivo `system_dic.def` é um arquivo de texto que descreve a lista dos dicionários do sistema a serem aplicados por default. Esse arquivo encontra-se na pasta da língua usual. Cada linha corresponde a um nome de arquivo `.bin`. Os dicionários do sistema devem encontrar-se na pasta do sistema, no interior da sub-pasta `(língua usual)/DELA`. A seguir um exemplo de arquivo:

```
delacf.bin¶
delaf.bin¶
```

### 10.9.3 Arquivo user_dic.def

O arquivo `user_dic.def` é um arquivo de texto que descreve a lista dos dicionários do usuário a serem aplicados por default. Esse arquivo encontra-se na pasta da língua usual e tem o mesmo formato que o arquivo `system_dic.def`. Os dicionários do usuário devem encontrar-se na sub-pasta `(língua usual)/DELA` da pasta pessoal do usuário.

### 10.9.4 Arquivo user.cfg

No Linux, o Unitex considera que a pasta pessoal do usuário tem o nome de unitex e que ela encontra-se em seu diretório raiz ($Home). No Windows, não é sempre possível associar uma pasta por default a um usuário. Para remediar isso, a Unitex criou para cada usuário um arquivo .cfg contendo o caminho de sua pasta pessoal. Esse arquivo é salvo com o nome `(login do usuário).cfg` na sub-pasta do sistema `Unitex/Users`.

ATENÇÃO: ESSE ARQUIVO NÃO ESTÁ EM UNICODE E O CAMINHO DA PASTA PESSOAL NÃO É SEGUIDO POR UMA QUEBRA DE LINHA.

## 10.10 Arquivos variados

Para cada texto, Unitex criou vários arquivos contendo as informações para serem mostradas na interface gráfica. Esta seção descreve esses diferentes arquivos.

### 10.10.1 Arquivos dlf.n, dlc.n e err.n

Esses três arquivos são arquivos de textos encontrados na pasta do texto. Contêm respectivamente os números de linhas de arquivos `dlf`, `dlc` e `err`. Esses números são seguidos por uma quebra de linha.

### 10.10.2 Arquivo stats_dic.n

Esse arquivo é um arquivo de texto encontrado na pasta do texto. É formado de três linhas, contendo os números de linhas dos arquivos `dlf`, `dlc` e `err`.

### 10.10.3 Arquivo stats.n

Esse arquivo texto encontra-se no diretório texto e contém uma linha da seguinte forma:

```
3949 sentence delimiters, 169394 (9428 diff) tokens, 73788 (9399) simple
 forms, 438 (10) digits¶
```

Os números indicados podem ser assim interpretados:

- `sentences delimiters`: número de separadores de frases (`{S}`);
- `tokens`: número total de unidades lexicais do texto. O número precedente `diff` indica o número de unidades diferentes;

- `simple forms`: número total no texto de unidades lexicais compostas de letras. O numero entre parênteses representa o número de unidades lexicais diferentes que são compostas de letras;

- `digits`: número total no texto de algarismos. O número entre parênteses indica o numero de algarismos diferentes utilizados (10 no máximo).

### 10.10.4 Arquivo concord.n

O arquivo `concord.n` é um arquivo texto que se encontra na pasta do texto. Contém informações sobre a última busca efetuada sobre esse texto e se apresenta da seguinte maneira:

```
6 matches¶
6 recognized units¶
(0.004% of the text is covered)¶
```

A primeira linha indica o número de ocorrências encontradas; a segunda, o número de unidades cobertas por essas ocorrências. A terceira linha indica a relação entre o número de unidades cobertas e o número total de unidades do texto.

# Anexo A  GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007



Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this

version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.
TERMS AND CONDITIONS
0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.
1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.
2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.
3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.
4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.
5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:
a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to “keep intact all notices”.
c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium,

is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.
6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:
a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the

product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.
7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.
8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”.

A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a

consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.
13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.
14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.
15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL,

INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS
How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

# Anexo B GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007





This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.
0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.
1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.
2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:
a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.
3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:
a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the object code with a copy of the GNU GPL and this license document.
4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:
a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
d) Do one of the following:
0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)
5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.
6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

# Anexo C Lesser General Public License For Linguistic Resources

Esta licença foi elaborada pela 'Université de Marne-la-Vallée e obteve a aprovação da Free Software Foundation ([1]).

Preamble

The licenses for most data are designed to take away your freedom to share and change it. By contrast, this License is intended to guarantee your freedom to share and change free data--to make sure the data are free for all their users.

This license, the Lesser General Public License for Linguistic Resources, applies to some specially designated linguistic resources -- typically lexicons, grammars, thesauri and textual corpora.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any Linguistic Resource which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License for Linguistic Resources (also called "this License"). Each licensee is addressed as "you".

A "linguistic resource" means a collection of data about language prepared so as to be used with application programs.

The "Linguistic Resource", below, refers to any such work which has been distributed under these terms. A "work based on the Linguistic Resource" means either the Linguistic Resource or any derivative work under copyright law: that is to say, a work containing the Linguistic Resource or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Legible form" for a linguistic resource means the preferred form of the resource for making modifications to it.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Linguistic Resource is not restricted, and output from such a program is covered only if its contents constitute a work based on the Linguistic Resource (independent of the use of the Linguistic Resource in a tool for writing it). Whether that is true depends on what the program that uses the Linguistic Resource does.

1. You may copy and distribute verbatim copies of the Linguistic Resource as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all

the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Linguistic Resource.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Linguistic Resource or any portion of it, thus forming a work based on the Linguistic Resource, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a linguistic resource.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Linguistic Resource, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Linguistic Resource, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Linguistic Resource.

In addition, mere aggregation of another work not based on the Linguistic Resource with the Linguistic Resource (or with a work based on the Linguistic Resource) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. A program that contains no derivative of any portion of the Linguistic Resource, but is designed to work with the Linguistic Resource (or an encrypted form of the Linguistic Resource) by reading it or being compiled or linked with it, is called a "work that uses the Linguistic Resource". Such a work, in isolation, is not a derivative work of the Linguistic Resource, and therefore falls outside the scope of this License.

However, combining a "work that uses the Linguistic Resource" with the Linguistic Resource (or an encrypted form of the Linguistic Resource) creates a package that is a derivative of the Linguistic Resource (because it contains portions of the Linguistic Resource), rather than a "work that uses the Linguistic Resource". If the package is a derivative of the Linguistic Resource, you may distribute the package under the terms of Section 4. Any works containing that package also fall under Section 4.

4. As an exception to the Sections above, you may also combine a "work that uses the Linguistic Resource" with the Linguistic Resource (or an encrypted form of the Linguistic Resource) to produce a package containing portions of the Linguistic Resource, and distribute that package under terms of your choice, provided that the terms permit modification of the package for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the package that the Linguistic Resource is used in it and that the Linguistic Resource and its use are covered by this License. You must supply a copy of this License. If the package during execution displays copyright notices, you must include the copyright notice for the Linguistic Resource among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the package with the complete corresponding machine-readable legible form of the Linguistic Resource including whatever changes were used in the package (which must be distributed under Sections 1 and 2 above); and, if the package contains an encrypted form of the Linguistic Resource, with the complete machine-readable "work that uses the Linguistic Resource", as object code and/or source code, so that the user can modify the Linguistic Resource and then encrypt it to produce a modified package containing the modified Linguistic Resource.

b) Use a suitable mechanism for combining with the Linguistic Resource. A suitable mechanism is one that will operate properly with a modified version of the Linguistic Resource, if the user installs one, as long as the modified version is interface-compatible with the version that the package was made with.

c) Accompany the package with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 4a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the package is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

If the package includes an encrypted form of the Linguistic Resource, the required form of the "work that uses the Linguistic Resource" must include any data and utility programs needed for reproducing the package from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of proprietary libraries that do not normally accompany the operating system. Such a contradiction

means you cannot use both them and the Linguistic Resource together in a package that you distribute.

5. You may not copy, modify, sublicense, link with, or distribute the Linguistic Resource except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Linguistic Resource is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Linguistic Resource or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Linguistic Resource (or any work based on the Linguistic Resource), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Linguistic Resource or works based on it.

7. Each time you redistribute the Linguistic Resource (or any work based on the Linguistic Resource), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Linguistic Resource subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Linguistic Resource at all. For example, if a patent license would not permit royalty-free redistribution of the Linguistic Resource by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Linguistic Resource.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free resource distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of data distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute resources through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Linguistic Resource is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Linguistic Resource under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License for Linguistic Resources from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Linguistic Resource specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Linguistic Resource does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Linguistic Resource into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission.

NO WARRANTY

12. BECAUSE THE LINGUISTIC RESOURCE IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LINGUISTIC RESOURCE, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LINGUISTIC RESOURCE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LINGUISTIC RESOURCE IS WITH YOU. SHOULD THE LINGUISTIC RESOURCE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LINGUISTIC RESOURCE AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LINGUISTIC RESOURCE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LINGUISTIC RESOURCE TO

OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
END OF TERMS AND CONDITIONS

# Bibliografia


[1] Free Software Foundation. http://www.fsf.org. 10.10.4

[2] Anna ANASTASSIADIS-SYMEONIDIS, Tita KYRIACOPOULOU, Elsa SKLAVOUNOU, Iasson THILIKOS, and Rania VOSKAKI. A system for analysing texts in modern greek : representing and solving ambiguities. In *Proceedings of COMLEX 2000, Workshop on Computational Lexicography and Multimedia Dictionaries*. Patras, 2000. 3.7

[3] Olivier BLANC and Anne DISTER. Automates lexicaux avec structure de traits. 2004. Actes RECITAL 2004. 7.3

[4] Xavier BLANCO. Noms composés et traduction français-espagnol. *Lingvisticæ Investigationes*, 21(1), 1997. Amsterdam-Philadelphia : John Benjamins Publishing Company. 3.7

[5] Xavier BLANCO. Les dictionnaires électroniques de l'espagnol (DELASs et DELACs). *Lingvisticæ Investigationes*, 23(2), 2000. Amsterdam-Philadelphia : John Benjamins Publishing Company. 3.7

[6] Jean-Paul BOONS, Alain GUILLET, and Christian LECLÈRE. La structure des phrases simples en français : classes de constructions transitives. Technical report, LADL, Paris, 1976. 8.1

[7] Jean-Paul BOONS, Alain GUILLET, and Christian LECLÈRE. *La structure des phrases simples en français : constructions intransitives*. Droz, Genève, 1976. 8.1

[8] Firefox.Web browser. http://www.mozilla.com/firefox/. 4.8.2

[9] Netscape. Web browser. http://www.netscape.com. 4.8.2

[10] Folker CAROLI. Les verbes transitifs à complément de lieu en allemand. *Lingvisticæ Investigationes*, 8(2) :225.267, 1984. Amsterdam-Philadelphia : John Benjamins Publishing Company. 8.1

[11] A. CHROBOT, B. COURTOIS, M. HAMMANI-MC CARTHY, M. GROSS, and K. ZELLAGUI. Dictionnaire electronique DELAC anglais : noms composés. Technical Report 59, LADL, Université Paris 7, 1999. 3.7

[12] Unicode Consortium. http://www.unicode.org. 2.2

[13] Matthieu CONSTANT and Anastasia YANNACOPOULOU. Le dictionnaire électronique du grec moderne : Conception et développement d'outils pour son enrichissement et sa validation. In *Studies in Greek Linguistics, Proceedings of the 23rd annual meeting of the Department of Linguistics*. Faculty of Philosophy, Aristotle University of Thessaloniki, 2002. 3.7

[14] Blandine COURTOIS. Formes ambiguës de la langue française. *Lingvisticæ Investigationes*, 20(1) :167.202, 1996. Amsterdam-Philadelphia : John Benjamins Publishing Company. 3.7

[15] Blandine Courtois and Max Silberztein, editors. *Les dictionnaires électroniques du français*. Larousse, Langue française, vol. 87, 1990. 3.7

[16] Anne DISTER, Nathalie FRIBURGER, and Denis MAUREL. Améliorer le découpage en phrases sous INTEX. In Anne Dister, editor, *Revue Informatique et Statistique dans les Sciences Humaines*, volume Actes des 3èmes Journées INTEX, pages 181.199, 2000. 2.5.2

[17] Anibale ELIA. *Le verbe italien. Les complétives dans les phrases à un complément*. Schena/Nizet, Fasano/Paris, 1984. 8.1

[18] Anibale ELIA. *Lessico-grammatica dei verbi italiani a completiva. Tavole e indice generale*. Liguori, Napoli, 1984. 8.1

[19] Anibale ELIA and Simoneta VIETRI. Electronic dictionaries and linguistic analysis of italian large corpora. In *Actes des 5es Journées internationales d'Analyse statistique des Données Textuelles*. Ecole Polytechnique fédérale de Lausanne, 2000. 3.7

[20] Anibale ELIA and Simoneta VIETRI. L'analisi automatica dei testi e i dizionari elettronici. In E. Burattini and R. Cordeschi, editors, *Manuale di Intelligenza Arti_ciale per le Scienze Umane*. Roma :Carocci, 2002. 3.7

[21] Jacqueline GIRY-SCHNEIDER. *Les nominalisations en français. L'opérateur faire dans le lexique*. Droz, Genève-Paris, 1978. 8.1

[22] Jacqueline GIRY-SCHNEIDER. *Les prédicats nominaux en français. Les phrases simples à verbe support*. Droz, Genève-Paris, 1987. 8.1

[23] GNU. General Public License. `http://www.gnu.org/licenses/gpl.html`. 1.1, 10.10.4

[24] GNU. Lesser General Public License. `http://www.gnu.org/licenses/lgpl.html`. 1.1, 10.10.4

[25] Gaston GROSS. *Les expressions _gées en français*. Ophrys, Paris, 1996. 3.7

[26] Maurice GROSS. *Méthodes en syntaxe*. Hermann, Paris, 1975. 8.1

[27] Maurice GROSS. *Grammaire transformationnelle du français. 3 - Syntaxe de l'adverbe*. ASSTRIL, Paris, 1986. 3.7, 8.1

[28] Alain GUILLET and Christian LECLÈRE. *La structure des phrases simples en français : les constructions transitives locatives*. Droz, Genève, 1992. 8.1

[29] IGM. Lesser General Public License for Linguistic Resources. `http://igm.univ-mlv/~unitex/lgpllr.html`. 1.1

[30] Gaby KLARSFLED and Mary HAMMANI-MC CARTHY. Dictionnaire électronique du ladl pour les mots simples de l'anglais (DELASa). Technical report, LADL, Université Paris 7, 1991. 3.7

[31] Tita KYRIACOPOULOU. *Les dictionnaires électroniques : la _exion verbale en grec moderne*, 1990. Thèse de doctorat. Université Paris 8. 3.7

[32] Tita KYRIACOPOULOU. Un système d'analyse de textes en grec moderne : représentation des noms composés. In *Actes du 5ème Colloque International de Linguistique Grecque, 13-15 septembre 2001*. Sorbonne, Paris, 2002. 3.7

[33] Tita KYRIACOPOULOU, Sa_a MRABTI, and AnastasiaYANNACOPOULOU. Le dictionnaire électronique des noms composés en grec moderne. *Lingvisticæ Investigationes*, 25(1) :7.28, 2002. Amsterdam-Philadelphia : John Benjamins Publishing Company. 3.7

[34] Jacques LABELLE. Le traitement automatique des variantes linguistiques en français : l'exemple des concrets. *Lingvisticæ Investigationes*, 19(1) :137.152, 1995. Amsterdam- Philadelphia : John Benjamins Publishing Company. 3.7

[35] Eric LAPORTE and Anne MONCEAUX. Elimination of lexical ambiguities by grammars : The ELAG system. *Lingvisticæ Investigationes*, 22 :341.367, 1998. Amsterdam-Philadelphia : John Benjamins Publishing Company. 7, 7.3

[36] Ville LAURIKARI. TRE home page. `http://laurikari.net/tre/`. 1.1, 4.7

[37] Annie MEUNIER. *Nominalisation d'adjectifs par verbes supports*, 1981. Thèse de doctorat. Université Paris 7. 8.1

[38] Sun Microsystems. Java. `http://java.sun.com`. 1.2

[39] Christian MOLINIER and Françoise LEVRIER. *Grammaire des adverbes : description des formes en -ment*. Droz, Genève, 2000. 8.1

[40] Anne MONCEAUX. Le dictionnaire des mots simples anglais : mots nouveaux et variantes orthographiques. Technical Report 15, IGM, Université de Marne-la-Vallée, 1995. 3.7

[41] OpenOffice.org. `http://www.openoffice.org`. 2.2, 8.2.2

[42] Dong-Ho PAK. *Lexique-grammaire comparé français-coréen. Syntaxe des constructions complétives*. PhD thesis, UQAM, Montréal, 1996. 8.1

[43] Soun-Nam PARK. *La construction des verbes neutres en coréen*, 1996. Thèse de doctorat. Université Paris 7. 8.1

[44] Sébastien PAUMIER and Harald ULLAND. Analyse automatique de mots polylexicaux en norvégien. *Lingvisticæ Investigationes*, 28(2), 2005. Amsterdam-Philadelphia : John Benjamins Publishing Company. 2.5.6

[45] Roger-Bruno RABENNILAINA. *Le verbe malgache*. AUPELF-UREF et Université Paris 13, Paris, 1991. 8.1

[46] Agata SAVARY. *Recensement et description des mots composés - méthodes et applications*, 2000. Thèse de doctorat. Université de Marne-la-Vallée. 3.7

[47] Max SILBERZTEIN. Les groupes nominaux productifs et les noms composés lexicalis és. *Lingvisticæ Investigationes*, 27(2) :405.426, 1999. Amsterdam-Philadelphia : John Benjamins Publishing Company. 3.7

[48] Carlos SUBIRATS-RÜGGEBERG. *Sentential complementation in Spanish. A lexicogrammatical study of three classes of verbs*. John Benjamins, Amsterdam/Philadelphia, 1987. 8.1

[49] Thomas TREIG. Complétives en allemand. classi_cation. Technical Report 7, LADL, 1977. 8.1

[50] Lidia VARGA. Classi_cation syntaxique des verbes de mouvement en hongrois dans l'optique d'un traitement automatique. In F. Kiefer, G. Kiss, and J. Pajzs, editors, *Papers in Computational Lexicography (COMPLEX)*, pages 257.265, Budapest, Research Institute for Linguistics, Hungarian Academy of Sciences, 1996. 8.1

[51] Simoneta VIETRI. On the study of idioms in italian. In *Sintassi e morfologia della lingua italiana, Congresso internazionale della Società di Linguistica Italiana*. Roma :Bulzoni, 1984. 3.7